Peter the Great St.Petersburg Polytechnic University
Institute of Computer Science & Technologys
Department of Computer Systems & Software Engineering

Laboratory №2 Report

Discipline: Information Security

Theme: Nmap utility

Made by student of group. 13541/3 _____ D.V. Filippov
(signature)

Lecturer _____ N.V. Bogach
(signature)

Saint-Petersburg
2017

Contents

Nmap (”Network Mapper”) – a free and open source utility for network discovery and security auditing

1.1 Objectives

After completing this module you will be able to:

1. perform network discovery with various TARGET SPECIFICATION (hostnames, IP addresses, networks, etc.);
2. perform HOST DISCOVERY;
3. apply a variety of SCAN TECHNIQUES;
4. perform PORT SPECIFICATION AND set SCAN ORDER;
5. perform SERVICE/VERSION DETECTION;
6. perform SCRIPT SCAN;
7. perform OS DETECTION;
8. manage TIMING AND PERFORMANCE.

1.2 Task

1. List targets to scan;
2. Probe open ports to determine service/version info;
3. Study nmap-services, nmap-os-db, nmap-service-probes;
4. (OPTIONAL) Add new service to nmap-service-probes (create a minimal tcp server, get its name and version by nmap);
5. Output to xml-format file;
6. Study nmap stages and modes using Wireshark.

Perform VM Metasploitable2 scanning using db_nmap from metasploitframework.

Get 5 records from nmap-service-probes and describe them. Choose one Nmap Script and describe it

Work Progress

2.1 Preparing

1. Download last kali linux and metasploitable2 distributions;
2. Install it in VMware Workstation;
3. Setting up common network;
4. Define the IP addresses of virtual machines;
5. Check ping request's in both ways.

As common network choosed - **VMnet8**.

