寻找 metasploitable 的 flag

实验环境

- 操作系统: macOS Sequoia 15.6.1
- 实验平台:
  - kali
  - Metasploitable3-Win2k8

安装 Metasploitable3

Mac M1 安装 metasploitable2
参考博客

不安装 **metasploitable3** 的原因:

1. VirtualBox 原生对 Apple Silicon 的支持不成熟/不可用（多数 VirtualBox 功能是为 x86 做的），无法运行 x86 虚拟机或不稳定。
2. Metasploitable3 官方 build.sh 期望 Packer 1.6.x + VirtualBox builder（内置 builder），而 Packer 的 1.6.x 只发布 x86_64 二进制，没有 arm64 版。Packer 新版（1.7+）是插件化的，但脚本并不兼容插件化流程。

metasploitable2 安装流程
这个链接 比直接 官网 下载快很多！也可能是因为连接了美国的 VPN。

1. 安装 homebrew
2. 安装 qemu

   ```
   brew install qemu
   ```
3. 安装 UTM
   - Apple Store 需要收费，可以去官网免费下载。
   - UTM 官网，点击“Download”即可。

安装完成后,
- 默认账户: `msfadmin`
- 默认密码: `msfadmin`

Metasploitable3 安装流程
发现“扑克牌”只在 Metasploitable3 存在，于是没办法，只能再下载一个 Metasploitable3 镜像了。

有两个镜像选择:

1. Metasploitable3-Ubuntu1404，走官网。这个太耗时间了
   尝试用别人构建的 vmware 镜像 Metasploitable3-ub1404 下载发现中途失败，应该是链接失效了
2. Metasploitable3-Win2k8

本人选择了 2.

下载完成后，按照这个博客操作即可

打开虚拟机后，是一个 Windows Server 2008 的系统。按下 `Ctrl + Alt + Del`（Mac 按下 `fn + Ctrl + Alt + Del`），输入用户名和密码均为 `vagrant` 即可登录。

靶机渗透测试

Metasploitable3 与 2 操作一致

1. 查看 kali 和 metasploitable2 的 IP 地址

```
ifconfig

# metasploitable2: 192.168.64.2
# metasploitable3: 192.168.64.3
```

2. 使用 nmap 扫描 metasploitable2 的开放端口

```
# nmap 扫描
nmap -p- -sS -sV -n -v --reason --open -oA scans/target_full.xml 192.168.64.2

# -p- 全端口扫描
# -sS Tcp SYN Scan (sS) 这是一个基本的扫描方式，它被称为半开放扫描 ，Nmap发送SYN包到远程主机，但是它不会产生任何会话，因此不会在目标主机上产生任何日志记录，因为没有形成会话
# -sV  版本检测是用来扫描目标主机和端口上运行的软件的版本
# --open 仅仅显示开启的端口
# --reason 显示端口处于特殊状态
# -n  不进行dns解析操作 （本地搭建环境，不用dns解析）
# -v  提高输出信息的详细度
# -oX XML XML格式 （我们需要导入到metasploit里面，以便我们更好的 下次查看）
```

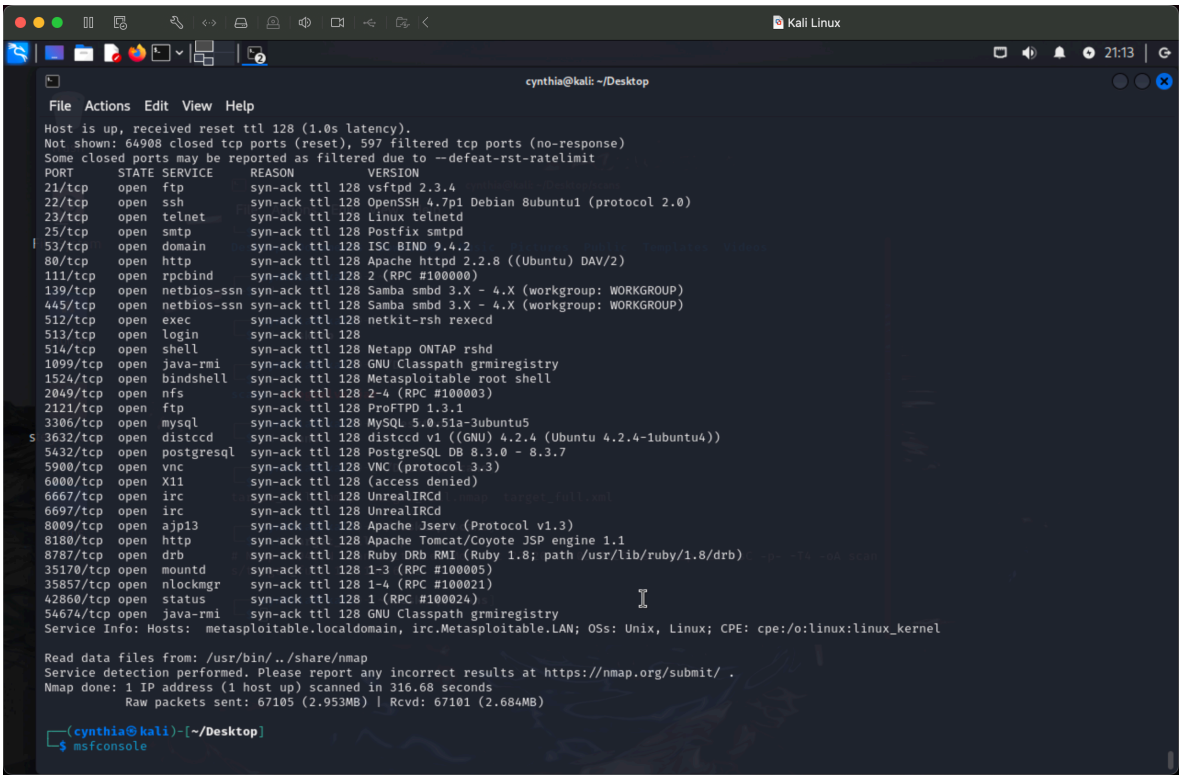


Figure 1: 扫描成功

3. kali 打开 postgresql 数据库

```
sudo systemctl enable --now postgresql

# 检查·postgresql是否启动
sudo systemctl status postgresql
```

4. 启动 metasploit 数据库

```
sudo msfdb init

# 自己配置
cd ~/.msf4
nano database.yml
```

配置内容如下:

```
production:
  adapter: postgresql
  database: msf
  username: msf
  password: Cynthia
  host: localhost
  port: 5432
  pool: 75
  timeout: 5
```

这会在 Kali 上创建 msf 数据库/用户并写入配置。同样在 Kali 中执行 并没有创建 db 和 role

5. 创建 msf 数据库和用户

```
# 创建 msf 角色
sudo -- sh -c "su - postgres -c \"psql -c \\\"DO \$\$ BEGIN IF NOT EXISTS (SELECT
FROM pg_roles WHERE rolname='msf') THEN CREATE ROLE msf LOGIN PASSWORD 'Cynthia';
ELSE ALTER ROLE msf WITH PASSWORD 'Cynthia'; END IF; END \$\$;\\\"\""

# 创建 msf 数据库
sudo -- sh -c "su - postgres -c \"psql -c \\\"CREATE DATABASE msf OWNER msf;\\\"\"
|| true"
```

6. 测试 msf 用户连接 postgresql

```
PASSWORD='youpwd' psql -U msf -h localhost -d msf -c '\l'
```

输入密码 youpwd，如果成功连接则说明配置正确。

7. 使用 Metasploit 进行漏洞利用，kali 输入命令

```
msfconsole
```

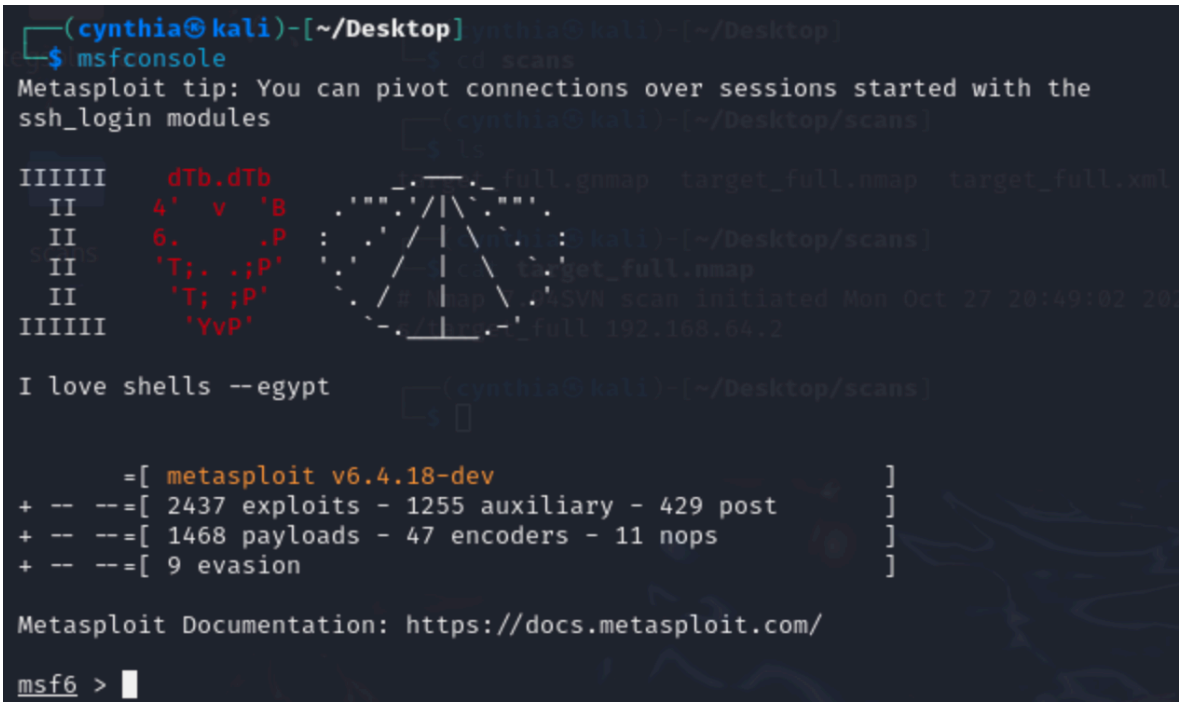


Figure 2: msfconsole

8. 创建工作环境并导入扫描结果

```
workspace -a metasploitable2
db_import /scans/demon.xml
services
```

