BÁO CÁO THỰC HÀNH

Môn học: Pháp chứng kỹ thuật số

Lab 6: CTF Final Test

GVHD: Đoàn Minh Trung

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT334.P11.ANTT

Nhóm 12

STT	Họ và tên	MSSV	Email
1	Nguyễn Lê Thảo Ngọc	21521191	21521191@gm.uit.edu.vn
2	Trần Lê Minh Ngọc	21521195	21521195@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá
1	Kịch bản 1	80%
2	Kịch bản 2	100%
3	Kịch bản 3	100%
4	Kịch bản 4	100%

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

[1] Ghi nội dung công việc, các kịch bản trong bài Thực hành

BÁO CÁO CHI TIẾT

A. KỊCH BẢN 1 - Memory

Đề bài:

a) *memory.dmp* - Có một command chứa thông báo lạ trong bash history, liệu bạn có thể khôi phục thông báo đó? Bạn phải xây dựng volatility và tìm profile.

Giải:

- Trước tiên, cũng như các lab khác em sử dụng volatility với plugin imageinfo nhưng không có kết quả.
- Em sử dụng theo câu lệnh trong đường dẫn được cung cấp để xác định phiên bản được sử dụng trong memory.dmp.
  Tham khảo: [Writeup DGA - CTF - Bwing]

```
┌──(bun㉿bun)-[~/Downloads/lab6_nt334/volatility_2.6_lin64_standalone]
└─$ strings memory.dmp | grep -i "distrib_description="
DISTRIB_DESCRIPTION="Ubuntu 20.04.2 LTS"
DISTRIB_DESCRIPTION="Ubuntu 20.04.2 LTS"
DISTRIB_DESCRIPTION="Ubuntu 20.04.2 LTS"
DISTRIB_DESCRIPTION=
DISTRIB_DESCRIPTION=%s
DISTRIB_DESCRIPTION=
DISTRIB_DESCRIPTION=%s
```

⇨ Hệ điều hành: Ubuntu 20.04.2 LTS.

- Tiếp theo là xác định version của kernel.

```
┌──(bun㉿bun)-[~/Downloads/lab6_nt334/volatility_2.6_lin64_standalone]
└─$ strings memory.dmp | grep -i "Linux version"
 o  The intent is to make the tool independent of Linux version dependencies,
 o  The intent is to make the tool independent of Linux version dependencies,
MESSAGE=Linux version 5.13.0-39-generic (buildd@lcy02-amd64-080) (gcc (Ubuntu 9.4.0-1ubuntu1~20
.04.1) 9.4.0, GNU ld (GNU Binutils for Ubuntu) 2.34) #44~20.04.1-Ubuntu SMP Thu Mar 24 16:43:35
 UTC 2022 (Ubuntu 5.13.0-39.44~20.04.1-generic 5.13.19)
Linux version 5.13.0-39-generic (buildd@lcy02-amd64-080) (gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9
.4.0, GNU ld (GNU Binutils for Ubuntu) 2.34) #44~20.04.1-Ubuntu SMP Thu Mar 24 16:43:35 UTC 202
2 (Ubuntu 5.13.0-39.44~20.04.1-generic 5.13.19)
Linux version 5.13.0-39-generic (buildd@lcy02-amd64-080) (gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9
.4.0, GNU ld (GNU Binutils for Ubuntu) 2.34) #44~20.04.1-Ubuntu SMP Thu Mar 24 16:43:35 UTC 202
2 (Ubuntu 5.13.0-39.44~20.04.1-generic 5.13.19)9)
 o  The intent is to make the tool independent of Linux version dependencies,
Linux version 5.13.0-39-generic (buildd@lcy02-amd64-080) (gcc (Ubuntu 9.4.0-1ubuntu1~20.04.1) 9
.4.0, GNU ld (GNU Binutils for Ubuntu) 2.34) #44~20.04.1-Ubuntu SMP Thu Mar 24 16:43:35 UTC 202
2 (Ubuntu 5.13.0-39.44~20.04.1-generic 5.13.19)9)

┌──(bun㉿bun)-[~/Downloads/lab6_nt334/volatility_2.6_lin64_standalone]
└─$
```

⇨ Kernel: 5.13.0-39-generic

- Em sẽ chuyển sang máy Ubuntu để làm. Tuy nhiên, phiên bản kernel hiện tại của máy em là 5.15.0-126-generic.

```
nt334@ubuntu:~$ uname -r
5.15.0-126-generic
nt334@ubuntu:~$
```

- Sử dụng câu lệnh bên dưới để cài đặt kernel mới cho máy Ubuntu

```
nt334@ubuntu:~$ sudo apt install linux-image-5.13.0-39-generic
[sudo] password for nt334:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  linux-modules-5.13.0-39-generic
Suggested packages:
  linux-doc | linux-hwe-5.13-source-5.13.0 linux-hwe-5.13-tools
  linux-headers-5.13.0-39-generic linux-modules-extra-5.13.0-39-generic
The following NEW packages will be installed:
```
Terminal

- Cài đặt các gói cần thiết như trong hướng dẫn.

```
nt334@ubuntu:~$ sudo apt install dwarfdump build-essential libelf-dev zip
Reading package lists... Done
Building dependency tree
Reading state information... Done
zip is already the newest version (3.0-11build1).
zip set to manually installed.
```

```
nt334@ubuntu:~/Downloads$ git clone https://github.com/volatilityfoundation/vol
atility.git
Cloning into 'volatility'...
remote: Enumerating objects: 27411, done.
remote: Total 27411 (delta 0), reused 0 (delta 0), pack-reused 27411 (from 1)
Receiving objects: 100% (27411/27411), 21.10 MiB | 1.89 MiB/s, done.
R      : 100% (19758/19758), done.
      ownloads$
```
Show Applications

- Tới lệnh make này nhóm không hiểu sao bị lỗi, cũng thử nhiều cách rồi nhưng không biết cách sửa.

```
nt334@ubuntu:~/Downloads/volatility/tools/linux$ make
make -C //lib/modules/5.15.0-126-generic/build CONFIG_DEBUG_INFO=y M="/home/nt3
34/Downloads/volatility/tools/linux" modules
make[1]: Entering directory '/usr/src/linux-headers-5.15.0-126-generic'
  CC [M]  /home/nt334/Downloads/volatility/tools/linux/module.o
  MODPOST /home/nt334/Downloads/volatility/tools/linux/Module.symvers
ERROR: modpost: missing MODULE_LICENSE() in /home/nt334/Downloads/volatility/to
ols/linux/module.o
make[2]: *** [scripts/Makefile.modpost:133: /home/nt334/Downloads/volatility/to
ols/linux/Module.symvers] Error 1
make[2]: *** Deleting file '/home/nt334/Downloads/volatility/tools/linux/Module
.symvers'
make[1]: *** [Makefile:1829: modules] Error 2
make[1]: Leaving directory '/usr/src/linux-headers-5.15.0-126-generic'
make: *** [Makefile:10: dwarf] Error 2
nt334@ubuntu:~/Downloads/volatility/tools/linux$
```

a) dump.raw - Dường như đã có hành vi bất thường trên laptop của NHK, bạn có thể giúp chúng tôi điều tra:

- Thu thập các file bất thường để ghép mảnh flag.
- Liệu họ có để lại những dấu vết trên trình duyệt web?
- Và hình như kẻ xâm nhập bằng một cách nào đó đã lấy được password laptop của NHK. Hãy tìm password đó.

Giải:

- **Thu thập các file bất thường để ghép mảnh flag.**
  - Đầu tiên, kiểm tra thông tin của file dump find-me.bin bằng lệnh "volatility -f dump.raw imageinfo"

```
┌──(ngoc㉿ngoc)-[~/Phap_chung]
└─$ vol -f dump.raw imageinfo
Volatility Foundation Volatility Framework 2.6
INFO    : volatility.debug    : Determining profile based on KDBG search...
          Suggested Profile(s) : Win7SP1x64, Win7SP0x64, Win2008R2SP0x64, Win2008R2SP1x64_23418, Win2008R2SP1x64, Win7
SP1x64_23418
                     AS Layer1 : WindowsAMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/home/ngoc/Phap_chung/dump.raw)
                      PAE type : No PAE
                           DTB : 0x187000L
                          KDBG : 0xf800029f2110L
          Number of Processors : 1
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0xfffff800029f3d00L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2022-04-08 19:05:12 UTC+0000
     Image local date and time : 2022-04-08 12:05:12 -0700
```

  ⇨ Dựa vào kết quả đưa ra, ta có thể xác định profile của hệ thống đã được dump là Win7SP0x64 hoặc Win7SP1x64.
  - Sử dụng các plugin khác của volatility như pslist, cmdscan,... nhưng em không tìm thấy gì đặc biệt.
  - Việc bây giờ là tìm flag nên em sẽ thử kiếm từ khóa "flag" trong các file của file dump.

```
┌──(ngoc㉿ngoc)-[~/Phap_chung]
└─$ vol -f dump.raw --profile=Win7SP1x64 filescan | grep "flag"
Volatility Foundation Volatility Framework 2.6
0x000000013fb0cf20     16      0 RW-r-- \Device\HarddiskVolume1\Users\TEMP\Desktop\flag.txt.txt
0x000000013fc30070     16      0 R--rwd \Device\HarddiskVolume1\Users\NHK-InsecLab\Desktop\flag.txt
0x000000013fc45350     16      0 R--rwd \Device\HarddiskVolume1\Users\NHK-InsecLab\AppData\Roaming\Microsoft\Windows\R
ecent\flag.txt.lnk
0x000000013ff104b0     16      0 RW-rw- \Device\HarddiskVolume1\Users\TEMP\AppData\Roaming\Microsoft\Windows\Recent\fl
ag.txt.lnk
```

  ⇨ Có tập tin flag.txt trong Desktop của tài khoản NHK-InsecLab.
  - Dump riêng file này ra bằng plugin dumpfiles.

```
┌──(ngoc㉿ngoc)-[~/Phap_chung]
└─$ vol -f dump.raw --profile=Win7SP1x64 dumpfiles -Q 0x000000013fc30070  -D ./
Volatility Foundation Volatility Framework 2.6
DataSectionObject 0x13fc30070   None   \Device\HarddiskVolume1\Users\NHK-InsecLab\Desktop\flag.txt
```

  - Sau khi dump file ra thì dùng lệnh strings để đọc nội dung file.

```
┌──(ngoc㉿ngoc)-[~/Phap_chung]
└─$ strings file.None.0xfffffa8003f10350.dat
inseclab{w3lcom3_t0
```

  ⇨ Tìm được nửa flag đầu.
  - Em thử dump tiếp file ở phía dưới.

```
┌──(ngoc㉿ngoc)-[~/Phap_chung]
└─$ vol -f dump.raw --profile=Win7SP1x64 filescan | grep "flag"
Volatility Foundation Volatility Framework 2.6
0x000000013fb0cf20     16      0 RW-r-- \Device\HarddiskVolume1\Users\TEMP\Desktop\flag.txt.txt
0x000000013fc30070     16      0 R--rwd \Device\HarddiskVolume1\Users\NHK-InsecLab\Desktop\flag.txt
0x000000013fc45350     16      0 R--rwd \Device\HarddiskVolume1\Users\NHK-InsecLab\AppData\Roaming\Microsoft\Windows\R
ecent\flag.txt.lnk
0x000000013ff104b0     16      0 RW-rw- \Device\HarddiskVolume1\Users\TEMP\AppData\Roaming\Microsoft\Windows\Recent\fl
ag.txt.lnk
```

  - Dump file ra thì em thấy có một đường dẫn khả nghi có thể chứa nửa flag còn lại. Tuy nhiên lúc nãy dùng lleenhjgrep "flag" thì không tìm thấy đường dẫn này nên có lẽ file đã bị đem đi chỗ khác.

```
┌──(ngoc㉿ngoc)-[~/Phap_chung]
└─$ strings file.None.0xfffffa8003e24240.dat
Thp
flag.txt
TWp*
Windows 7
C:\Users\IEUser\Desktop\flag.txt
iewin7
```

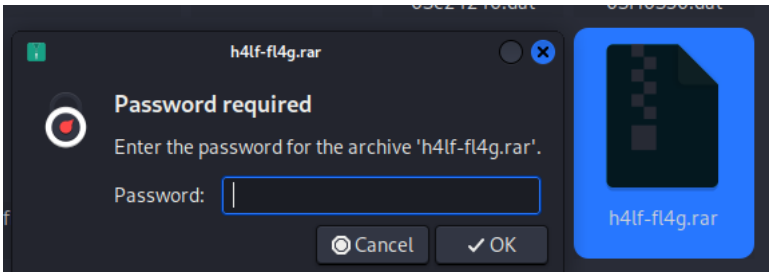
- Thử tìm kiếm với các từ khóa như ".txt", "IEUser", "Desktop", ... thì em tìm được file "h4lf-fl4g.txt" với cái tên như là nửa flag còn lại.

```
┌──(ngoc㉿ngoc)-[~/Phap_chung]
└─$ vol -f dump.raw --profile=Win7SP1x64 filescan | grep "Desktop"
Volatility Foundation Volatility Framework 2.6
0x00000000071f3a10     16      0 RW---- \Device\HarddiskVolume1\Users\TEMP\Desktop\h4lf-fl4g.rar
0x0000000061b6a950     15      0 R--rwd \Device\HarddiskVolume1\Users\IEUser\Desktop\desktop.ini
0x0000000063556330     16      0 R--rwd \Device\HarddiskVolume1\Users\NHK-InsecLan\AppData\Roaming\Microsoft\Windows\SendTo\Desktop.ini
0x000000013abb4c40     16      0 R--rwd \Device\HarddiskVolume1\Windows\Web\Wallpaper\Architecture\Desktop.ini
0x000000013d8546c0     15      0 R--rwd \Device\HarddiskVolume1\ProgramData\Microsoft\Windows\Start Menu\Programs\Maintenance\Desktop.ini
0x000000013d85dd00     15      0 R--rwd \Device\HarddiskVolume1\ProgramData\Microsoft\Windows\Start Menu\Programs\Accessories\Desktop.ini
0x000000013d875dd0     15      0 R--rwd \Device\HarddiskVolume1\ProgramData\Microsoft\Windows\Start Menu\Programs\Accessories\System Tools\Desktop.i
ni
0x000000013d875f20     15      0 R--rwd \Device\HarddiskVolume1\ProgramData\Microsoft\Windows\Start Menu\Programs\Accessories\Accessibility\Desktop.
ini
```

- Dump riêng file này ra.

```
┌──(ngoc㉿ngoc)-[~/Phap_chung]
└─$ vol -f dump.raw --profile=Win7SP1x64 dumpfiles -Q 0x00000000071f3a10 -D ./
Volatility Foundation Volatility Framework 2.6
DataSectionObject 0x071f3a10   None   \Device\HarddiskVolume1\Users\TEMP\Desktop\h4lf-fl4g.rar
```

- Vì đây là file rar nên cần phải giải nén. Tuy nhiên để giải nén được file này lại cần phải có mật khẩu.



- Cài đặt rarfile trên kali để chuẩn bị viết script kiểm tra mật khẩu cho file rar.

```
┌──(ngoc㉿ngoc)-[~/Phap_chung]
└─$ python3 -m venv ~/myenv

┌──(ngoc㉿ngoc)-[~/Phap_chung]
└─$ source ~/myenv/bin/activate


┌──(myenv)─(ngoc㉿ngoc)-[~/Phap_chung]
└─$ pip3 install rarfile
Collecting rarfile
  Downloading rarfile-4.2-py3-none-any.whl.metadata (4.4 kB)
Downloading rarfile-4.2-py3-none-any.whl (29 kB)
Installing collected packages: rarfile
Successfully installed rarfile-4.2

┌──(myenv)─(ngoc㉿ngoc)-[~/Phap_chung]
└─$ sudo apt-get install unrar
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
unrar is already the newest version (1:7.1.2-3).
```

- Đoạn code kiểm tra mật khẩu bằng file mật khẩu rockyou.txt có sẵn.

```
Open   kb1b.py  ~/Phap_chung   Save

1 import rarfile
2
3 # Đường dẫn tới file .rar và file danh sách mật khẩu
4 rar_file_path = "/home/ngoc/Phap_chung/h4lf-fl4g.rar"
5 password_list_path = "/usr/share/wordlists/rockyou.txt"
6
7 # Mở file .rar
8 rf = rarfile.RarFile(rar_file_path)
9
0 # Kiểm tra mật khẩu
1 with open(password_list_path, 'r') as password_list:
2     for password in password_list:
3         password = password.strip()  # Loại bỏ khoảng trắng hoặc ký tự xuống dòng
4         try:
5             rf.extractall(pwd=password)
6             print(f"[+] Password found: {password}")
7             break
8         except rarfile.BadRarFile:
9             print(f"[-] Password incorrect: {password}")
0         except Exception as e:
1             print(f"[!] Error: {e}")
2
```

- Sau khoảng hơn 10p chạy script thì em nhận được mật khẩu là r0cky0u.

```
[-] Password incorrect: sarajevo
[-] Password incorrect: sammyjo
[-] Password incorrect: reloaded
[-] Password incorrect: reinaldo
[-] Password incorrect: rangga
[-] Password incorrect: rampage
[-] Password incorrect: rahayu
[+] Password found: r0cky0u
```

- Dùng mật khẩu trên để giải nén và em có được nửa flag còn lại.

```
Open   h4lf-fl4g.txt (2)  ~/Phap_chung

1 _th3_w0rld_NHK}
```

⇨ **Flag**: inseclab{w3lcom3_t0_th3_w0rld_NHK}

- **Liệu họ có để lại những dấu vết trên trình duyệt web?**
  - Sử dụng plugin pslist để kiểm tra các tiến trình đã chạy trên máy người dùng thì em thấy người này sử dụng chrome là trình duyệt web.

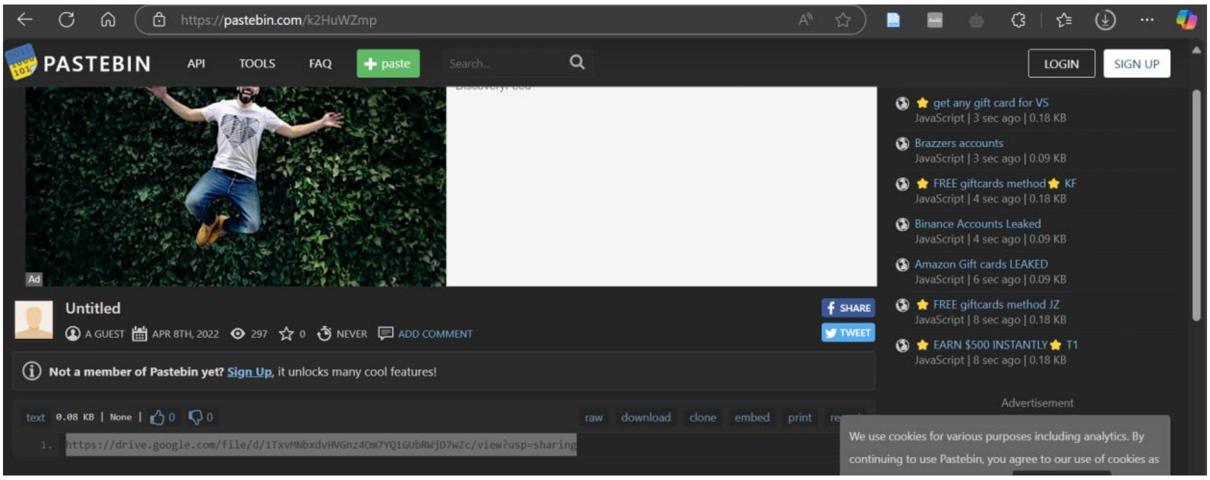
```
0xfffffa80062689c0 GoogleCrashHan        2924   2884      5       83      0      0 2022-04-08 17:44:41 UTC+0000
0xfffffa800629e060 SearchIndexer.        1336    456     13      712      0      0 2022-04-08 17:44:45 UTC+0000
0xfffffa80063ff600 vmtoolsd.exe          1732   2528      8      278      1      0 2022-04-08 17:44:47 UTC+0000
0xfffffa80061375c0 svchost.exe           2840    456     14      384      0      0 2022-04-08 17:46:26 UTC+0000
0xfffffa800629a600 taskhost.exe          3228    456      5      141      1      0 2022-04-08 18:44:26 UTC+0000
0xfffffa8004220060 chrome.exe            2332   2528      0 ------      1      0 2022-04-08 19:02:52 UTC+0000   2022-04-08 19:05:17 UTC+0000
0xfffffa80053fa610 SearchProtocol        1756   1336      8      282      0      0 2022-04-08 19:05:04 UTC+0000
0xfffffa8004eebb10 SearchFilterHo        4172   1336      5      101      0      0 2022-04-08 19:05:04 UTC+0000
0xfffffa8005461700 DumpIt.exe            4512   2332      5       46      1      1 2022-04-08 19:05:10 UTC+0000
0xfffffa80062eab10 conhost.exe           4352    404      2       52      1      0 2022-04-08 19:05:10 UTC+0000
0xfffffa80061e3b10 WMIADAP.exe           1656    924      6      769 ------      0 2022-04-08 19:05:25 UTC+0000
0xfffffa800548e650 WmiPrvSE.exe          4380    616      8       24 ------      0 2022-04-08 19:05:25 UTC+0000
```