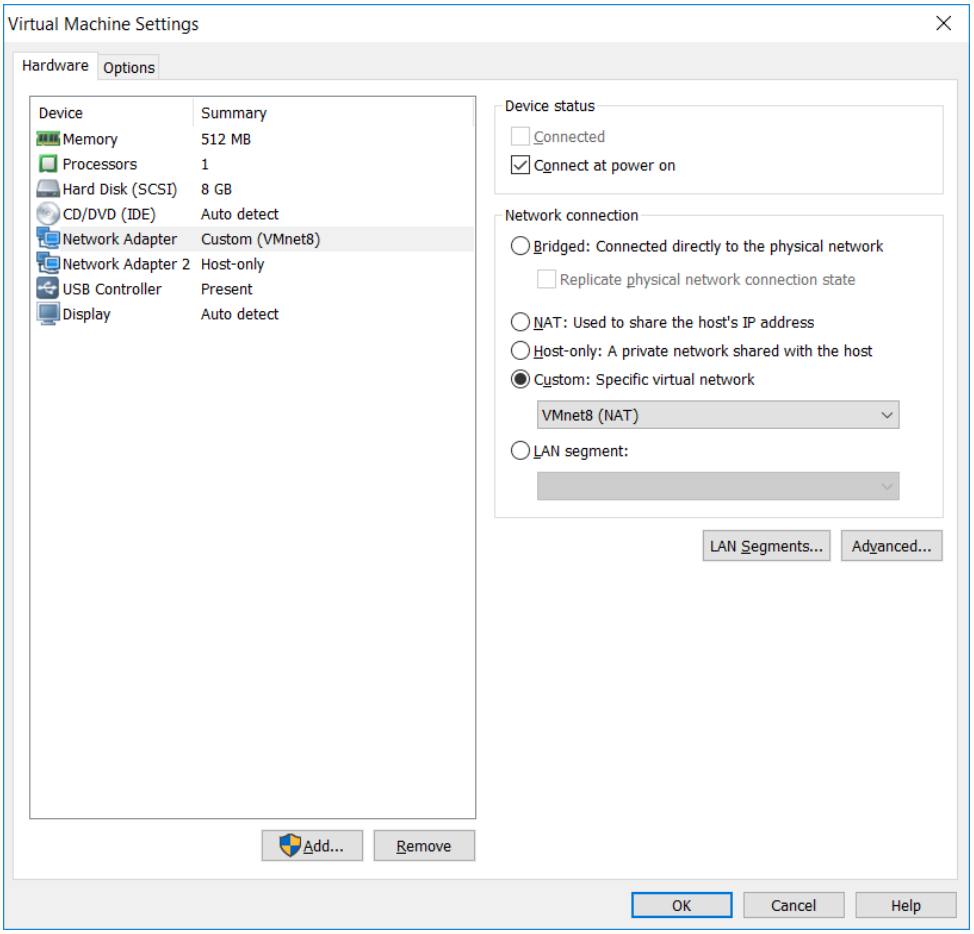


Figure 2.1: Metasploitable2 settings

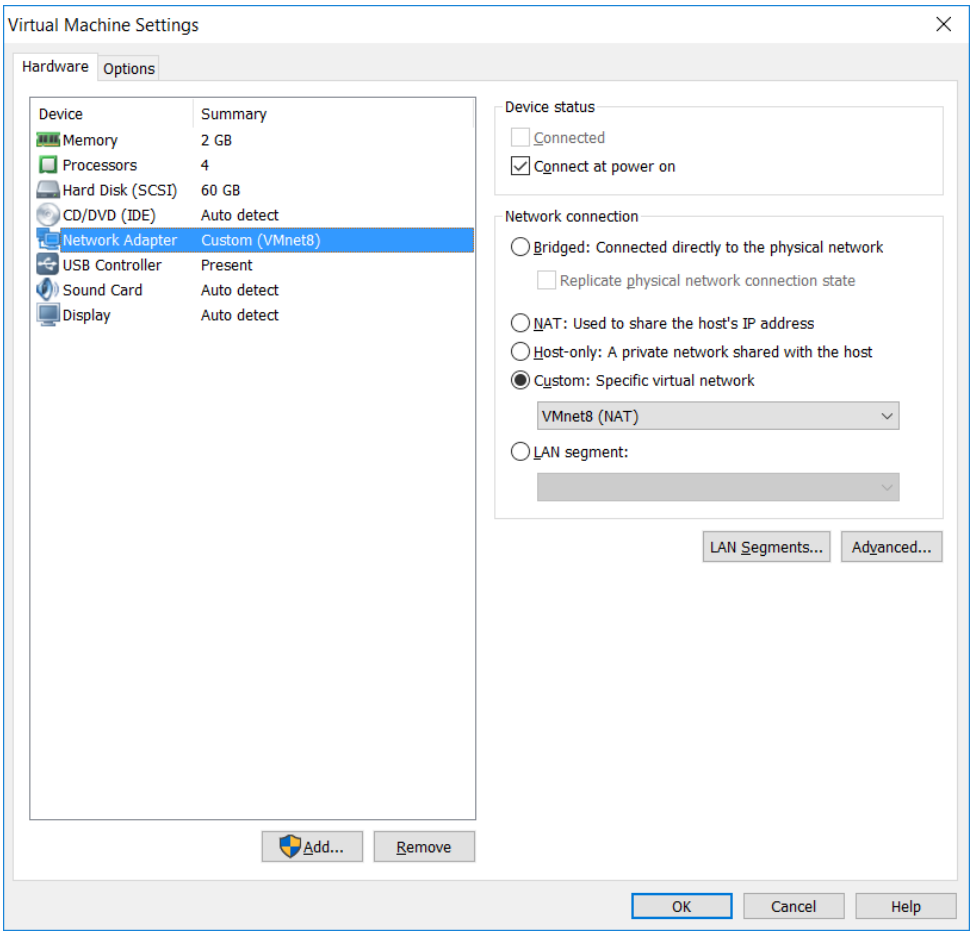


Figure 2.2: Kali settings

Now power on VM's and define their IP addresses.

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:88:7b:e8
          inet addr:192.168.81.130  Bcast:192.168.81.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe88:7be8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:41 errors:0 dropped:0 overruns:0 frame:0
          TX packets:68 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4783 (4.6 KB)  TX bytes:7140 (6.9 KB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:94 errors:0 dropped:0 overruns:0 frame:0
          TX packets:94 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:19577 (19.1 KB)  TX bytes:19577 (19.1 KB)
```

Figure 2.3: Metasploitable2 ifconfig

```
1 root@kali:~# ifconfig
2 eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
3         inet 192.168.81.129  netmask 255.255.255.0  broadcast
    ↪ 192.168.81.255
```

```
4           inet6 fe80::20c:29ff:feed:c99e  prefixlen 64  scopeid 0x20
      ↪ <link>
5           ether 00:0c:29:ed:c9:9e  txqueuelen 1000  (Ethernet)
6           RX packets 69  bytes 7784 (7.6 KiB)
7           RX errors 0  dropped 0  overruns 0  frame 0
8           TX packets 36  bytes 3002 (2.9 KiB)
9           TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
10
11 lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
12          inet 127.0.0.1  netmask 255.0.0.0
13          inet6 ::1  prefixlen 128  scopeid 0x10<host>
14          loop  txqueuelen 1000  (Local Loopback)
15          RX packets 26  bytes 1518 (1.4 KiB)
16          RX errors 0  dropped 0  overruns 0  frame 0
17          TX packets 26  bytes 1518 (1.4 KiB)
18          TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Listing 2.1: Kali ifconfig

Using **ifconfig** command, i defined IP addresses:

Metasploitable2 IP - **192.168.81.130**

Kali IP - **192.168.81.129**

To test that VM see each other, use the **ping** command.

```
msfadmin@metasploitable:~$ ping 192.168.81.129
PING 192.168.81.129 (192.168.81.129) 56(84) bytes of data.
64 bytes from 192.168.81.129: icmp_seq=1 ttl=64 time=0.012 ms
64 bytes from 192.168.81.129: icmp_seq=2 ttl=64 time=0.366 ms
64 bytes from 192.168.81.129: icmp_seq=3 ttl=64 time=0.544 ms
64 bytes from 192.168.81.129: icmp_seq=4 ttl=64 time=0.534 ms

--- 192.168.81.129 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.012/0.364/0.544/0.215 ms
```

Figure 2.4: Metasploitable2 ping

```
1  root@kali:~# ping 192.168.81.130
2  PING 192.168.81.130 (192.168.81.130) 56(84) bytes of data.
3  64 bytes from 192.168.81.130: icmp_seq=1 ttl=64 time=2.08 ms
4  64 bytes from 192.168.81.130: icmp_seq=2 ttl=64 time=0.542 ms
5  64 bytes from 192.168.81.130: icmp_seq=3 ttl=64 time=0.579 ms
6  64 bytes from 192.168.81.130: icmp_seq=4 ttl=64 time=0.669 ms
7  ^C
8  --- 192.168.81.130 ping statistics ---
9  4 packets transmitted, 4 received, 0% packet loss, time 3061ms
10 rtt min/avg/max/mdev = 0.542/0.968/2.085/0.647 ms
```

Listing 2.2: Kali ping

Ping requests were successfully completed, which means that the network is working successfully.

2.2 List targets to scan

The following command will be used to scan the network:

nmap -sn 192.168.81.0-255

Key **sn** means only ping scan, without port scan. 0-255 means in what range nmap should find target's.

```
root@kali:~# nmap -sn 192.168.81.0-255

Starting Nmap 7.60 ( https://nmap.org ) at 2017-11-04 06:13 EDT
Nmap scan report for 192.168.81.1
Host is up (0.00050s latency).
MAC Address: 00:50:56:C0:00:08 (VMware)
Nmap scan report for 192.168.81.2
Host is up (0.00033s latency).
MAC Address: 00:50:56:EF:06:7B (VMware)
Nmap scan report for 192.168.81.130
Host is up (-0.10s latency).
MAC Address: 00:0C:29:88:7B:E8 (VMware)
Nmap scan report for 192.168.81.254
Host is up (-0.12s latency).
MAC Address: 00:50:56:E1:C6:C7 (VMware)
Nmap scan report for 192.168.81.129
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 3.23 seconds
```

Listing 2.3: NMAP scanning result

Metasploitable2 was successfully found.

2.3 Probe open ports to determine service/version info

Now scan 10 most popular ports at Metasploitable2 IP address.

```
root@kali:~# nmap -top-ports 10 192.168.81.130

Starting Nmap 7.60 ( https://nmap.org ) at 2017-11-04 08:37 EDT
Nmap scan report for 192.168.81.130
Host is up (0.00045s latency).
PORT     STATE  SERVICE
21/tcp   open   ftp
22/tcp   open   ssh
23/tcp   open   telnet
25/tcp   open   smtp
80/tcp   open   http
110/tcp  closed pop3
139/tcp  open   netbios-ssn
443/tcp  closed https
445/tcp  open   microsoft-ds
```

```
16 3389/tcp closed ms-wbt-server
17 MAC Address: 00:0C:29:88:7B:E8 (VMware)
18
19 Nmap done: 1 IP address (1 host up) scanned in 0.71 seconds
```

Listing 2.4: NMAP scanning ports

Now determine service and version info with the **sV** key.

```
 1 root@kali:~# nmap -sV 192.168.81.130
 2
 3 Starting Nmap 7.60 ( https://nmap.org ) at 2017-11-04 08:40 EDT
 4 Nmap scan report for 192.168.81.130
 5 Host is up (0.00033s latency).
 6 Not shown: 977 closed ports
 7 PORT     STATE SERVICE     VERSION
 8 21/tcp   open  ftp         vsftpd 2.3.4
 9 22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol
     ↪ 2.0)
10 23/tcp   open  telnet      Linux telnetd
11 25/tcp   open  smtp        Postfix smtpd
12 53/tcp   open  domain      ISC BIND 9.4.2
13 80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
14 111/tcp  open  rpcbind     2 (RPC #100000)
15 139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup:
     ↪ WORKGROUP)
16 445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup:
     ↪ WORKGROUP)
17 512/tcp  open  exec        netkit-rsh rexecd
18 513/tcp  open  login       OpenBSD or Solaris rlogind
19 514/tcp  open  tcpwrapped
20 1099/tcp open  rmiregistry GNU Classpath grmiregistry
21 1524/tcp open  shell       Metasploitable root shell
22 2049/tcp open  nfs         2-4 (RPC #100003)
23 2121/tcp open  ftp         ProFTPD 1.3.1
24 3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
25 5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
26 5900/tcp open  vnc         VNC (protocol 3.3)
27 6000/tcp open  X11         (access denied)
28 6667/tcp open  irc         UnrealIRCd
29 8009/tcp open  ajp13       Apache Jserv (Protocol v1.3)
30 8180/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
31 MAC Address: 00:0C:29:88:7B:E8 (VMware)
32 Service Info: Hosts:  metasploitable.localdomain, localhost, irc.
     ↪ Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:
     ↪ linux_kernel
33
34 Service detection performed. Please report any incorrect results
     ↪ at https://nmap.org/submit/ .
35 Nmap done: 1 IP address (1 host up) scanned in 14.48 seconds
```

