Journal scientific and applied research, vol. 25, 2023
International Journal

Original Contribution

ISSN 1314-6289 (Print)
ISSN 2815-4622 (Online)

INVESTIGATING THE NETWORK TRAFFIC USING THE COMMAND-LINE PACKETS SNIFFER TCPDUMP IN KALI LINUX

Petar Kr. Boyanov

COMMUNICATION AND COMPUTER TECHNOLOGIES, FACULTY OF TECHNICAL SCIENCES, KONSTANTIN PRESLAVSKY UNIVERSITY OF SHUMEN, SHUMEN 9712,115, UNIVERSITETSKA STR.,
E-MAIL: peshoaikido@gmail.com

ABSTRACT: *In this scientific article a comprehensive investigation of the network traffic using the command-line packets sniffer Tcpdump in kali Linux is made.*

KEY WORDS: *Analyzing, Connection, Command-line, Flag, Investigation, Kali Linux, Monitoring, Packet, Port, Scanning, Security, Sniffer, TCP, Traffic, UDP.*

1. Introduction

In the modern computer practice, various types of network programs are used to analyze and monitor the network packets that are sent between the individual hosts in a local and global computer networks. Most of the network packet analysis and monitoring programs are made so that the system administrator enters commands with options and parameters in the command line (terminal) [1,4,5,7,9,10,13,18,19,20,21]. Another part of them with a graphical user interface are made. In this scientific article a research about the capabilities of the Tcpdump network packet analyzer will be conducted. In its essence, it is a command-line network program through which the user can real-time inspect, sniff and monitor the incoming and outgoing network traffic, investigate the sending of each network packet, and the entire result of the achieved scan and trace to a text file can be saved. After that it can be in more detail analyzed and reviewed [8,10,16,17,18,21]. The special feature of this scientific article is that the different network states of the bits (flags) in the header of the TCP protocol will be considered. The flags that are used in the TCP protocol header are the following [1,2,3,4,5,6,11,12,14,15,20,21]:

- Flag SYN (SYNchronisation) with bit number in Tcpdump – 2.
- Flag ACK (ACKnowledgement) with bit number in Tcpdump – 16.
- Flag FIN (FINished) with bit number in Tcpdump – 1.

- Flag RST (Reset) with bit number in Tcpdump – 4.
- Flag PSH (Push) with bit number in Tcpdump – 8.
- Flag URG (Urgent Pointer) with bit number in Tcpdump – 32.

One or combinations of several flags define the function and task of each incoming and outgoing network packet. In this regard, once it is understood which flags in the package are enabled, the system administrator will be able to analyze and determine whether the package is normal (not dangerous) or is being used to perform scanning and exploitation of the resources of the respective operating system [8,9,13,14,15,20,21].

The performed network traffic analysis, sniffing and monitoring without the host's permission is considered as a crime and, if proven, is punishable to the full extent of the law of the respective country [2,3,8,11,12,13,15]. Everything illustrated and explained in this scientific article is only for research work and educational purposes and the author is not responsible in cases of abuse.

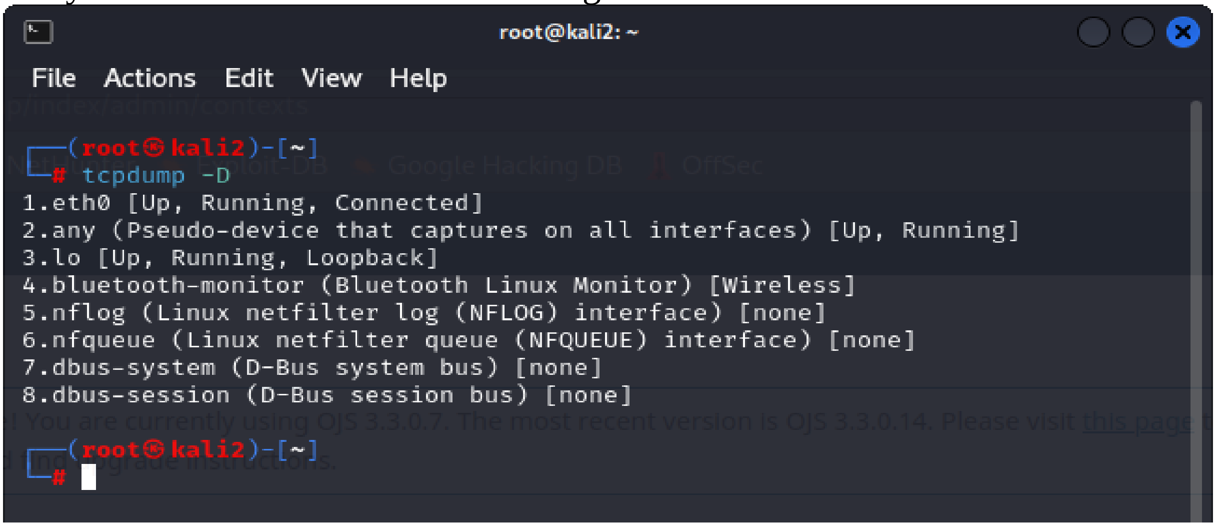
2. Experiment

In this article the scientific experiments and research works in a specialized computer network laboratory in the Faculty of Technical Sciences of the Konstantin Preslavsky University of Shumen are conducted.

The installed operating system for the two hosts used in Local Area Network is respectively Kali Linux - 6.0.0-kali6-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.0.12-1kali1 (2022-12-19) x86_64 GNU/Linux.

A local computer network of two hosts is built. The IPv4 address of the first host is 192.168.80.130 and IPv4 address of the second host is 192.168.80.132. The network mask is 24-bit.

After starting the terminal, the command "Tcpdump -D" is entered. The function of this command is to determine which network interface will be analyzed and sniff. This is shown on fig. 1.



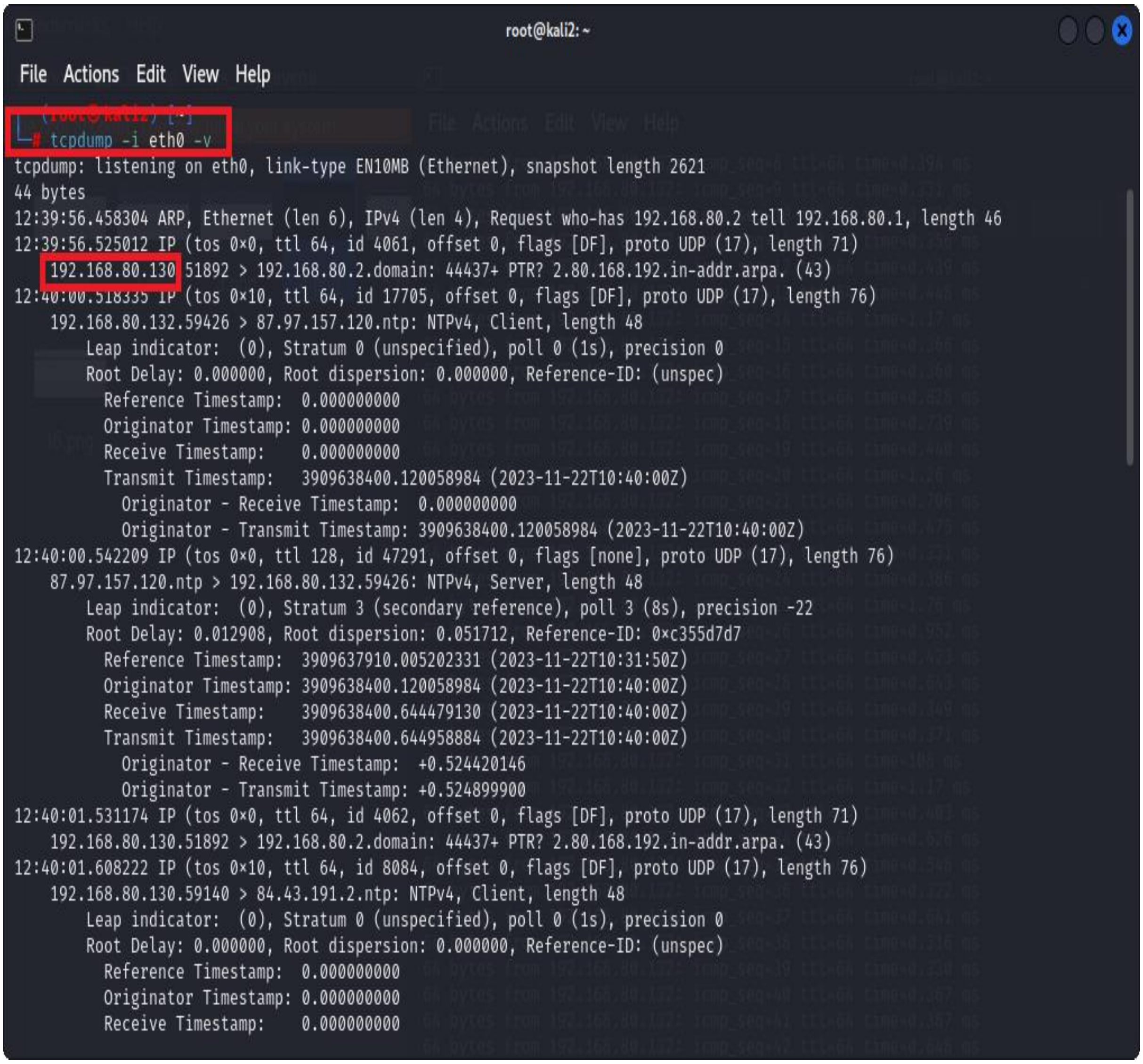
```
┌──(root㉿kali2)-[~]
└─# tcpdump -D
1.eth0 [Up, Running, Connected]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.bluetooth-monitor (Bluetooth Linux Monitor) [Wireless]
5.nflog (Linux netfilter log (NFLOG) interface) [none]
6.nfqueue (Linux netfilter queue (NFQUEUE) interface) [none]
7.dbus-system (D-Bus system bus) [none]
8.dbus-session (D-Bus session bus) [none]

┌──(root㉿kali2)-[~]
└─# 
```

Fig. 1. The result of the executed command "Tcpdump -D"

The network interfaces that can be monitored are respectively:

- eth0 - [up, running and connected network state].
- any (Pseudo-device that captures on all interfaces) [up and running state].
- lo (Localhost) [up and running network state].
- bluetooth-monitor [wireless network interface].
- nflog (Linux netfilter log network interface) [wireless network interface].
- nfqueue (Linux netfilter queue network interface) [none network state].
- dbus-system (D-Bus system bus) [none network state].
- dbus-session [D-Bus session bus) [none network state].