  - Sử dụng plugin chromehistory để xem lịch sử duyệt web (vì không thể thêm plugin mới bằng volatility standard nên em chuyển sang volatility git để sử dụng).
    Tham khảo:
    superponible/volatility-plugins: Plugins I've written for Volatility

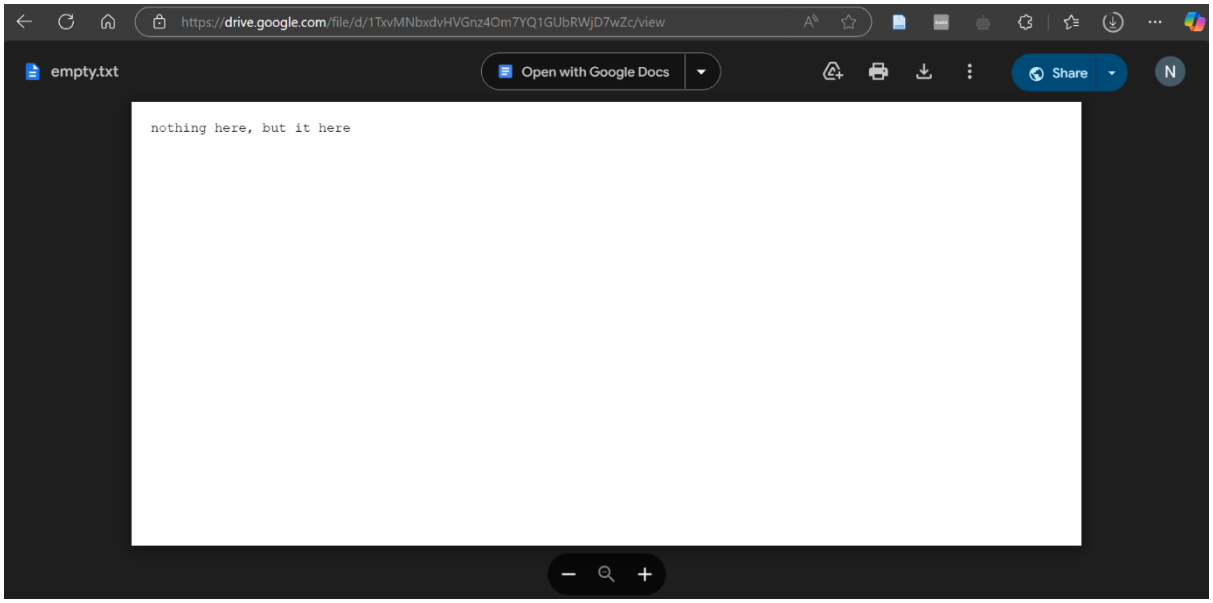
```
┌──(ngoc㉿ngoc)-[~/Phap_chung/volatility-master]
└─$ python vol.py --plugins=volatility-plugins-master -f ~/Phap_chung/dump.raw --profile=Win7SP1x64 chromehistory
Volatility Foundation Volatility Framework 2.6.1
Index  URL                                                                    Title
                                                       Visits Typed Last Visit Time            Hidden Favicon ID
______ ______________________________________________ ______ _____ __________________________ ______ __________

    32 https://www.google.com/search?q=downlao ... 3015.5015j0j7&sourceid=chrome&ie=UTF-8 download dumpit - Google 搜尋
                                                        6      0 2022-04-08 19:04:24.443945        N/A
    29 https://www.google.com/search?q=how+to+ ... 9i57.9678j0j9&sourceid=chrome&ie=UTF-8 how to pass digital forensics
- Google 搜尋                                          2      0 2022-04-08 19:02:40.139948        N/A
    25 https://www.google.com/search?q=wannaon ... 4xmAEAoAEBsAEAyAEJwAEB&sclient=gws-wiz wannaone - Google 搜尋
                                                        2      0 2022-04-08 19:01:32.828397        N/A
    19 https://www.google.com/search?q=insecla ... 4ymAEAoAEBsAEKyAEKwAEB&sclient=gws-wiz inseclab - Google 搜尋
                                                        3      0 2022-04-08 19:01:16.406905        N/A
    18 https://www.google.com/search?q=conan&o ... 6i512.897j0j7&sourceid=chrome&ie=UTF-8 conan - Google 搜尋
                                                        2      0 2022-04-08 19:00:25.836168        N/A
    16 https://www.google.com/search?q=downlao ... 3017.1911j0j7&sourceid=chrome&ie=UTF-8 downlaod hxd - Google 搜尋
                                                        2      0 2022-04-08 18:58:06.184801        N/A
     9 https://www.google.com/search?q=downloa ... 8i30.4374j0j7&sourceid=chrome&ie=UTF-8 download hidden tear - Google
搜尋                                                   2      0 2022-04-08 18:47:57.622508        N/A
     6 https://www.google.com/search?q=goliate ... 58.216279j0j9&sourceid=chrome&ie=UTF-8 goliate/hidden-tear: ransomwar
e open-sources (github.com) - Google 搜尋              2      0 2022-04-08 18:44:39.388108        N/A
     5 https://www.google.com/search?q=a&oq=a& ... i60l2.189j0j9&sourceid=chrome&ie=UTF-8 a - Google 搜尋
                                                        2      0 2022-04-08 18:41:05.127330        N/A
     1 https://www.google.com/search?q=downloa ... 12l9.3638j0j7&sourceid=chrome&ie=UTF-8 download winrar - Google 搜尋
                                                        2      0 2022-04-08 18:05:14.499225        N/A
    26 https://ctftime.org/                                                   CTFtime.org / All about CTF (C
apture The Flag)                                         1      1 2022-04-08 19:01:40.851134        N/A
```

```
    15 https://github.com/goliate/hidden-tear/ ... -tear/bin/Debug/hidden-tear.vshost.exe hidden-tear/hidden-tear.vshost
.exe at master · goliate/hidden-tear · GitHub          1      0 2022-04-08 18:51:31.159613        N/A
    14 https://github.com/goliate/hidden-tear/ ... ster/hidden-tear/hidden-tear/bin/Debug hidden-tear/hidden-tear/hidden
-tear/bin ... aster · goliate/hidden-tear · GitHub     2      0 2022-04-08 18:51:33.424394        N/A
    13 https://github.com/goliate/hidden-tear/ ... ter/hidden-tear/hidden-tear/Properties hidden-tear/hidden-tear/hidden
-tear/Pro ... aster · goliate/hidden-tear · GitHub     2      0 2022-04-08 18:51:21.710010        N/A
    12 https://github.com/goliate/hidden-tear/tree/master/hidden-tear/hidden-tear        hidden-tear/hidden-tear/hidden
-tear at master · goliate/hidden-tear · GitHub          5      0 2022-04-08 18:51:33.934445        N/A
    11 https://github.com/goliate/hidden-tear/tree/master/hidden-tear                    hidden-tear/hidden-tear/hidden
-tear at master · goliate/hidden-tear · GitHub          4      0 2022-04-08 18:51:35.373730        N/A
    10 https://github.com/goliate/hidden-tear                                        GitHub - goliate/hidden-tear:
ransomware open-sources                                  2      0 2022-04-08 18:51:13.252715        N/A
     8 https://pastebin.com/k2HuWZmp                                                 https://drive.google.com/file/
d/1TxvMNb ... RWjD7wZc/view?usp=shari - Pastebin.com    1      0 2022-04-08 18:46:46.246539        N/A
     3 https://www.win-rar.com/download.html?&L=0                                    WinRAR download free and suppo
rt: WinRAR Download Latest Version                       1      0 2022-04-08 18:05:17.080333        N/A
     2 https://www.win-rar.com/download.html                                         WinRAR download free and suppo
rt: WinRAR Download Latest Version                       1      0 2022-04-08 18:05:17.080333        N/A

┌──(ngoc㉿ngoc)-[~/Phap_chung/volatility-master]
└─$ █
```

⇨ Có một đường dẫn đến drive khả nghi. Thử theo đường dẫn này.



- Mở theo đường dẫn trên và trong đó cũng có cả đường dẫn đến drive.



⇨ Theo dòng chữ bên trên thì có vẻ thông tin đã được ẩn trong tập tin này.

- Tập tin có tên là empty.txt. Sau khi thử các cách thì em lấy được flag thông qua công cụ Stegsnow.

```
┌──(ngoc㉿ngoc)-[~/Phap_chung/volatility-master]
└─$ stegsnow -C ~/Phap_chung/empty.txt
inseclab{y0u_c4n_s33_fl4g}
```

⇨ **Flag:** inseclab{y0u_c4n_s33_fl4g}

- **Và hình như kẻ xâm nhập bằng một cách nào đó đã lấy được password laptop của NHK. Hãy tìm password đó.**
  - Chúng ta có thể sử dụng plugin hivelist. Hiểu đơn giản thì đây là công đoạn lấy ra trường địa chỉ bắt đầu trọng bộ nhớ của nơi lưu trữ thông tin đăng ký và quản lý về tài khoản người dùng Windows.

```
┌──(ngoc㉿ngoc)-[~/Phap_chung]
└─$ vol -f dump.raw --profile=Win7SP1x64 hivelist
Volatility Foundation Volatility Framework 2.6
Virtual            Physical           Name
------------------ ------------------ ----
0xfffff8a0012a6010 0x000000009e18b010 \??\C:\Users\sshd_server\ntuser.dat
0xfffff8a0012bb270 0x000000004829e270 \??\C:\Users\sshd_server\AppData\Local\Microsoft\Windows\UsrClass.dat
0xfffff8a0017f4010 0x0000000019cda010 \??\C:\Users\TEMP\ntuser.dat
0xfffff8a001882410 0x0000000021a41410 \??\C:\Users\TEMP\AppData\Local\Microsoft\Windows\UsrClass.dat
0xfffff8a0032eb010 0x000000011ff7a010 \??\C:\Windows\AppCompat\Programs\Amcache.hve
0xfffff8a00484c010 0x00000000a8ca5010 \Device\HarddiskVolume1\Boot\BCD
0xfffff8a004ecd010 0x00000000529bb010 \SystemRoot\System32\Config\DEFAULT
0xfffff8a004ed7010 0x0000000052913010 \SystemRoot\System32\Config\SAM
0xfffff8a00000e010 0x00000000a9537010 [no name]
0xfffff8a000024010 0x00000000a9742010 \REGISTRY\MACHINE\SYSTEM
0xfffff8a000063010 0x00000000a9683010 \REGISTRY\MACHINE\HARDWARE
0xfffff8a0005dc010 0x0000000054799010 \SystemRoot\System32\Config\SECURITY
0xfffff8a0005e6010 0x0000000013a00010 \SystemRoot\System32\Config\SOFTWARE
0xfffff8a000e2b010 0x00000000a4cc8010 \??\C:\System Volume Information\Syscache.hve
0xfffff8a000e61010 0x000000000dc00010 \??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
0xfffff8a000ef1010 0x000000004b8d9010 \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
```

- Trong kết quả hiện ra, ta có được danh sách các key về user trên Windows đang được lưu trữ trên RAM. Công đoạn tiếp theo chỉ là tìm ra mã băm của mật khẩu dựa vào giá trị key của hệ thống [system key] và giá trị key của tập tin SAM [SAM key].

```
┌──(ngoc㉿ngoc)-[~/Phap_chung]
└─$ vol -f dump.raw --profile=Win7SP1x64 hashdump -y 0xfffff8a000024010 -s 0xfffff8a004ed7010
Volatility Foundation Volatility Framework 2.6
Administrator:500:aad3b435b51404eeaad3b435b51404ee:fc525c9683e8fe067095ba2ddc971889:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
NHK-InsecLab:1000:aad3b435b51404eeaad3b435b51404ee:141be588e38b145c4e1f274b646898eb:::
sshd:1001:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
sshd_server:1002:aad3b435b51404eeaad3b435b51404ee:8d0a16cfc061c3359db455d00ec27035:::
```

- Tuy nhiên là các mật khẩu này đều đã bị hash. Theo như em tìm hiểu, đây là một hash NTLM thường được sử dụng trong Windows Security Accounts Manager (SAM) để lưu trữ mật khẩu đã mã hóa của tài khoản người dùng và có thể giải mã bằng các công cụ như Mimikatz, hashcat, hoặc pwdump.
- Em sẽ thêm plugin mimikatz vào volatility để giải mã các mật khẩu. Tham khảo đường dẫn: community/FrancescoPicasso at master · volatilityfoundation/community

```
┌──(ngoc㉿ngoc)-[~/Phap_chung/volatility-master]
└─$ python vol.py --plugins=volatility-plugins-master -f ~/Phap_chung/dump.raw --profile=Win7SP1x64 mimikatz

Volatility Foundation Volatility Framework 2.6.1
Module   User             Domain           Password
-------- ---------------- ---------------- ------------------------------------------
wdigest  NHK-InsecLab     IEWIN7           AntiNHK
wdigest  sshd_server      IEWIN7           D@rj33l1ng
wdigest  IEWIN7$          WORKGROUP
```

⇨ Mật khẩu của tài khoảng NHK-InsecLab là AntiNHK.

B. KỊCH BẢN 2 – Network

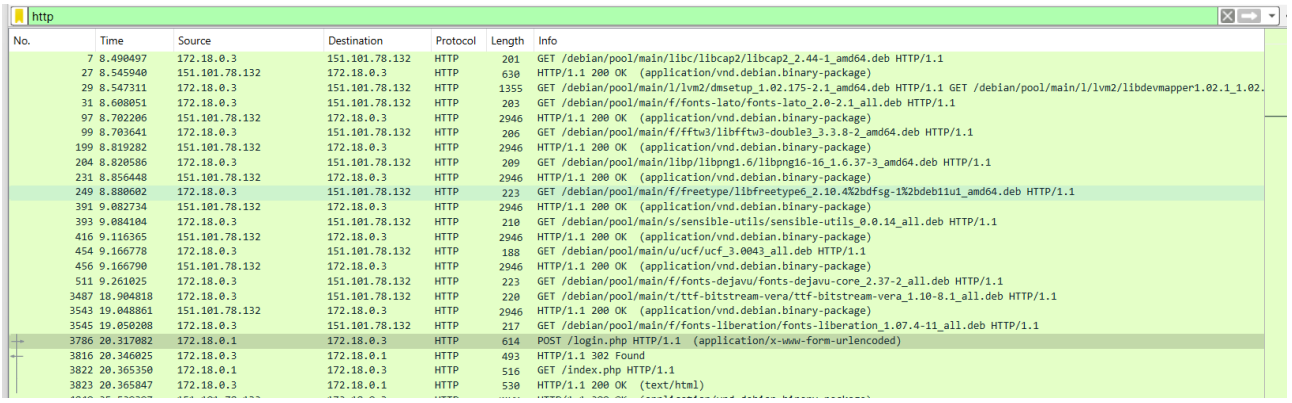
Đề bài:

pcap.pcap - Vào lúc 3h sáng, trong khi NHK đang ngủ thì IDS của NHK cảnh báo rằng web server đang có cuộc tấn công, NHK nghe loáng thoáng mấy anh dev nói về việc "mù mờ" gì đó, nhưng họ không đủ năng lực để điều tra. Hãy giúp mấy anh dev nhà NHK nhé :)

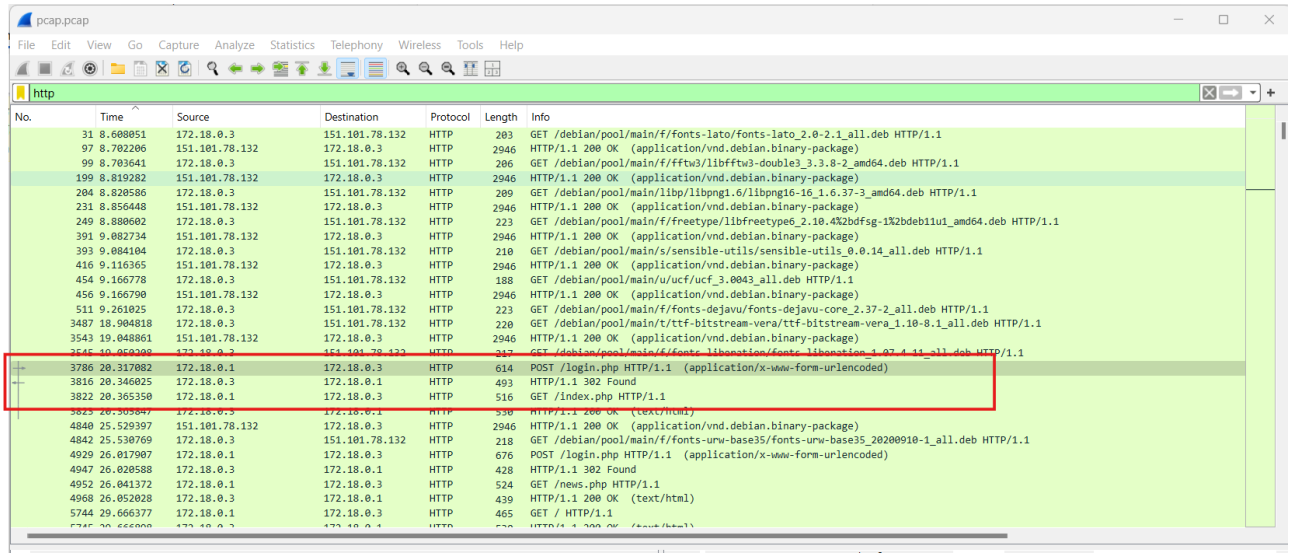
Giải:

a) Tìm IP của web server

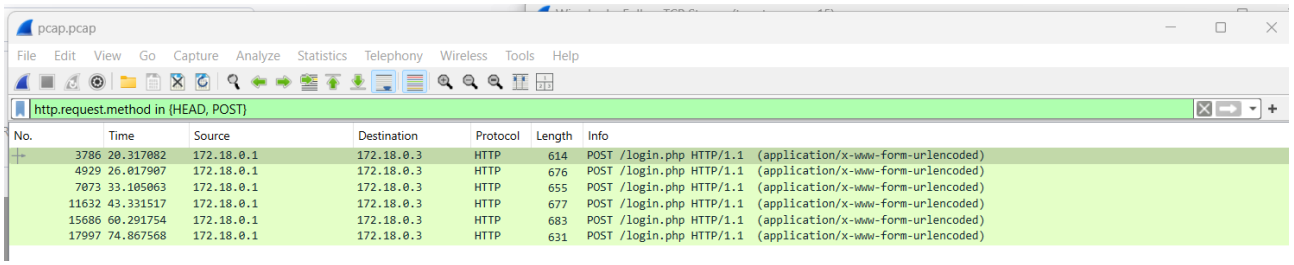
- Vì đây là 1 web server nên em sẽ lọc các gói tin giao thức HTTP.