Listing 2.5: NMAP version info

2.4 Study nmap-services, nmap-os-db, nmap-service-probes

Theese files can be found in the directory **/usr/share/nmap**.

The **nmap-services** file is a registry of port names to their corresponding number and protocol. Each entry has a number representing how likely that port is to be found open. Most lines have a comment as well.

```
1  tcpmux  1/tcp   0.001995    # TCP Port Service Multiplexer [rfc
       ↪ -1078] | TCP Port Service Multiplexer
2  tcpmux  1/udp   0.001236    # TCP Port Service Multiplexer
3  compressnet 2/tcp   0.000013    # Management Utility
4  compressnet 2/udp   0.001845    # Management Utility
5  compressnet 3/tcp   0.001242    # Compression Process
6  compressnet 3/udp   0.001532    # Compression Process
7  unknown 4/tcp   0.000477
8  rje 5/tcp   0.000000    # Remote Job Entry
9  rje 5/udp   0.000593    # Remote Job Entry
10 unknown 6/tcp   0.000502
11 echo    7/sctp  0.000000
12 echo    7/tcp   0.004855
13 echo    7/udp   0.024679
14 unknown 8/tcp   0.000013
15 discard 9/sctp  0.000000    # sink null
16 discard 9/tcp   0.003764    # sink null
17 discard 9/udp   0.015733    # sink null
18 unknown 10/tcp  0.000063
19 systat  11/tcp  0.000075    # Active Users
20 systat  11/udp  0.000577    # Active Users
21 unknown 12/tcp  0.000063
22 daytime 13/tcp  0.003927
23 daytime 13/udp  0.004827
24 unknown 14/tcp  0.000038
25 netstat 15/tcp  0.000038
26 unknown 16/tcp  0.000050
```

Listing 2.6: Some lines of nmap-services

The **nmap-os-db** data file contains hundreds of examples of how different operating systems respond to Nmap's specialized OS detection probes. It is divided into blocks known as fingerprints, with each fingerprint containing an operating system's name.

```
53775 # Linux 3.5.0-17-generic #28-Ubuntu SMP Tue Oct 9 19:31:23 UTC
       ↪ 2012 x86_64 x86_64 x86_64 GNU/Linux, Linux Mint 14
53776 # Google Chromecast
53777 # MikroTik RouterOS 6.0 rc 14
53778 # Aastra SIP-DECT 4.0SP1
53779 # Android version 4.1.2, Kernel 3.4.0
53780 # Amazon FireStick and FireTV
53781 Fingerprint Linux 2.6.32 - 3.10
53782 Class Linux | Linux | 2.6.X | general purpose
53783 CPE cpe:/o:linux:linux_kernel:2.6 auto
53784 Class Linux | Linux | 3.X | general purpose
```

```
53785 CPE cpe:/o:linux:linux_kernel:3 auto
53786 SEQ(SP=ED-10B%GCD=1-6%ISR=F0-114%TI=Z%CI=Z%II=I%TS=7|8|9|A|B)
53787 OPS(O1=M5B4ST11NW1|M5B4ST11NW2|M5B4ST11NW3|M5B4ST11NW4|M5B4ST11NW5
      ↪ |M5B4ST11NW6|M5B4ST11NW7|M5B4ST11NW8|M5B4ST11NW9|M5B4ST11NWA
      ↪ %O2=M5B4ST11NW1|M5B4ST11NW2|M5B4ST11NW3|M5B4ST11NW4|
      ↪ M5B4ST11NW5|M5B4ST11NW6|M5B4ST11NW7|M5B4ST11NW8|M5B4ST11NW9|
      ↪ M5B4ST11NWA%O3=M5B4NNT11NW1|M5B4NNT11NW2|M5B4NNT11NW3|
      ↪ M5B4NNT11NW4|M5B4NNT11NW5|M5B4NNT11NW6|M5B4NNT11NW7|
      ↪ M5B4NNT11NW8|M5B4NNT11NW9|M5B4NNT11NWA%O4=M5B4ST11NW1|
      ↪ M5B4ST11NW2|M5B4ST11NW3|M5B4ST11NW4|M5B4ST11NW5|M5B4ST11NW6|
      ↪ M5B4ST11NW7|M5B4ST11NW8|M5B4ST11NW9|M5B4ST11NWA%O5=
      ↪ M5B4ST11NW1|M5B4ST11NW2|M5B4ST11NW3|M5B4ST11NW4|M5B4ST11NW5|
      ↪ M5B4ST11NW6|M5B4ST11NW7|M5B4ST11NW8|M5B4ST11NW9|M5B4ST11NWA%
      ↪ O6=M5B4ST11)
53788 WIN(W1=3890%W2=3890%W3=3890%W4=3890%W5=3890%W6=3890)
53789 ECN(R=Y%DF=Y%T=3B-45%TG=40%W=3908%O=M5B4NNSNW1|M5B4NNSNW2|
      ↪ M5B4NNSNW3|M5B4NNSNW4|M5B4NNSNW5|M5B4NNSNW6|M5B4NNSNW7|
      ↪ M5B4NNSNW8|M5B4NNSNW9|M5B4NNSNWA%CC=N|Y)
53790 T1(R=Y%DF=Y%T=3B-45%TG=40%S=O%A=S+%F=AS%RD=0)
53791 T2(R=N)
53792 T3(R=N)
53793 T4(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=A%A=Z%F=R%RD=0)
53794 T5(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=Z%A=S+%F=AR%RD=0)
53795 T6(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=A%A=Z%F=R%RD=0)
53796 T7(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=Z%A=S+%F=AR%RD=0)
53797 U1(DF=N%T=3B-45%TG=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G|I%RUCK=G%
      ↪ RUD=G)
53798 IE(DFI=N%T=3B-45%TG=40%CD=S)
```