In this scientific article only network interface “eth0” for both hosts will be used. Fig. 2 shows that the host with IPv4 address 192.168.80.130 executed the command “tcpdump -i eth0 -v”. The option “-v” for showing detailed network information is used.

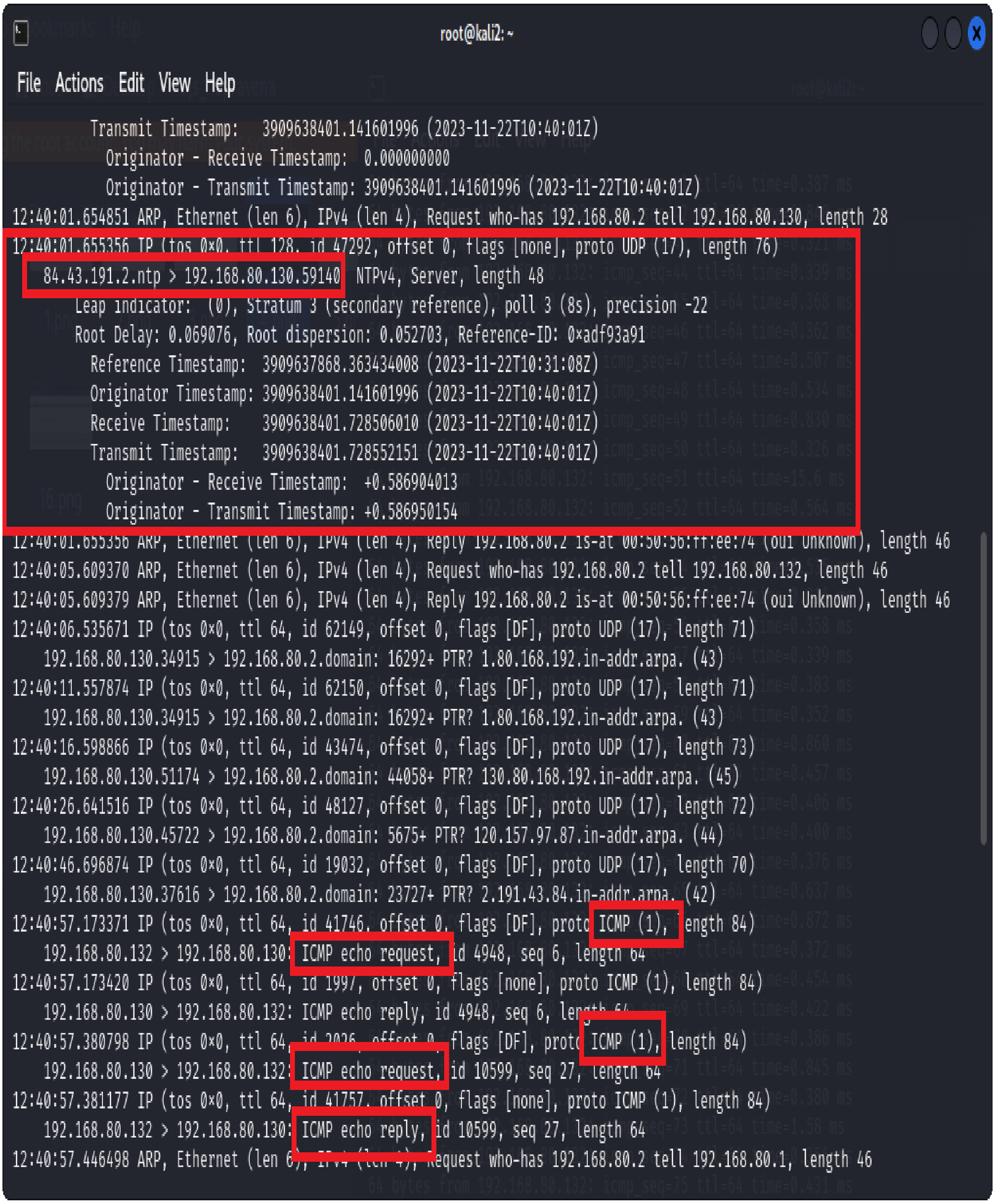


```
┌──(root@kali2)-[~]
└─# tcpdump -i eth0 -v
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 2621
44 bytes
12:39:56.458304 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.80.2 tell 192.168.80.1, length 46
12:39:56.525012 IP (tos 0x0, ttl 64, id 4061, offset 0, flags [DF], proto UDP (17), length 71)
    192.168.80.130 51892 > 192.168.80.2.domain: 44437+ PTR? 2.80.168.192.in-addr.arpa. (43)
12:40:00.518335 IP (tos 0x10, ttl 64, id 17705, offset 0, flags [DF], proto UDP (17), length 76)
    192.168.80.132.59426 > 87.97.157.120.ntp: NTPv4, Client, length 48
        Leap indicator:  (0), Stratum 0 (unspecified), poll 0 (1s), precision 0
        Root Delay: 0.000000, Root dispersion: 0.000000, Reference-ID: (unspec)
          Reference Timestamp:  0.000000000
          Originator Timestamp: 0.000000000
          Receive Timestamp:    0.000000000
          Transmit Timestamp:   3909638400.120058984 (2023-11-22T10:40:00Z)
            Originator - Receive Timestamp:  0.000000000
            Originator - Transmit Timestamp: 3909638400.120058984 (2023-11-22T10:40:00Z)
12:40:00.542209 IP (tos 0x0, ttl 128, id 47291, offset 0, flags [none], proto UDP (17), length 76)
    87.97.157.120.ntp > 192.168.80.132.59426: NTPv4, Server, length 48
        Leap indicator:  (0), Stratum 3 (secondary reference), poll 3 (8s), precision -22
        Root Delay: 0.012908, Root dispersion: 0.051712, Reference-ID: 0xc355d7d7
          Reference Timestamp:  3909637910.005202331 (2023-11-22T10:31:50Z)
          Originator Timestamp: 3909638400.120058984 (2023-11-22T10:40:00Z)
          Receive Timestamp:    3909638400.644479130 (2023-11-22T10:40:00Z)
          Transmit Timestamp:   3909638400.644958884 (2023-11-22T10:40:00Z)
            Originator - Receive Timestamp:  +0.524420146
            Originator - Transmit Timestamp: +0.524899900
12:40:01.531174 IP (tos 0x0, ttl 64, id 4062, offset 0, flags [DF], proto UDP (17), length 71)
    192.168.80.130.51892 > 192.168.80.2.domain: 44437+ PTR? 2.80.168.192.in-addr.arpa. (43)
12:40:01.608222 IP (tos 0x10, ttl 64, id 8084, offset 0, flags [DF], proto UDP (17), length 76)
    192.168.80.130.59140 > 84.43.191.2.ntp: NTPv4, Client, length 48
        Leap indicator:  (0), Stratum 0 (unspecified), poll 0 (1s), precision 0
        Root Delay: 0.000000, Root dispersion: 0.000000, Reference-ID: (unspec)
          Reference Timestamp:  0.000000000
          Originator Timestamp: 0.000000000
          Receive Timestamp:    0.000000000
```

Fig. 2. Selection for network interface “eth0”

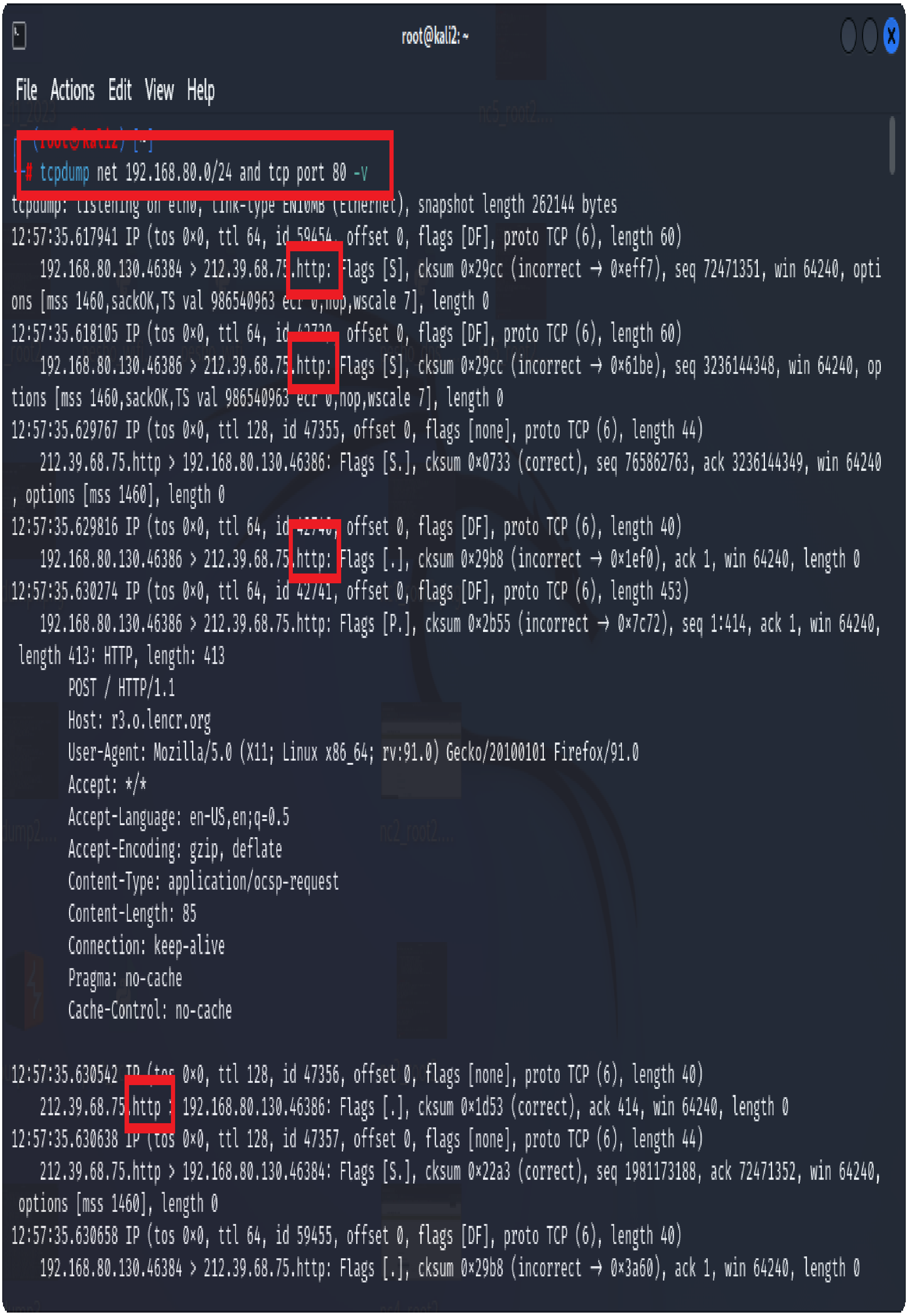
3. Results

Fig. 3 shows the host with address 192.168.80.130 receive the actual time and date from host with IP address 84.43.191.2 via Network Time Protocol (NTP). Additionally, fig. 3 presents that host with IP address 192.168.80.132 sends ICMP echo request to the other host with IP address 192.168.80.132. From his side it returns ICMP echo reply to host with IP address 192.168.80.130.