- Để ý gói tin 3786 và gói 3822, có vẻ đối tượng có IP 172.18.0.1 đang thực hiện đăng nhập vào 1 trang web có địa chỉ 172.18.0.3



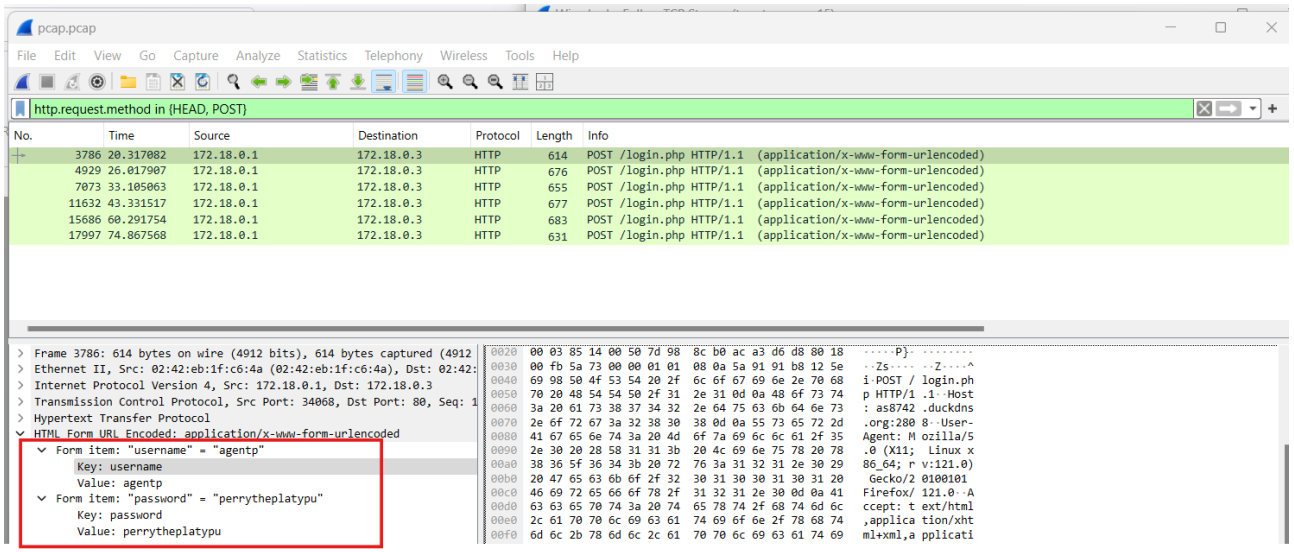
- Ngoài ra khi lọc ra các gói có request header là POST, em chỉ thấy IP đích của gói tin là 172.18.0.3



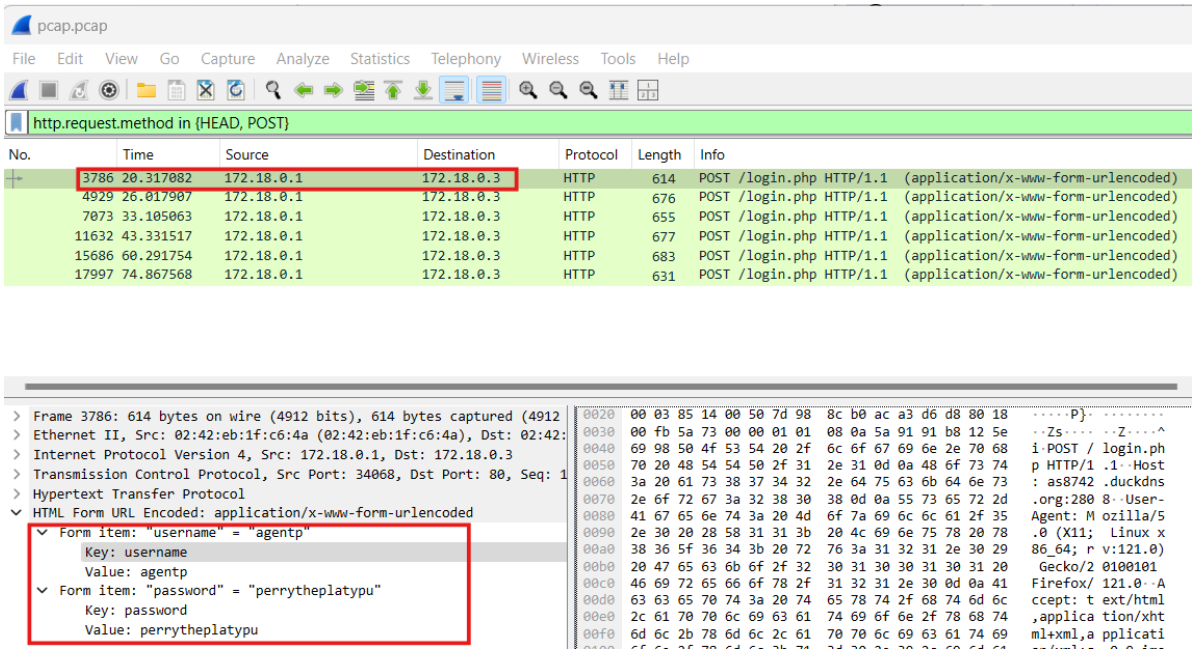
⇨ Địa chỉ web server cần tìm là : 172.18.0.3

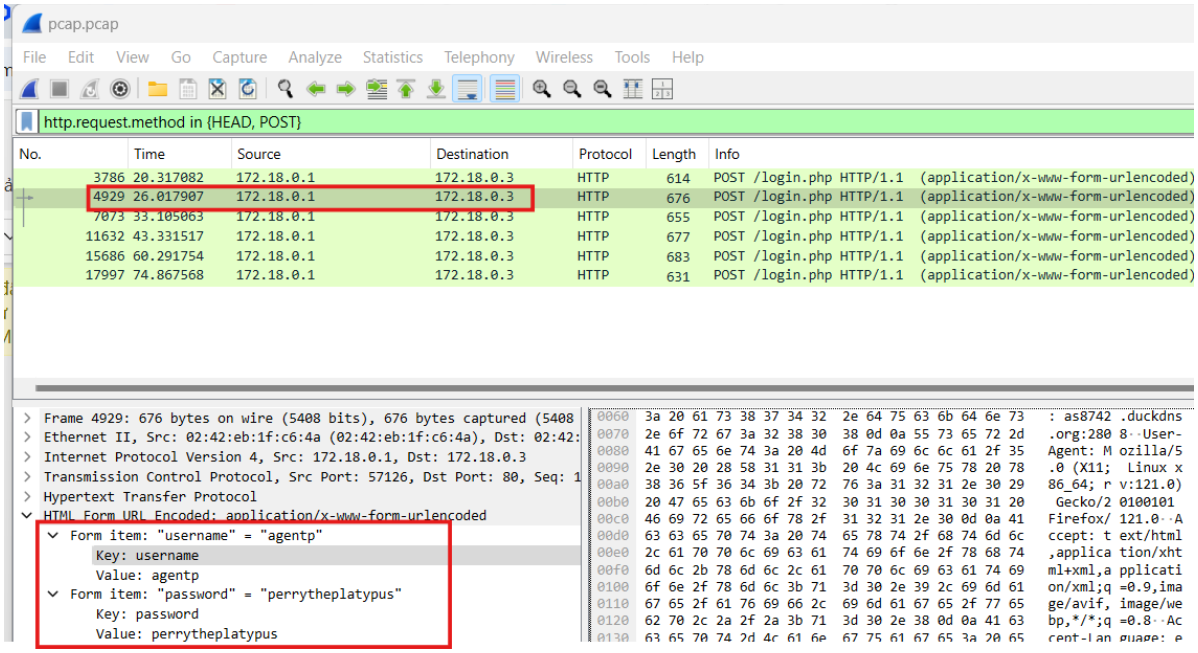
b) Tìm username và password của một tài khoản sử dụng server. NHK nghe nói anh ta là một đặc vụ mật.

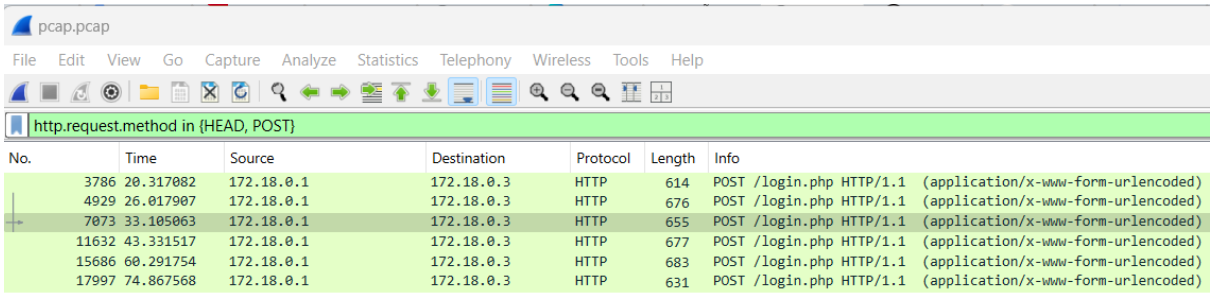
- Vì thao tác đăng nhập vào trang web sẽ gửi gói tin có request header gồm phương thức POST với url: /login.php nên em lọc ra các gói tin có request header là POST

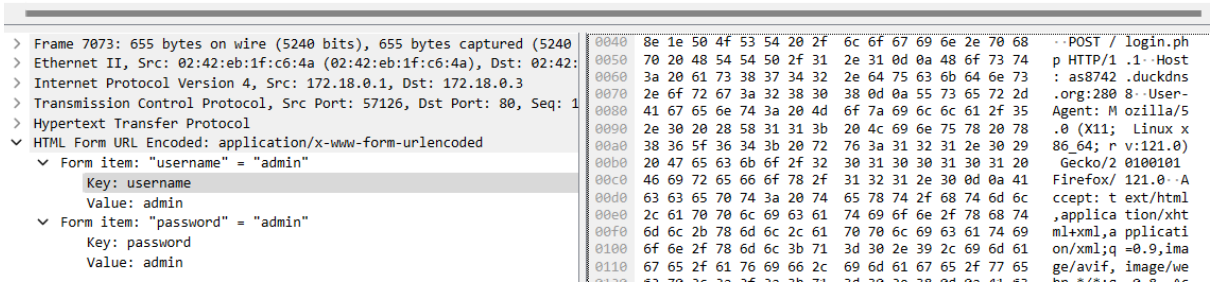


- Kết quả em tìm được 6 gói chứa username và password đều đến từ 172.18.0.1. Trong đó 3 gói tin sau cùng cho thấy đây là dạng tấn công nên sẽ chỉ có 3 tài khoản hợp lệ là:





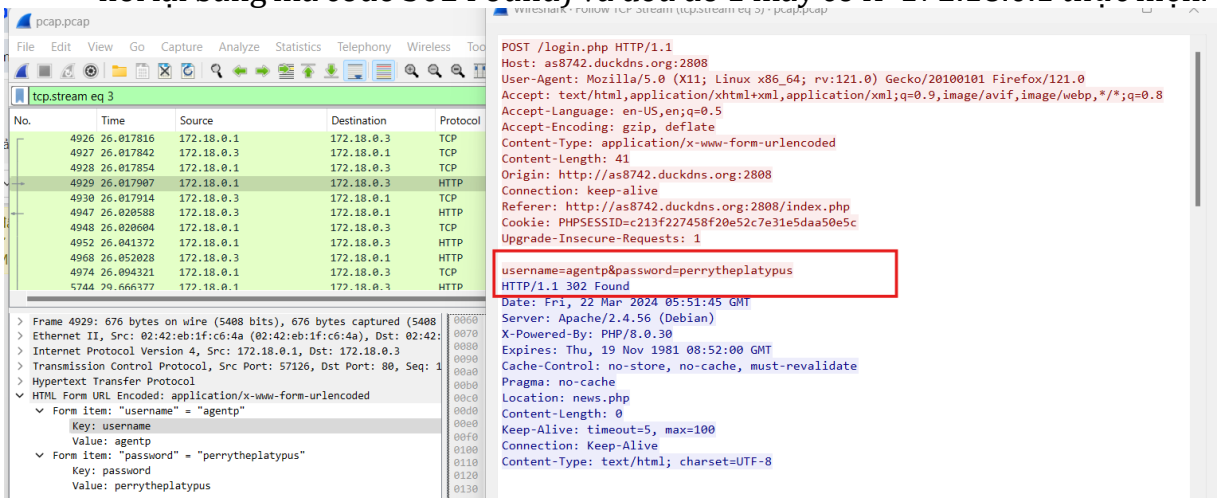


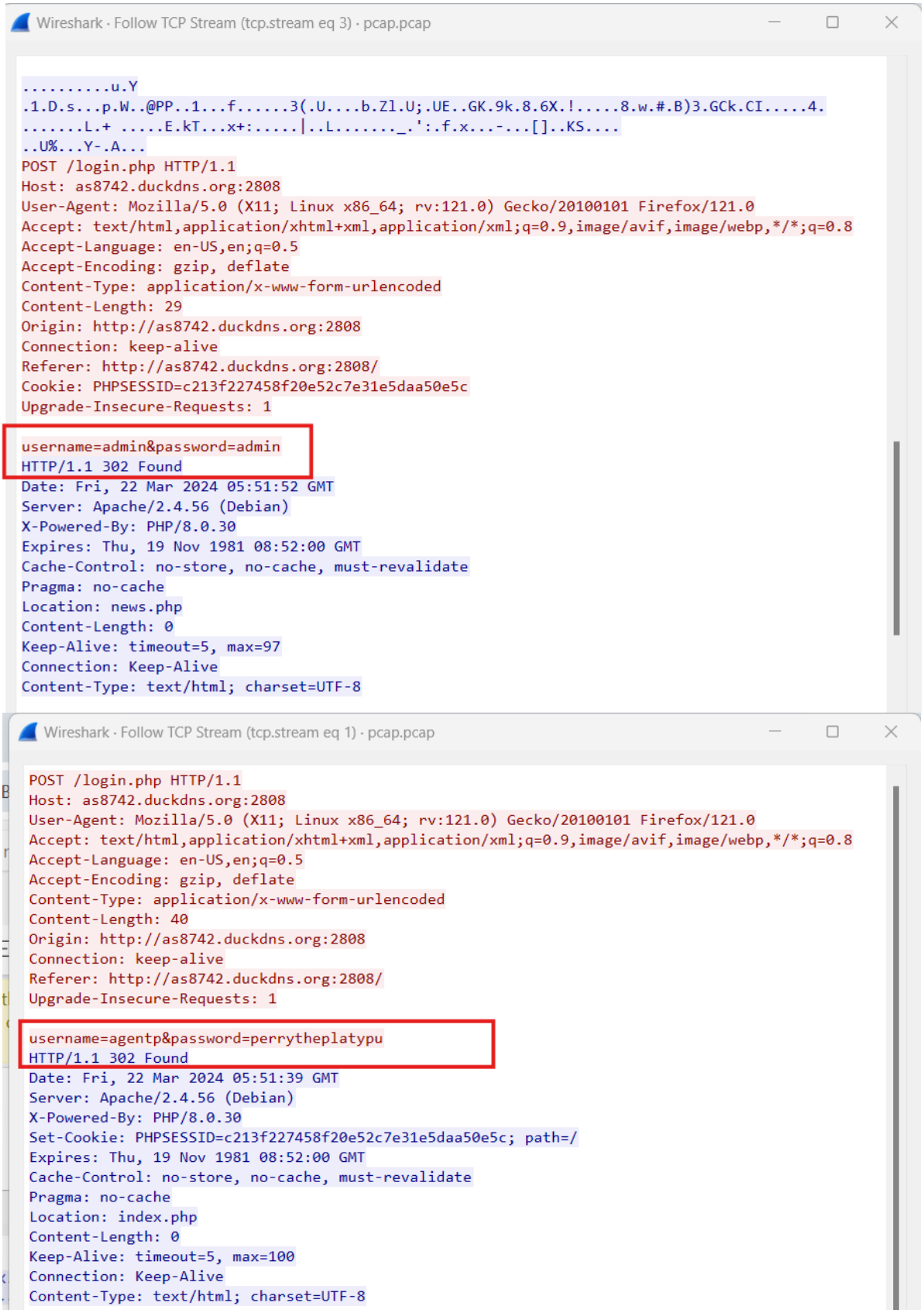


⇨ Tìm được 3 tài khoản thực hiện đăng nhập vào web server:

- Tài khoản 1:
  Username: agentp
  Password: perrytheplatypu
- Tài khoản 2:
  Username: agentp
  Password: perrytheplatypus
- Tài khoản 3:
  Username: admin
  Password: admin

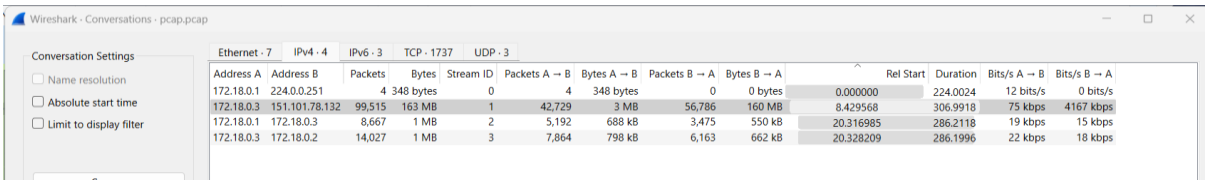
- Cả 3 tài khoản này đều đăng nhập được vào trang web (vì web server phản hồi lại bằng mã code 302 Found) và đều do 1 máy có IP 172.18.0.1 thực hiện.



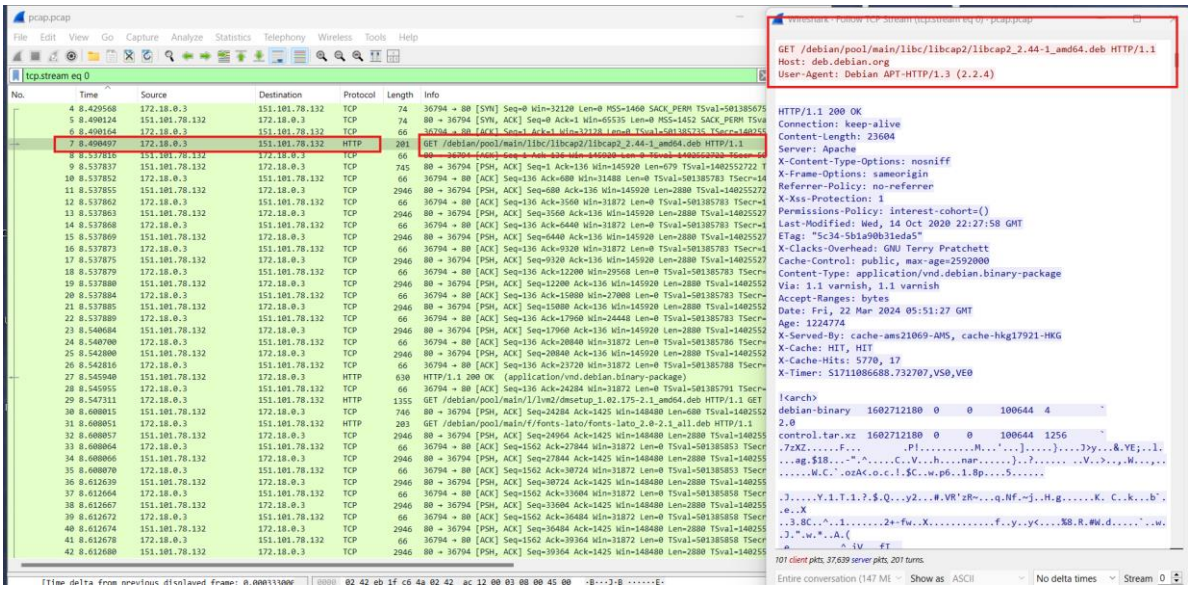


c) Hacker tấn công từ bên ngoài mạng hay là từ bên trong mạng.

- Chọn Statistics -> Conversations để xem thông tin mà Wireshark thống kê được:

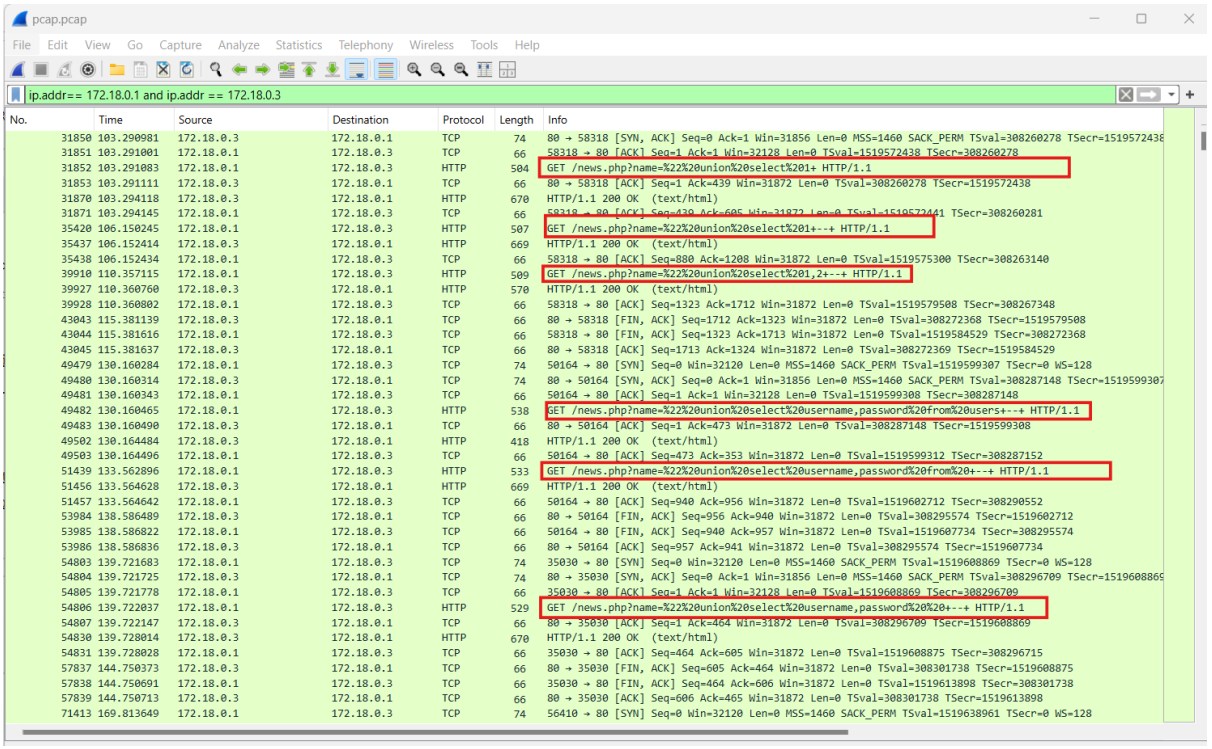


- Dễ thấy server (172.18.0.3) giao tiếp rất nhiều với đối tượng có địa chỉ IP là 151.101.78.132 nên em sẽ tập trung phân tích đối tượng này vì có vẻ đây là attacker.
- Theo dõi luồng giao tiếp giữa web server với 151.101.78.132



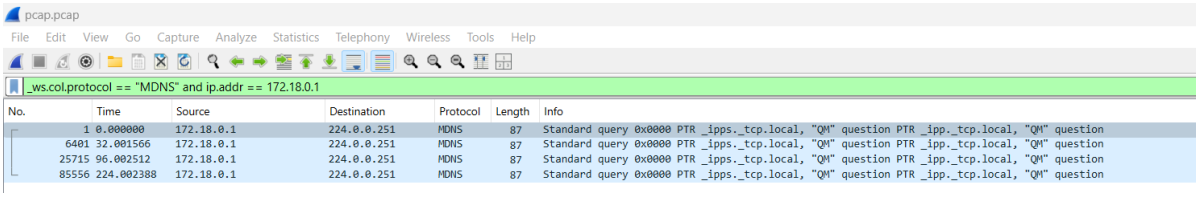
⇨ Chỉ biết được Web server thực hiện tải file .deb từ máy có IP 151.101.78.132

- Em chuyển sang theo dõi luồng giao tiếp giữa web server với 3 đối tượng còn lại (trước đó em đã phát hiện có 4 đối tượng tương ứng với 4 IP khác nhau giao tiếp với sever nhờ bảng Conversation phía trên)
- Khi theo dõi các gói tin giao tiếp giữa webserver với IP 172.18.0.1 bằng cú pháp lọc: ip.addr== 172.18.0.1 and ip.addr == 172.18.0.3
- Em phát hiện rất nhiều gói tin chứa nội dung bất thường trong phần requestHeader, theo như các dòng truy vấn web thì có vẻ đối tượng 172.18.0.1 đang tấn công web server để lấy đi các dữ liệu lưu trên web server này.