9. 使用 telnet 连接 metasploitable2

```
telnet <metasploitable2的IP地址>
msfadmin
msfadmin
```

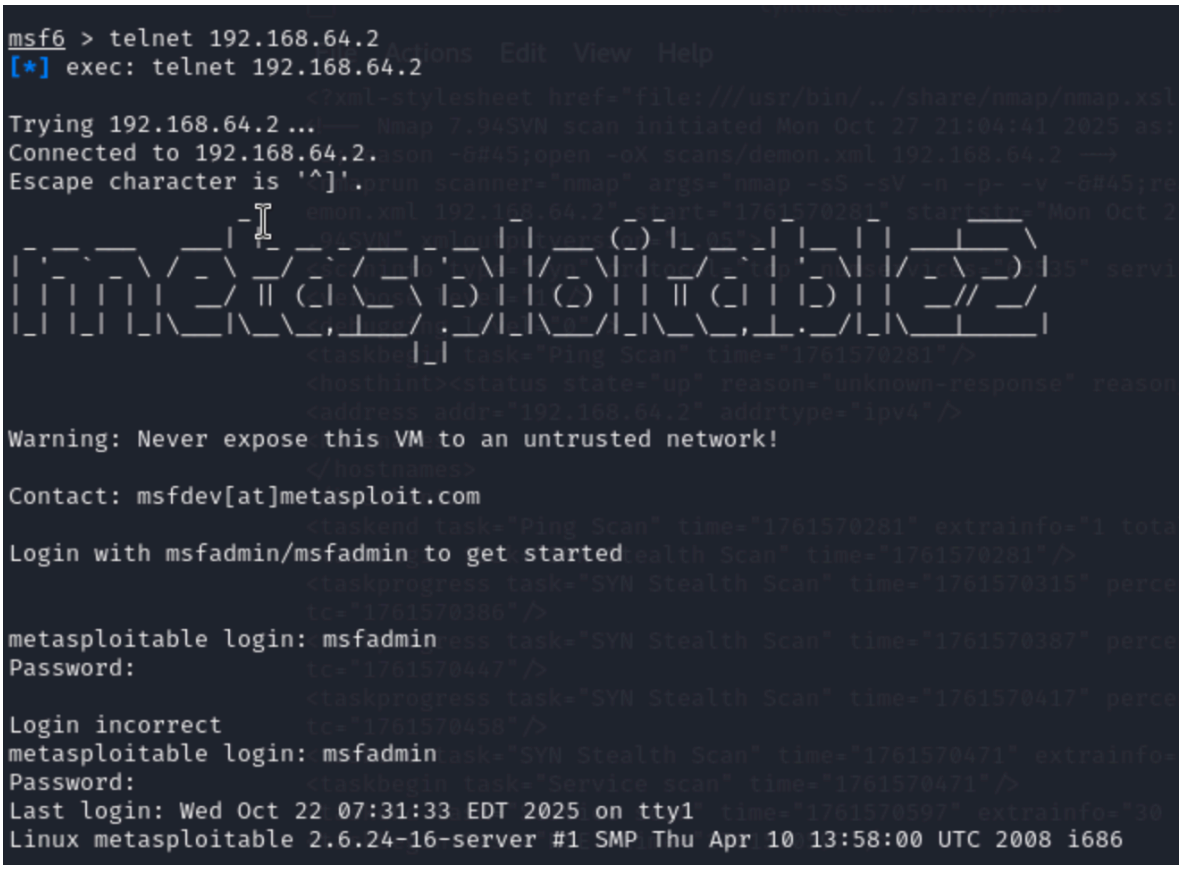
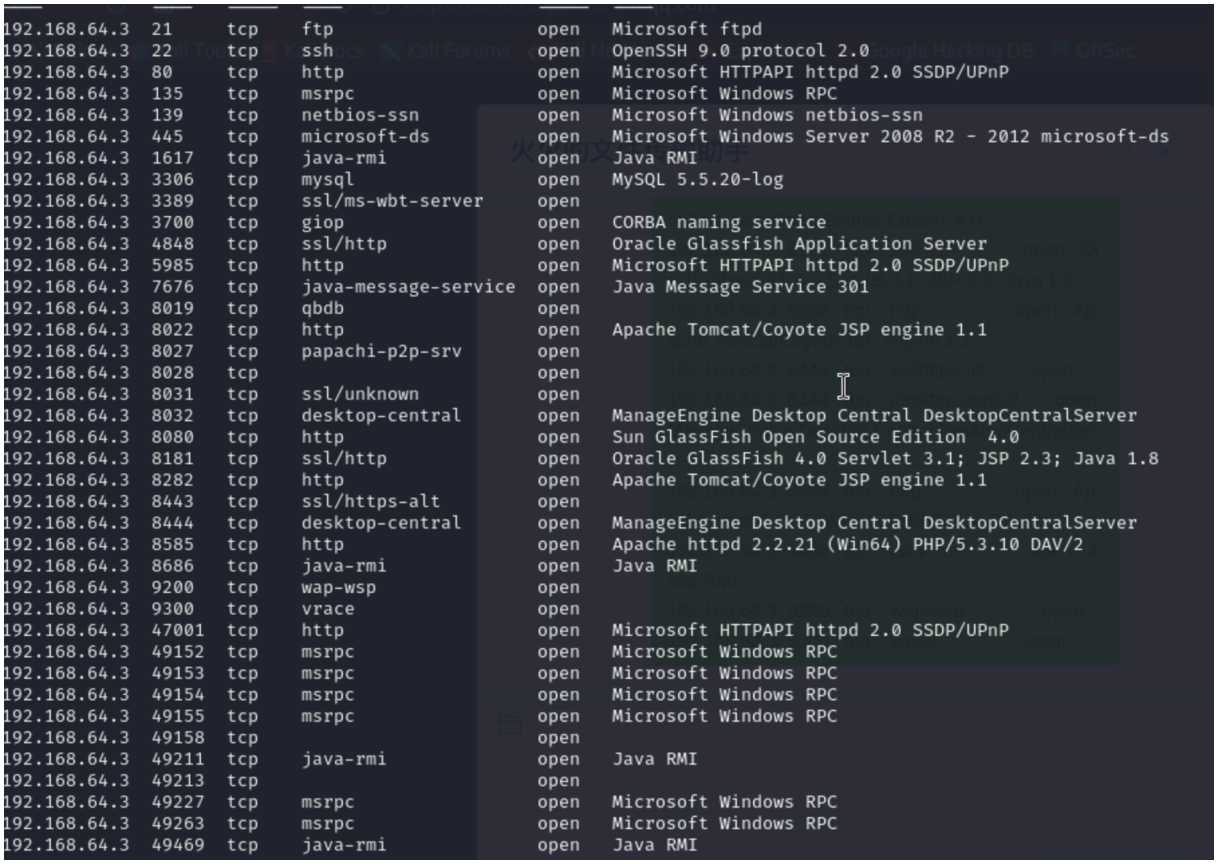


Figure 3: telnet 连接成功

15 个 flag

six_of_diamonds

我们输入 `msfconsole` 后，输入命令 `work space -a metasploitable3` 创建新的工作环境。导入新的 `demo.xml` 扫描结果，输入命令 `db_import /scans/demo.xml`。 输入 `services` 查看服务。



```
192.168.64.3  21     tcp    ftp                  open   Microsoft ftpd
192.168.64.3  22     tcp    ssh                  open   OpenSSH 9.0 protocol 2.0
192.168.64.3  80     tcp    http                 open   Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
192.168.64.3  135    tcp    msrpc                open   Microsoft Windows RPC
192.168.64.3  139    tcp    netbios-ssn          open   Microsoft Windows netbios-ssn
192.168.64.3  445    tcp    microsoft-ds         open   Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
192.168.64.3  1617   tcp    java-rmi             open   Java RMI
192.168.64.3  3306   tcp    mysql                open   MySQL 5.5.20-log
192.168.64.3  3389   tcp    ssl/ms-wbt-server    open
192.168.64.3  3700   tcp    giop                 open   CORBA naming service
192.168.64.3  4848   tcp    ssl/http             open   Oracle Glassfish Application Server
192.168.64.3  5985   tcp    http                 open   Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
192.168.64.3  7676   tcp    java-message-service open   Java Message Service 301
192.168.64.3  8019   tcp    qbdb                 open
192.168.64.3  8022   tcp    http                 open   Apache Tomcat/Coyote JSP engine 1.1
192.168.64.3  8027   tcp    papachi-p2p-srv      open
192.168.64.3  8028   tcp                         open
192.168.64.3  8031   tcp    ssl/unknown          open
192.168.64.3  8032   tcp    desktop-central      open   ManageEngine Desktop Central DesktopCentralServer
192.168.64.3  8080   tcp    http                 open   Sun GlassFish Open Source Edition  4.0
192.168.64.3  8181   tcp    ssl/http             open   Oracle GlassFish 4.0 Servlet 3.1; JSP 2.3; Java 1.8
192.168.64.3  8282   tcp    http                 open   Apache Tomcat/Coyote JSP engine 1.1
192.168.64.3  8443   tcp    ssl/https-alt        open
192.168.64.3  8444   tcp    desktop-central      open   ManageEngine Desktop Central DesktopCentralServer
192.168.64.3  8585   tcp    http                 open   Apache httpd 2.2.21 (Win64) PHP/5.3.10 DAV/2
192.168.64.3  8686   tcp    java-rmi             open   Java RMI
192.168.64.3  9200   tcp    wap-wsp              open
192.168.64.3  9300   tcp    vrace                open
192.168.64.3  47001  tcp    http                 open   Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
192.168.64.3  49152  tcp    msrpc                open   Microsoft Windows RPC
192.168.64.3  49153  tcp    msrpc                open   Microsoft Windows RPC
192.168.64.3  49154  tcp    msrpc                open   Microsoft Windows RPC
192.168.64.3  49155  tcp    msrpc                open   Microsoft Windows RPC
192.168.64.3  49158  tcp                         open
192.168.64.3  49211  tcp    java-rmi             open   Java RMI
192.168.64.3  49213  tcp                         open
192.168.64.3  49227  tcp    msrpc                open   Microsoft Windows RPC
192.168.64.3  49263  tcp    msrpc                open   Microsoft Windows RPC
192.168.64.3  49469  tcp    java-rmi             open   Java RMI
```