Listing 2.7: One of many fingerprint's

nmap-service-probes file contains the probes that the Nmap service/version detection system (-sV or -A options) uses during port interrogation to determine what program is listening on a port.

```
12975 ##############################NEXT PROBE
      ↪ ##############################
12976 # SSLv3 ClientHello probe. Will be able to reliably identify the
      ↪ SSL version
12977 # used, unless the server is running SSLv2 only. Note that it will
      ↪  also detect
12978 # TLSv1-only servers, based on a failed handshake alert.
12979 Probe TCP SSLSessionReq q|\x16\x03\0\0S\x01\0\0O\x03\0?G\xd7\xf7\
      ↪ xba,\xee\xea\xb2'~\xf3\0\xfd\x82{\xb9\xd5\x96\xc8w\x9b\xe6\
      ↪ xc4\xdb<=\xdbo\xef\x10n\0\0(\0\x16\0\x13\0\x0a\0f\0\x05\0\
      ↪ x04\0e\0d\0c\0b\0a\0`\0\x15\0\x12\0\x09\0\x14\0\x11\0\x08\0\
      ↪ x06\0\x03\x01\0|
12980 rarity 1
12981 ports 322,443,444,465,548,636,989,990,992,993,994,995,1241,1311,
      ↪ 1443,2000,2252,2443,3443,4433,4443,4444,4911,5061,5443,5550,
      ↪
      ↪ 6443,6679,6697,7000,7210,7272,7443,8009,8181,8194,8443,8531,
```

```
    ↪ 8883,9001,9443,10443,14443,44443,60443
12982 fallback GetRequest
12983
12984 # OpenSSL/0.9.7aa, 0.9.8e
12985 match ssl m|^\x16\x03\0\0J\x02\0\0F\x03\0| p/OpenSSL/ i/SSLv3/ cpe
    ↪ :/a:openssl:openssl/
12986
12987 # Microsoft-IIS/5.0 - note that OpenSSL must go above this one
    ↪ because this is more general
12988 match ssl m|^\x16\x03\0..\x02\0\0F\x03\0|s p/Microsoft IIS SSL/ o/
    ↪ Windows/ cpe:/a:microsoft:iis/ cpe:/o:microsoft:windows/a
12989 # Novell Netware 6 Enterprise Web server 5.1 https
12990 # Novell Netware Ldap over SSL or enterprise web server 5.1 over
    ↪ SSL
12991 match ssl m|^\x16\x03\0\0:\x02\0\x006\x03\0| p/Novell NetWare SSL/
    ↪ o/NetWare/ cpe:/o:novell:netware/a
12992 # Cisco IDS 4.1 Appliance
12993 match ssl m|^\x16\x03\0\0\*\x02\0\0&\x03\0\xd10:\xbd\\\x8e\xe3\x15
    ↪ \x1c\x0fZ\xe4\x04\x87\x07\xc0\x82\xa9\xd4\x0e\x9c1LXk\xd1\
    ↪ xd2\x0b\x1a\xc6/p\0\0\n\0\x16\x03\0\x026\x0b\0\x022\0| p/
    ↪ Cisco IDS SSL/ d/firewall/
12994 # PGP Corporation Keyserver Web Console 7.0 - custom Apache 1.3
12995 # PGP LDAPS Keyserver 8.X
12996 match ssl m|^\x16\x03\0\0\+\x02\0\0'\x03\0...\?|s p/PGP
    ↪ Corporation product SSL/
12997 # Unreal IRCd SSL
12998 # RemotelyAnywhere
12999 match ssl m|^\x16\x03\0\0\*\x02\0\0&\x03\0\?|
13000 # Tumbleweed SecureTransport 4.1.1 Transaction Manager Secure Port
    ↪ on Solaris
13001 # Dell Openmanage
13002 match ssl m|^\x15\x03[\x01\x00]\0\x02\x01\0$| p/multi-vendor SSL/
13003 # Probably Oracle https?
13004 match ssl m|^}\0\x02\0\0\0\0\0\0\0\0\0\0\0\0| p/Oracle https/
13005 match ssl m|^\x15\x03\0\0\x02\x02\(31666:error:1408A0C1:SSL
    ↪ routines:SSL3_GET_CLIENT_HELLO:no shared cipher:s3_srvr\.c
    ↪ :881:\n| p/Webmin SSL Control Panel/
13006 match ssl m|^20928:error:140760FC:SSL routines:
    ↪ SSL23_GET_CLIENT_HELLO:unknown protocol:s23_srvr\.c:565:\n|
    ↪ p/qmail-pop3d behind stunnel/ cpe:/a:djb:qmail/
```