⇨ Địa chỉ IP mà attacker dùng là: 172.18.0.1 (lúc này vẫn chưa thể biết được đặc vụ có phải là hacker không)

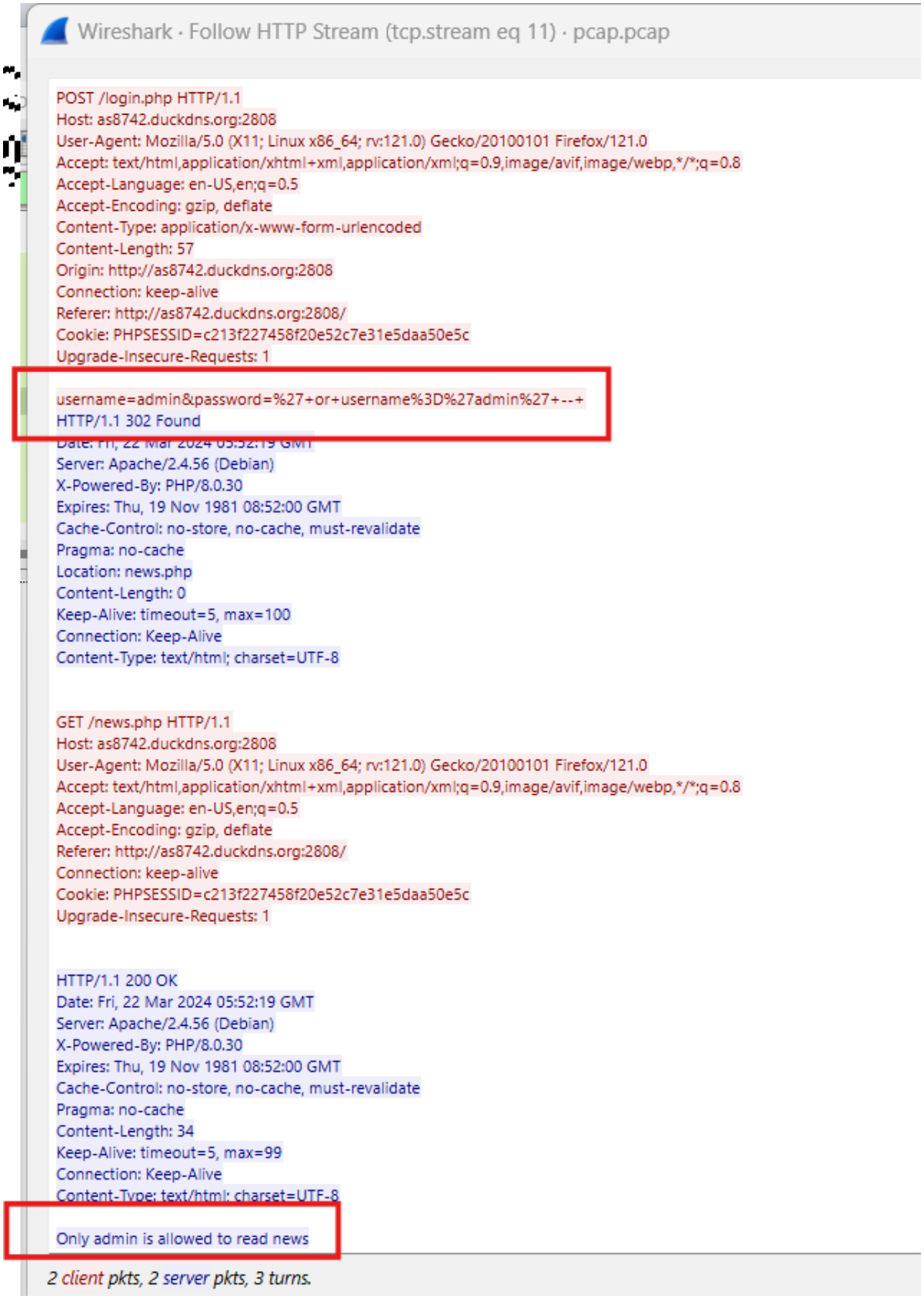
- Thử lọc các gói tin có IP 172.18.0.1, em thấy có địa chỉ lạ (không thuộc mạng 172.18.0.0/24) giao tiếp với IP này bằng giao thức MDNS (1 giao thức mà hacker có thể lợi dụng để giả mạo IP). Trong trường hợp này em nghi ngờ hacker đã giả mạo IP 172.18.0.1



- Mà địa chỉ IP của web server là 172.18.0.3, còn địa chỉ lạ lại khác phân vùng mạng nên có khả năng rất cao đây là cuộc tấn công từ bên ngoài mạng. Hacker đã giả mạo địa chỉ IP của một đối tượng trong mạng này để thực hiện tấn công.

d) Lỗ hổng mà hacker dùng để khai thác là gì?

- Từ câu a, có thể thấy lúc đầu attacker cố gắng khai thác lỗ hổng SQL để thực hiện bypass login và xem trang news nhưng đều thất bại vì cần phải có tài khoản quyền administrator thì mới xem được trang này

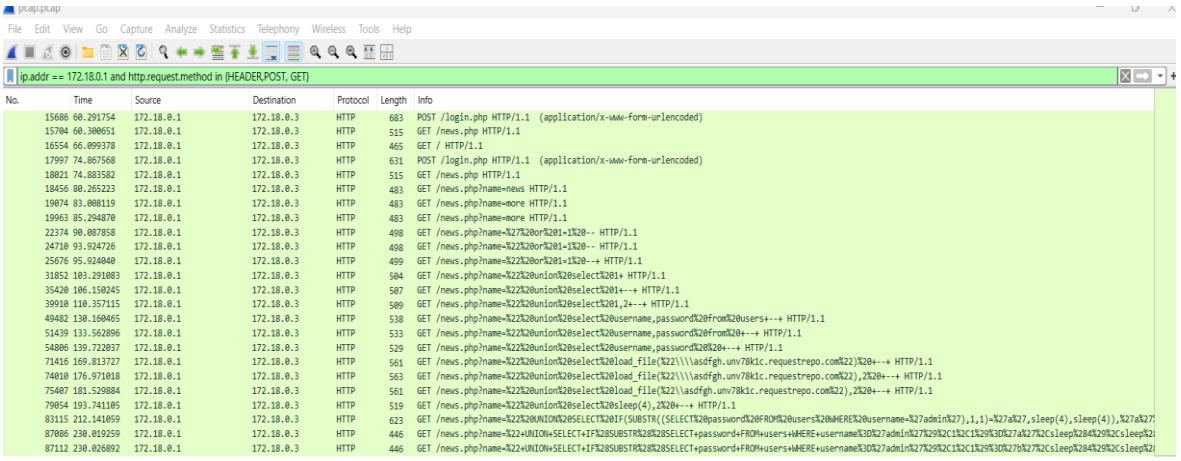


Wireshark · Follow HTTP Stream (tcp.stream eq 11) · pcap.pcap

```
POST /login.php HTTP/1.1
Host: as8742.duckdns.org:2808
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 57
Origin: http://as8742.duckdns.org:2808
Connection: keep-alive
Referer: http://as8742.duckdns.org:2808/
Cookie: PHPSESSID=c213f227458f20e52c7e31e5daa50e5c
Upgrade-Insecure-Requests: 1

username=admin&password=%27+or+username%3D%27admin%27+--+
HTTP/1.1 302 Found
Date: Fri, 22 Mar 2024 05:52:19 GMT
Server: Apache/2.4.56 (Debian)
X-Powered-By: PHP/8.0.30
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: news.php
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8


GET /news.php HTTP/1.1
Host: as8742.duckdns.org:2808
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://as8742.duckdns.org:2808/
Connection: keep-alive
Cookie: PHPSESSID=c213f227458f20e52c7e31e5daa50e5c
Upgrade-Insecure-Requests: 1


HTTP/1.1 200 OK
Date: Fri, 22 Mar 2024 05:52:19 GMT
Server: Apache/2.4.56 (Debian)
X-Powered-By: PHP/8.0.30
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 34
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

Only admin is allowed to read news
```

2 client pkts, 2 server pkts, 3 turns.

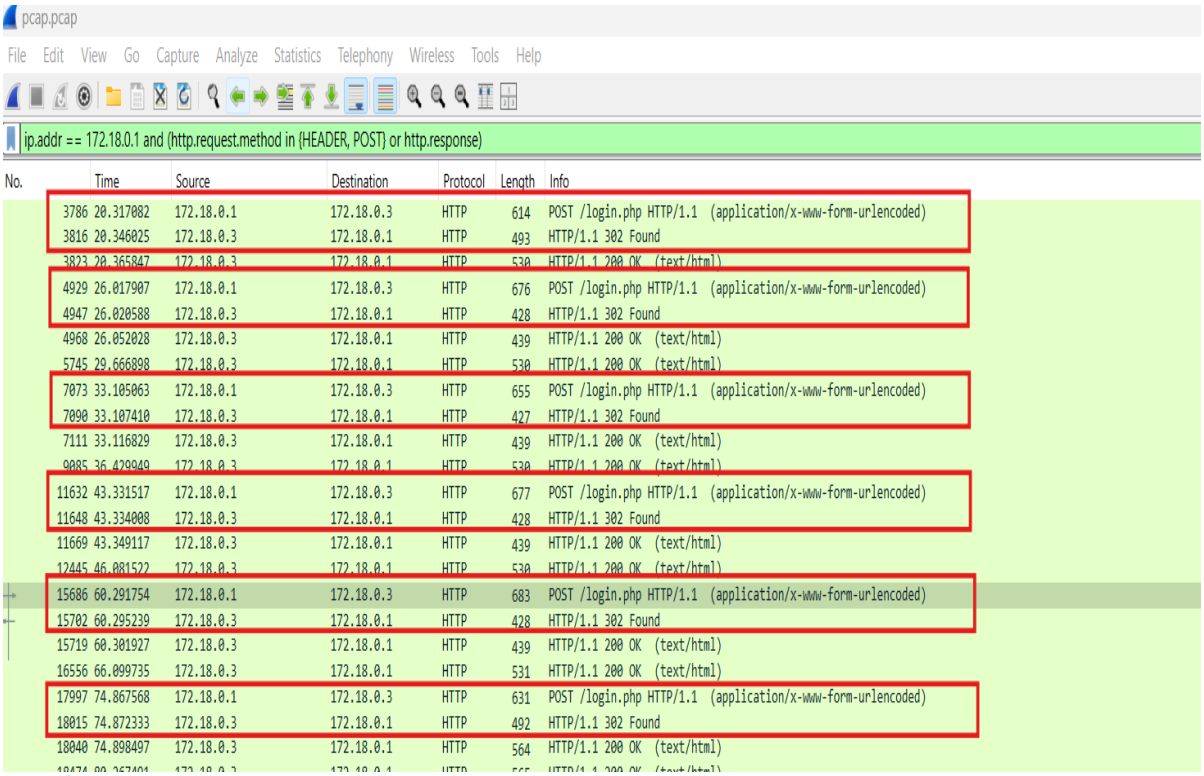
- Do đó, attacker chuyển sang tấn công trực tiếp vào trang news.php thay vì tiếp tục tấn công trang login.php



- Nhìn vào trường Info của mỗi gói tin, em thấy đây chắn chắn là tấn công SQL injection

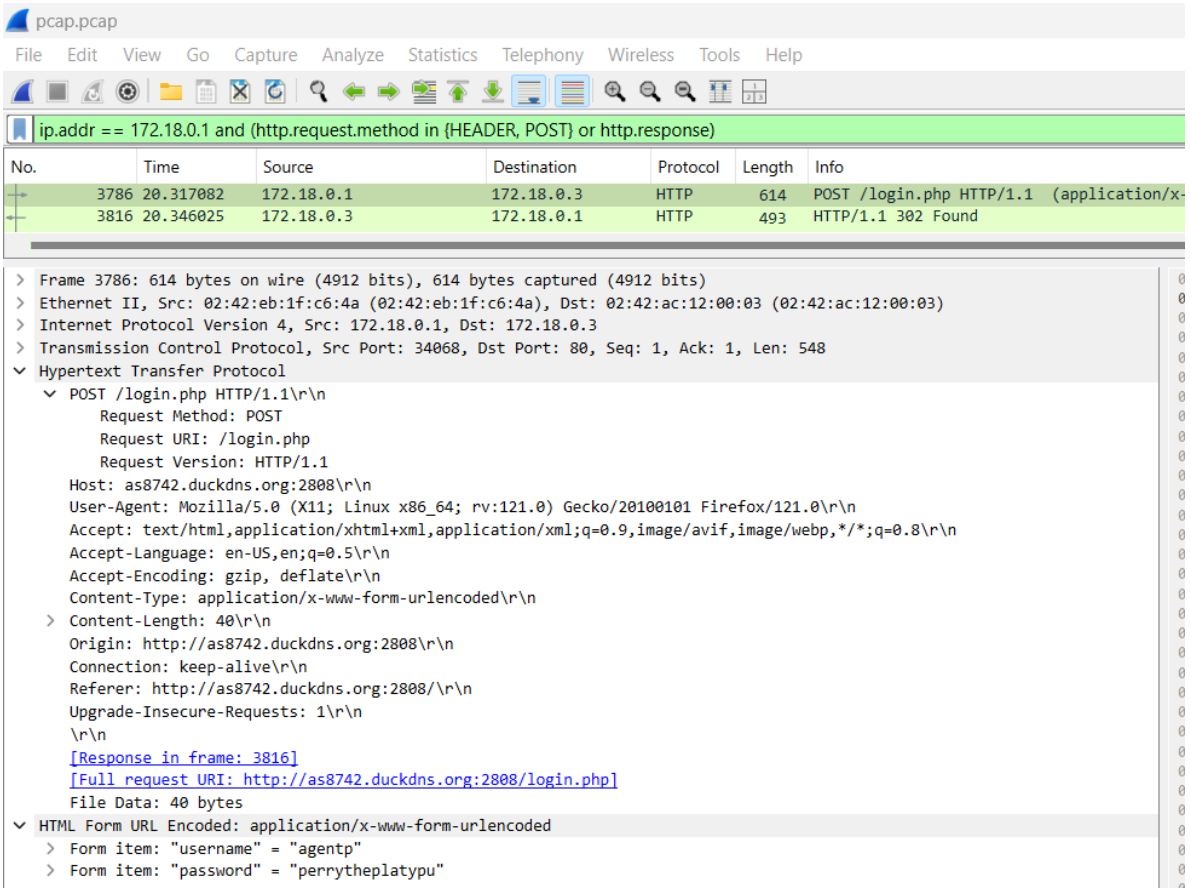
e) Hacker đã login vào tài khoản nào?

- Trong câu c, em đã phát hiện được địa chỉ IP mà attacker giả mạo là 172.18.0.1 và khi thực hiện đăng nhập, attcker phải gửi gói tin POST tới web server.
- Do đó em sẽ lọc ra các gói tin liên quan và các gói response mà attacker nhận được bằng cú pháp: ip.addr == 172.18.0.1 and (http.request.method in {HEADER, POST} or http.response)

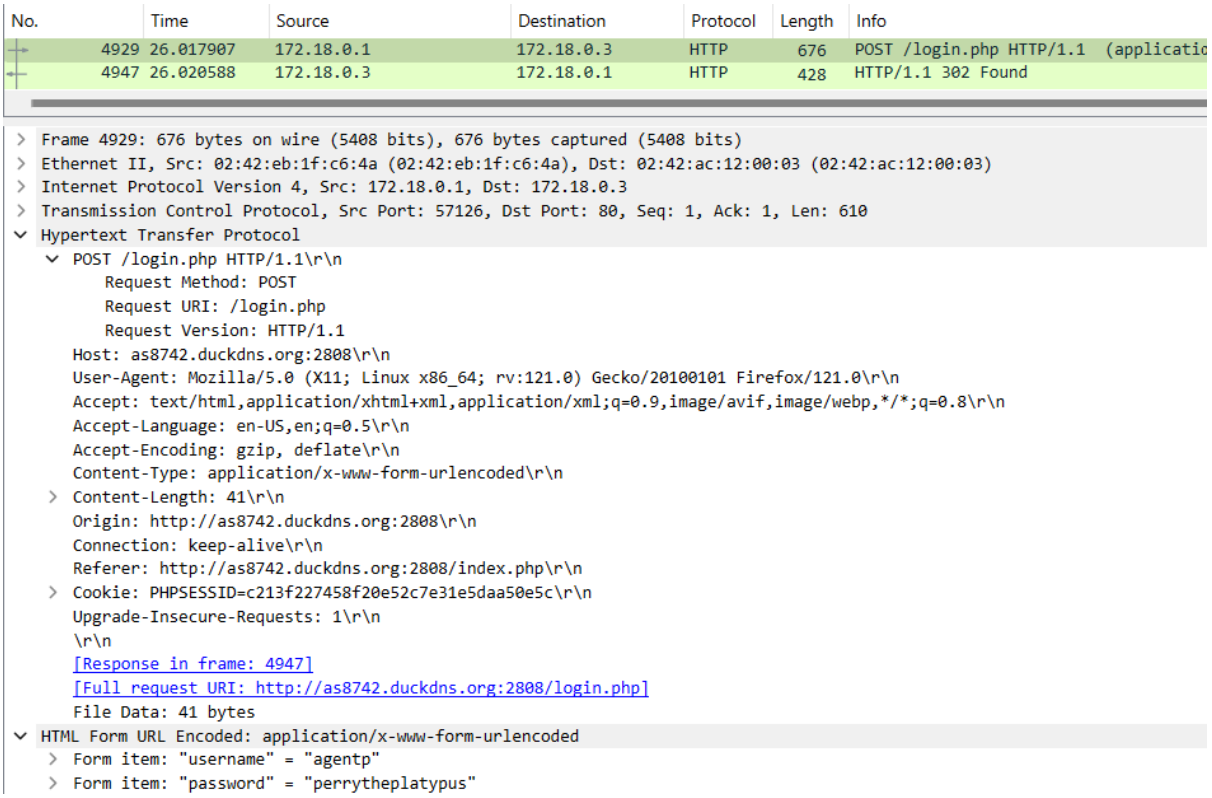


- Atacker hoặc đặc vụ (người dùng thực sự) đã thực hiện tổng cộng 6 lần đăng nhập vào trang web với các account sau:

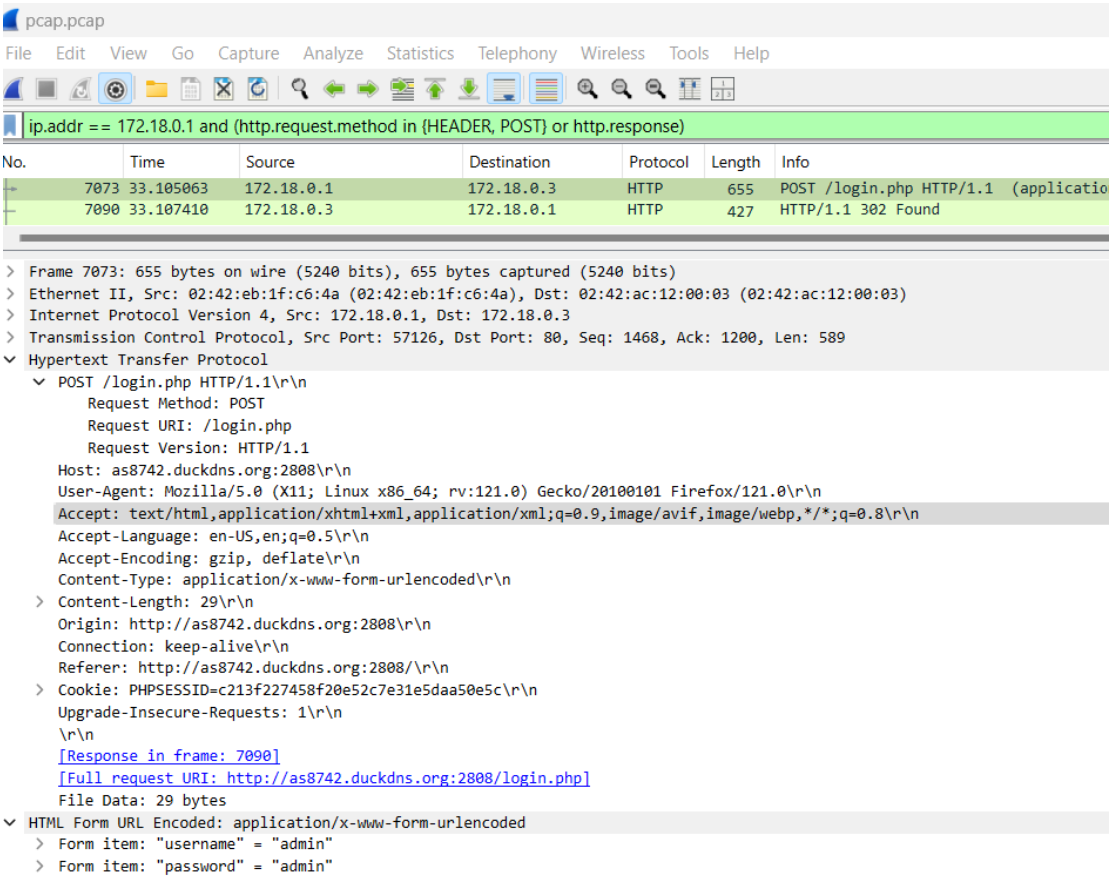
- Account 1



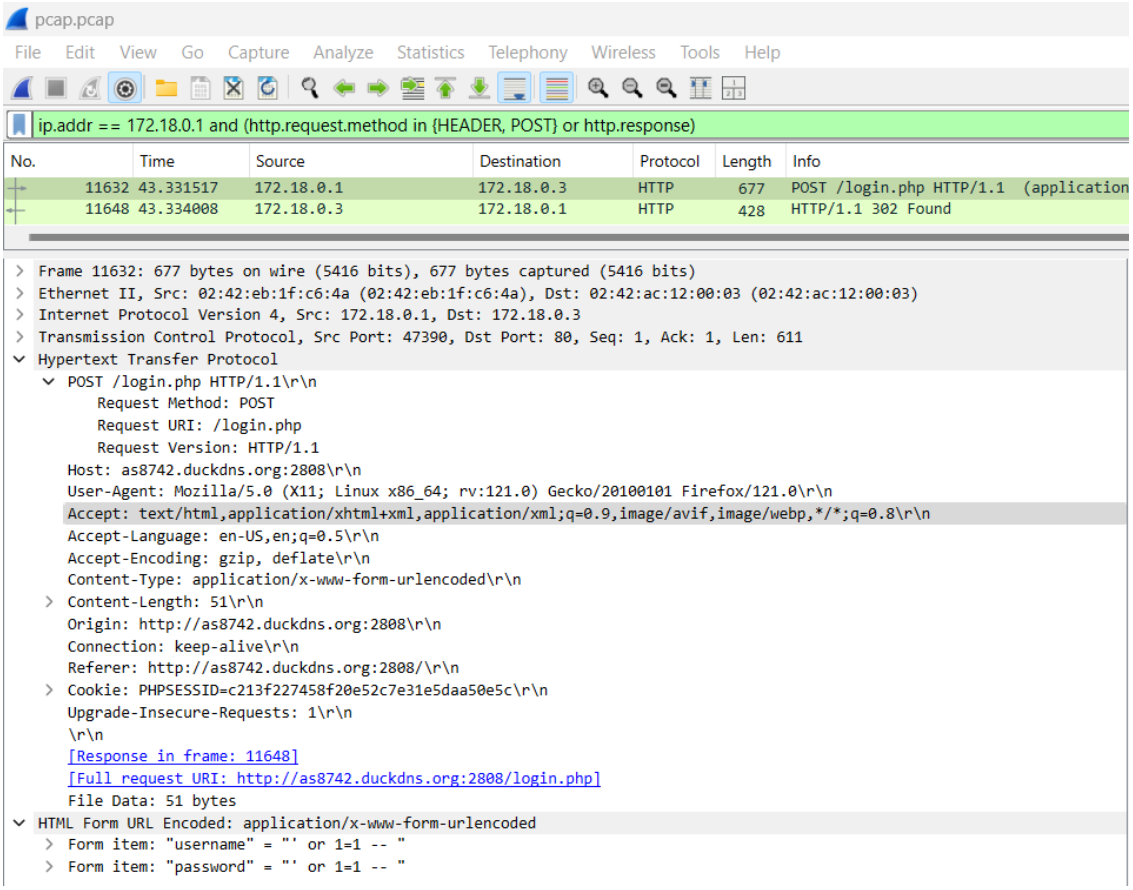
- Account 2



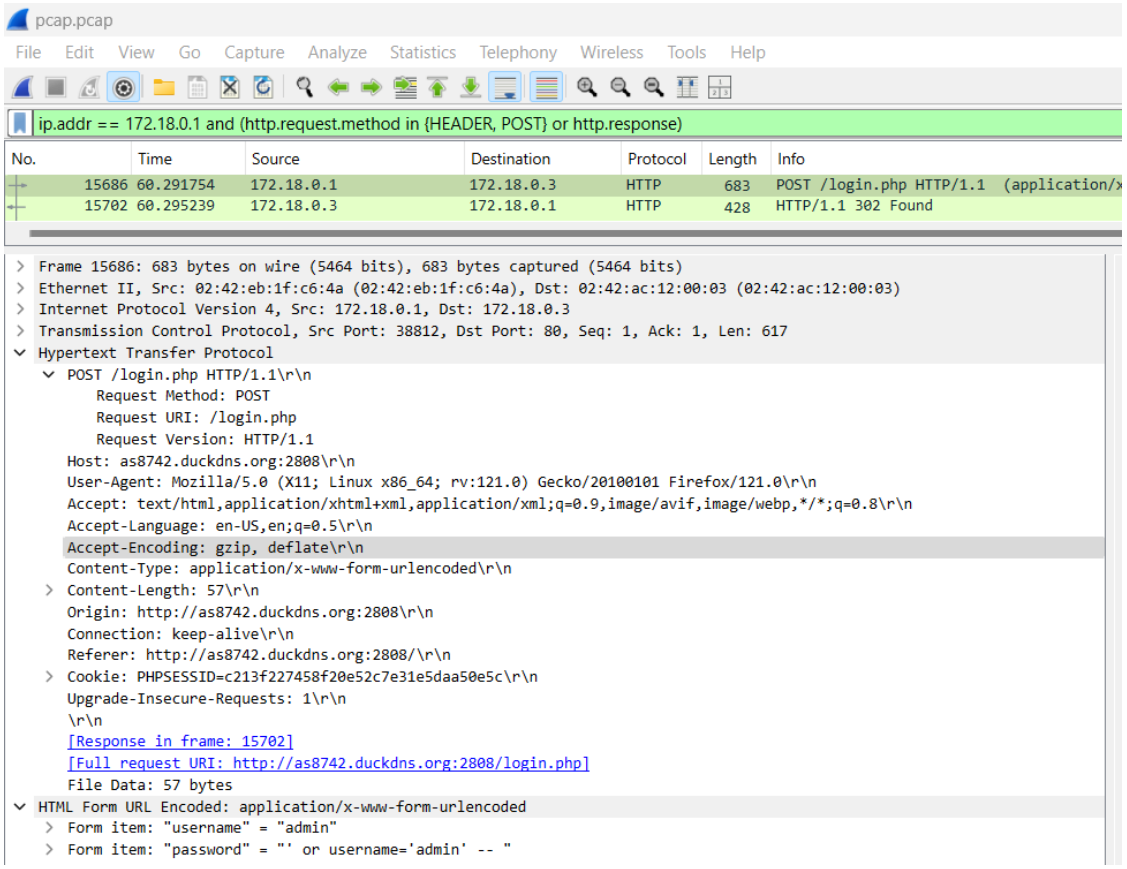
- Account 3



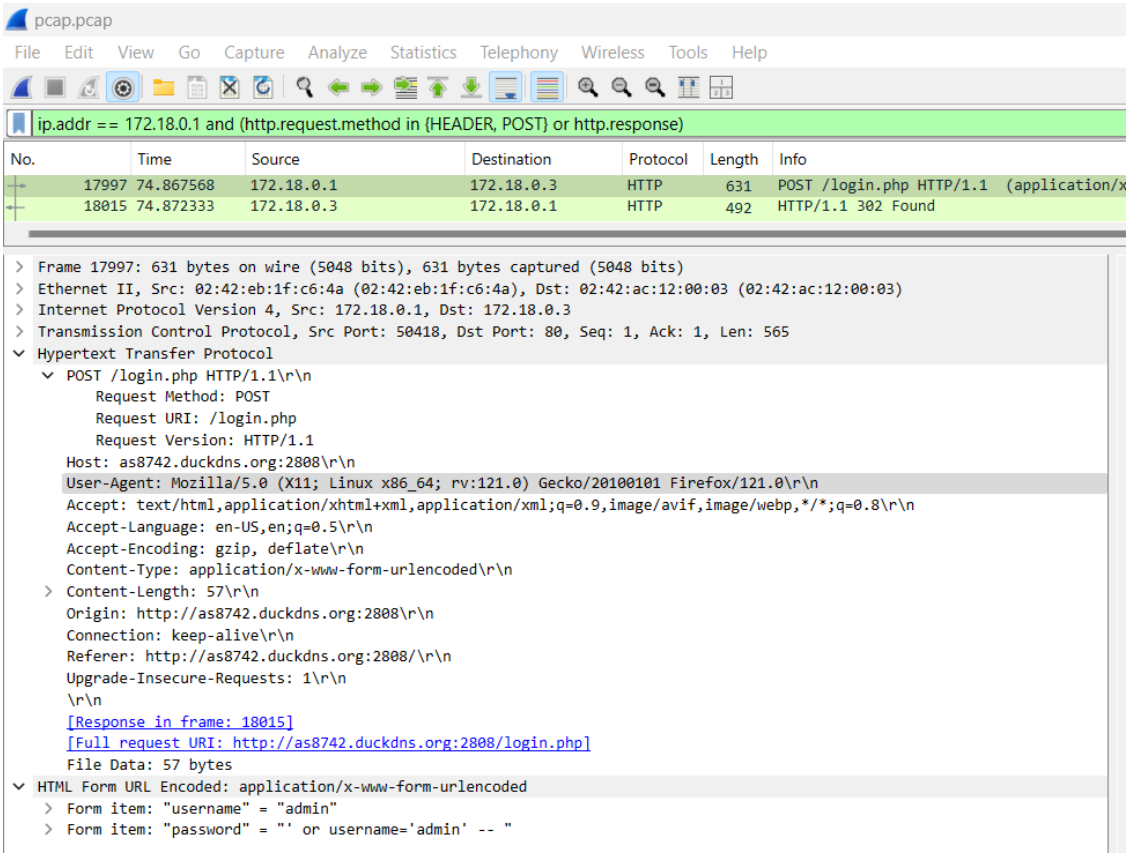
- Account 4



- Account 5

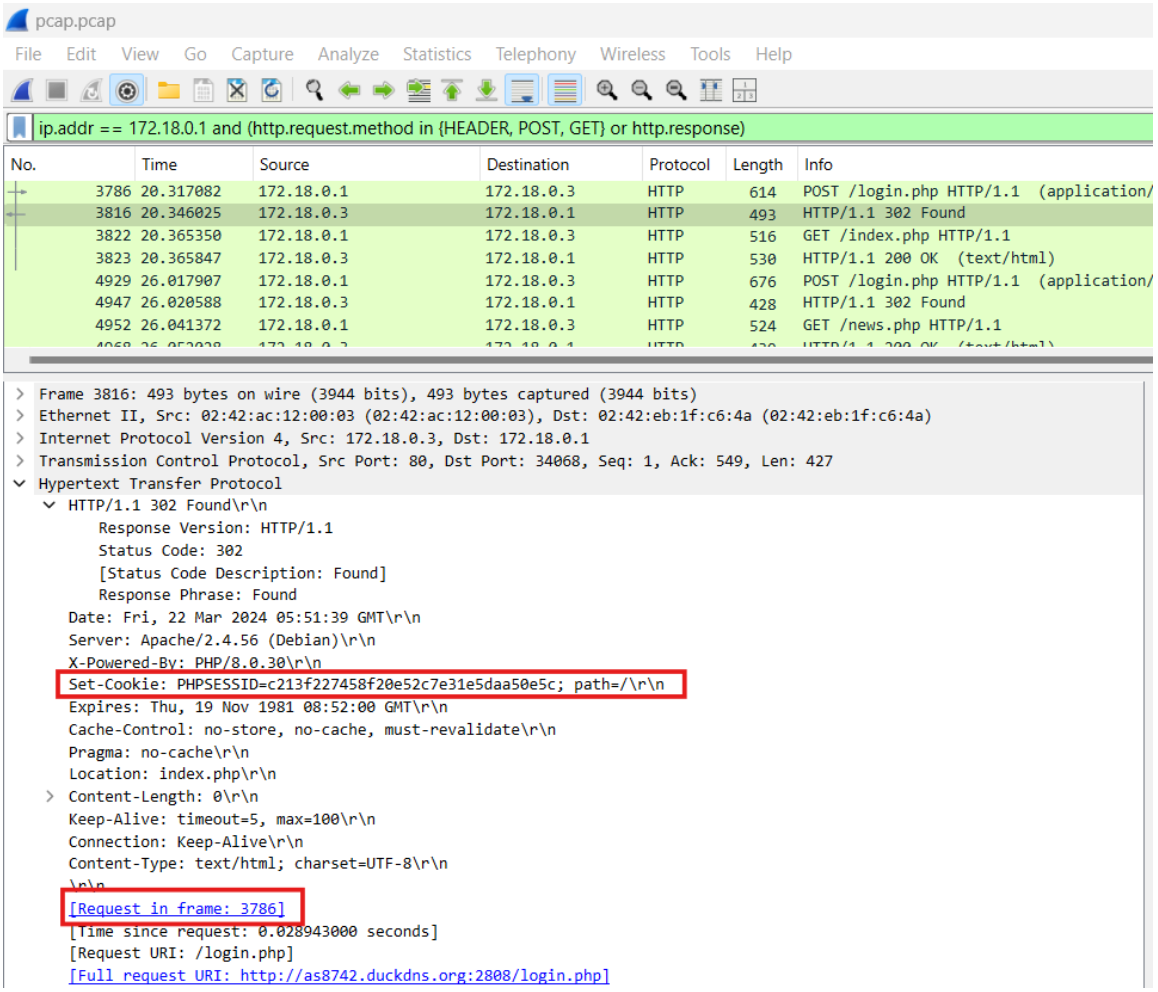


- Account 6



- Tất cả đều nhận được phản hồi 302 Found và truy cập được trang index.php nhưng account 1 và 6 lại không có cookie nên khá đáng ngờ.
- Tuy nhiên sau khi phân tích thêm thì có vẻ từ lần đăng nhập account 1 tới lần đăng nhập account 5 đều thực hiện trên 1 thiết bị và dùng 1 cookie giống nhau trong suốt quá trình tương tác với web nên nguyên nhân không có cookie là do đây là lần đầu đặc vụ (người dùng thực) đăng nhập vào web.

Cookie được tạo ra sau khi đăng nhập Account 1



Cookie của tương tác khác giữa máy này và web server

No.	Time	Source	Destination	Protocol	Length	Info
3786	20.317082	172.18.0.1	172.18.0.3	HTTP	614	POST /login.php HTTP/1.1 (application/
3816	20.346025	172.18.0.3	172.18.0.1	HTTP	493	HTTP/1.1 302 Found
3822	20.365350	172.18.0.1	172.18.0.3	HTTP	516	GET /index.php HTTP/1.1
3823	20.365847	172.18.0.3	172.18.0.1	HTTP	530	HTTP/1.1 200 OK (text/html)
4929	26.017907	172.18.0.1	172.18.0.3	HTTP	676	POST /login.php HTTP/1.1 (application/
4947	26.020588	172.18.0.3	172.18.0.1	HTTP	428	HTTP/1.1 302 Found
4952	26.041372	172.18.0.1	172.18.0.3	HTTP	524	GET /news.php HTTP/1.1

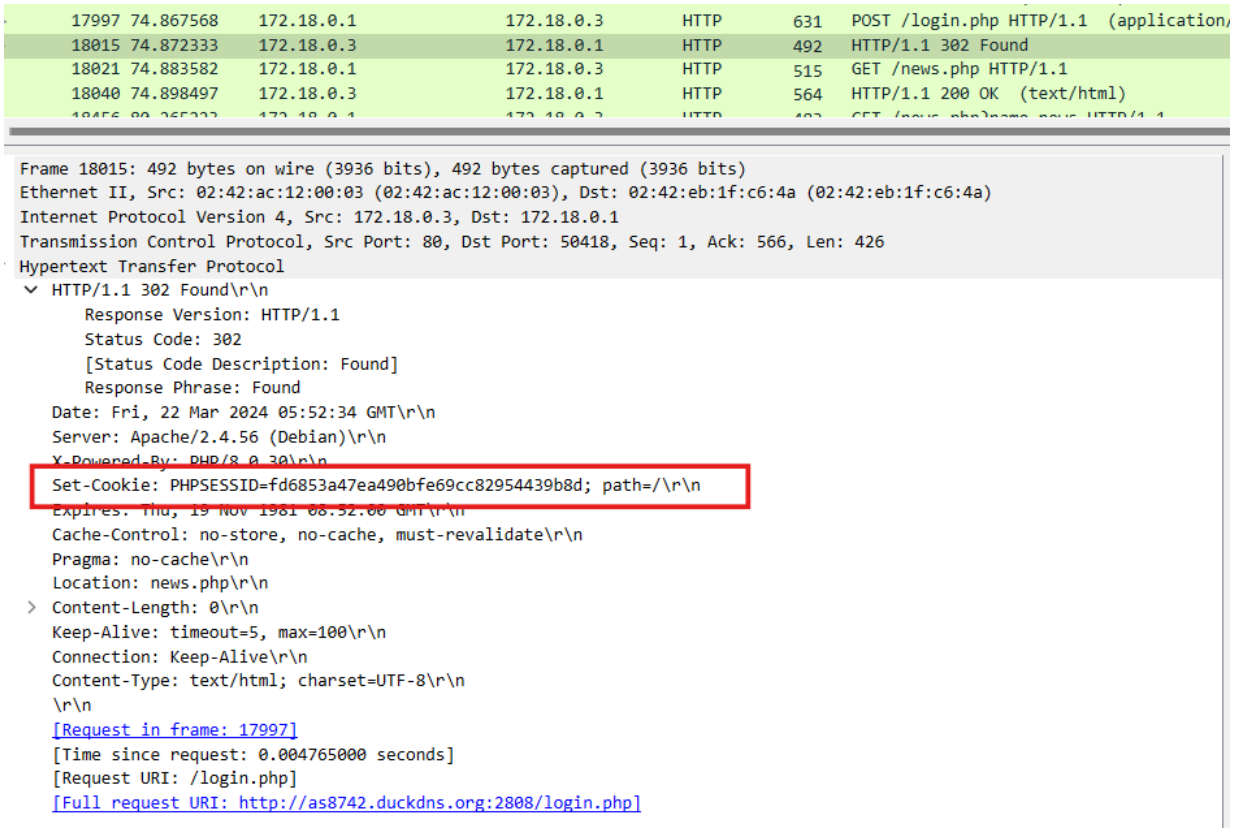
```
> Frame 3822: 516 bytes on wire (4128 bits), 516 bytes captured (4128 bits)
> Ethernet II, Src: 02:42:eb:1f:c6:4a (02:42:eb:1f:c6:4a), Dst: 02:42:ac:12:00:03 (02:42:ac:12:00:03)
> Internet Protocol Version 4, Src: 172.18.0.1, Dst: 172.18.0.3
> Transmission Control Protocol, Src Port: 34068, Dst Port: 80, Seq: 549, Ack: 428, Len: 450
v Hypertext Transfer Protocol
  v GET /index.php HTTP/1.1\r\n
      Request Method: GET
      Request URI: /index.php
      Request Version: HTTP/1.1
    Host: as8742.duckdns.org:2808\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Referer: http://as8742.duckdns.org:2808/\r\n
    Connection: keep-alive\r\n
  > Cookie: PHPSESSID=c213f227458f20e52c7e31e5daa50e5c\r\n
    Upgrade-Insecure-Requests: 1\r\n
    \r\n
    [Response in frame: 3823]
    [Full request URI: http://as8742.duckdns.org:2808/index.php]
```

No.	Time	Source	Destination	Protocol	Length	Info
15686	60.291754	172.18.0.1	172.18.0.3	HTTP	683	POST /login.php HTTP/1.1 (appli
15702	60.295239	172.18.0.3	172.18.0.1	HTTP	428	HTTP/1.1 302 Found
15704	60.300651	172.18.0.1	172.18.0.3	HTTP	515	GET /news.php HTTP/1.1
15719	60.301927	172.18.0.3	172.18.0.1	HTTP	439	HTTP/1.1 200 OK (text/html)
16554	66.099378	172.18.0.1	172.18.0.3	HTTP	465	GET / HTTP/1.1
16556	66.099735	172.18.0.3	172.18.0.1	HTTP	531	HTTP/1.1 200 OK (text/html)
17997	74.867568	172.18.0.1	172.18.0.3	HTTP	631	POST /login.php HTTP/1.1 (appli