Figure 4: services 查看服务

暴力破解 ftp 服务:

1. 安装 `hydra`

```
sudo apt update
sudo apt install hydra
```

2. 使用 `hydra` 进行暴力破解

```
# 查找 password 文件在哪里
sudo find /usr -type f -iname '*password*' 2>/dev/null

# 使用  hydra
hydra -L ftpusername.txt -P /usr/share/metasploit-framework/wordlists/password.lst
-f <metasploitable3的IP地址> ftp -v
```

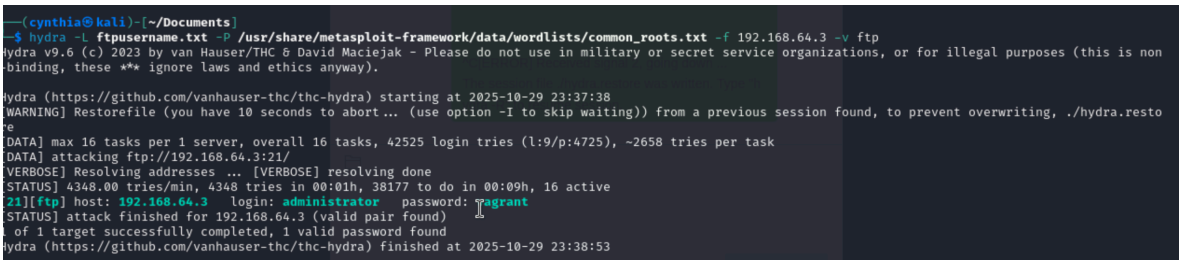


Figure 5: hydra 破解成功

3. 登陆 ftp

```
ftp <metasploitable3的IP地址>

# 输入用户名和密码
ftpusername
ftppassword

# ls 查看文件
ls
# mget 下载全部文件
mget *
```

4. 发现 six+of_diomonds.zip， unzip 发现文件损坏，尝试恢复文件

```
cp six_of_diamonds.zip cp_six_of_diamonds.zip
zip -FF cp_six_of_diamonds.zip --out fixed_six_of_diamonds.zip
unzip fixed_six_of_diamonds.zip
```

发现能够解压，但是文件还是损坏，只能得到一小部分 six_of_diamonds.png

5. 尝试别的方法。点击 index.html，发现在 ftp 中看到的 hahaha.jpg

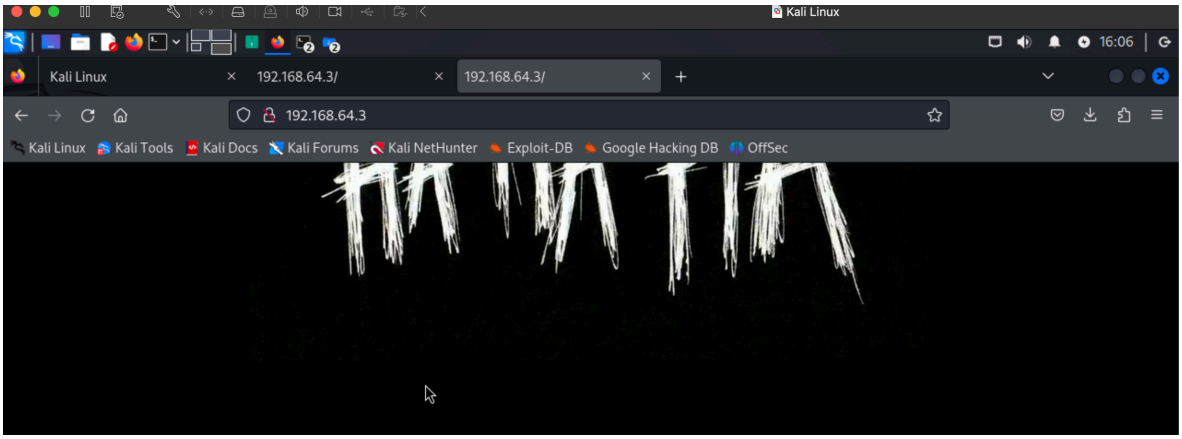


Figure 6: hahaha.jpg

6. 右键 open image in new tab，发现 url = http://<metasploitable3 的 IP 地址>/hahaha.jpg，从这里访问 six_of_diamonds.zip，成功下载完整的压缩包。

7. 解压 six_of_diamonds.zip，得到 six_of_diamonds.png

```
unzip -P vagrant six_of_diamonds.zip
```

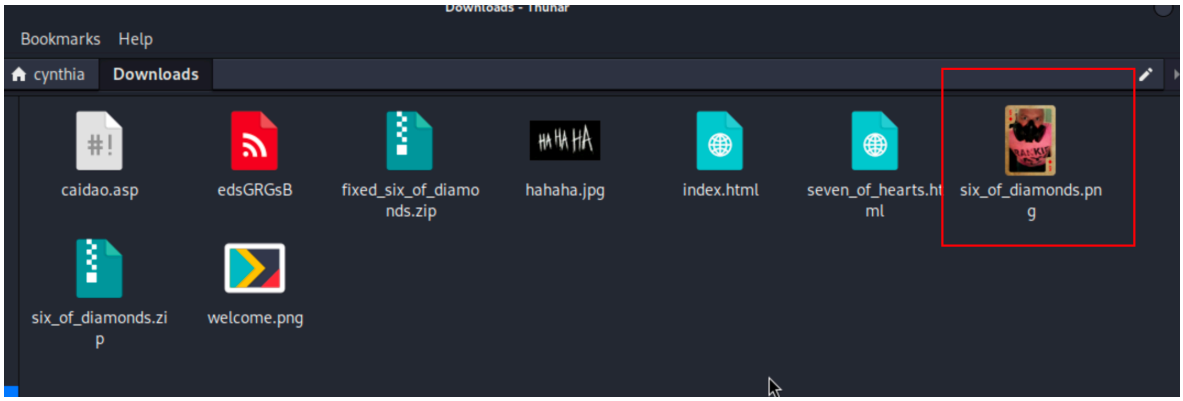


Figure 7: six_of_diamonds.png

seven of hearts

需要用 burpsuite，需要要升级 jdk 到 21+　　具体参考博客

1. 打开 seven_of_hearts.html，发现网页有一张打不开的图片
2. F12 查看网页源代码，标签下面的 src 很长，像 base64 编码
3. 用 burpsuite 拦截响应，访问 192.168.64.3/seven_of_hearts.html

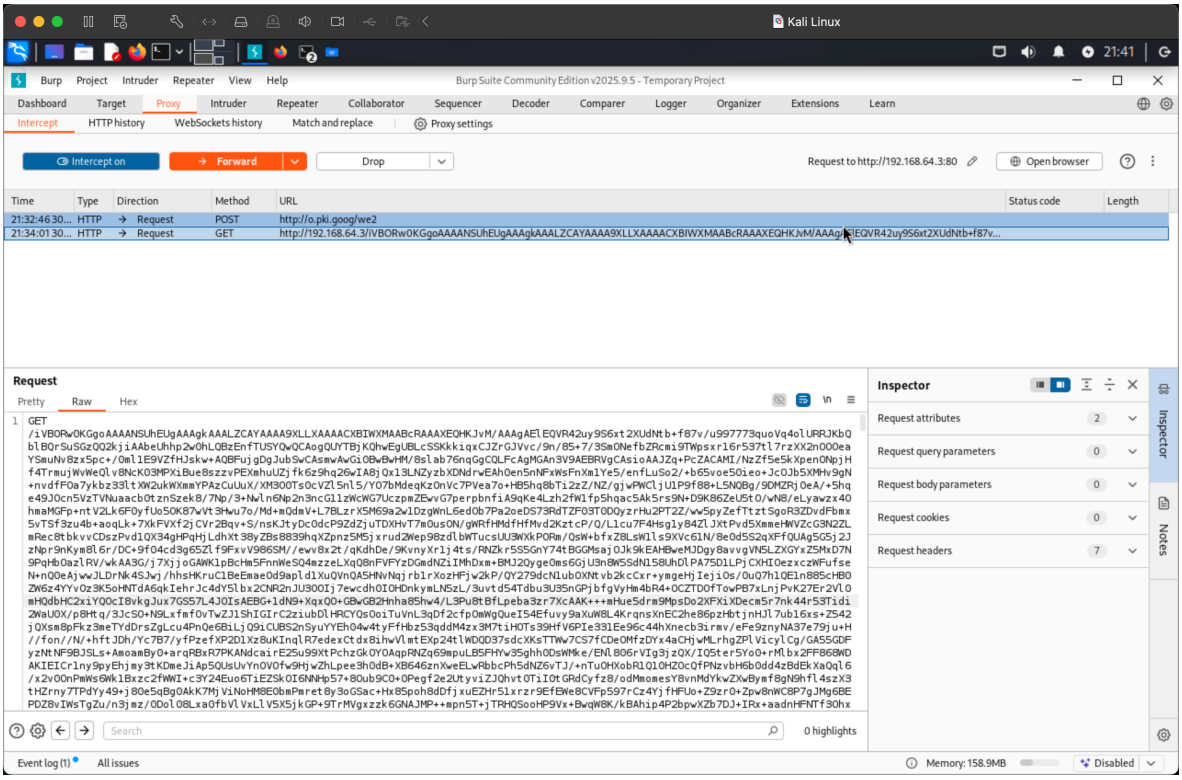


Figure 8: burpsuite 得到 url

4. 导出文件为 seven_of_hearts.url
5. 过滤掉 GET 等字符，得到 seven_of_hearts.b64

```
# url -> b64
cat seven_of_hearts2.url | perl -ne 'print $1 if(/GET \/(.*)HTTP/g)' >
seven_of_hearts.b64

# base64 解码
base64 -d seven_of_hearts.b64 > seven_of_hearts.png

# 查看文件信息
file seven_of_hearts.png
```