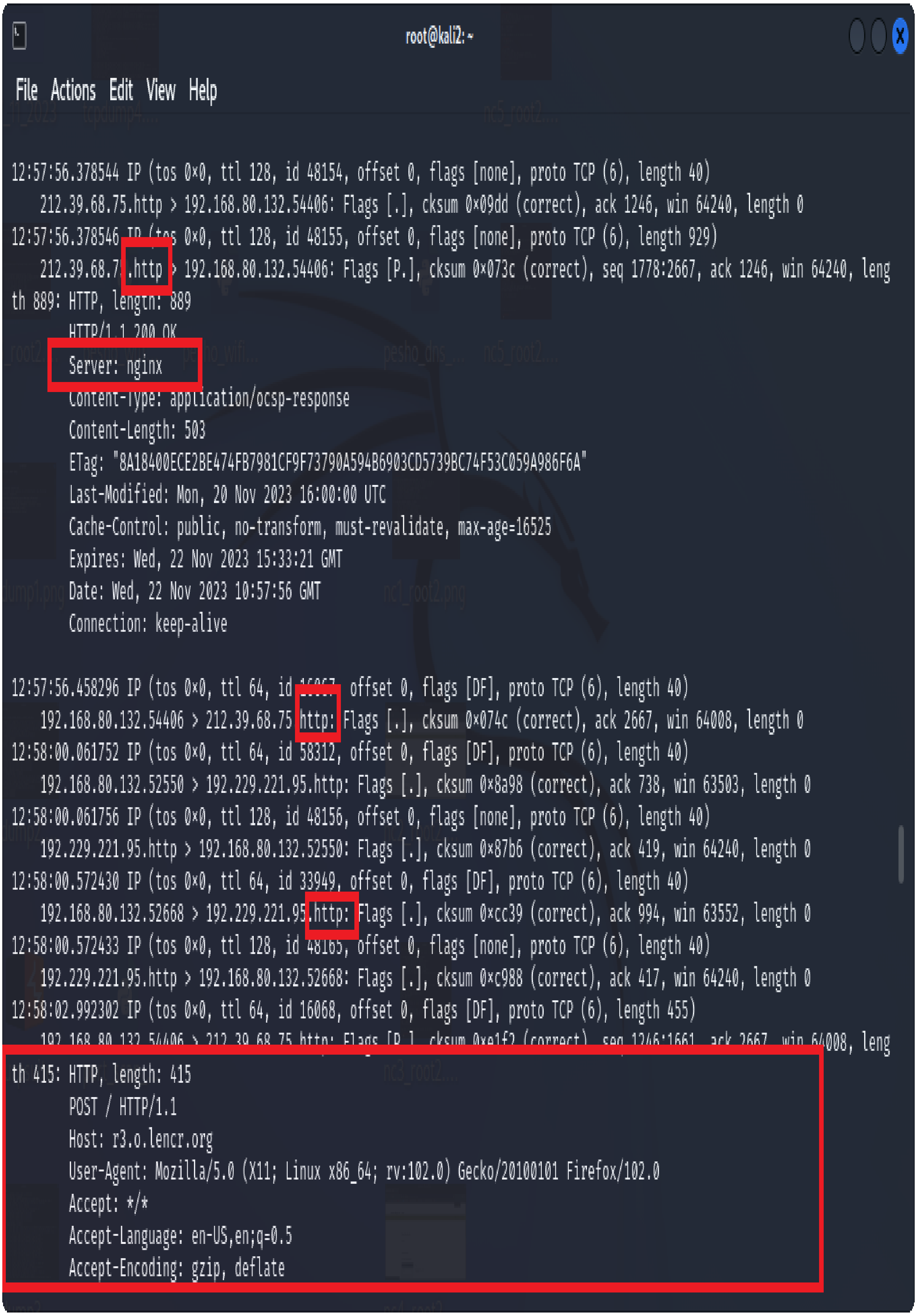
```
          Transmit Timestamp:   3909638401.141601996 (2023-11-22T10:40:01Z)
            Originator - Receive Timestamp:  0.000000000
            Originator - Transmit Timestamp: 3909638401.141601996 (2023-11-22T10:40:01Z)
12:40:01.654851 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.80.2 tell 192.168.80.130, length 28
12:40:01.655356 IP (tos 0x0, ttl 128, id 47292, offset 0, flags [none], proto UDP (17), length 76)
    84.43.191.2.ntp > 192.168.80.130.59140  NTPv4, Server, length 48
        Leap indicator:  (0), Stratum 3 (secondary reference), poll 3 (8s), precision -22
        Root Delay: 0.069076, Root dispersion: 0.052703, Reference-ID: 0xadf93a91
          Reference Timestamp:  3909637868.363434008 (2023-11-22T10:31:08Z)
          Originator Timestamp: 3909638401.141601996 (2023-11-22T10:40:01Z)
          Receive Timestamp:    3909638401.728506010 (2023-11-22T10:40:01Z)
          Transmit Timestamp:   3909638401.728552151 (2023-11-22T10:40:01Z)
            Originator - Receive Timestamp:  +0.586904013
            Originator - Transmit Timestamp: +0.586950154
12:40:01.655356 ARP, Ethernet (len 6), IPv4 (len 4), Reply 192.168.80.2 is-at 00:50:56:ff:ee:74 (oui Unknown), length 46
12:40:05.609370 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.80.2 tell 192.168.80.132, length 46
12:40:05.609379 ARP, Ethernet (len 6), IPv4 (len 4), Reply 192.168.80.2 is-at 00:50:56:ff:ee:74 (oui Unknown), length 46
12:40:06.535671 IP (tos 0x0, ttl 64, id 62149, offset 0, flags [DF], proto UDP (17), length 71)
    192.168.80.130.34915 > 192.168.80.2.domain: 16292+ PTR? 1.80.168.192.in-addr.arpa. (43)
12:40:11.557874 IP (tos 0x0, ttl 64, id 62150, offset 0, flags [DF], proto UDP (17), length 71)
    192.168.80.130.34915 > 192.168.80.2.domain: 16292+ PTR? 1.80.168.192.in-addr.arpa. (43)
12:40:16.598866 IP (tos 0x0, ttl 64, id 43474, offset 0, flags [DF], proto UDP (17), length 73)
    192.168.80.130.51174 > 192.168.80.2.domain: 44058+ PTR? 130.80.168.192.in-addr.arpa. (45)
12:40:26.641516 IP (tos 0x0, ttl 64, id 48127, offset 0, flags [DF], proto UDP (17), length 72)
    192.168.80.130.45722 > 192.168.80.2.domain: 5675+ PTR? 120.157.97.87.in-addr.arpa. (44)
12:40:46.696874 IP (tos 0x0, ttl 64, id 19032, offset 0, flags [DF], proto UDP (17), length 70)
    192.168.80.130.37616 > 192.168.80.2.domain: 23727+ PTR? 2.191.43.84.in-addr.arpa. (42)
12:40:57.173371 IP (tos 0x0, ttl 64, id 41746, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.80.132 > 192.168.80.130: ICMP echo request, id 4948, seq 6, length 64
12:40:57.173420 IP (tos 0x0, ttl 64, id 1997, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.80.130 > 192.168.80.132: ICMP echo reply, id 4948, seq 6, length 64
12:40:57.380798 IP (tos 0x0, ttl 64, id 2026, offset 0, flags [DF], proto ICMP (1), length 84)
    192.168.80.130 > 192.168.80.132: ICMP echo request, id 10599, seq 27, length 64
12:40:57.381177 IP (tos 0x0, ttl 64, id 41757, offset 0, flags [none], proto ICMP (1), length 84)
    192.168.80.132 > 192.168.80.130: ICMP echo reply, id 10599, seq 27, length 64
12:40:57.446498 ARP, Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.80.2 tell 192.168.80.1, length 46
```

Fig. 3. Results showing the functions of the NTP and ICMP protocols



```
┌──(root㉿kali2)-[~]
└─# tcpdump net 192.168.80.0/24 and tcp port 80 -v
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
12:57:35.617941 IP (tos 0x0, ttl 64, id 59454, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.80.130.46384 > 212.39.68.75.http: Flags [S], cksum 0x29cc (incorrect → 0xeff7), seq 72471351, win 64240, opti
ons [mss 1460,sackOK,TS val 986540963 ecr 0,nop,wscale 7], length 0
12:57:35.618105 IP (tos 0x0, ttl 64, id 42739, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.80.130.46386 > 212.39.68.75.http: Flags [S], cksum 0x29cc (incorrect → 0x61be), seq 3236144348, win 64240, op
tions [mss 1460,sackOK,TS val 986540963 ecr 0,nop,wscale 7], length 0
12:57:35.629767 IP (tos 0x0, ttl 128, id 47355, offset 0, flags [none], proto TCP (6), length 44)
    212.39.68.75.http > 192.168.80.130.46386: Flags [S.], cksum 0x0733 (correct), seq 765862763, ack 3236144349, win 64240
, options [mss 1460], length 0
12:57:35.629816 IP (tos 0x0, ttl 64, id 42740, offset 0, flags [DF], proto TCP (6), length 40)
    192.168.80.130.46386 > 212.39.68.75.http: Flags [.], cksum 0x29b8 (incorrect → 0x1ef0), ack 1, win 64240, length 0
12:57:35.630274 IP (tos 0x0, ttl 64, id 42741, offset 0, flags [DF], proto TCP (6), length 453)
    192.168.80.130.46386 > 212.39.68.75.http: Flags [P.], cksum 0x2b55 (incorrect → 0x7c72), seq 1:414, ack 1, win 64240,
 length 413: HTTP, length: 413
        POST / HTTP/1.1
        Host: r3.o.lencr.org
        User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
        Accept: */*
        Accept-Language: en-US,en;q=0.5
        Accept-Encoding: gzip, deflate
        Content-Type: application/ocsp-request
        Content-Length: 85
        Connection: keep-alive
        Pragma: no-cache
        Cache-Control: no-cache

12:57:35.630542 IP (tos 0x0, ttl 128, id 47356, offset 0, flags [none], proto TCP (6), length 40)
    212.39.68.75.http > 192.168.80.130.46386: Flags [.], cksum 0x1d53 (correct), ack 414, win 64240, length 0
12:57:35.630638 IP (tos 0x0, ttl 128, id 47357, offset 0, flags [none], proto TCP (6), length 44)
    212.39.68.75.http > 192.168.80.130.46384: Flags [S.], cksum 0x22a3 (correct), seq 1981173188, ack 72471352, win 64240,
 options [mss 1460], length 0
12:57:35.630658 IP (tos 0x0, ttl 64, id 59455, offset 0, flags [DF], proto TCP (6), length 40)
    192.168.80.130.46384 > 212.39.68.75.http: Flags [.], cksum 0x29b8 (incorrect → 0x3a60), ack 1, win 64240, length 0
```