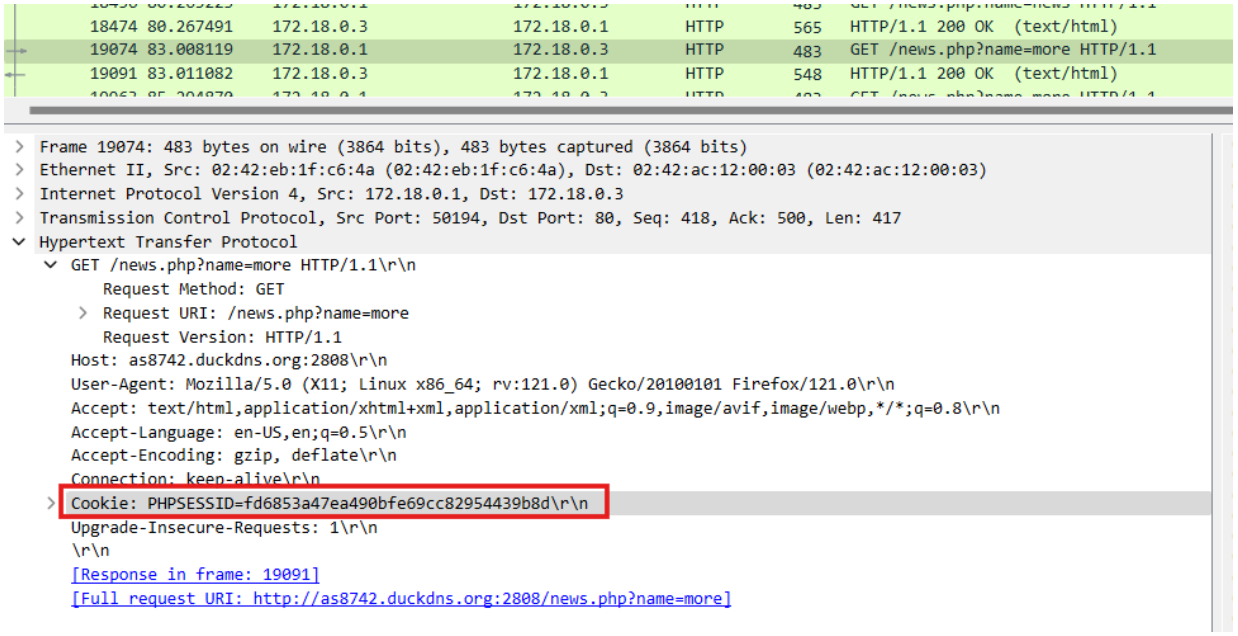
```
> Frame 15686: 683 bytes on wire (5464 bits), 683 bytes captured (5464 bits)
> Ethernet II, Src: 02:42:eb:1f:c6:4a (02:42:eb:1f:c6:4a), Dst: 02:42:ac:12:00:03 (02:42:ac:12:00:03)
> Internet Protocol Version 4, Src: 172.18.0.1, Dst: 172.18.0.3
> Transmission Control Protocol, Src Port: 38812, Dst Port: 80, Seq: 1, Ack: 1, Len: 617
v Hypertext Transfer Protocol
  v POST /login.php HTTP/1.1\r\n
      Request Method: POST
      Request URI: /login.php
      Request Version: HTTP/1.1
    Host: as8742.duckdns.org:2808\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
  > Content-Length: 57\r\n
    Origin: http://as8742.duckdns.org:2808\r\n
    Connection: keep-alive\r\n
    Referer: http://as8742.duckdns.org:2808/\r\n
  > Cookie: PHPSESSID=c213f227458f20e52c7e31e5daa50e5c\r\n
    Upgrade-Insecure-Requests: 1\r\n
    \r\n
    [Response in frame: 15702]
    [Full request URI: http://as8742.duckdns.org:2808/login.php]
    File Data: 57 bytes
v HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "username" = "admin"
  > Form item: "password" = "' or username='admin' -- "
```

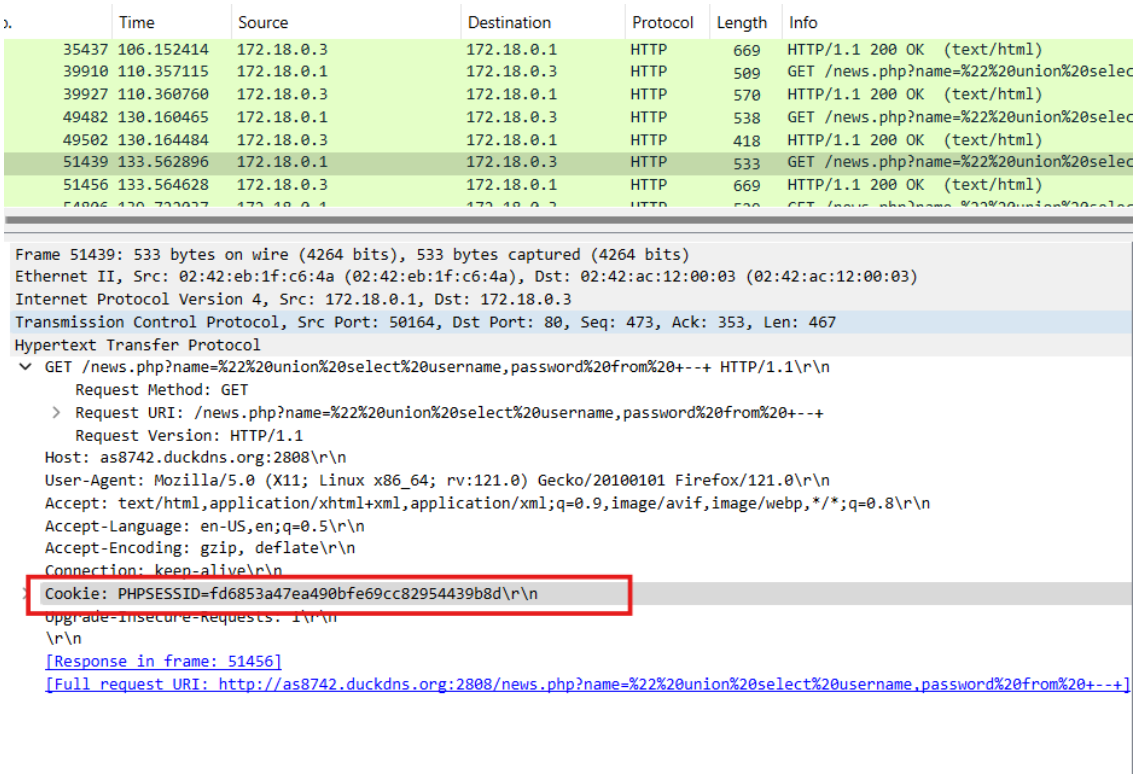
- Ngoài ra nguyên nhân mà em xác định account 6 là tài khoản mà hacker đăng nhập là do cookie được tạo ra sau lần đăng nhập này cũng chính là cookie tồn tại xuyên suốt quá trình diễn ra quá trình tấn công web (cụ thể là tấn công SQL injection)

Cookie tạo ra sau lần đăng nhập account 6



Cookie trong suốt quá trình diễn ra tấn công web

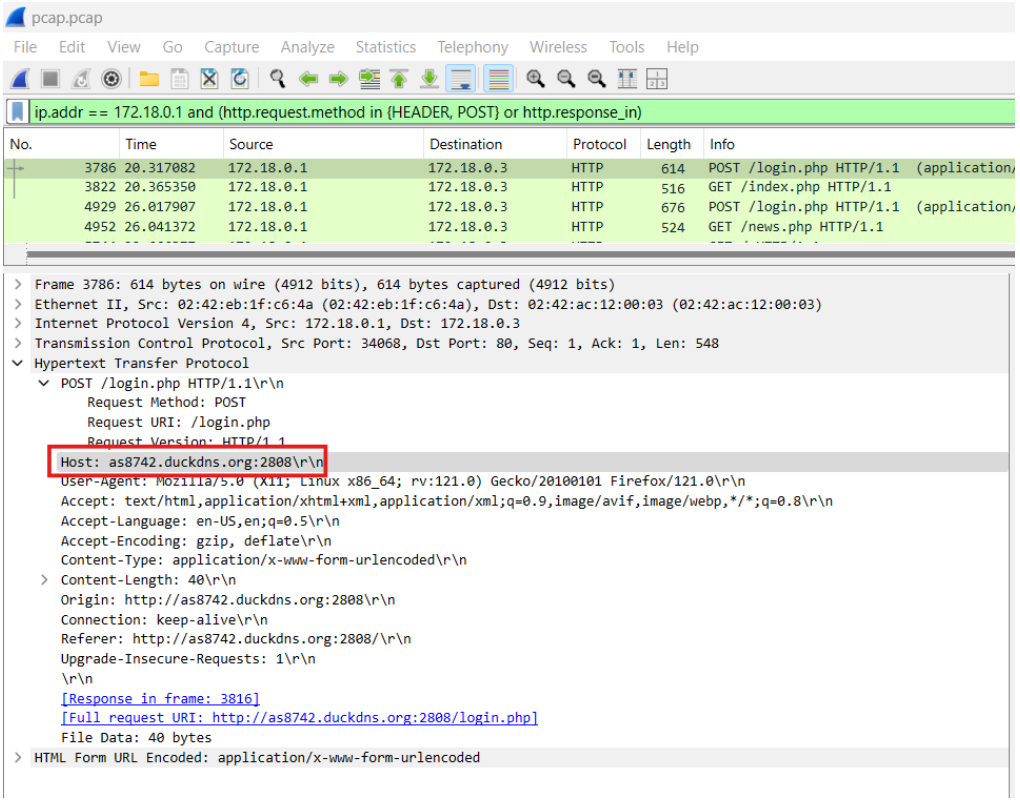




- Vậy tài khoản mà hacker dùng để đăng nhập là
  Username: admin
  Password: ‘ or username=’admin’ --

f) Server mà hacker dùng để test là gì?

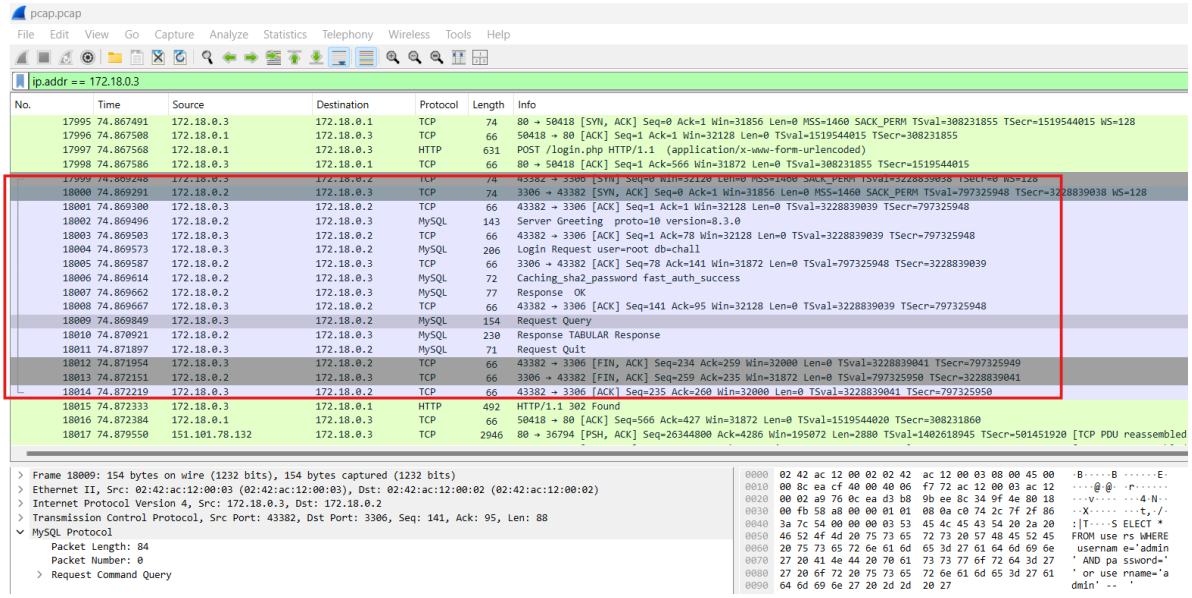
- Xem ngẫu nhiên 1 gói tin POST, em thấy trong trường Hyper Text Transfer Protocol có các thông tin cụ thể về server



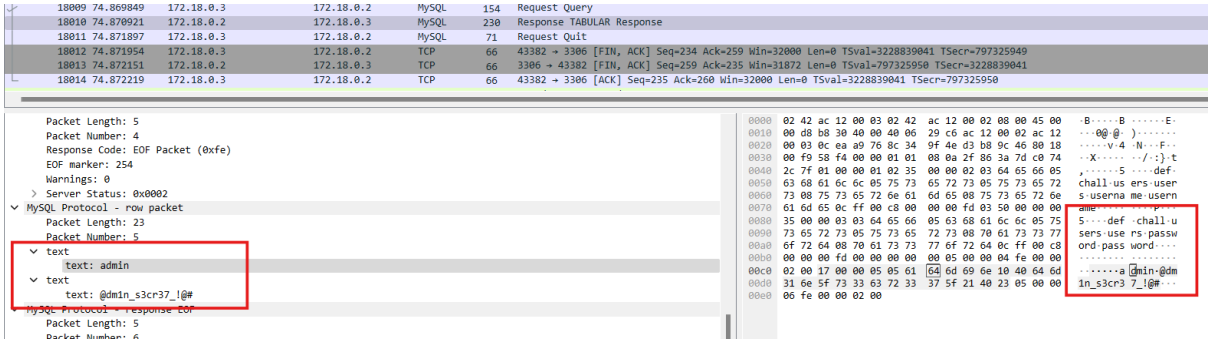
⇨ Server mà hacker đang test có domain: as8742.duckdns.org:2808

g) Có vẻ là hacker đã lấy được mật khẩu của admin. Nhưng có người lại bảo là chưa. Vậy hacker đã lấy được mật khẩu của admin chưa? Mật khẩu của admin là gì? Hacker đã lấy được gì?

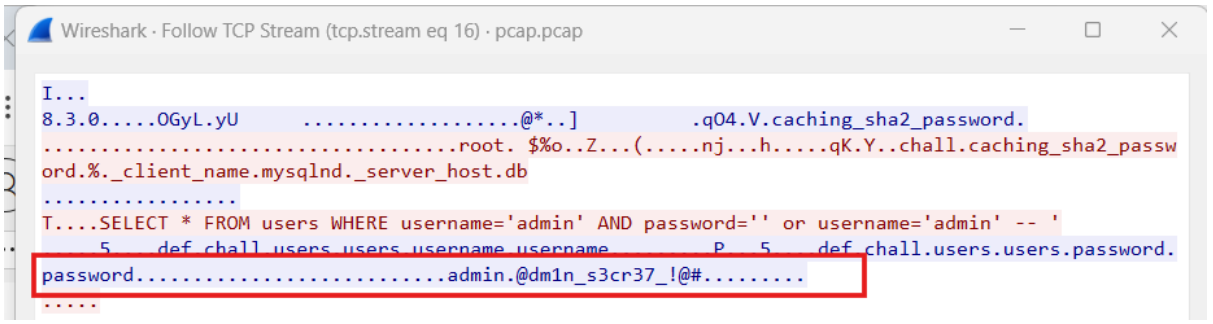
- Để xác thực tin đồn, em sẽ tiếp tục phân tích các gói tin giao tiếp giữa web server (172.18.0.3) và database (172.18.0.2) ngay sau khi hackertruy cập được vào trang web. Có thể thấy ngay sau khi hacker login thành công thì server lặp tức giao tiếp với database để xác thực tài khoản (từ gói 17999 - 18040)



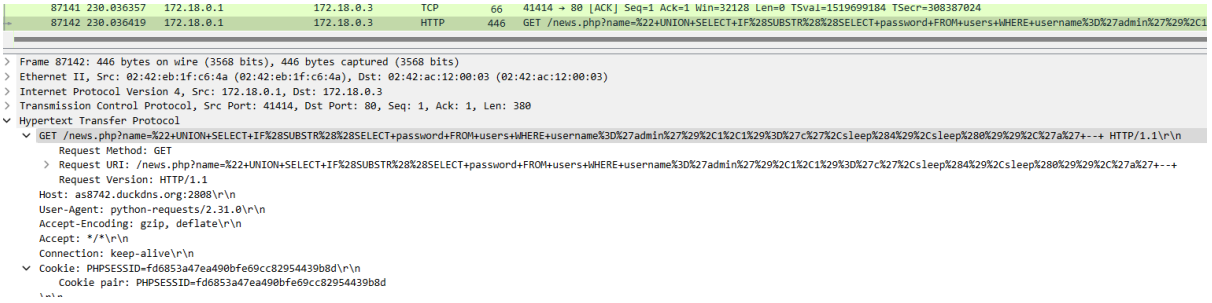
- Ngay khi xem nội dung gói 18010 thì em tìm được thông tin khá đáng nghi



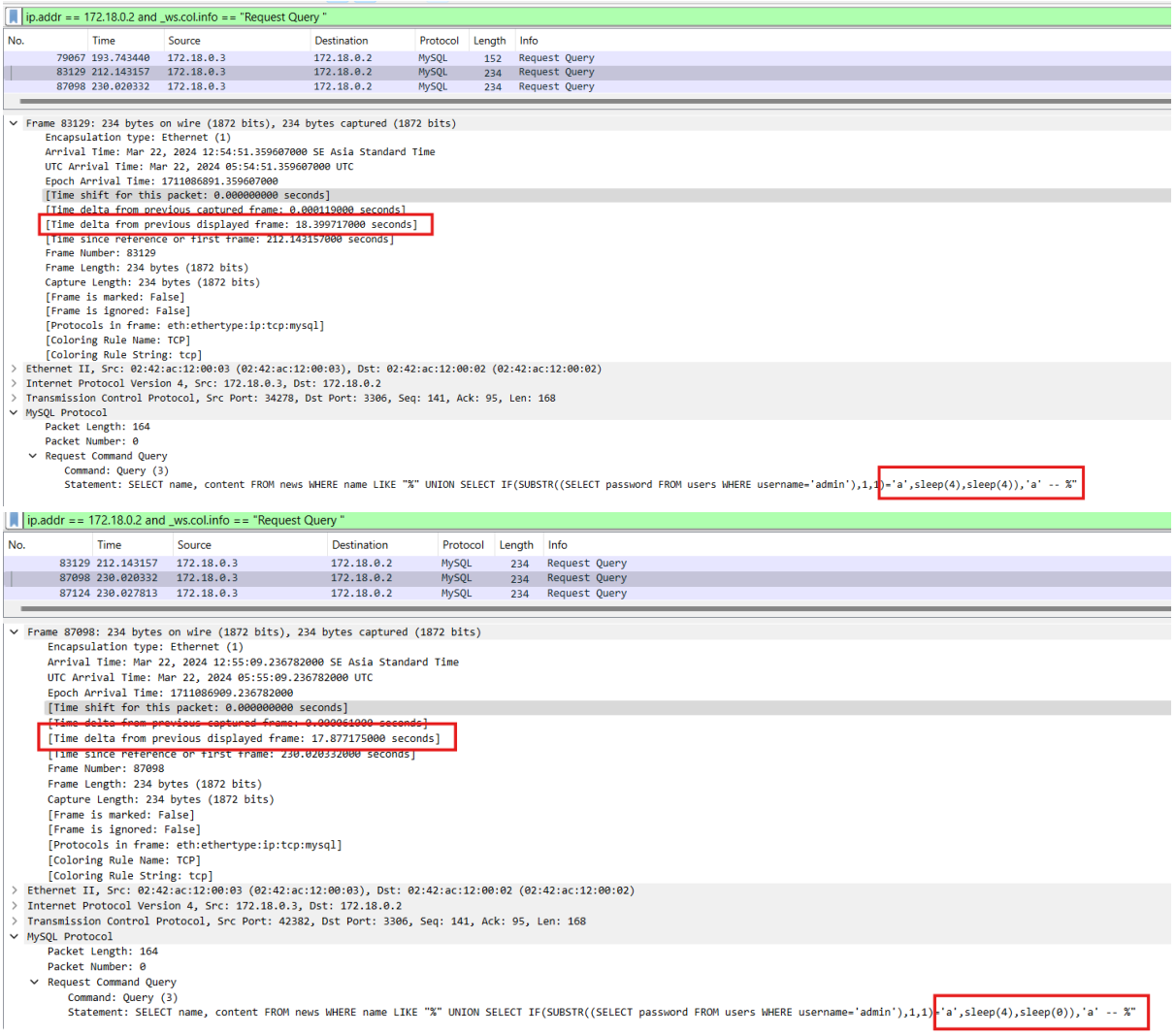
- Em tiến hành Follow TCP Stream với gói tin này thì tìm được 1 password như sau:

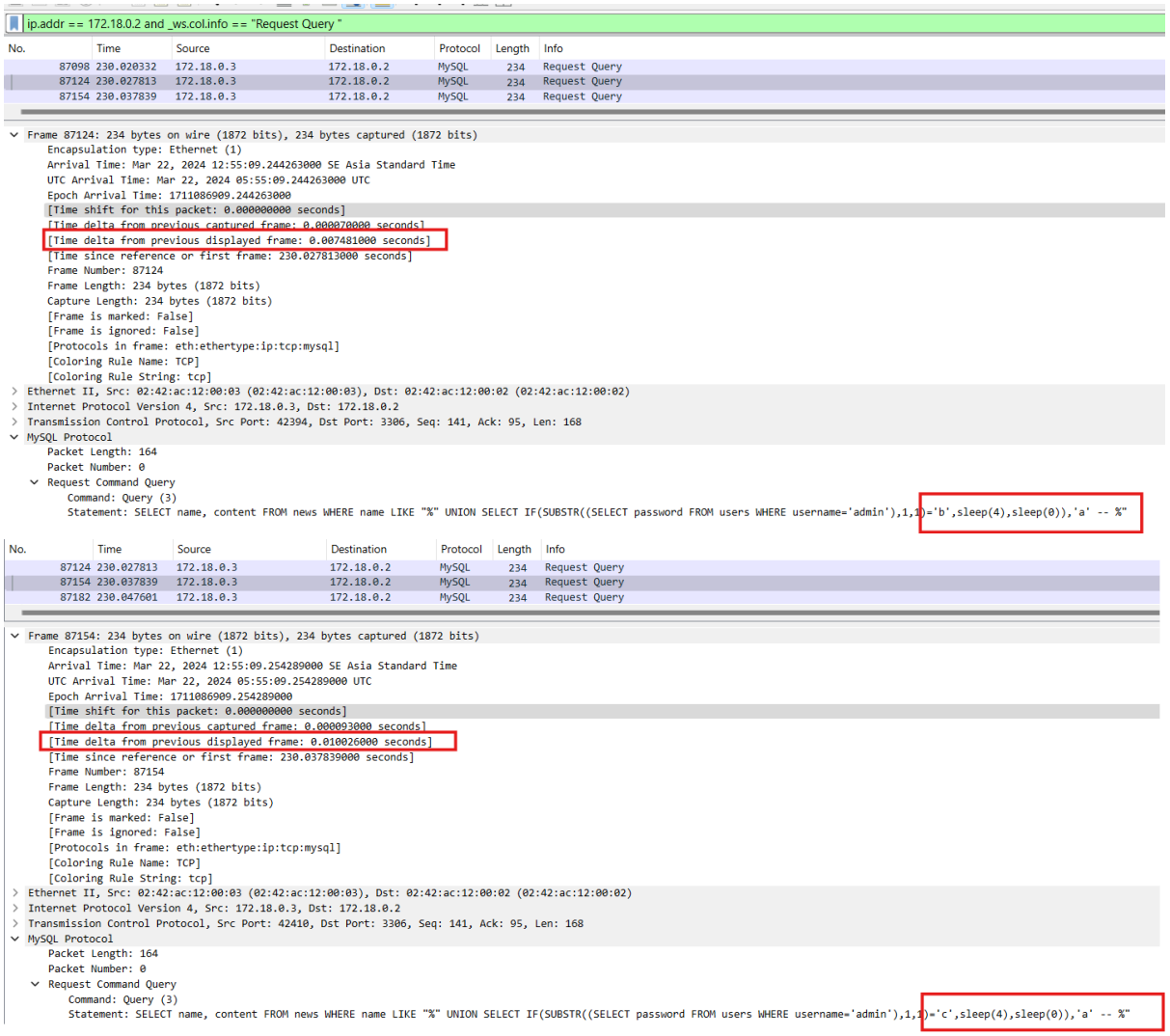


- Vì username là admin nên có vẻ đây là password của tài khoản admin. Giờ để kiểm tra xem hacker có biết được password đúng chưa, em sẽ tìm các password mà Hacker đã thử và so sánh với password đúng này.
- Hacker đã gửi đi rất nhiều câu truy vấn tới web server nhưng có vẻ trong đó không có password của attacker nên em sẽ phân tích thêm các câu truy vấn này. Trước tiên các câu truy vấn trong request header đều bị encode hết nên để dễ nhìn hơn em sẽ tìm cách decode chúng.



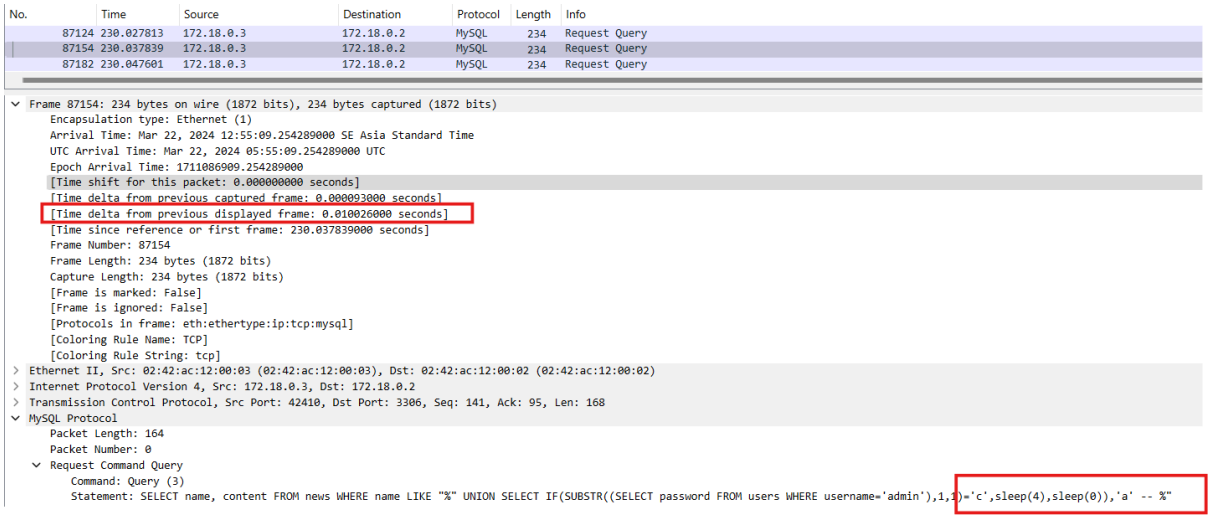
- Nhưng may mắn thay, em tìm được bản rõ của các câu truy vấn này trong các gói tin Request query mà web server (IP là 172.18.0.1) gửi cho hệ thống database (có IP là 172.18.0.2)



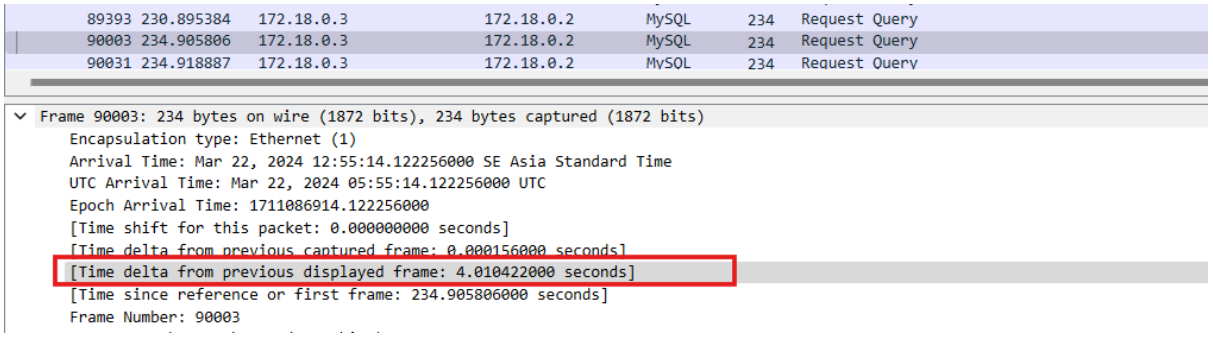


- Giá trị Statement chính là chuỗi URL mà hacker gửi cho web server nhằm tìm password cho tài khoản admin.
- Sau khi phân tích các giá trị Statement mà hacker gửi đi thì em thấy hacker thực hiện SQL injection tấn công webserver để lấy password của tài khoản admin bằng cách chèn các câu lệnh truy vấn SQL tích hợp với điều kiện if else.
- Có thể hiểu ý tưởng tấn công SQLi như sau:

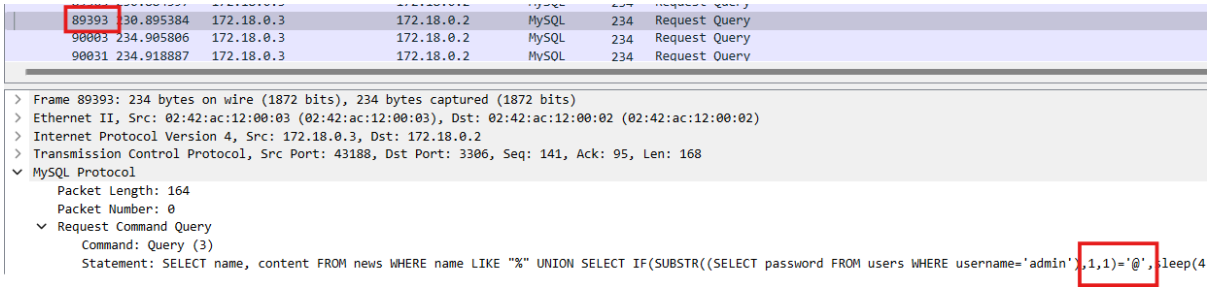
Thực hiện bruteforce từng ký tự một trong password, mỗi khi tìm được 1 ký tự tại 1 vị trí cụ thể trong password thì gọi hàm sleep(4) để ngủ 4 giây tức là trong 4 giây này sẽ không gửi truy vấn nữa, ngược lại gọi hàm sleep(0) để ngủ 0 giây.
Ví dụ:



- Từ lệnh này: database sẽ kiểm tra xem ký tự thứ 1 trong password có phải ‘c’ không. Nếu đúng thì chương trình gửi truy vấn ngủ 4 giây, ngược lại ngủ 0 giây hay tiếp tục công việc gửi truy vấn tới web server.
  ‘a’ + --: Kết thúc câu lệnh SQL với dấu "--" để bỏ qua mọi thứ còn lại
- Do đó để tìm được passord mà hacker đã tìm ra thì em sẽ tìm các gói tin mà gói tin ngay sau nó có Time delta from previous displayed frame >= 4 giây (vì giá trị này cho biết khoảng chênh lệch thời gian giữa 2 gói tin liên tiếp trong cửa sổ chính sau khi đã thực hiện filter)/
- Dựa trên chênh lêch thời gian giữa các gói tin, em tìm các gói có khoảng chênh lênh 4 giây
- Vì gói 90003 có Time > 4s nên gói trước N0.89393 chứa ký tự đúng đầu tiên



Xem nội dung gói 89393 thì tìm được ký tự @



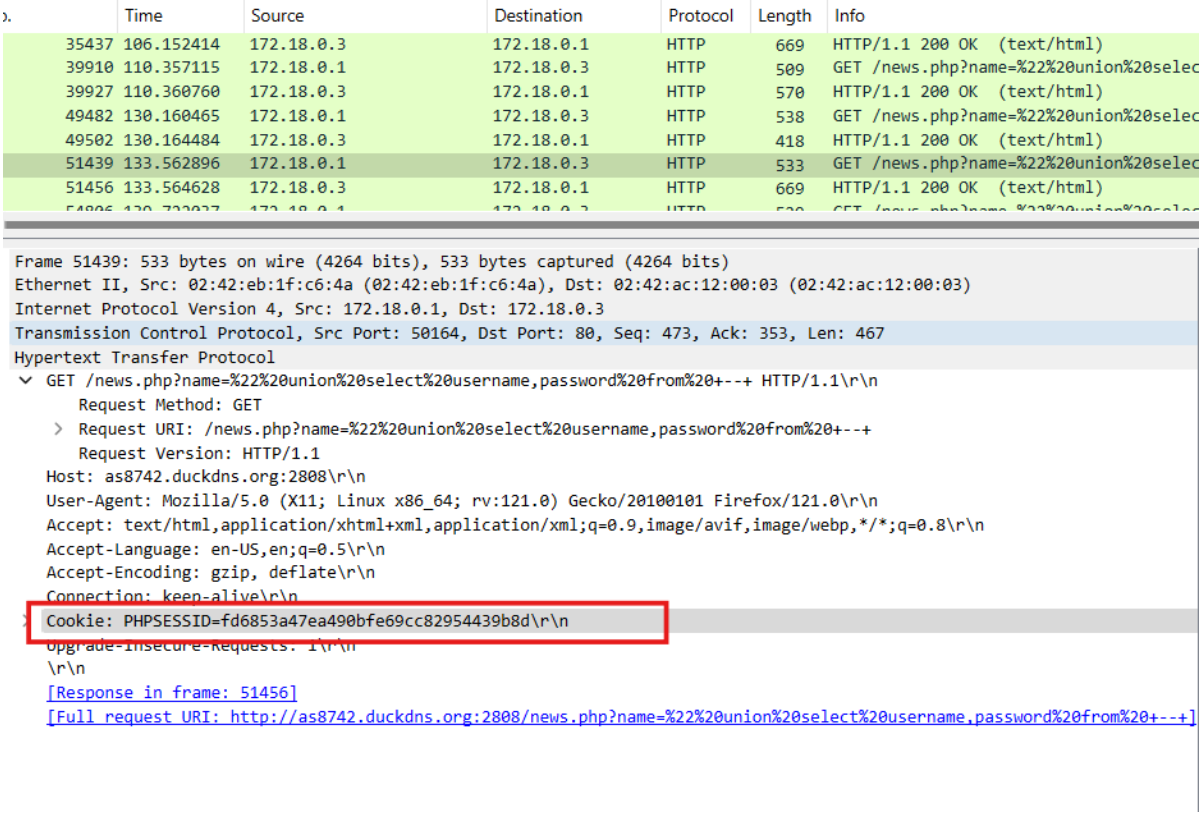
- Cứ tiếp tục như vậy em tìm được chuỗi: @dm1n_a3cr37_!@#.
- So sánh với password gốc thì khác nhau ở vị trí thứ 7. Thay vì ‘s’, attcker chọn ‘a’ nên bị sai password.

- Vậy Hacker vẫn chưa biết được password của tài khoản admin

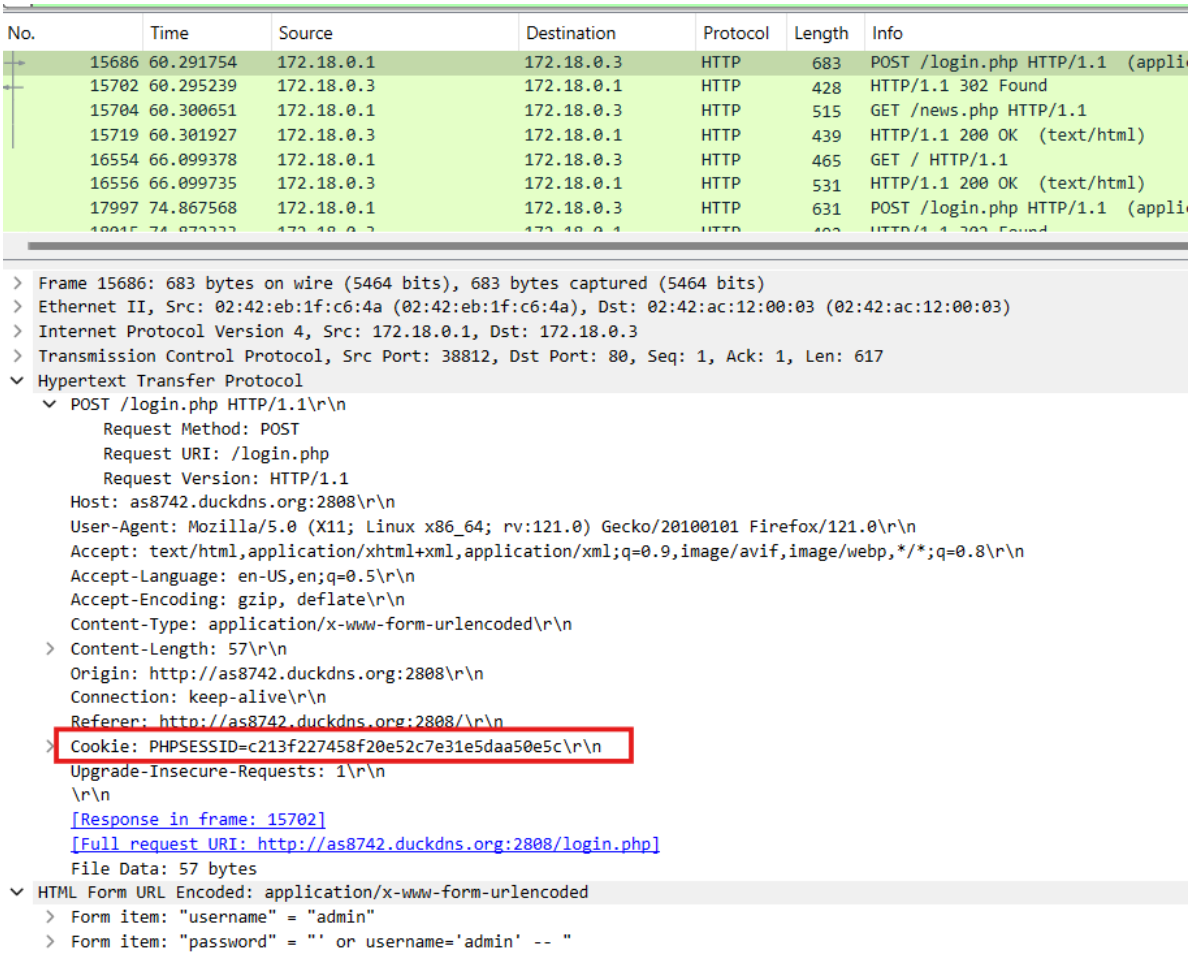
h) Có nên tình nghi đặc vụ đó là người đã thực hiện cuộc tấn công không? Tại sao?

- Từ các kết quả, thông tin thu thập được từ câu E, em nghĩ khả năng đặc vụ là người thực hiện cuộc tấn công là cực kỳ thấp nên có thể tạm coi như đặc vụ vô tội. Bởi vì như trình bày trong câu E thì ai đó đã dùng MDNS để mạo danh IP 172.18.0.1 rồi thực hiện tấn công thành công bằng SQL injection nhằm vào web server. Ngoài ra cookie giữa lần diễn ra tấn công và cookie khi đặc vụ tương tác với web là hoàn toàn khác nhau.

Đây là cookie khi diễn ra tấn công:



Đây là cookie khi đặc vụ tương tác với web:



No.	Time	Source	Destination	Protocol	Length	Info
15686	60.291754	172.18.0.1	172.18.0.3	HTTP	683	POST /login.php HTTP/1.1 (appli
15702	60.295239	172.18.0.3	172.18.0.1	HTTP	428	HTTP/1.1 302 Found
15704	60.300651	172.18.0.1	172.18.0.3	HTTP	515	GET /news.php HTTP/1.1
15719	60.301927	172.18.0.3	172.18.0.1	HTTP	439	HTTP/1.1 200 OK (text/html)
16554	66.099378	172.18.0.1	172.18.0.3	HTTP	465	GET / HTTP/1.1
16556	66.099735	172.18.0.3	172.18.0.1	HTTP	531	HTTP/1.1 200 OK (text/html)
17997	74.867568	172.18.0.1	172.18.0.3	HTTP	631	POST /login.php HTTP/1.1 (appli

```
> Frame 15686: 683 bytes on wire (5464 bits), 683 bytes captured (5464 bits)
> Ethernet II, Src: 02:42:eb:1f:c6:4a (02:42:eb:1f:c6:4a), Dst: 02:42:ac:12:00:03 (02:42:ac:12:00:03)
> Internet Protocol Version 4, Src: 172.18.0.1, Dst: 172.18.0.3
> Transmission Control Protocol, Src Port: 38812, Dst Port: 80, Seq: 1, Ack: 1, Len: 617
v Hypertext Transfer Protocol
  v POST /login.php HTTP/1.1\r\n
      Request Method: POST
      Request URI: /login.php
      Request Version: HTTP/1.1
    Host: as8742.duckdns.org:2808\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Content-Type: application/x-www-form-urlencoded\r\n
  > Content-Length: 57\r\n
    Origin: http://as8742.duckdns.org:2808\r\n
    Connection: keep-alive\r\n
    Referer: http://as8742.duckdns.org:2808/\r\n
  > Cookie: PHPSESSID=c213f227458f20e52c7e31e5daa50e5c\r\n
    Upgrade-Insecure-Requests: 1\r\n
    \r\n
    [Response in frame: 15702]
    [Full request URI: http://as8742.duckdns.org:2808/login.php]
    File Data: 57 bytes
v HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "username" = "admin"
  > Form item: "password" = "' or username='admin' -- "
```

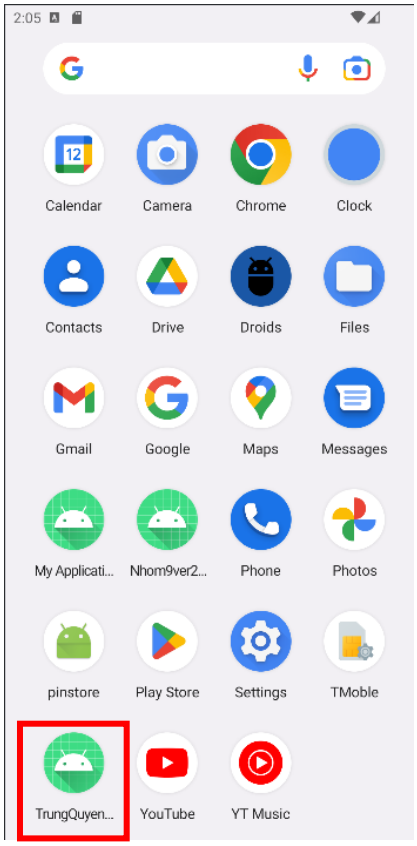
C. KỊCH BẢN 3 - Android

Đề bài:

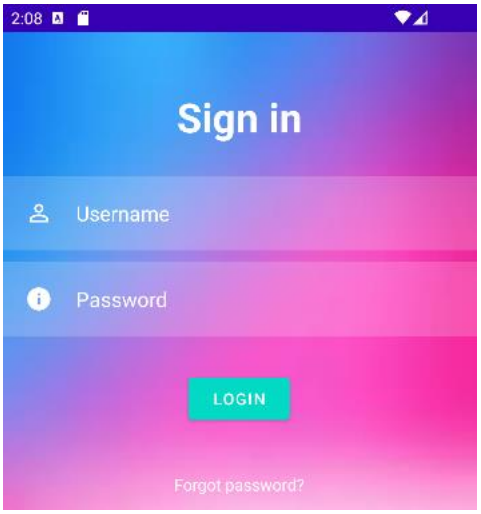
bypass_login.apk - Như tên file, để tìm được flag hãy dịch ngược ứng dụng này và tìm cách bypass login. Đồng thời, cho biết dev đã mắc lỗi gì khi lập trình dẫn đến dễ dàng để lộ thông tin nhạy cảm như vậy. Đề xuất biện pháp khắc phục.