king_of_hearts

1. 打开 `192.168.64.3:8585`

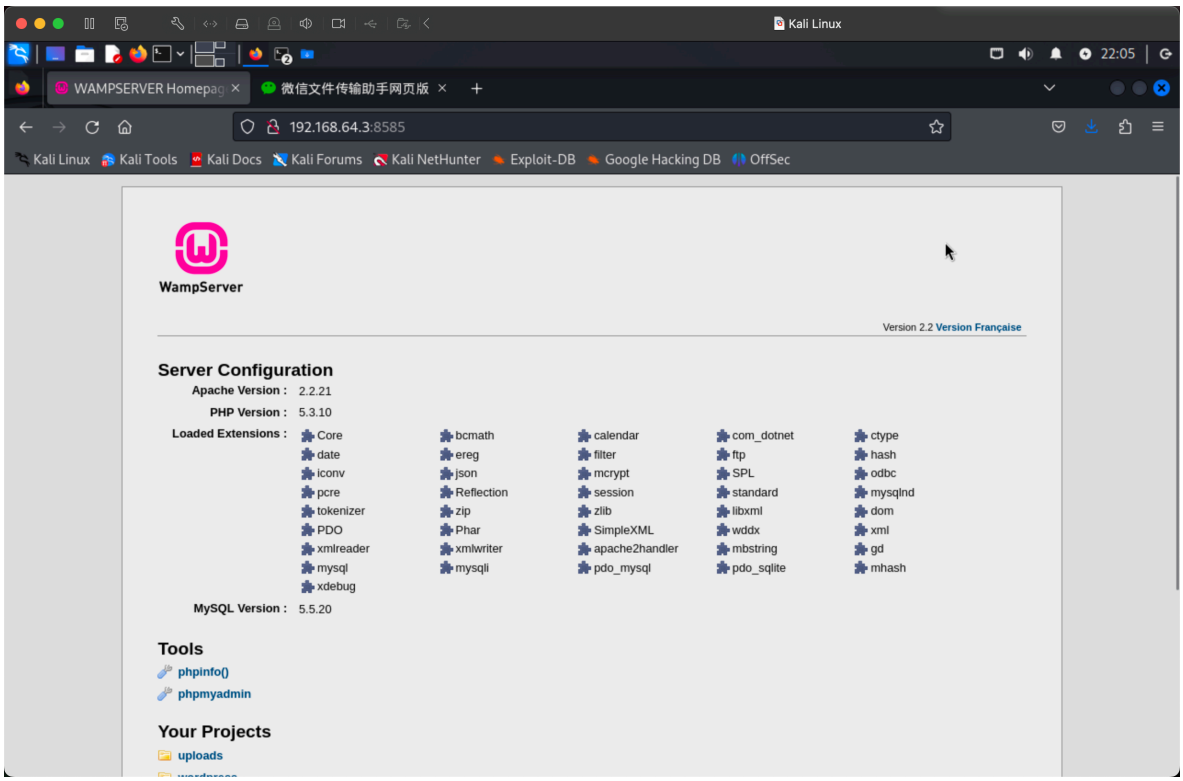


Figure 9: 8585 端口页面

2. 依次点击，要么没权限，要么看不出什么，但是进入 `wordpress` 发现右上角有 `king_of_hearts`

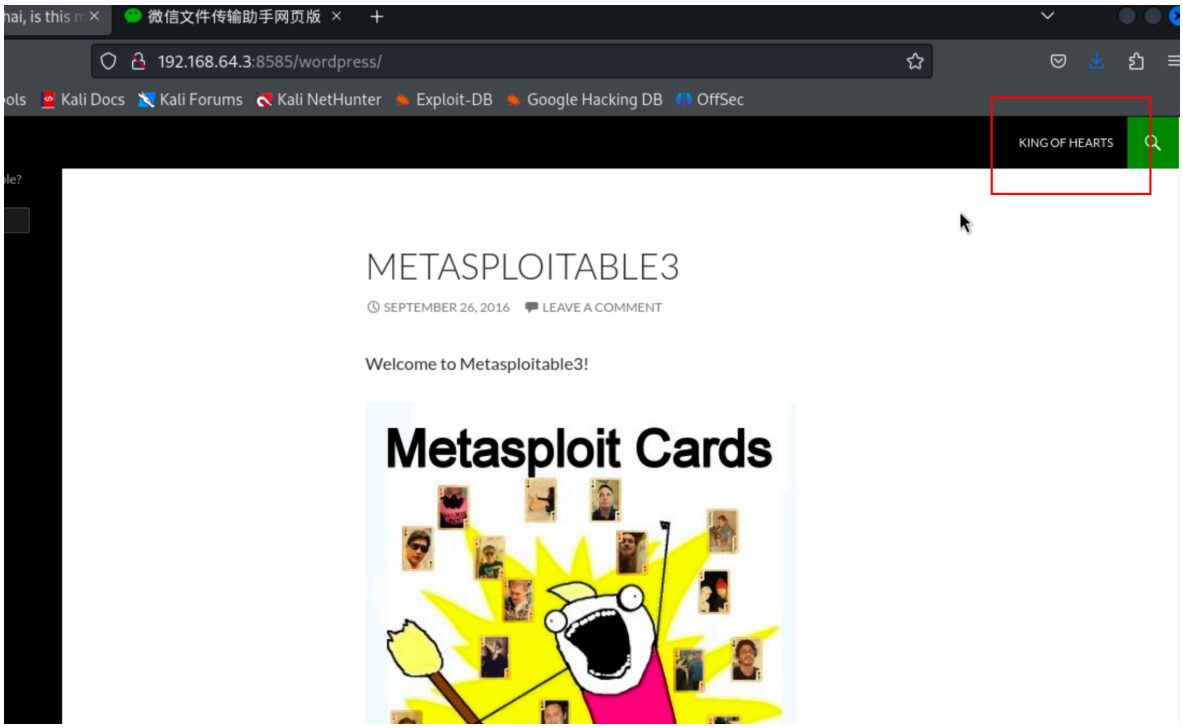


Figure 10: king of hearts

3. 直接保存就是 .png

JOKER

1. `192.168.63.3:80` 直接用 F12 看源代码，发现注释说 `hidden`

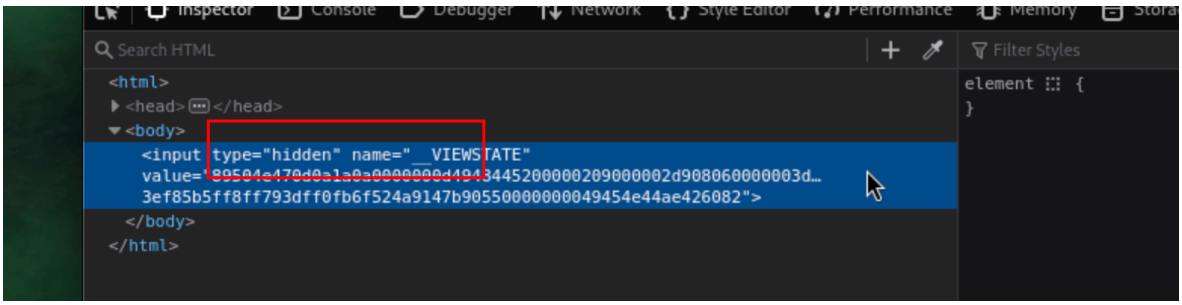


Figure 11: hidden hex

2. 把 `value` 保存为 `.hex`，转成二进制后发现是 `png data`，直接重命名即可

```
curl http://192.168.64.3:80/ > JOKER.html
cat data.html | perl -ne 'print $1 if(/value\=\"(\w+)\"\>/g)' > JOKER.hex

# 转二进制
sed 's/[^0-9A-Fa-f]//g' JOKER.hex | xxd -r -p > JOKER.bin

# png data
file JOKER.bin

# 重命名
mv JOKER.bin JOKER.png
```

ace of hearts

1. 使用 hydra 进行 ssh 暴力破解

```
# 查找 password 文件在哪里
sudo find /usr -type f -iname '*usernames*' 2>/dev/null

# 使用  hydra
hydra -L ftpusername.txt -P /usr/share/seclists/Usernames/top-usernames-
shortlist.txt -f ssh://<metasploitable3的IP地址>
```