Fig. 4. Results of the executed command “tcpdump net 192.168.80.0/24 and tcp port 80 -v“



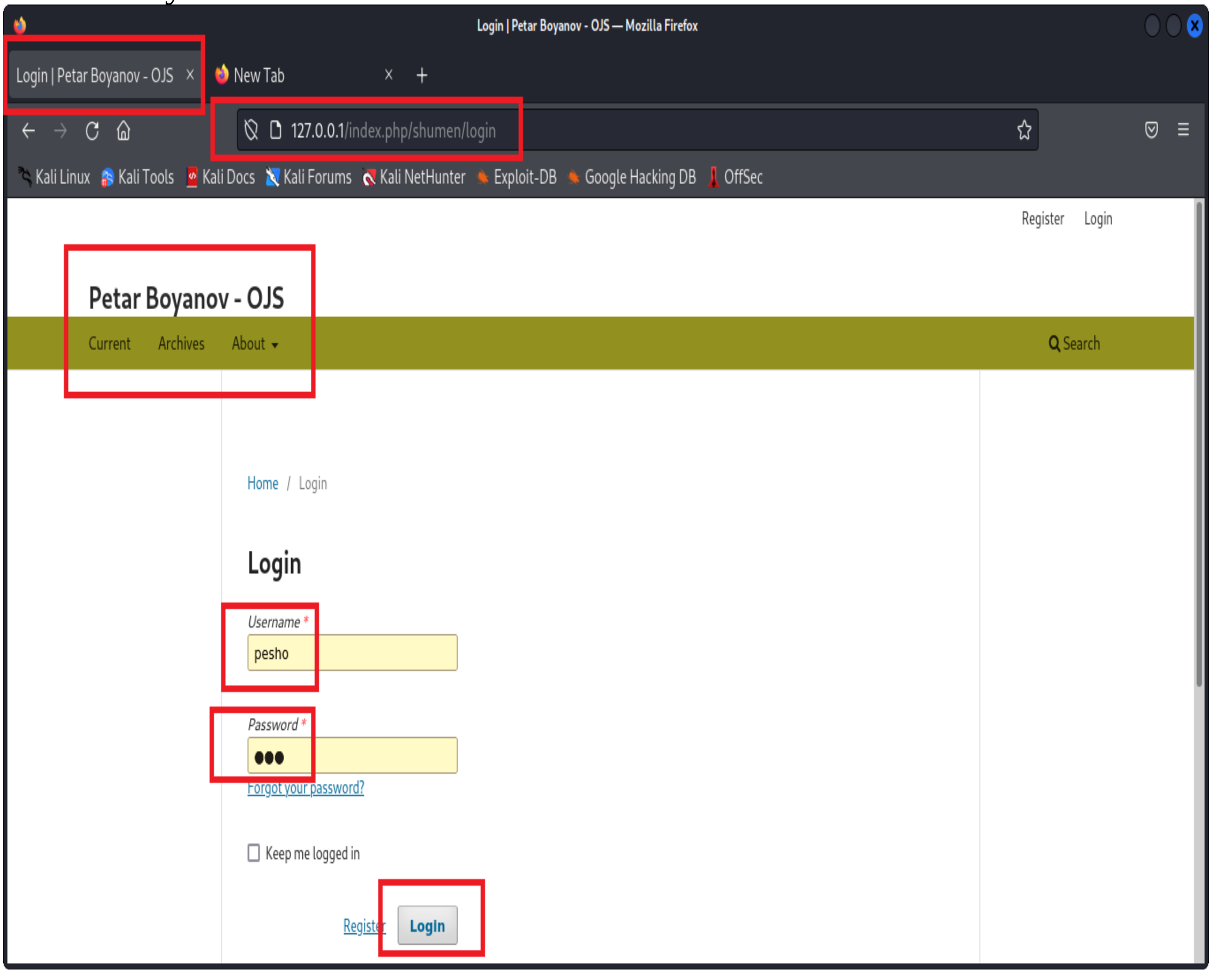
```
12:57:56.378544 IP (tos 0×0, ttl 128, id 48154, offset 0, flags [none], proto TCP (6), length 40)
    212.39.68.75.http > 192.168.80.132.54406: Flags [.], cksum 0×09dd (correct), ack 1246, win 64240, length 0
12:57:56.378546 IP (tos 0×0, ttl 128, id 48155, offset 0, flags [none], proto TCP (6), length 929)
    212.39.68.75.http > 192.168.80.132.54406: Flags [P.], cksum 0×073c (correct), seq 1778:2667, ack 1246, win 64240, leng
th 889: HTTP, length: 889
        HTTP/1.1 200 OK
        Server: nginx
        Content-Type: application/ocsp-response
        Content-Length: 503
        ETag: "8A18400ECE2BE474FB7981CF9F73790A594B6903CD5739BC74F53C059A986F6A"
        Last-Modified: Mon, 20 Nov 2023 16:00:00 UTC
        Cache-Control: public, no-transform, must-revalidate, max-age=16525
        Expires: Wed, 22 Nov 2023 15:33:21 GMT
        Date: Wed, 22 Nov 2023 10:57:56 GMT
        Connection: keep-alive

12:57:56.458296 IP (tos 0×0, ttl 64, id 16067, offset 0, flags [DF], proto TCP (6), length 40)
    192.168.80.132.54406 > 212.39.68.75.http: Flags [.], cksum 0×074c (correct), ack 2667, win 64008, length 0
12:58:00.061752 IP (tos 0×0, ttl 64, id 58312, offset 0, flags [DF], proto TCP (6), length 40)
    192.168.80.132.52550 > 192.229.221.95.http: Flags [.], cksum 0×8a98 (correct), ack 738, win 63503, length 0
12:58:00.061756 IP (tos 0×0, ttl 128, id 48156, offset 0, flags [none], proto TCP (6), length 40)
    192.229.221.95.http > 192.168.80.132.52550: Flags [.], cksum 0×87b6 (correct), ack 419, win 64240, length 0
12:58:00.572430 IP (tos 0×0, ttl 64, id 33949, offset 0, flags [DF], proto TCP (6), length 40)
    192.168.80.132.52668 > 192.229.221.95.http: Flags [.], cksum 0×cc39 (correct), ack 994, win 63552, length 0
12:58:00.572433 IP (tos 0×0, ttl 128, id 48165, offset 0, flags [none], proto TCP (6), length 40)
    192.229.221.95.http > 192.168.80.132.52668: Flags [.], cksum 0×c988 (correct), ack 417, win 64240, length 0
12:58:02.992302 IP (tos 0×0, ttl 64, id 16068, offset 0, flags [DF], proto TCP (6), length 455)
    192.168.80.132.54406 > 212.39.68.75.http: Flags [P.], cksum 0×e1f2 (correct), seq 1246:1661, ack 2667, win 64008, leng
th 415: HTTP, length: 415
        POST / HTTP/1.1
        Host: r3.o.lencr.org
        User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
        Accept: */*
        Accept-Language: en-US,en;q=0.5
        Accept-Encoding: gzip, deflate
```

Fig. 5. Results of the executed command “tcpdump net 192.168.80.0/24 and tcp port 80 -v“

Fig. 4 and 5 show the achieved results after execution the command “tcpdump net 192.168.80.0/24 and tcp port 80 -v“. After executing this command, only network packets that use the protocol HTTP are displayed and filtered [3,4,12,13,15,16,19]. If the system administrator wants to receive only network information about the HTTPS protocol, then instead of port 80, it must be written port 443.

It was additionally Open Journal System (OJS) version 3.3.0.7 on the host with IP address 192.168.80.130 installed and configured. The purpose is to show that there is an apache server and database platform installed. The direct URL link to the platform is “127.0.0.1/index.php/shumen/login”. This is shown on fig.6. The IPv4 address 127.0.0.1 is called “localhost” and in this regard the following command “tcpdump -i lo” is executed (shown on fig.7). This command only scans the network traffic that passes through the interface loopback (localhost). On this localhost interface FTP, MAIL, HTTP, DNS servers can be installed and configured. Another important function for this address is to verify that the device's network card is working properly.

Fig. 7 shows the sniffed network traffic via loopback interface. The flags “[S]”, [S.]” and “[.]” indicates that the three-way handshake connection is successfully established.