Listing 2.8: Some lines of nmap-service-probes

The Probe directive tells Nmap what string to send to recognize various services. All of the directives discussed later operate on the most recent Probe statement. The arguments are as follows:

1. <protocol>
    - This must be either TCP or UDP.
2. <probename>

- Plain English name for the probe.

3. <probestring>

   - Tells Nmap what to send. It must start with a q, then a delimiter character which begins and ends the string. Between the delimiter characters is the string that is actually sent.

2.5 Add new service to nmap-service-probes (create a minimal tcp server, get its name and version by nmap)

The source code of echo-server is shown below.

```
1  package server;
2
3  import java.net.*;
4  import java.io.*;
5
6  public class Server {
7
8      private static int SERVER_PORT=8090;
9      private static String SERVER_IP="127.0.0.1";
10
11     private static String VERSION = "1.0";
12
13     public static void main(String[] args) {
14         try {
15             ServerSocket serverSocket = new ServerSocket(
   ↪ SERVER_PORT,0,InetAddress.getByName(SERVER_IP));
16             System.out.println("Server started.");
17             Socket clientSocket = serverSocket.accept();
18             PrintWriter out = new PrintWriter(clientSocket.
   ↪ getOutputStream(), true);
19             BufferedReader in = new BufferedReader(new
   ↪ InputStreamReader(clientSocket.getInputStream()));
20
21             String inputLine;
22             while (true) {
23                 inputLine = in.readLine();
24                 if(inputLine != null){
25                     if(inputLine.equals("version"))
26                         out.println(VERSION);
27                     else
28                         out.println(inputLine);
29                 }
30             }
31         } catch (IOException e) {
32             System.out.println("Exception caught when trying to
   ↪ listen on port " + SERVER_PORT + " or listening for a
   ↪ connection");
```

```
33                 System.out.println(e.getMessage());
34             }
35         }
36 }
```

Listing 2.9: Server.java

```
1 root@kali:~/Desktop/tcpServer# javac server/Server.java
2 root@kali:~/Desktop/tcpServer# java server/Server
3 Server started.
```

Listing 2.10: Compiling and starting server

Echo server was successfully detected, but nmap did not determine what it is, not even that this is echo-server.

```
1 root@kali:~# nmap -sV -p 8090 127.0.0.1
2
3 Starting Nmap 7.60 ( https://nmap.org ) at 2017-11-04 12:55 EDT
4 Nmap scan report for localhost (127.0.0.1)
5 Host is up (0.000045s latency).
6
7 PORT     STATE SERVICE       VERSION
8 8090/tcp open  opsmessaging?
9 1 service unrecognized despite returning data. If you know the
    ↪ service/version, please submit the following fingerprint at
    ↪ https://nmap.org/cgi-bin/submit.cgi?new-service :
10 SF-Port8090-TCP:V=7.60%I=7%D=11/4%Time=59FDF102%P=x86_64-pc-linux-
    ↪ gnu%r(si
11 SF:mple-tcp-server-ver,8,"version\n");
12
13 Service detection performed. Please report any incorrect results
    ↪ at https://nmap.org/submit/ .
14 Nmap done: 1 IP address (1 host up) scanned in 11.61 seconds
```