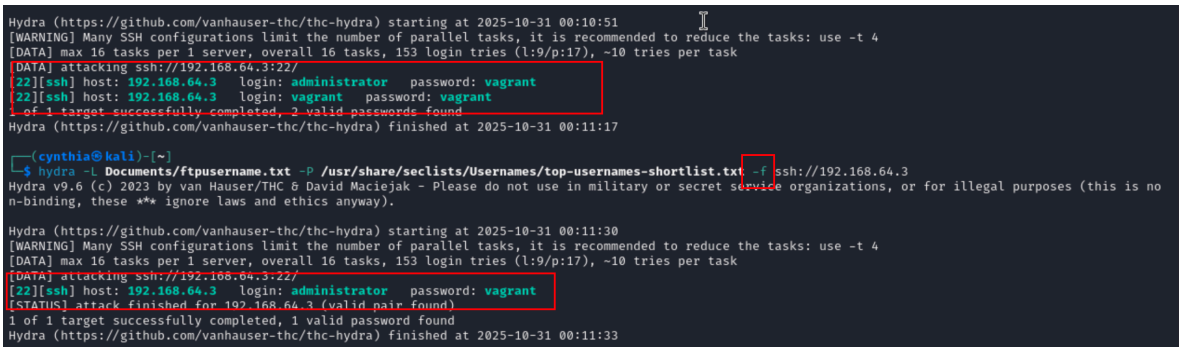


Figure 12: ssh 爆破

得到两组:

username	password
administrator	vagrant
vagrant	vagrant

2. ssh 登陆 vagrant

```
ssh vagrant@192.168.64.3
```

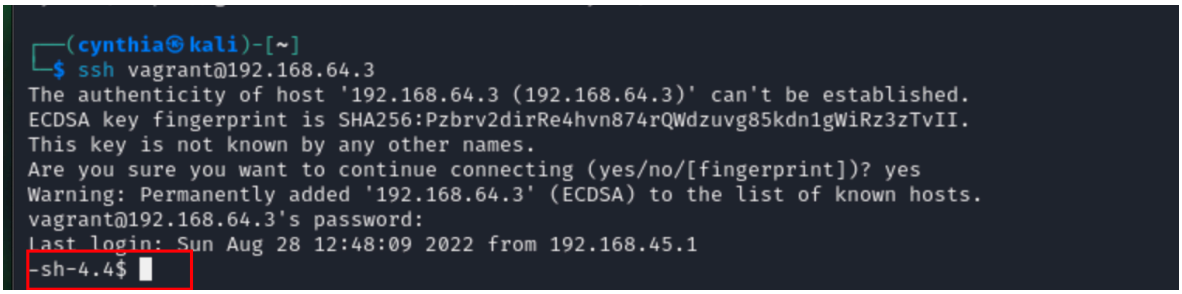


Figure 13: ssh 登陆成功

3. 进入 Users/Public/Pictures 目录，发现有两张图片

这个还是进入输入 cmd，用命令 `C:> dir /s /b *hearts*` 查找吧，其余也是如此。找完文件再解密效率更高。

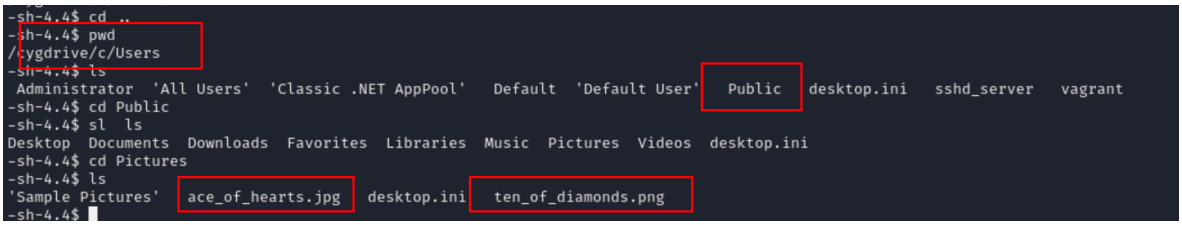


Figure 14: ace of hearts & ten of diamonds

4. 从远程服务器下载到本地

```
# 注：有空格的path用单引号括起来
scp adiministrator@192.168.64.3:/path/to/dir Downloads

# 之后输入靶机用户admin的密码
```

5. 首先查看 ace_of_hearts.jpg，格式不对，binwlk 查看一下，发现有压缩包

```
binwalk ace_of_hearts.jpg
```

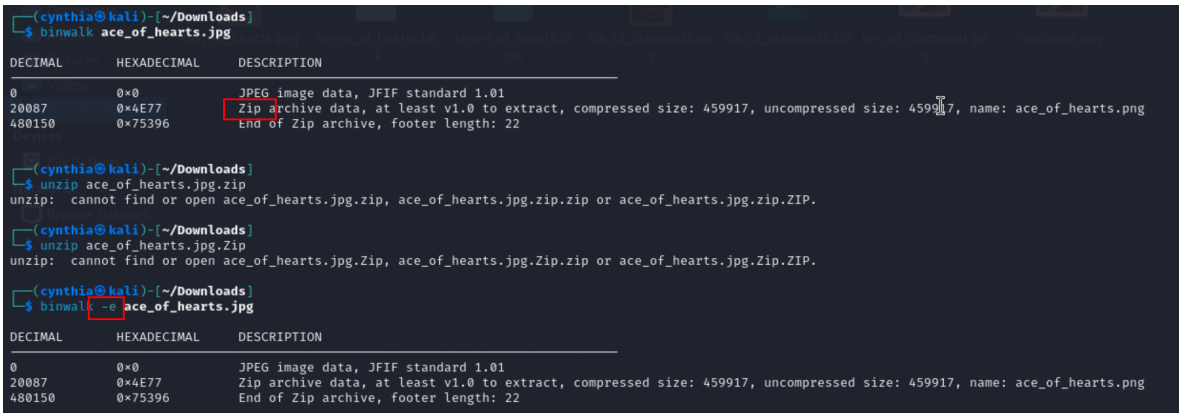


Figure 15: jpg 有压缩包

6. 直接 unzip 会破坏 zip，因为 jpg 偏移了 20087 才嵌入了压缩包，要么从 20087 切出来 zip，要 binwalk 自动提取

```
# 切片
dd if=ace_of_hearts.jpg of=embedded.zip bs=1 skip=20087 status=progress

# 解压
unzip embedded.zip

# binwalk
binwalk -e ace_of_hearts.jpg
```

7. 解压即可看到 ace_of_hearts.png

ten of diamonds

1. `ten_of_diamonds.png` 没有扑克牌样式，binwalk 看一下

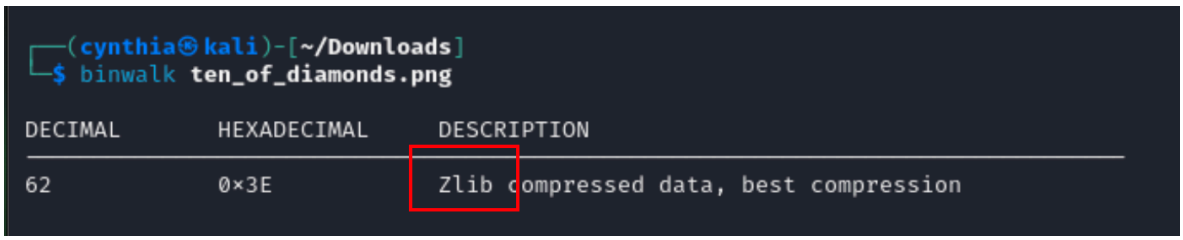


Figure 16: png 是 Zlib 压缩包

2. 显示 `ten_of_diamonds.png` 发现文件二进制数据错误

Figure 17: png 二进制数据错误

3. 查看 png 头，发现头错误

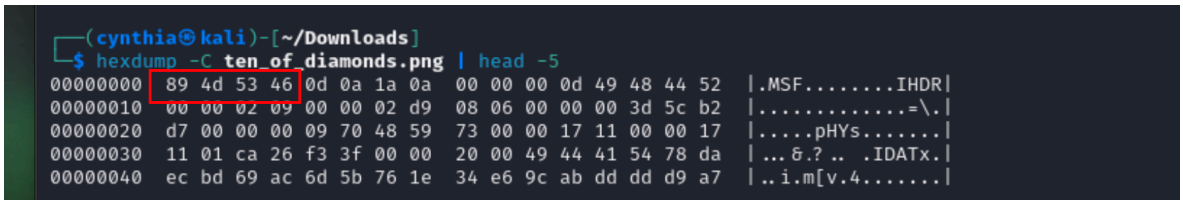


Figure 18: error png header

发现文件签名是 `89 4d 53 46 0d 1a 0a`，正确 png 签名因该是 `89 50 4E 0D 0A 1A 0A`

4. 修改 header

```
cp ten_of_diamonds.png cp_ten_of_diamonds.png

hexedit ten_of_diamonds.png
```

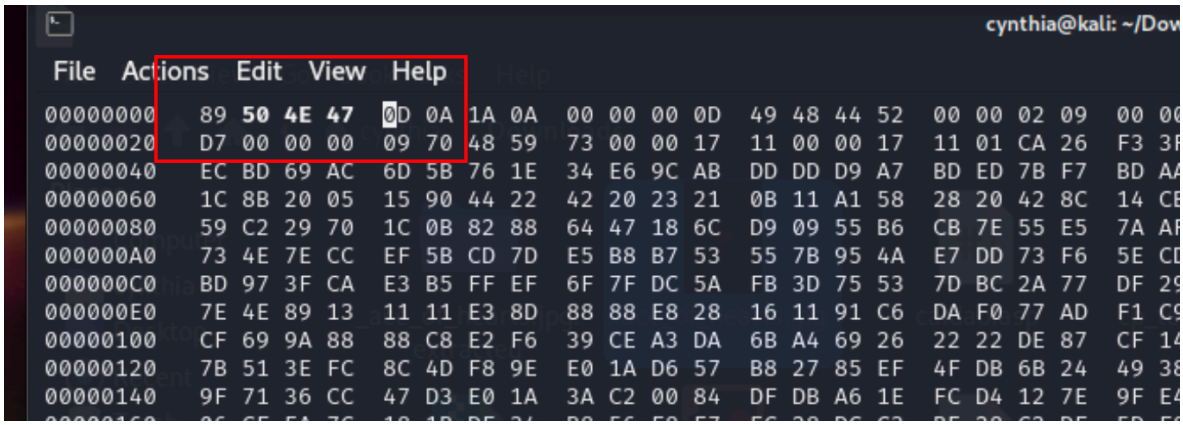


Figure 19: 修改为正确 png 头

Ctrl+X 修改完保存，即可查看正确 png

jack of diamonds

1. 在靶机 System 下直接就有 `jack_of_diamonds.png`

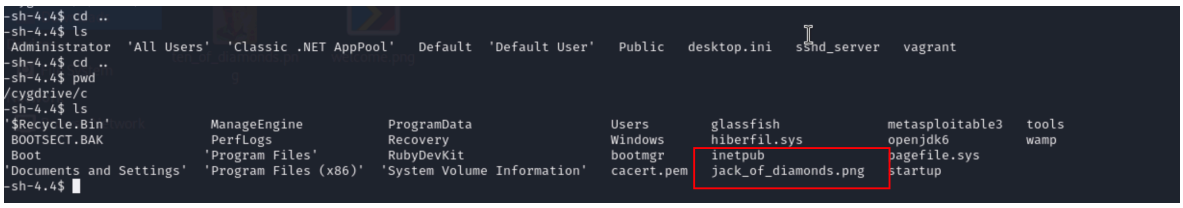


Figure 20: jack_of_diamonds.png

2. 发现 png 是空的，png 格式也不正确。登陆到靶机，调用 cmd.exe，查看是否有隐藏数据流

```
cmd.exe

dir /R jack_of_diamonds.png
```