Login | Petar Boyanov - OJS — Mozilla Firefox

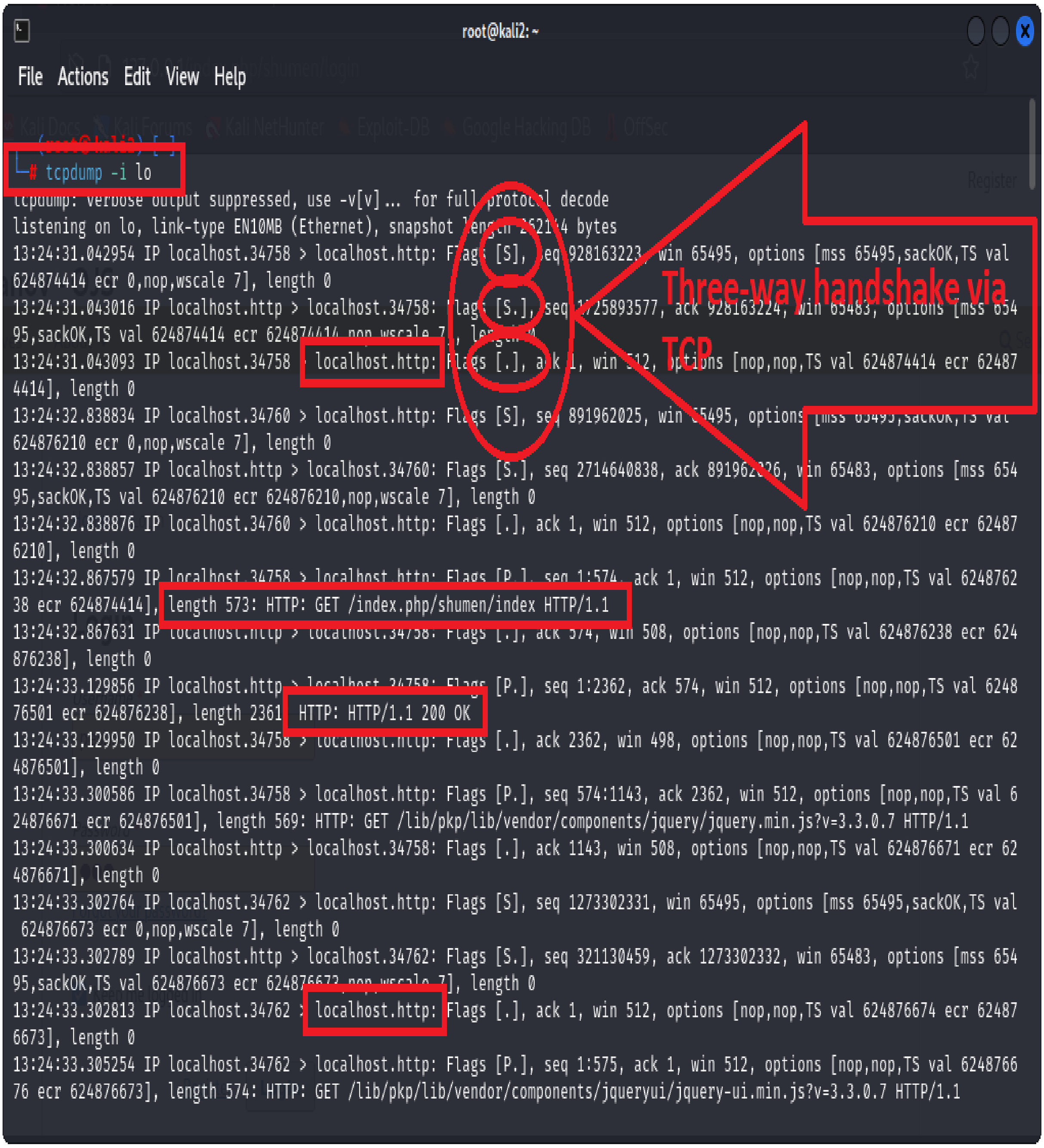
127.0.0.1/index.php/shumen/login

Username * pesho

Password * ●●●

□ Keep me logged in

Fig. 6. The login form page for the custom OJS platform

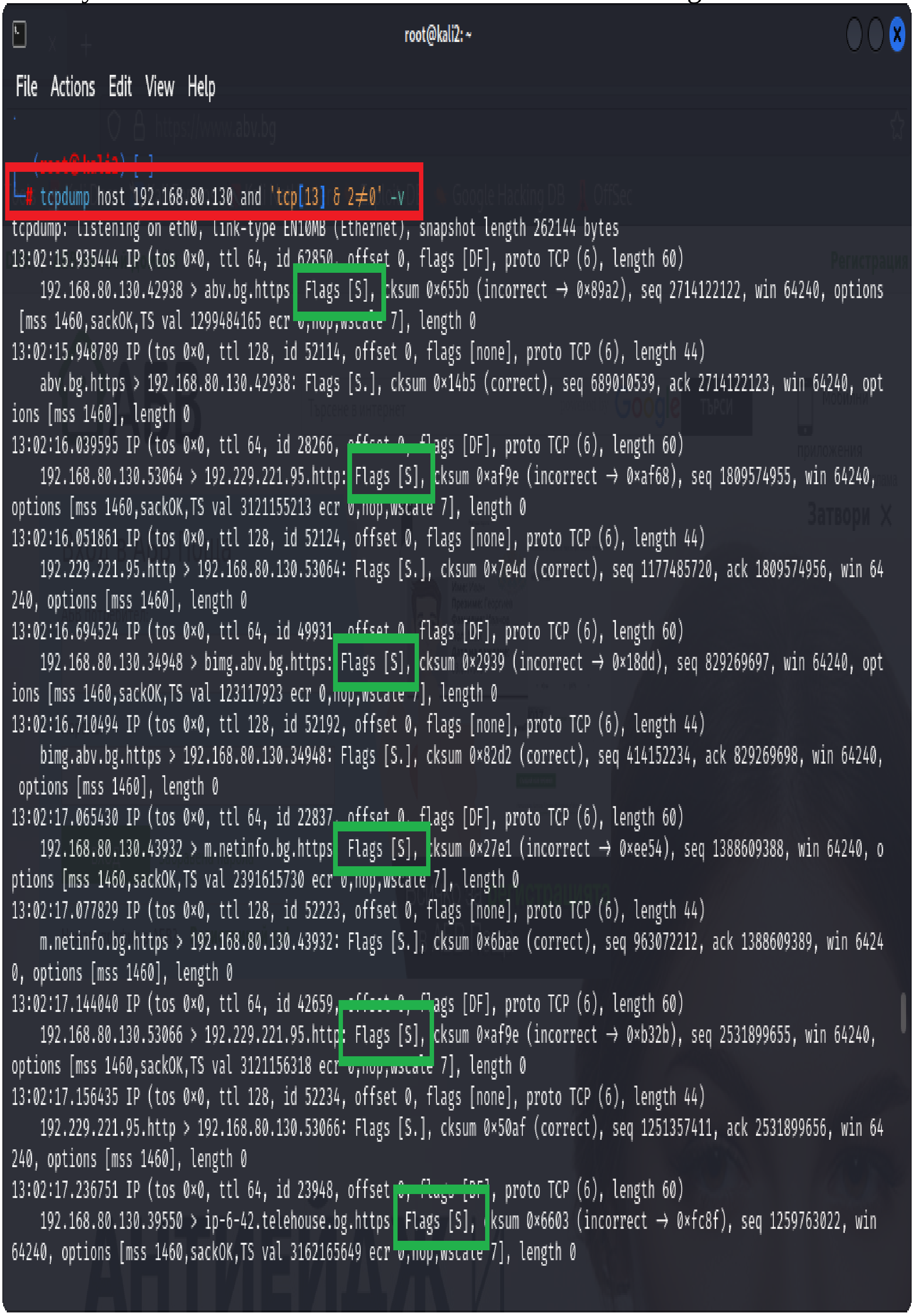


```
└─# tcpdump -i lo
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on lo, link-type EN10MB (Ethernet), snapshot length 262144 bytes
13:24:31.042954 IP localhost.34758 > localhost.http: Flags [S], seq 928163223, win 65495, options [mss 65495,sackOK,TS val
624874414 ecr 0,nop,wscale 7], length 0
13:24:31.043016 IP localhost.http > localhost.34758: Flags [S.], seq 1725893577, ack 928163224, win 65483, options [mss 654
95,sackOK,TS val 624874414 ecr 624874414,nop,wscale 7], length 0
13:24:31.043093 IP localhost.34758 > localhost.http: Flags [.], ack 1, win 512, options [nop,nop,TS val 624874414 ecr 62487
4414], length 0
13:24:32.838834 IP localhost.34760 > localhost.http: Flags [S], seq 891962025, win 65495, options [mss 65495,sackOK,TS val
624876210 ecr 0,nop,wscale 7], length 0
13:24:32.838857 IP localhost.http > localhost.34760: Flags [S.], seq 2714640838, ack 891962026, win 65483, options [mss 654
95,sackOK,TS val 624876210 ecr 624876210,nop,wscale 7], length 0
13:24:32.838876 IP localhost.34760 > localhost.http: Flags [.], ack 1, win 512, options [nop,nop,TS val 624876210 ecr 62487
6210], length 0
13:24:32.867579 IP localhost.34758 > localhost.http: Flags [P.], seq 1:574, ack 1, win 512, options [nop,nop,TS val 6248762
38 ecr 624874414], length 573: HTTP: GET /index.php/shumen/index HTTP/1.1
13:24:32.867631 IP localhost.http > localhost.34758: Flags [.], ack 574, win 508, options [nop,nop,TS val 624876238 ecr 624
876238], length 0
13:24:33.129856 IP localhost.http > localhost.34758: Flags [P.], seq 1:2362, ack 574, win 512, options [nop,nop,TS val 6248
76501 ecr 624876238], length 2361: HTTP: HTTP/1.1 200 OK
13:24:33.129950 IP localhost.34758 > localhost.http: Flags [.], ack 2362, win 498, options [nop,nop,TS val 624876501 ecr 62
4876501], length 0
13:24:33.300586 IP localhost.34758 > localhost.http: Flags [P.], seq 574:1143, ack 2362, win 512, options [nop,nop,TS val 6
24876671 ecr 624876501], length 569: HTTP: GET /lib/pkp/lib/vendor/components/jquery/jquery.min.js?v=3.3.0.7 HTTP/1.1
13:24:33.300634 IP localhost.http > localhost.34758: Flags [.], ack 1143, win 508, options [nop,nop,TS val 624876671 ecr 62
4876671], length 0
13:24:33.302764 IP localhost.34762 > localhost.http: Flags [S], seq 1273302331, win 65495, options [mss 65495,sackOK,TS val
 624876673 ecr 0,nop,wscale 7], length 0
13:24:33.302789 IP localhost.http > localhost.34762: Flags [S.], seq 321130459, ack 1273302332, win 65483, options [mss 654
95,sackOK,TS val 624876673 ecr 624876673,nop,wscale 7], length 0
13:24:33.302813 IP localhost.34762 > localhost.http: Flags [.], ack 1, win 512, options [nop,nop,TS val 624876674 ecr 62487
6673], length 0
13:24:33.305254 IP localhost.34762 > localhost.http: Flags [P.], seq 1:575, ack 1, win 512, options [nop,nop,TS val 6248766
76 ecr 624876673], length 574: HTTP: GET /lib/pkp/lib/vendor/components/jqueryui/jquery-ui.min.js?v=3.3.0.7 HTTP/1.1
```