Giải:

- Cài đặt file bypass_login.apk vào máy ảo Android. Giao diện chính của ứng dụng.



- Khi vào ứng dụng sẽ phải nhập mật khẩu.

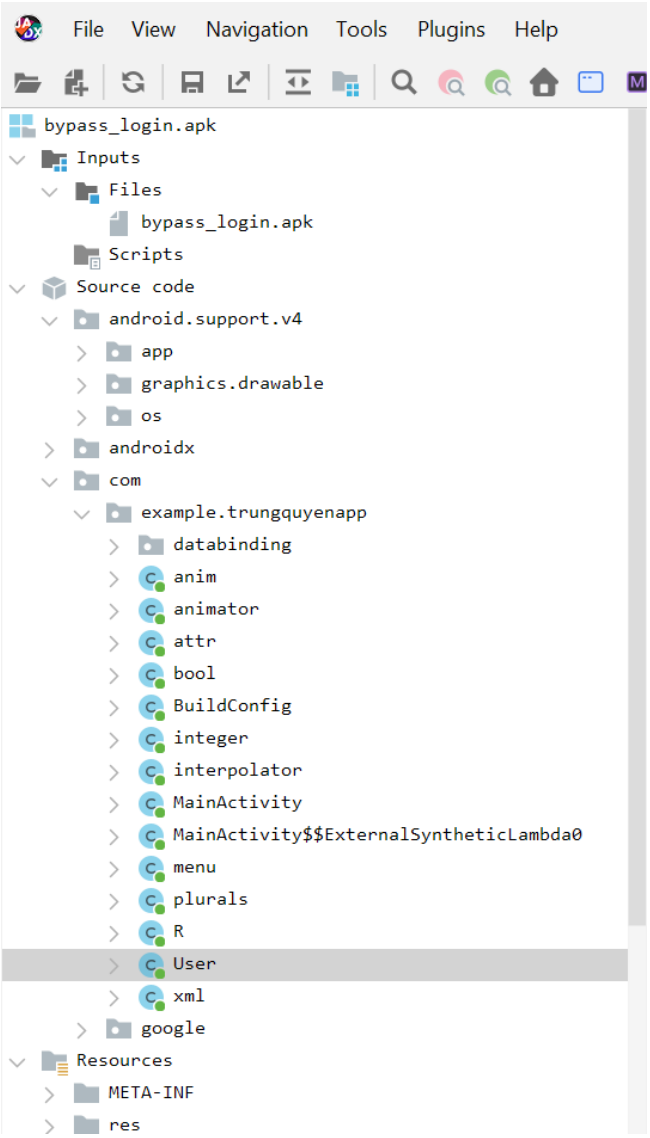


- Tải công cụ Jadx để tiến hành xem code của file apk. Nhóm sẽ thử vào file AndroidManifest.xml để xem bên trong có khai báo gì đặc biệt không.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0"
    android:compileSdkVersion="32"
    android:compileSdkVersionCodename="12"
    package="com.example.trungquyenapp"
    platformBuildVersionCode="32"
    platformBuildVersionName="12">
    <uses-sdk
        android:minSdkVersion="21"
        android:targetSdkVersion="32"/>
    <application
        android:theme="@style/Theme.TrungQuyenApp"
        android:label="@string/app_name"
        android:icon="@mipmap/ic_launcher"
        android:debuggable="true"
        android:testOnly="true"
        android:allowBackup="true"
        android:supportsRtl="true"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:appComponentFactory="androidx.core.app.CoreComponentFactory">
        <activity
            android:theme="@style/Theme.TrungQuyenApp.NoActionBar"
            android:label="@string/title_activity_user"
            android:name="com.example.trungquyenapp.User"
            android:exported="false"/>
        <activity
            android:name="com.example.trungquyenapp.MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <provider
            android:name="androidx.startup.InitializationProvider"
            android:exported="false"
            android:authorities="com.example.trungquyenapp.androidx-startup">
            <meta-data
                android:name="androidx.emoji2.text.EmojiCompatInitializer"
                android:value="androidx.startup"/>
            <meta-data
                android:name="androidx.lifecycle.ProcessLifecycleInitializer"
                android:value="androidx.startup"/>
        </provider>
    </application>
</manifest>
```

⇨ Có vẻ không có gì đặc biệt ở đây. Nhóm sẽ thử các file khác.

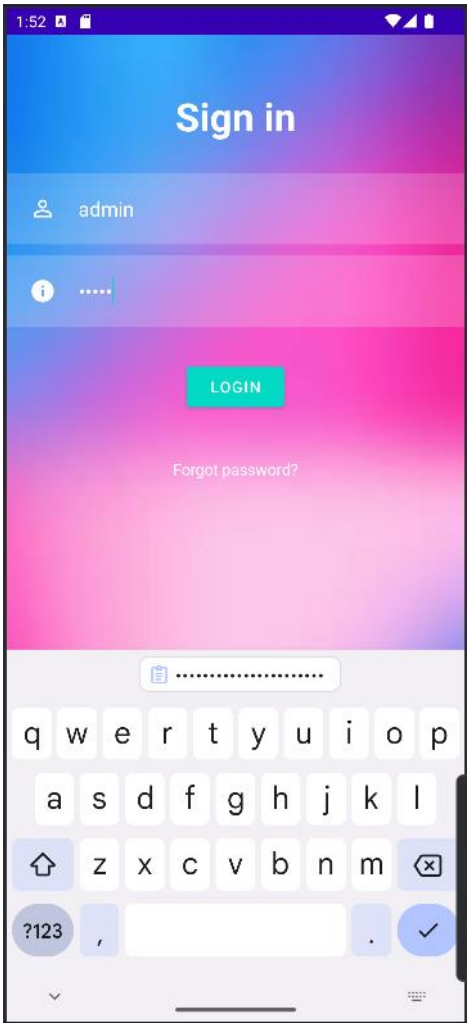


- Sau khi thử các file như BuildConfig, User, ... thì nhóm phát hiện được tài khoản và mật khẩu gắn cứng trong file MainActivity.

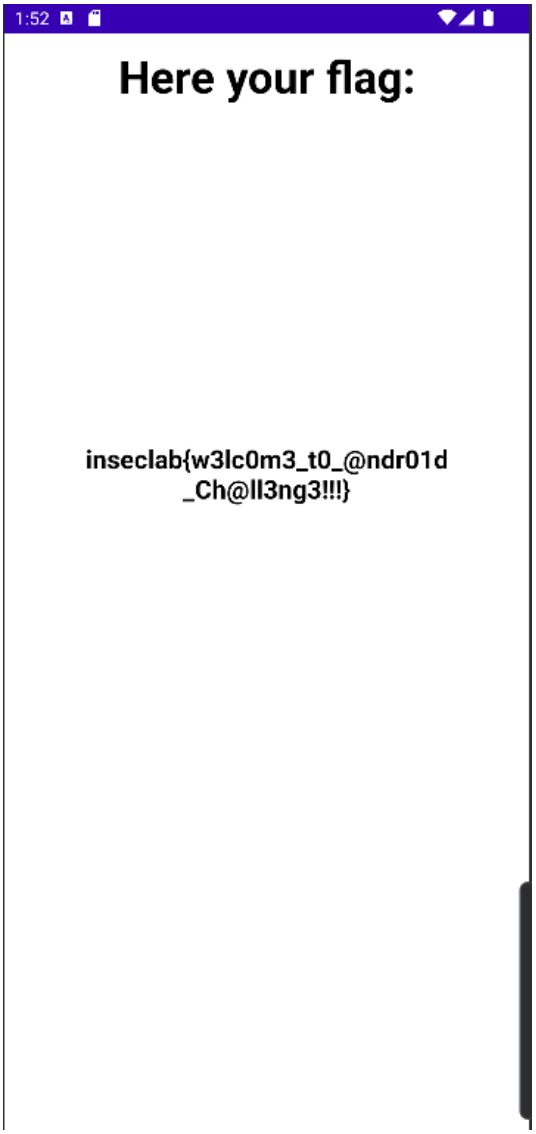
AndroidManifest.xml | MainActivity | User

```
package com.example.trungquyenapp;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import com.google.android.material.button.MaterialButton;

/* loaded from: classes3.dex */
public class MainActivity extends AppCompatActivity {
    @Override // androidx.fragment.app.FragmentActivity, androidx.activity.ComponentActivity, androidx.core.app.ComponentActivity, android.app.Activity
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final TextView username = (TextView) findViewById(R.id.username);
        final TextView password = (TextView) findViewById(R.id.password);
        MaterialButton loginbtn = (MaterialButton) findViewById(R.id.loginbtn);
        loginbtn.setOnClickListener(new View.OnClickListener() { // from class: com.example.trungquyenapp.MainActivity$$ExternalSyntheticLambda0
            @Override // android.view.View.OnClickListener
            public final void onClick(View view) {
                MainActivity.this.m26lambda$onCreate$0$comexampletrungquyenappMainActivity(username, password, view);
            }
        });
    }

    /* renamed from: lambda$onCreate$0$com-example-trungquyenapp-MainActivity, reason: not valid java name */
    public /* synthetic */ void m26lambda$onCreate$0$comexampletrungquyenappMainActivity(TextView username, TextView password, View v) {
        if (username.getText().toString().equals("admin") && password.getText().toString().equals("admin")) {
            Toast.makeText(this, "LOGIN SUCCESSFUL", 0).show();
            Intent intent = new Intent(this, (Class<?>) User.class);
            startActivity(intent);
            return;
        }
        Toast.makeText(this, "LOGIN FAILED !!!", 0).show();
    }
}
```

⇨ Tài khoản: **admin** & mật khẩu: **admin**



- Đăng nhập thành công và lấy được flag



⇨ **Flag:** inseclab{w3lc0m3_t0_@ndr01d_Ch@ll3ng3!!!}

D. KỊCH BẢN 4 - Steganography

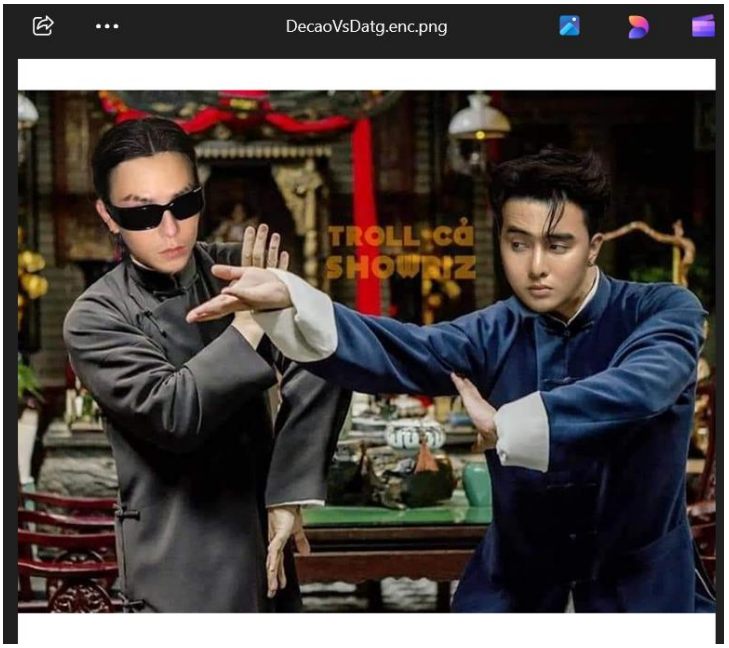
Đề bài:

DecaoVsDatg.enc.png - Hiện tại cộng đồng mạng chưa xác định được ai là võ sĩ mạnh nhất thời đại. Hãy tìm flag được giấu trong bức ảnh để xác định được ai là võ sĩ đấm đau nhất nhé :))

Hint: Một câu danh ngôn mà ai cũng biết…

Giải:

- Bước ảnh đề cung cấp.



- Đầu tiên, để xem có thông tin gì được giấu trong ảnh không thì nhóm sẽ trích xuất từng pixel có trong ảnh. Dưới đây là đoạn code nhóm sử dụng thư viện Pillow (PIL) để đọc ảnh và lấy các pixel.

Open getpixel.py ~/Phap_chung

```
1 from PIL import Image
2
3 # Mở ảnh
4 img = Image.open('DecaoVsDatg.enc.png')
5
6 # Lấy các pixel
7 pixels = img.load()
8
9 # Lấy kích thước ảnh
10 width, height = img.size
11
12 # In giá trị RGB của từng pixel
13 for y in range(height):
14     for x in range(width):
15         r, g, b = pixels[x, y]  # Lấy giá trị RGB của pixel
16         print(f"Pixel at ({x},{y}): R={r}, G={g}, B={b}")
17
```

- Kết quả nhóm lấy được các pixel của ảnh với các điểm RGB tương ứng.
  - Đầu bức ảnh

```
┌──(ngoc㉿ngoc)-[~/Phap_chung]
└─$ python3 getpixel.py
Pixel at (0,0): R=254, G=254, B=255
Pixel at (1,0): R=255, G=254, B=255
Pixel at (2,0): R=255, G=254, B=255
Pixel at (3,0): R=255, G=254, B=254
Pixel at (4,0): R=254, G=254, B=254
Pixel at (5,0): R=254, G=255, B=255
Pixel at (6,0): R=255, G=254, B=254
Pixel at (7,0): R=254, G=254, B=255
Pixel at (8,0): R=254, G=254, B=255
Pixel at (9,0): R=254, G=255, B=255
Pixel at (10,0): R=255, G=254, B=255
Pixel at (11,0): R=255, G=254, B=254
Pixel at (12,0): R=254, G=255, B=255
Pixel at (13,0): R=254, G=255, B=254
Pixel at (14,0): R=254, G=255, B=254
Pixel at (15,0): R=255, G=255, B=254
Pixel at (16,0): R=254, G=254, B=255
Pixel at (17,0): R=255, G=255, B=255
Pixel at (18,0): R=255, G=254, B=255
Pixel at (19,0): R=255, G=254, B=254
```

- Giữa bức ảnh

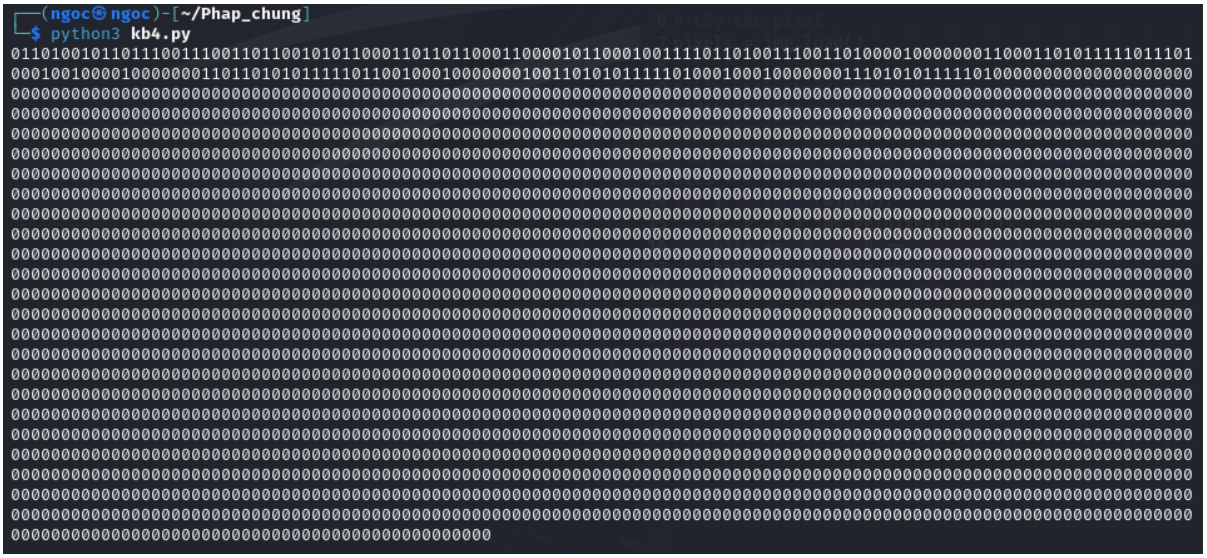
```
Pixel at (956,42): R=255, G=255, B=255
Pixel at (957,42): R=255, G=255, B=255
Pixel at (958,42): R=255, G=255, B=255
Pixel at (959,42): R=255, G=255, B=255
Pixel at (0,43): R=39, G=39, B=15
Pixel at (1,43): R=46, G=46, B=22
Pixel at (2,43): R=52, G=52, B=28
Pixel at (3,43): R=52, G=51, B=30
Pixel at (4,43): R=45, G=47, B=26
Pixel at (5,43): R=43, G=44, B=26
Pixel at (6,43): R=46, G=47, B=31
Pixel at (7,43): R=50, G=52, B=38
Pixel at (8,43): R=50, G=52, B=39
Pixel at (9,43): R=30, G=33, B=22
Pixel at (10,43): R=10, G=13, B=4
Pixel at (11,43): R=4, G=7, B=0
Pixel at (12,43): R=10, G=15, B=9
Pixel at (13,43): R=16, G=21, B=15
Pixel at (14,43): R=12, G=17, B=13
Pixel at (15,43): R=4, G=9, B=5
```

- Điều bất thường nhóm nhận ra ở đây là các giá trị RGB ở phần đầu bức ảnh đều là 254 và 255, trong khi các giá trị ở giữa và trở về sau có vẻ bình thường. Một bức ảnh tự nhiên thường không có mẫu lặp lại của 254 và 255 ở phần đầu, đây có thể là một dấu hiệu của kỹ thuật **LSB steganography** (giấu thông tin ẩn trong ảnh).
- Để trích xuất thông tin ẩn, nhóm sẽ lấy từng pixel từ ảnh và trích xuất bit cuối cùng (LSB) của mỗi thành phần R, G, B, sau đó ghép các bit này lại với nhau để tạo thành chuỗi nhị phân.
- Cụ thể, nhóm sẽ viết code để các giá trị RGB có giá trị bằng 255 sẽ biến thành 0 và còn lại sẽ biến thành 1.

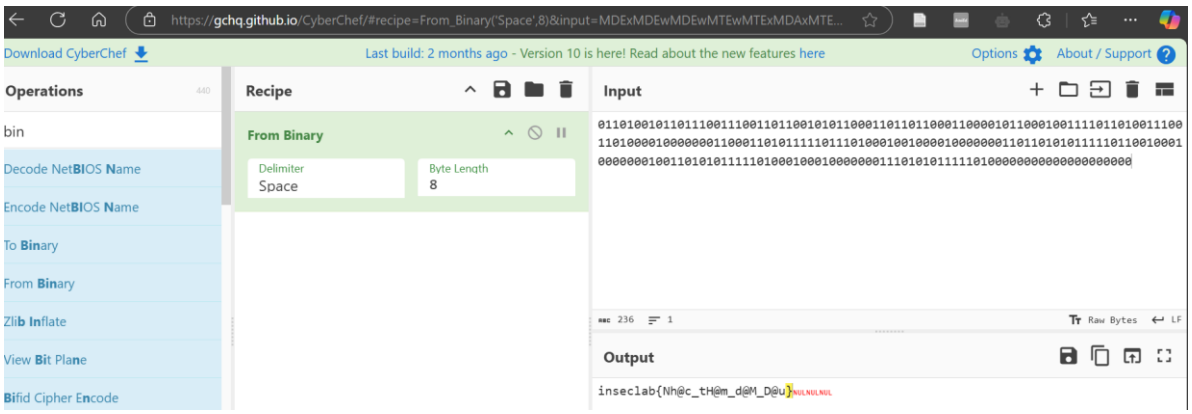
kb4.py

```python
from PIL import Image

# Mở ảnh
img = Image.open('DecaoVsDatg.enc.png')

# Lấy các pixel
pixels = img.load()

# Lấy kích thước ảnh
width, height = img.size

#lấy các pixel của ảnh và tiến hành chuyển đổi
bits = ''
for i in range(width):
    r, g, b = img.getpixel((i, 0))
    bits += '0' if b == 255 else '1'
    bits += '0' if g == 255 else '1'
    bits += '0' if r == 255 else '1'

print(bits)
```

- Kết quả nhóm nhận được chuỗi nhị phân như sau:



- Giờ chỉ cần đem chuỗi nhị phân này chuyển thành text.



⇨ **Flag:** inseclab{Nh@c_tH@m_d@M_D@u}