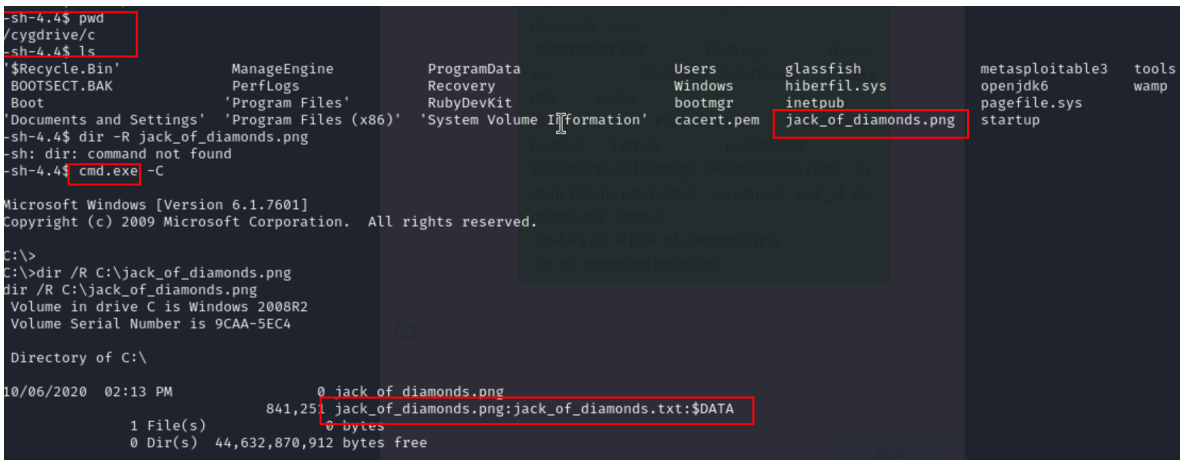


Figure 21: 调用 Windows 的 cmd

发现有隐藏数据流

3. 导出数据流为 base64 编码格式

```
# 下面操作会 not enough memory
more < jack_of_diamonds.png:jack_of_diamonds.txt > jack_of_diamonds.b64

# powershell（自动分块处理）导出命令流
powershell.exe -NoProfile -Command "Get-Content -Path 'jack_of_diamonds.png' -
Stream 'jack_of_diamonds.txt' -Raw | Set-Content -Path 'jack_of_diamonds.b64' -
Encoding Ascii"
```

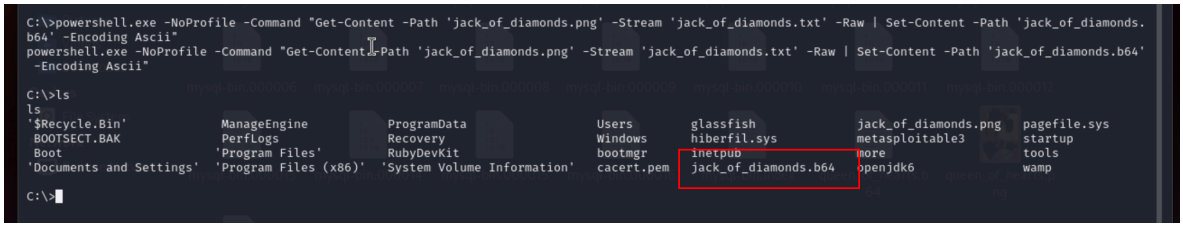


Figure 22: powshell 成功导出 .b64

4. 下载并解码

```
# 下载
scp vagrant@ip:/cygdrive/c/jack_of_diamonds.b64 .

# 解码
base64 -d jack_of_diamonds.b64 > jack_of_diamonds.png
```

可见 png

jack of hearts

1. `/xxx/Users/Public/Documents/` 下面还有两个 flag

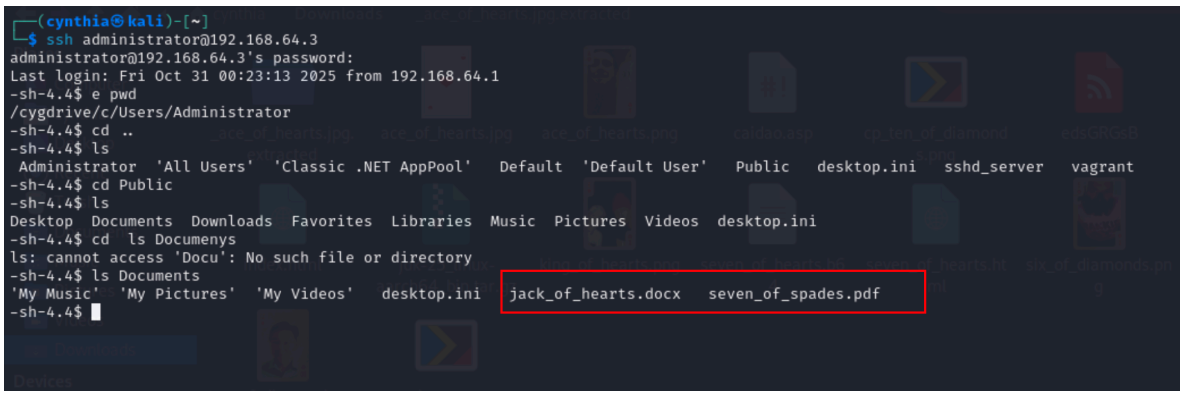


Figure 23: Documents 的 flag

2. binwalk 一下发现有 zip 存在，提取后直接在 word/media 里面发现 png

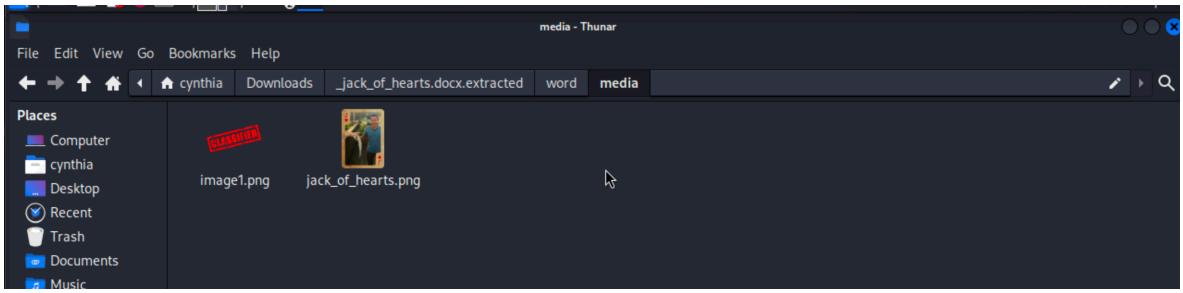


Figure 24: jack of hearts

seven of spades

1. 提取 pdf 中图片

```
# 提取
pdfimages -all -p seven_of_spades.pdf .
# 查看
ls -la
# 查看图片
display .-001-000.png
# 重命名
mv .-001-000.png seven_of_spades.png
```