Listing 2.11: Nmap scanning(unsuccessful)

Now add the server description to the nmap-service-probes.

```
1 root@kali:~# tail -n 7 /usr/share/nmap/nmap-service-probes
2 ##############################NEXT PROBE
    ↪ ##############################
3 # Echo TSP server.
4 Probe TCP echo-tcp-server-ver q|version\r\n|
5 rarity 9
6 ports 8090
7 match stcps m|^1\.0$| p/Echo TCP Server/ v/1.0/
```

Listing 2.12: Adding description

And scan with nmap again.

```
1 root@kali:~# nmap -sV -p 8090 127.0.0.1
2
3 Starting Nmap 7.60 ( https://nmap.org ) at 2017-11-04 13:02 EDT
```

```
4 Nmap scan report for localhost (127.0.0.1)
5 Host is up (0.000065s latency).
6
7 PORT     STATE SERVICE VERSION
8 8090/tcp open  stcps   Echo TCP Server 1.0
9
10 Service detection performed. Please report any incorrect results
     ↪ at https://nmap.org/submit/ .
11 Nmap done: 1 IP address (1 host up) scanned in 6.54 seconds
```

Listing 2.13: Nmap scanning(successful)

As expected, nmap successfully determined my echo server.

2.6 Output to xml-format file

Adding **oX** key for output to xml file.

root@kali: # nmap -sV -p 8090 -oX myOutput.xml 127.0.0.1

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE nmaprun>
3 <?xml-stylesheet href="file:///usr/bin/../share/nmap/nmap.xsl"
     ↪ type="text/xsl"?>
4 <!-- Nmap 7.60 scan initiated Sat Nov  4 13:26:07 2017 as: nmap -
     ↪ sV -p 8090 -oX myOutput.xml 127.0.0.1 -->
5 <nmaprun scanner="nmap" args="nmap -sV -p 8090 -oX myOutput.xml
     ↪ 127.0.0.1" start="1509816367" startstr="Sat Nov  4 13:26:07
     ↪ 2017" version="7.60" xmloutputversion="1.04">
6 <scaninfo type="syn" protocol="tcp" numservices="1" services
     ↪ ="8090"/>
7 <verbose level="0"/>
8 <debugging level="0"/>
9 <host starttime="1509816367" endtime="1509816373"><status state="
     ↪ up" reason="localhost-response" reason_ttl="0"/>
10 <address addr="127.0.0.1" addrtype="ipv4"/>
11 <hostnames>
12 <hostname name="localhost" type="PTR"/>
13 </hostnames>
14 <ports><port protocol="tcp" portid="8090"><state state="open"
     ↪ reason="syn-ack" reason_ttl="64"/><service name="stcps"
     ↪ product="Echo TCP Server" version="1.0" method="probed" conf
     ↪ ="10"/></port>
15 </ports>
16 <times srtt="48" rttvar="5000" to="100000"/>
17 </host>
18 <runstats><finished time="1509816373" timestr="Sat Nov  4 13:26:13
     ↪  2017" elapsed="6.57" summary="Nmap done at Sat Nov  4
     ↪ 13:26:13 2017; 1 IP address (1 host up) scanned in 6.57
     ↪ seconds" exit="success"/><hosts up="1" down="0" total="1"/>
```

```
19 </runstats>
20 </nmaprun>
```

Listing 2.14: myOutput.xml

2.7 Study nmap stages and modes using Wireshark

Scanning port 22