Fig. 7. Successfully established three-way handshake

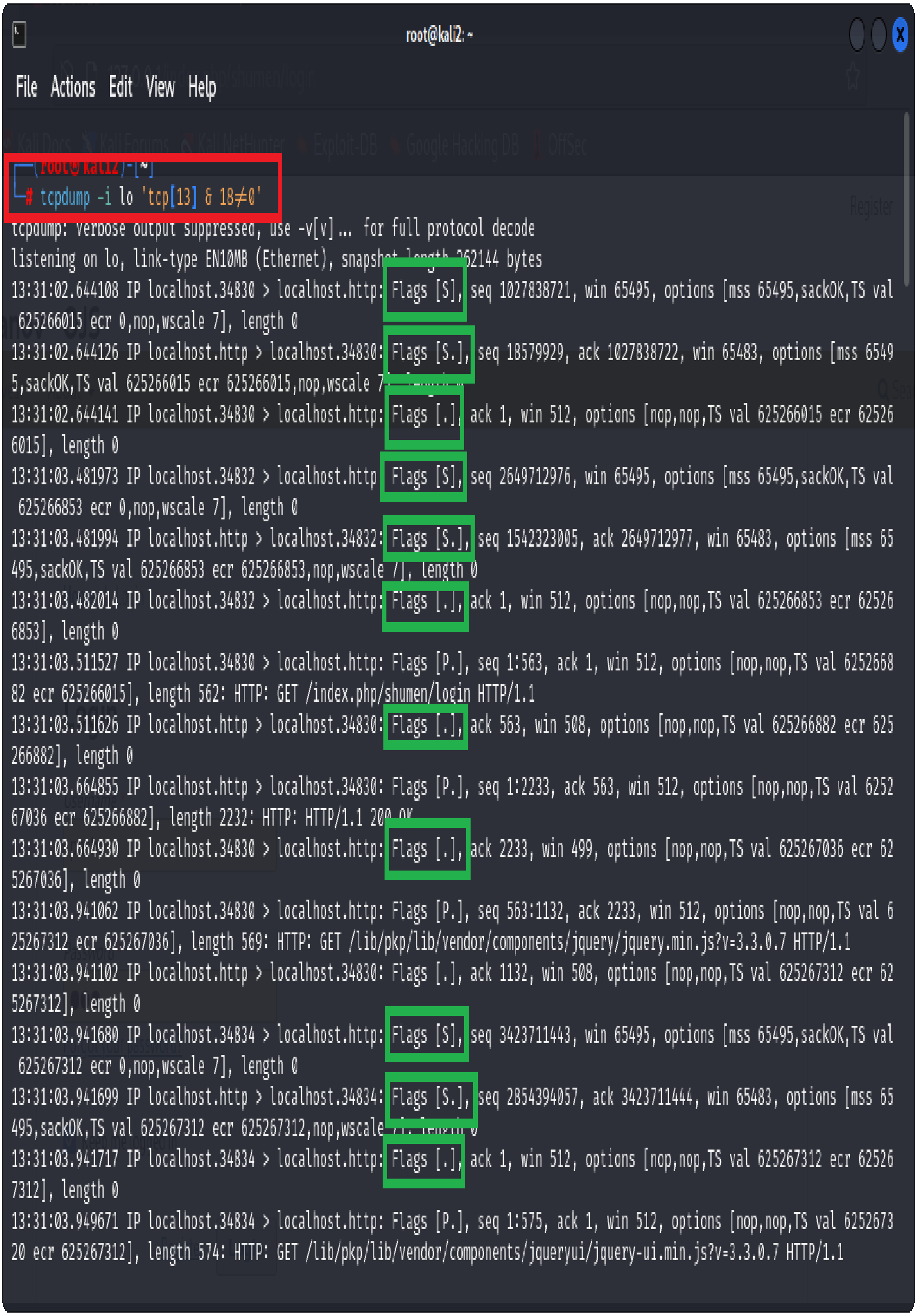
The command “tcpdump host 192.168.80.130 and 'tcp[13] & 2 !=0' -v” filters and sniffs only TCP connections with activated SYN bit (flag) in verbose mode. This is shown on fig. 8.

The command “tcpdump -i lo 'tcp[13] & 18 !=0'” filters and sniffs only TCP connections with activated both SYN-ACK bits (flags) on the loopback interface. In this case localhost:34830 (192.168.80.130) sends SYN flag to host localhost:http (OJS platform). After that the localhost:http sends SYN-ACK flags to the host localhost:34830 [3,4,5,6,15,16,17]. Finally, localhost:34830 sends ACK flag to the host localhost:http in order to finish the establishment of three-way handshake between the hosts. This is shown on fig. 9.



```
┌──(root㉿kali)-[~]
└─# tcpdump host 192.168.80.130 and 'tcp[13] & 2≠0' -v
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
13:02:15.935444 IP (tos 0x0, ttl 64, id 62850, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.80.130.42938 > abv.bg.https: Flags [S], cksum 0x655b (incorrect → 0x89a2), seq 2714122122, win 64240, options
 [mss 1460,sackOK,TS val 1299484165 ecr 0,nop,wscale 7], length 0
13:02:15.948789 IP (tos 0x0, ttl 128, id 52114, offset 0, flags [none], proto TCP (6), length 44)
    abv.bg.https > 192.168.80.130.42938: Flags [S.], cksum 0x14b5 (correct), seq 689010539, ack 2714122123, win 64240, opt
ions [mss 1460], length 0
13:02:16.039595 IP (tos 0x0, ttl 64, id 28266, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.80.130.53064 > 192.229.221.95.http: Flags [S], cksum 0xaf9e (incorrect → 0xaf68), seq 1809574955, win 64240,
options [mss 1460,sackOK,TS val 3121155213 ecr 0,nop,wscale 7], length 0
13:02:16.051861 IP (tos 0x0, ttl 128, id 52124, offset 0, flags [none], proto TCP (6), length 44)
    192.229.221.95.http > 192.168.80.130.53064: Flags [S.], cksum 0x7e4d (correct), seq 1177485720, ack 1809574956, win 64
240, options [mss 1460], length 0
13:02:16.694524 IP (tos 0x0, ttl 64, id 49931, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.80.130.34948 > bimg.abv.bg.https: Flags [S], cksum 0x2939 (incorrect → 0x18dd), seq 829269697, win 64240, opt
ions [mss 1460,sackOK,TS val 123117923 ecr 0,nop,wscale 7], length 0
13:02:16.710494 IP (tos 0x0, ttl 128, id 52192, offset 0, flags [none], proto TCP (6), length 44)
    bimg.abv.bg.https > 192.168.80.130.34948: Flags [S.], cksum 0x82d2 (correct), seq 414152234, ack 829269698, win 64240,
 options [mss 1460], length 0
13:02:17.065430 IP (tos 0x0, ttl 64, id 22837, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.80.130.43932 > m.netinfo.bg.https: Flags [S], cksum 0x27e1 (incorrect → 0xee54), seq 1388609388, win 64240, o
ptions [mss 1460,sackOK,TS val 2391615730 ecr 0,nop,wscale 7], length 0
13:02:17.077829 IP (tos 0x0, ttl 128, id 52223, offset 0, flags [none], proto TCP (6), length 44)
    m.netinfo.bg.https > 192.168.80.130.43932: Flags [S.], cksum 0x6bae (correct), seq 963072212, ack 1388609389, win 6424
0, options [mss 1460], length 0
13:02:17.144040 IP (tos 0x0, ttl 64, id 42659, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.80.130.53066 > 192.229.221.95.http: Flags [S], cksum 0xaf9e (incorrect → 0xb32b), seq 2531899655, win 64240,
options [mss 1460,sackOK,TS val 3121156318 ecr 0,nop,wscale 7], length 0
13:02:17.156435 IP (tos 0x0, ttl 128, id 52234, offset 0, flags [none], proto TCP (6), length 44)
    192.229.221.95.http > 192.168.80.130.53066: Flags [S.], cksum 0x50af (correct), seq 1251357411, ack 2531899656, win 64
240, options [mss 1460], length 0
13:02:17.236751 IP (tos 0x0, ttl 64, id 23948, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.80.130.39550 > ip-6-42.telehouse.bg.https: Flags [S], cksum 0x6603 (incorrect → 0xfc8f), seq 1259763022, win
64240, options [mss 1460,sackOK,TS val 3162165649 ecr 0,nop,wscale 7], length 0
```