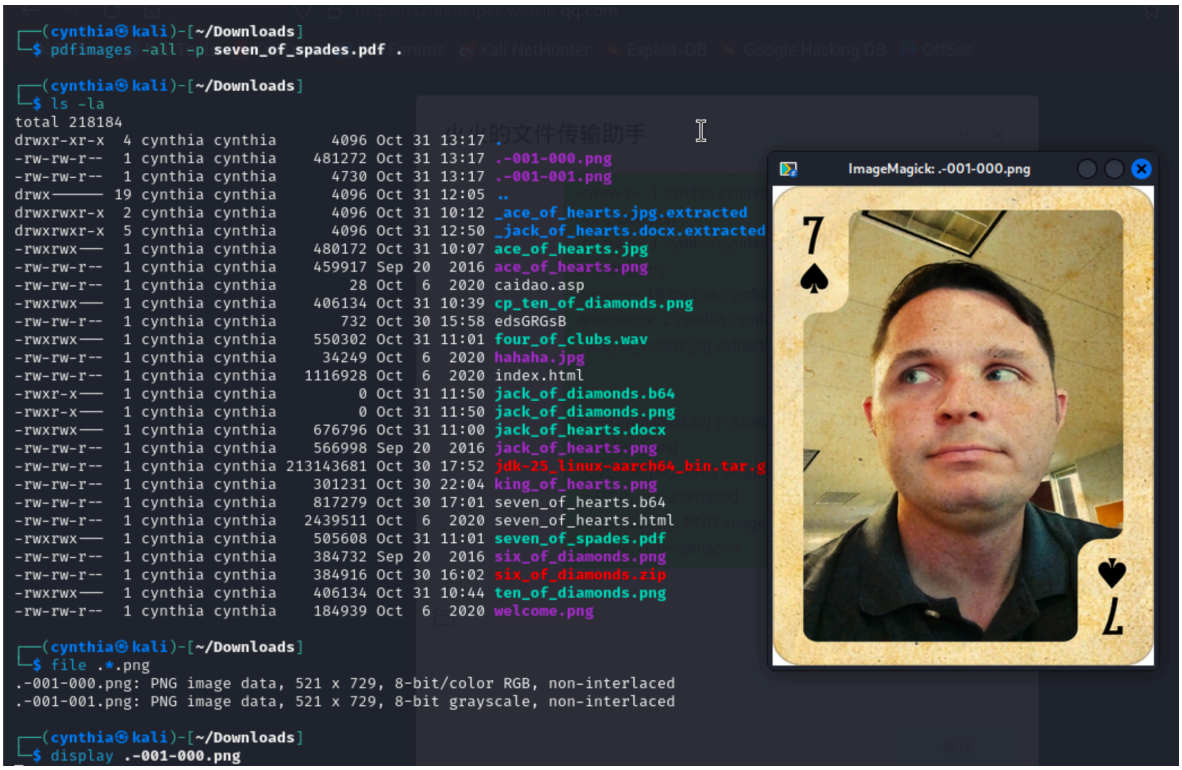


Figure 25: seven of spades

four of clubs

1. binwalk 看一下，发现附带了 png，foremost 分离文件

   ```
   foremost four_of_clubs.wav
   ```

   得到 output 文件夹，在 output/png/中可见 flag

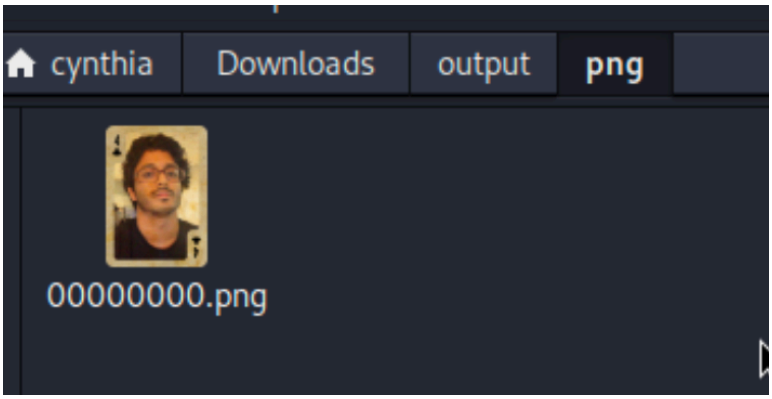


Figure 26: four of clubs

king of diamonds

1. 访问 `192.168.64.3:8585/wordpress`，点击 login
2. 需要登陆，随便试了一下，用 `vagrant:vagrant` 进去了
3. 随便看看，点击 Media，里面有三张图片，其中就有 `king_of_diamonds.png`

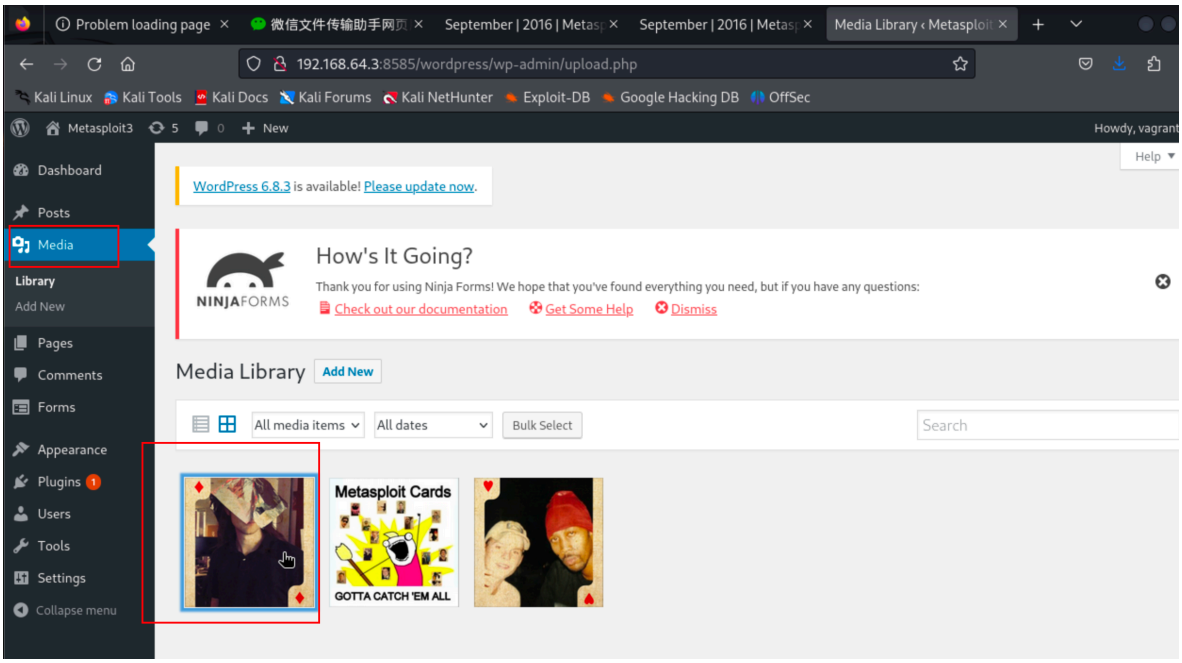


Figure 27: 找到 king_of_diamonds

4. 点击图片，得到 url，打开下载即可

three of spades

1. ssh 连接，继续翻找，发现 `/cygdrive/c/Windows` 下有 `three_of_spades.png`（事实上，它是隐藏文件，如果在 cmd 查找，需要查找隐藏文件）
2. 下载到 kali，binwalk 发现语法 Warning，`invalid escape sequence '\.'  self.period = re.compile("\.)`
3. 查看原始十六进制

   ```
   hexdump -C three_of_spades.png | head -n 5
   ```

   发现不是 png 文件

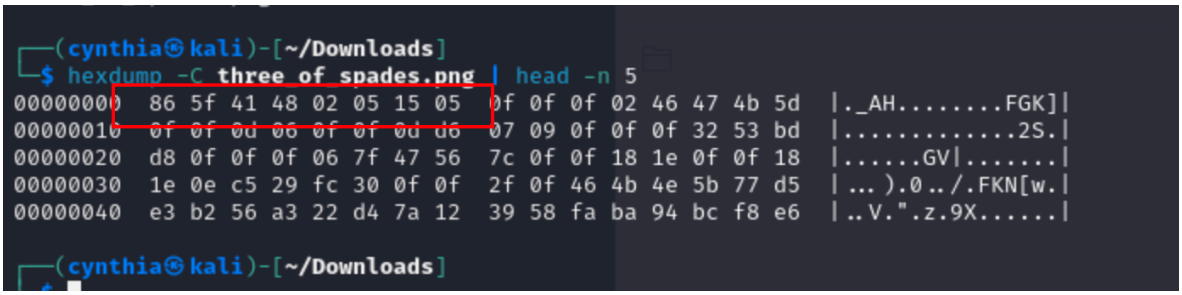


Figure 28: 文件被修改为 png

4. 但是这个文件的十六进制正常显示，对比两个 header（转二进制），每 4 位异或 4 位，即，密钥是 0f！写个 python 脚本转换这个 png。

   ```
   # 创建并编辑脚本
   nano xor_png.py

   # 脚本内容
   #!/usr/bin/python
   fin = 'three_of_spades.png'
   fout = 'three_of_spades_out.png'

   ba = bytearray(open(fin 'rb').read())
   for i in range(len(ba)):
     ba[i] ^= 0x0f
   open(fout, 'wb').write(ba)

   # 运行脚本
   python xor_png.py
   ```

5. 运行结束后，生成的 `three_of_spades_out.png` 即为 flag

Figure 29: three of spades

jack of clubs

1. 在 `C:\Windows\System32` 下发现了 `JACK_OF_CLUBS.PNG`
   `where *.png`

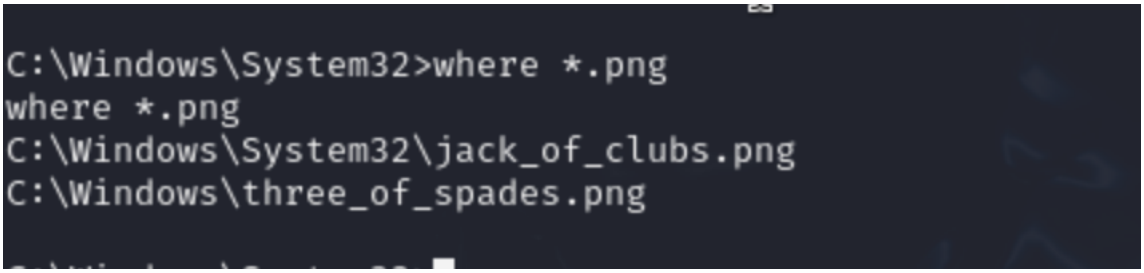


Figure 30: 翻找到 JACK_OF_CLUBS.PNG

2. 下载过来直接打开就是

king of clubs

UPX 是一个开源的可执行文件压缩工具。它的作用是:

- 主要目的: 减小可执行文件的体积，方便网络传输和存储。
- 工作原理: 将原始的可执行文件“打包”或“压缩”到一个新的外壳里。当你运行这个被压缩的文件时，这个外壳会先在内存中将原始程序解压，然后再执行它。