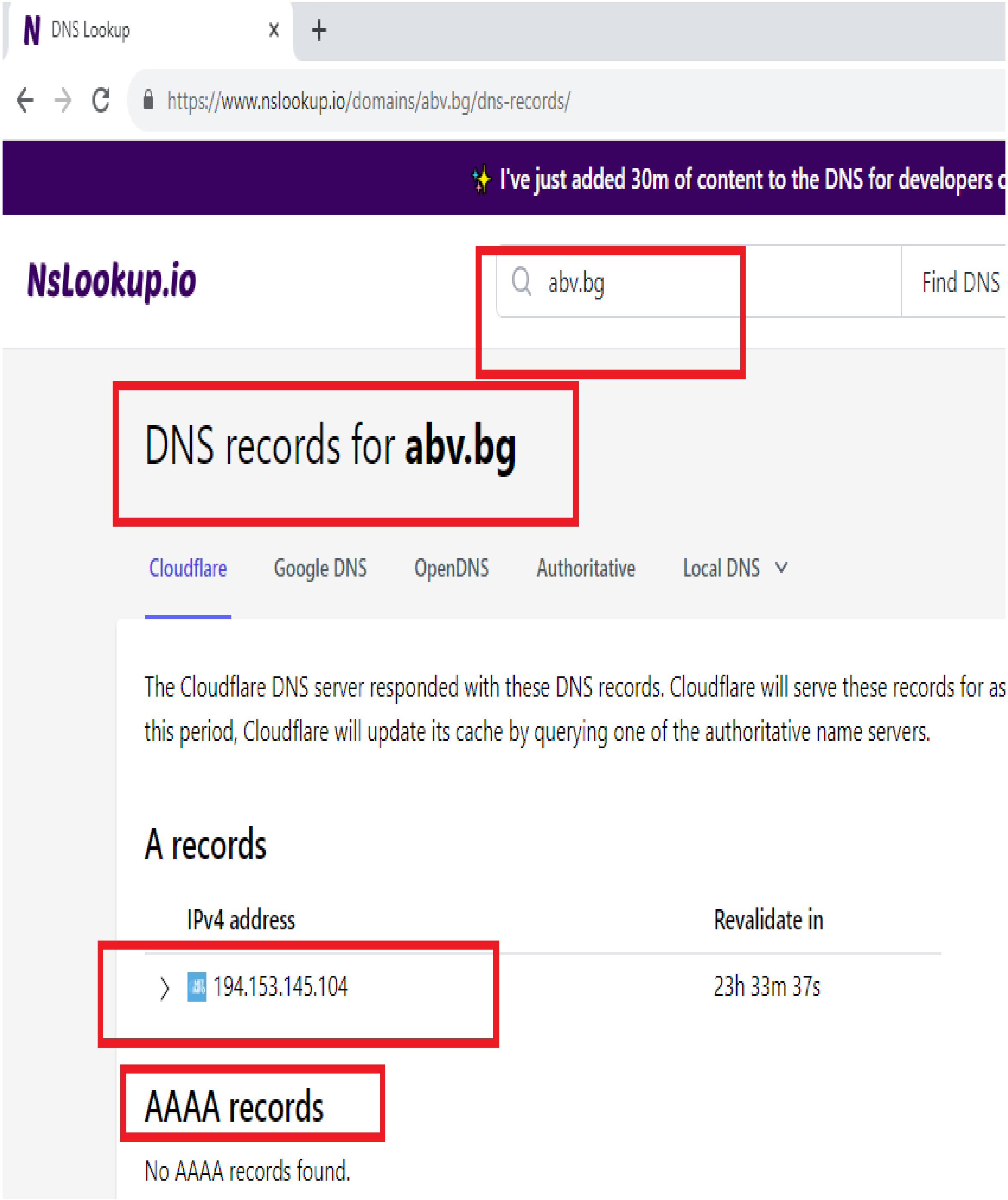
Fig. 8. Results of the executed command “tcpdump host 192.168.80.130 and 'tcp[13] & 2 !=0' -v”



```
┌──(root㉿kali2)-[~]
└─# tcpdump -i lo 'tcp[13] & 18≠0'
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on lo, link-type EN10MB (Ethernet), snapshot length 262144 bytes
13:31:02.644108 IP localhost.34830 > localhost.http: Flags [S], seq 1027838721, win 65495, options [mss 65495,sackOK,TS val
 625266015 ecr 0,nop,wscale 7], length 0
13:31:02.644126 IP localhost.http > localhost.34830: Flags [S.], seq 18579929, ack 1027838722, win 65483, options [mss 6549
5,sackOK,TS val 625266015 ecr 625266015,nop,wscale 7], length 0
13:31:02.644141 IP localhost.34830 > localhost.http: Flags [.], ack 1, win 512, options [nop,nop,TS val 625266015 ecr 62526
6015], length 0
13:31:03.481973 IP localhost.34832 > localhost.http: Flags [S], seq 2649712976, win 65495, options [mss 65495,sackOK,TS val
 625266853 ecr 0,nop,wscale 7], length 0
13:31:03.481994 IP localhost.http > localhost.34832: Flags [S.], seq 1542323005, ack 2649712977, win 65483, options [mss 65
495,sackOK,TS val 625266853 ecr 625266853,nop,wscale 7], length 0
13:31:03.482014 IP localhost.34832 > localhost.http: Flags [.], ack 1, win 512, options [nop,nop,TS val 625266853 ecr 62526
6853], length 0
13:31:03.511527 IP localhost.34830 > localhost.http: Flags [P.], seq 1:563, ack 1, win 512, options [nop,nop,TS val 6252668
82 ecr 625266015], length 562: HTTP: GET /index.php/shumen/login HTTP/1.1
13:31:03.511626 IP localhost.http > localhost.34830: Flags [.], ack 563, win 508, options [nop,nop,TS val 625266882 ecr 625
266882], length 0
13:31:03.664855 IP localhost.http > localhost.34830: Flags [P.], seq 1:2233, ack 563, win 512, options [nop,nop,TS val 6252
67036 ecr 625266882], length 2232: HTTP: HTTP/1.1 200 OK
13:31:03.664930 IP localhost.34830 > localhost.http: Flags [.], ack 2233, win 499, options [nop,nop,TS val 625267036 ecr 62
5267036], length 0
13:31:03.941062 IP localhost.34830 > localhost.http: Flags [P.], seq 563:1132, ack 2233, win 512, options [nop,nop,TS val 6
25267312 ecr 625267036], length 569: HTTP: GET /lib/pkp/lib/vendor/components/jquery/jquery.min.js?v=3.3.0.7 HTTP/1.1
13:31:03.941102 IP localhost.http > localhost.34830: Flags [.], ack 1132, win 508, options [nop,nop,TS val 625267312 ecr 62
5267312], length 0
13:31:03.941680 IP localhost.34834 > localhost.http: Flags [S], seq 3423711443, win 65495, options [mss 65495,sackOK,TS val
 625267312 ecr 0,nop,wscale 7], length 0
13:31:03.941699 IP localhost.http > localhost.34834: Flags [S.], seq 2854394057, ack 3423711444, win 65483, options [mss 65
495,sackOK,TS val 625267312 ecr 625267312,nop,wscale 7], length 0
13:31:03.941717 IP localhost.34834 > localhost.http: Flags [.], ack 1, win 512, options [nop,nop,TS val 625267312 ecr 62526
7312], length 0
13:31:03.949671 IP localhost.34834 > localhost.http: Flags [P.], seq 1:575, ack 1, win 512, options [nop,nop,TS val 6252673
20 ecr 625267312], length 574: HTTP: GET /lib/pkp/lib/vendor/components/jqueryui/jquery-ui.min.js?v=3.3.0.7 HTTP/1.1
```

Fig. 9. Results of the executed command “tcpdump -i lo 'tcp[13] & 18 !=0'”

Fig. 10 shows the IPv4 address (194.153.145.104) of the DNS records for the Bulgarian mail server „abv.bg“. This information thanks to the website NsLookup.io is revealed [1,2,3,4,15,17,20]. The command “tcpdump portrange 1-79” filters the network connections from port 1 to 79. On fig. 11 the IPv4 address of the mail server “abv.bg” is fetched and additionally all other interrelated network connections with the domain “abv.bg” are filtered and illustrated.



DNS Lookup
https://www.nslookup.io/domains/abv.bg/dns-records/
I've just added 30m of content to the DNS for developers c
NsLookup.io
abv.bg
Find DNS
Cloudflare
Google DNS
OpenDNS
Authoritative
Local DNS

DNS records for **abv.bg**

The Cloudflare DNS server responded with these DNS records. Cloudflare will serve these records for as this period, Cloudflare will update its cache by querying one of the authoritative name servers.

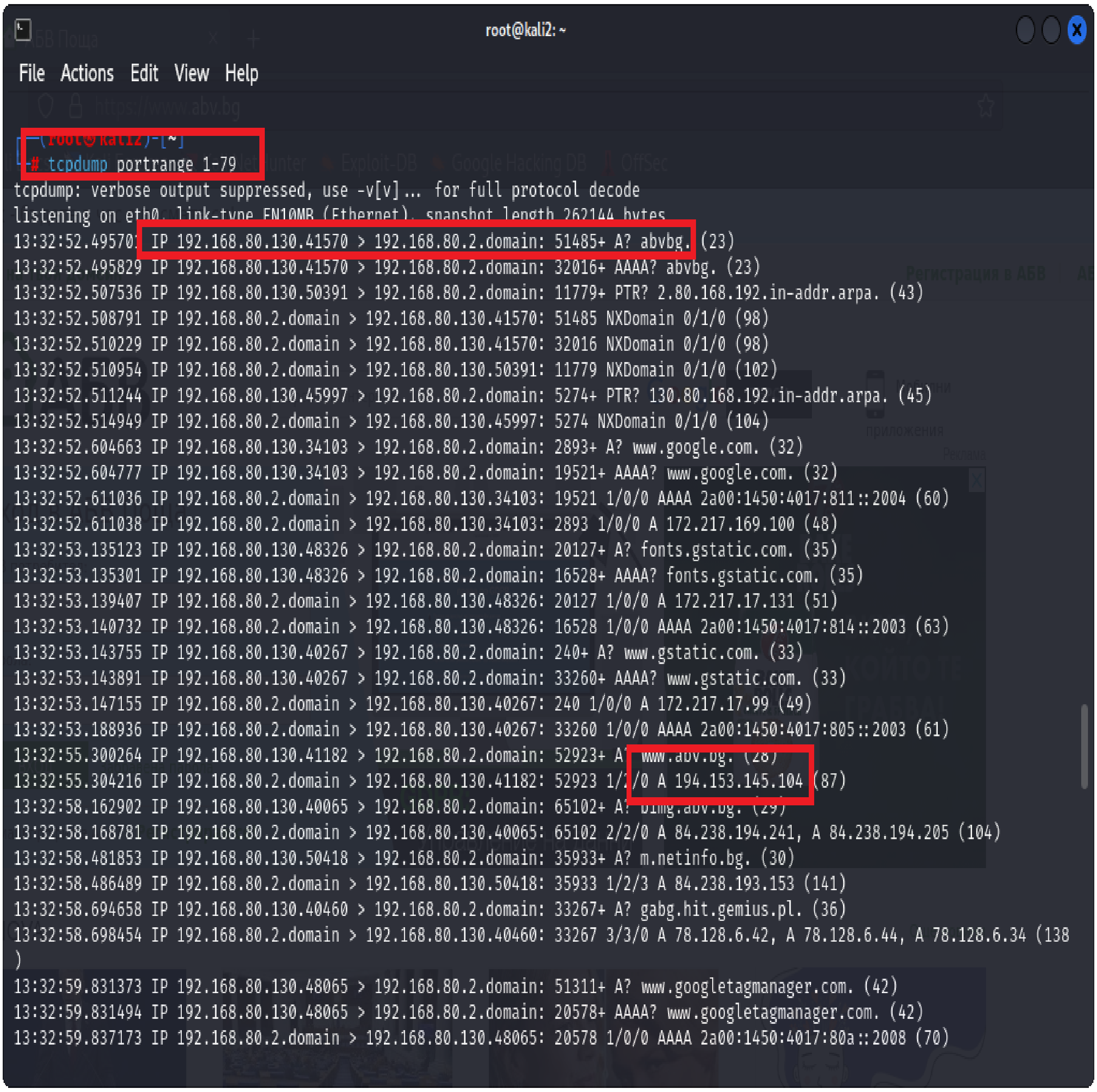
A records

IPv4 address	Revalidate in
194.153.145.104	23h 33m 37s

AAAA records

No AAAA records found.

Fig. 10. DNS records for the mail server „abv.bg“



```
┌──(root㉿kali2)-[~]
└─# tcpdump portrange 1-79
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
13:32:52.49570 IP 192.168.80.130.41570 > 192.168.80.2.domain: 51485+ A? abvbg. (23)
13:32:52.495829 IP 192.168.80.130.41570 > 192.168.80.2.domain: 32016+ AAAA? abvbg. (23)
13:32:52.507536 IP 192.168.80.130.50391 > 192.168.80.2.domain: 11779+ PTR? 2.80.168.192.in-addr.arpa. (43)
13:32:52.508791 IP 192.168.80.2.domain > 192.168.80.130.41570: 51485 NXDomain 0/1/0 (98)
13:32:52.510229 IP 192.168.80.2.domain > 192.168.80.130.41570: 32016 NXDomain 0/1/0 (98)
13:32:52.510954 IP 192.168.80.2.domain > 192.168.80.130.50391: 11779 NXDomain 0/1/0 (102)
13:32:52.511244 IP 192.168.80.130.45997 > 192.168.80.2.domain: 5274+ PTR? 130.80.168.192.in-addr.arpa. (45)
13:32:52.514949 IP 192.168.80.2.domain > 192.168.80.130.45997: 5274 NXDomain 0/1/0 (104)
13:32:52.604663 IP 192.168.80.130.34103 > 192.168.80.2.domain: 2893+ A? www.google.com. (32)
13:32:52.604777 IP 192.168.80.130.34103 > 192.168.80.2.domain: 19521+ AAAA? www.google.com. (32)
13:32:52.611036 IP 192.168.80.2.domain > 192.168.80.130.34103: 19521 1/0/0 AAAA 2a00:1450:4017:811::2004 (60)
13:32:52.611038 IP 192.168.80.2.domain > 192.168.80.130.34103: 2893 1/0/0 A 172.217.169.100 (48)
13:32:53.135123 IP 192.168.80.130.48326 > 192.168.80.2.domain: 20127+ A? fonts.gstatic.com. (35)
13:32:53.135301 IP 192.168.80.130.48326 > 192.168.80.2.domain: 16528+ AAAA? fonts.gstatic.com. (35)
13:32:53.139407 IP 192.168.80.2.domain > 192.168.80.130.48326: 20127 1/0/0 A 172.217.17.131 (51)
13:32:53.140732 IP 192.168.80.2.domain > 192.168.80.130.48326: 16528 1/0/0 AAAA 2a00:1450:4017:814::2003 (63)
13:32:53.143755 IP 192.168.80.130.40267 > 192.168.80.2.domain: 240+ A? www.gstatic.com. (33)
13:32:53.143891 IP 192.168.80.130.40267 > 192.168.80.2.domain: 33260+ AAAA? www.gstatic.com. (33)
13:32:53.147155 IP 192.168.80.2.domain > 192.168.80.130.40267: 240 1/0/0 A 172.217.17.99 (49)
13:32:53.188936 IP 192.168.80.2.domain > 192.168.80.130.40267: 33260 1/0/0 AAAA 2a00:1450:4017:805::2003 (61)
13:32:55.300264 IP 192.168.80.130.41182 > 192.168.80.2.domain: 52923+ A? www.abv.bg. (28)
13:32:55.304216 IP 192.168.80.2.domain > 192.168.80.130.41182: 52923 1/1/0 A 194.153.145.104 (87)
13:32:58.162902 IP 192.168.80.130.40065 > 192.168.80.2.domain: 65102+ A? bimg.abv.bg. (29)
13:32:58.168781 IP 192.168.80.2.domain > 192.168.80.130.40065: 65102 2/2/0 A 84.238.194.241, A 84.238.194.205 (104)
13:32:58.481853 IP 192.168.80.130.50418 > 192.168.80.2.domain: 35933+ A? m.netinfo.bg. (30)
13:32:58.486489 IP 192.168.80.2.domain > 192.168.80.130.50418: 35933 1/2/3 A 84.238.193.153 (141)
13:32:58.694658 IP 192.168.80.130.40460 > 192.168.80.2.domain: 33267+ A? gabg.hit.gemius.pl. (36)
13:32:58.698454 IP 192.168.80.2.domain > 192.168.80.130.40460: 33267 3/3/0 A 78.128.6.42, A 78.128.6.44, A 78.128.6.34 (138
)
13:32:59.831373 IP 192.168.80.130.48065 > 192.168.80.2.domain: 51311+ A? www.googletagmanager.com. (42)
13:32:59.831494 IP 192.168.80.130.48065 > 192.168.80.2.domain: 20578+ AAAA? www.googletagmanager.com. (42)
13:32:59.837173 IP 192.168.80.2.domain > 192.168.80.130.48065: 20578 1/0/0 AAAA 2a00:1450:4017:80a::2008 (70)
```

Fig. 11. Results of the executed command “tcpdump portrange 1-79”

4. Conclusion

The program Tcpdump is very similar in functionality and working way to the network analyzer Wireshark. Thanks to this network analyzer, the system administrator can filter the network connections by port, protocol, source and destination address of the network packet, network interface, TCP flags, network number, etc. After the filtering and interception of the network packets is done, all this information in a text file can be saved in order to be analyzed in detail for network anomalies and suspicious network activity the selected computer network. In this regard the exceptionally well-equipped laboratories at the Faculty of Technical Sciences at the Konstantin Preslavsky University of Shumen give great opportunities to students majoring in "Communication and Information Systems", "Computer Technologies in Automated Manufacturing" and "Signal Security Systems and Technologies" to gain extensive theoretical and practical experience in the work with the network command-line network tool - Tcpdump.

References:

[1] Arlos, P. and Fiedler, M., 2016. A comparison of measurement accuracy for DAG, tcpdump and windump. available online at Blekinge Institute of Technology (Sweden)< www. its. bth. se/staft/pca.

[2] Bachl, M., Fabini, J. and Zseby, T., 2021. A flow-based IDS using Machine Learning in eBPF. arXiv preprint arXiv:2102.09980.

[3] Boyanov, P., Implementation of modified script for Linux based operating systems using a linear algorithm for network port scanning. A refereed Journal Scientific and Applied Research, Konstantin Preslavsky University Press, Vol. 23, Shumen, 2022, ISSN 1314-6289 (Print), ISSN 2815-4622 (Online), pp. 48-59, DOI: https://doi.org/10.46687/jsar.v23i1.353.

[4] Boyanov, P., A comprehensive scanning for open, closed and filtered ports in the computer systems and networks. A refereed Journal Scientific and Applied Research, Konstantin Preslavsky University Press, Vol. 23, Shumen, 2022, ISSN 1314-6289 (Print), ISSN 2815-4622 (Online), pp. 85-98, DOI: https://doi.org/10.46687/jsar.v23i1.356.

[5] Corey, V., Peterman, C., Shearin, S., Greenberg, M.S. and Van Bokkelen, J., 2002. Network forensics analysis. IEEE Internet computing, 6(6), pp.60-66.

[6] Forte, D., 2001. Using tcpdump and sanitize for system security. login Usenix Mag., 26(3).

[7] Fuentes, F. and Kar, D.C., 2005. Ethereal vs. Tcpdump: a comparative study on packet sniffing tools for educational purpose. Journal of Computing Sciences in Colleges, 20(4), pp.169-176.

[8] Goyal, P. and Goyal, A., 2017, September. Comparative study of two most popular packet sniffing tools-Tcpdump and Wireshark. In 2017 9th International Conference on Computational Intelligence and Communication Networks (CICN) (pp. 77-81). IEEE.

[9] Guan, X., Ma, Y., Shao, Z. and Cao, W., 2020, July. Design and Application of Concurrent Test Scheme for Heartbeat Message of Mobile Terminal Based on Tcpdump and LoadRunner. In 2020 IEEE 10th International Conference on Electronics Information and Emergency Communication (ICEIEC) (pp. 232-235). IEEE.

[10] Gupta, A., 2013. A research study on packet sniffing tool TCPDUMP. International Journal of Communication and Computer Technologies, 1(49), pp.172-174.

[11] Iliev, R., K. Ignatova. Cloud technologies for building data center system for defense and security. T. Tagarev et al. (eds.), Digital Transformation, Cyber Security and Resilience of Modern Societies, Studies in Big Data 84, , ISBN 978-3-030-65721-5, Springer 2020, pp. 13-24, https://doi.org/10.1007/978-3-030-65722-2.

[12] Jandaeng, C., 2016. Embedded packet logger for network monitoring system. In Advanced Computer and Communication Engineering Technology: Proceedings of ICOCOE 2015 (pp. 1093-1102). Springer International Publishing.

[13] Joseph, D.A., Paxson, V. and Kim, S., 2006. tcpdump Tutorial. University of California, EE122 Fall.

[14] Mai, Y., Upadrashta, R. and Su, X., 2004, April. J-Honeypot: a Java-based network deception tool with monitoring and intrusion detection. In International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. (Vol. 1, pp. 804-808). IEEE.

[15] Makan, K., 2014. Penetration Testing with the Bash shell. Packt Publishing Ltd., ISBN 978-1-84969-510-7.

[16] Nauta, K. and Lieble, F., 1999, September. Offline network intrusion detection: Mining tcpdump data to identify suspicious activity. In Proceedings of the AFCEA Federal Database Colloquium.

[17] Pradhan, P. and Mannepalli, P., 2021. Machine Leaning for Flow Based Intrusion Detection Using Extended Berkley Packet Filter. Int. J. Eng. Res. Curr. Trends, 3, pp.5-7.

[18] Rohani, M.F.A., Maarof, M.A. and Selamat, A., 2005. Security Awareness: A Lesson from Tcpdump and Ethereal. In Proceedings of the Postgraduate Annual Research Seminar (p. 270).

[19] Therdphapiyanak, J. and Piromsopa, K., 2013, May. An analysis of suitable parameters for efficiently applying K-means clustering to large TCPdump data set using Hadoop framework. In 2013 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (pp. 1-6). IEEE.

[20] Van Der Merwe, J., Caceres, R., Chu, Y.H. and Sreenan, C., 2000. mmdump: A tool for monitoring Internet multimedia traffic. ACM SIGCOMM Computer Communication Review, 30(5), pp.48-59.

[21] Yurcik, W., Woolam, C., Hellings, G., Khan, L. and Thuraisingham, B., 2007, September. Scrub-tcpdump: A multi-level packet anonymizer demonstrating privacy/analysis tradeoffs. In 2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops-SecureComm 2007 (pp. 49-56). IEEE.