1. 在 `C:\Windows\System32` 下发现了 `kingofclubs.exe`
   `where "*of*"`

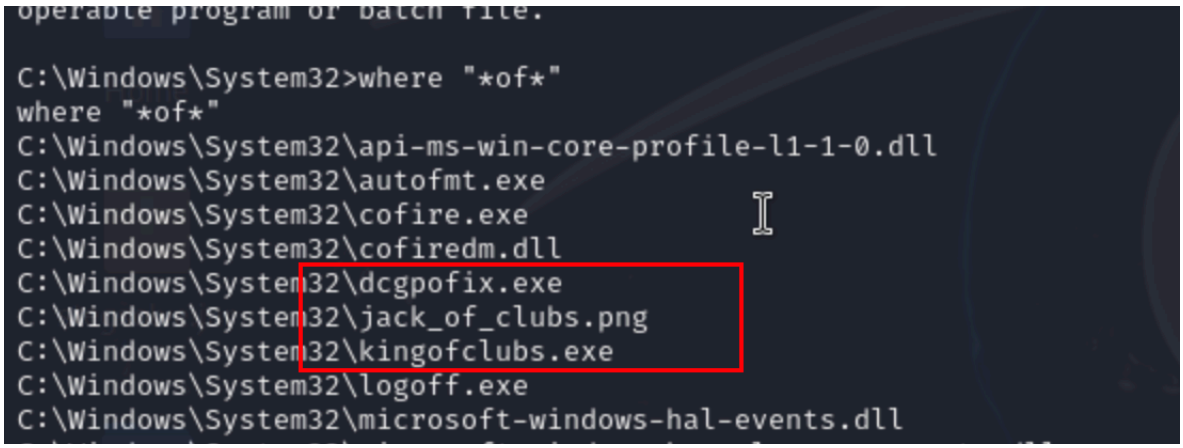


Figure 31: 翻找到 kingofclubs.exe

2. file 一下，感觉可能套 UPX 壳

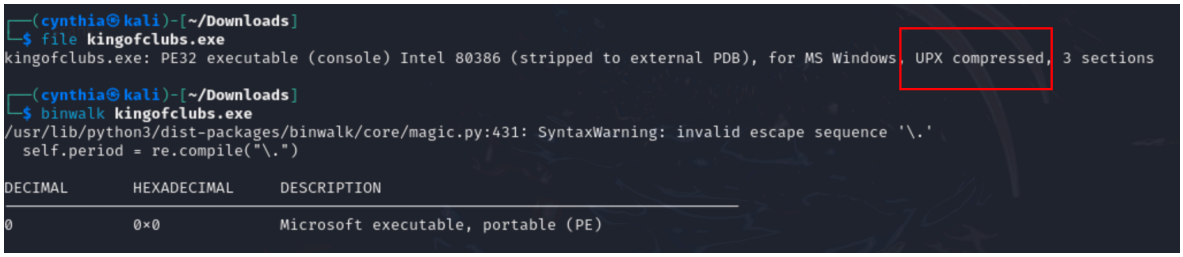


Figure 32: UPX Compressed

3. 拆壳 `upx -d kingofclubs.exe`

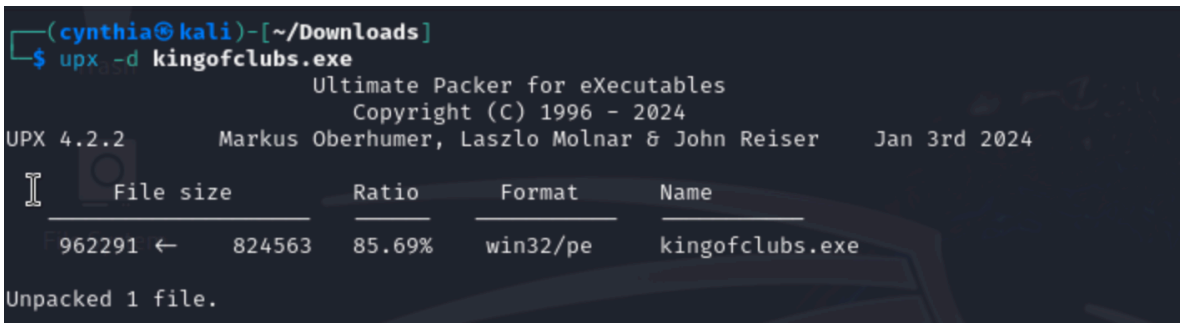


Figure 33: 解压

4. 查看十六进制数据，并查找 86 5f ... 出现的地方

```
hexdump -C kingofclubs.exe | grep -A 1 _AH
```

5. 找到位置后截断，把后面的内容写进 `kingofclubs.bin`

```
dd if=kingofclubs.exe of=kingofclubs.bin bs=1 skip=$((0x3e000))
```

6. 查看.bin 头是否是 png 头，若是，将其用 `xor_png.py` 转成真正的 png 即可

queen of hearts

1. ssh 进入
   `C:\wamp\bin\mysql\mysql5.5.20\data`
2. 把里面的文件都下载下来

```
scp -r vagrant@ip:/xx/c/wamp/x/data/ .
```

1. 依次查看日志

```
# 翻看每一个 .00000x 文件
mysqlbinlog mysql-bin.000003 | less
```

Figure 34: 有关 queen_of_hearts 操作内容

日志看到插入 queen_of_hearts 表的内容

2. 把日志保存进 `binlog.txt`

```
# filter
mysqlbinlog mysql-bin.000003 > binlog.txt
```

3. 过滤出 base64 的内容

```
# filter
grep -o "insert into queen_of_hearts values ('[^']*')" log.txt \
| sed "s/insert into queen_of_hearts values ('//" \
| sed "s/')//" > queen_of_hearts.b64
```

4. 解析 base64 为 png

```
base64 -d queen_of_hearts.b64 > queen_of_hearts.png
```

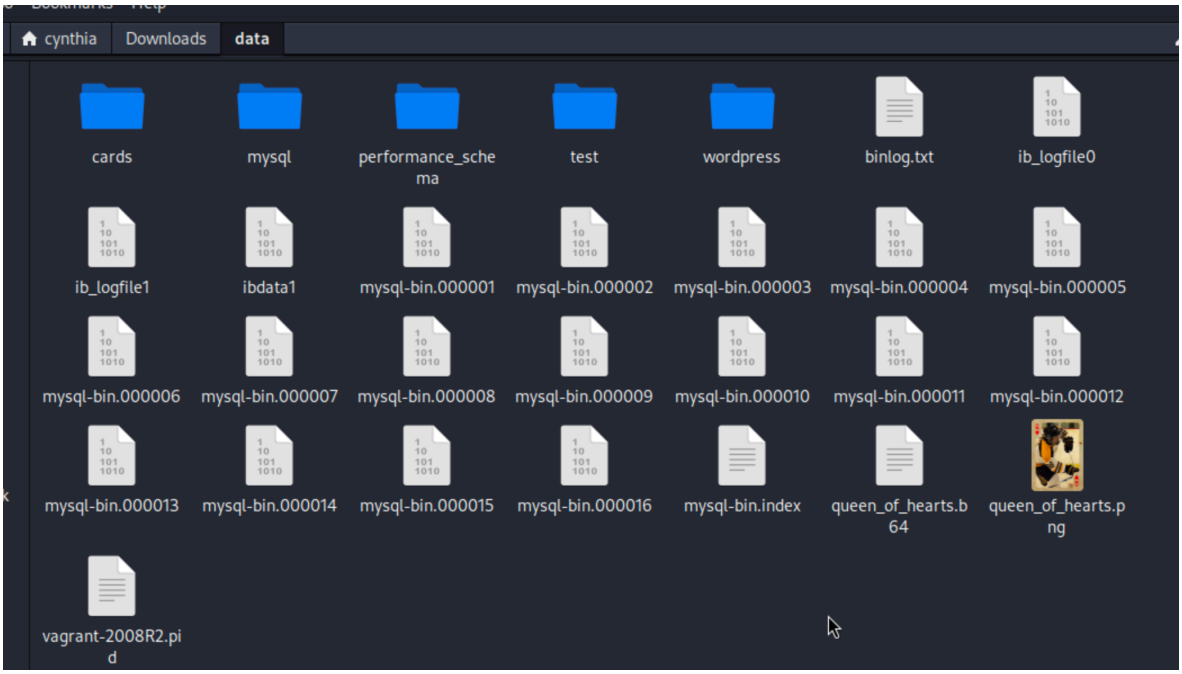


Figure 35: queen of hearts

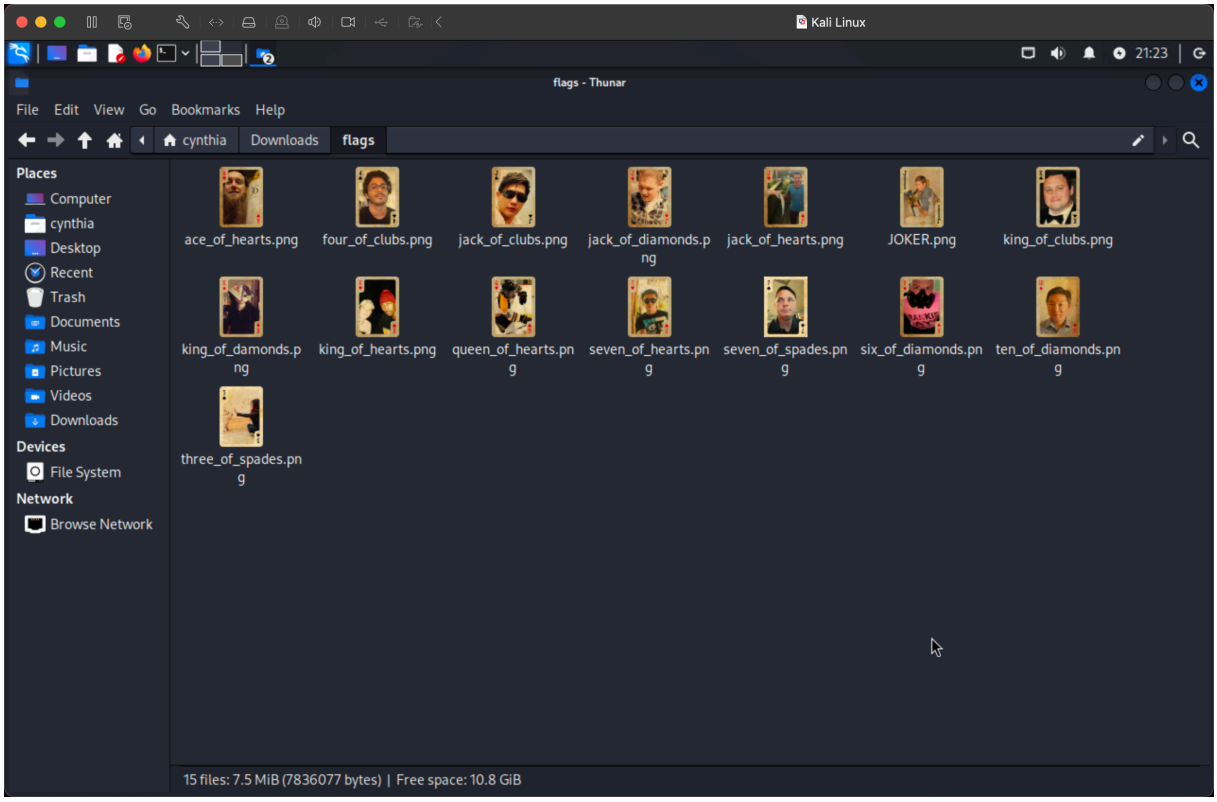


Figure 36: done