```
1  root@kali:~# nmap -sV 192.168.81.130 -p 22
2
3  Starting Nmap 7.60 ( https://nmap.org ) at 2017-11-04 14:23 EDT
4  Nmap scan report for 192.168.81.130
5  Host is up (0.00041s latency).
6
7  PORT   STATE SERVICE VERSION
8  22/tcp open  ssh     OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
9  MAC Address: 00:0C:29:88:7B:E8 (VMware)
10 Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
11
12 Service detection performed. Please report any incorrect results
   ↪ at https://nmap.org/submit/ .
13 Nmap done: 1 IP address (1 host up) scanned in 0.97 seconds
```

Listing 2.15: Scanning port 22

Let's see what is in the packets, that captured with wireshark.

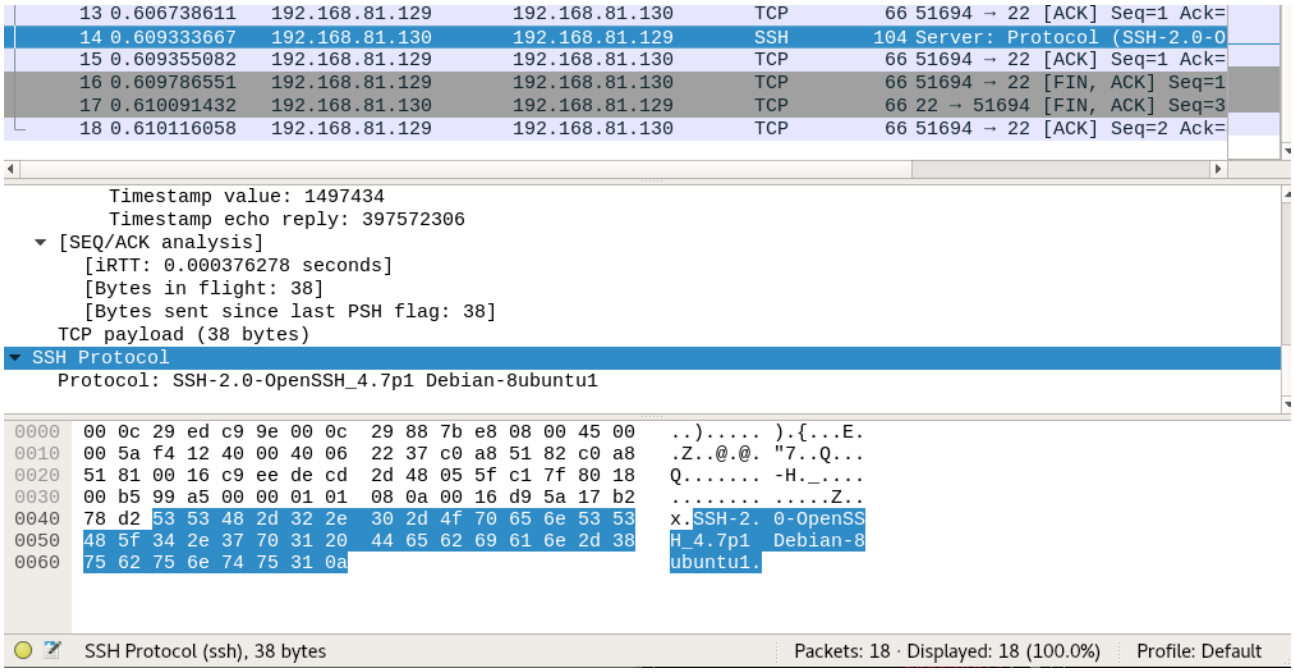


Figure 2.5: Wireshark packets

In response message, we see name of service at this port.
Now let's see what happens when the command below is used.

```
root@kali:~# nmap --top-ports 10 192.168.81.130

Starting Nmap 7.60 ( https://nmap.org ) at 2017-11-04 14:57 EDT
Nmap scan report for 192.168.81.130
Host is up (0.00080s latency).

PORT     STATE  SERVICE
21/tcp   open   ftp
22/tcp   open   ssh
23/tcp   open   telnet
25/tcp   open   smtp
80/tcp   open   http
110/tcp  closed pop3
139/tcp  open   netbios-ssn
443/tcp  closed https
445/tcp  open   microsoft-ds
3389/tcp closed ms-wbt-server
MAC Address: 00:0C:29:88:7B:E8 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.57 seconds
```

Listing 2.16: 10 most popular ports

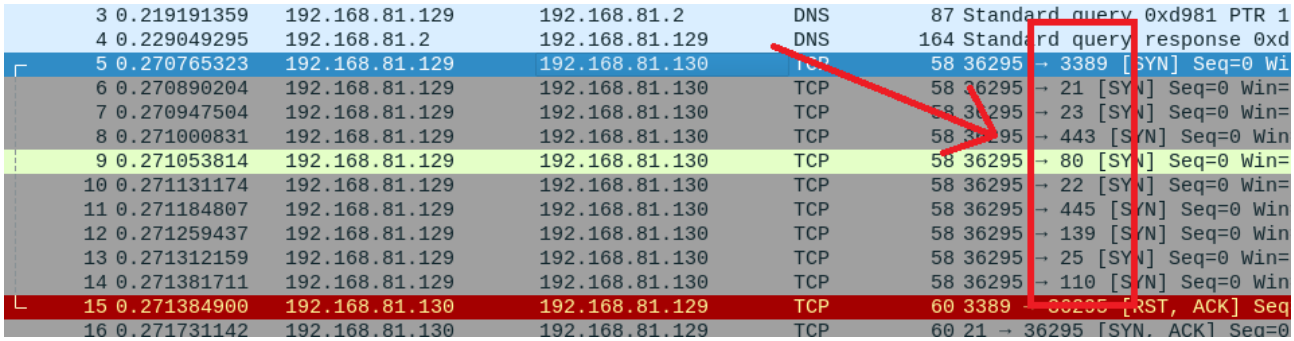
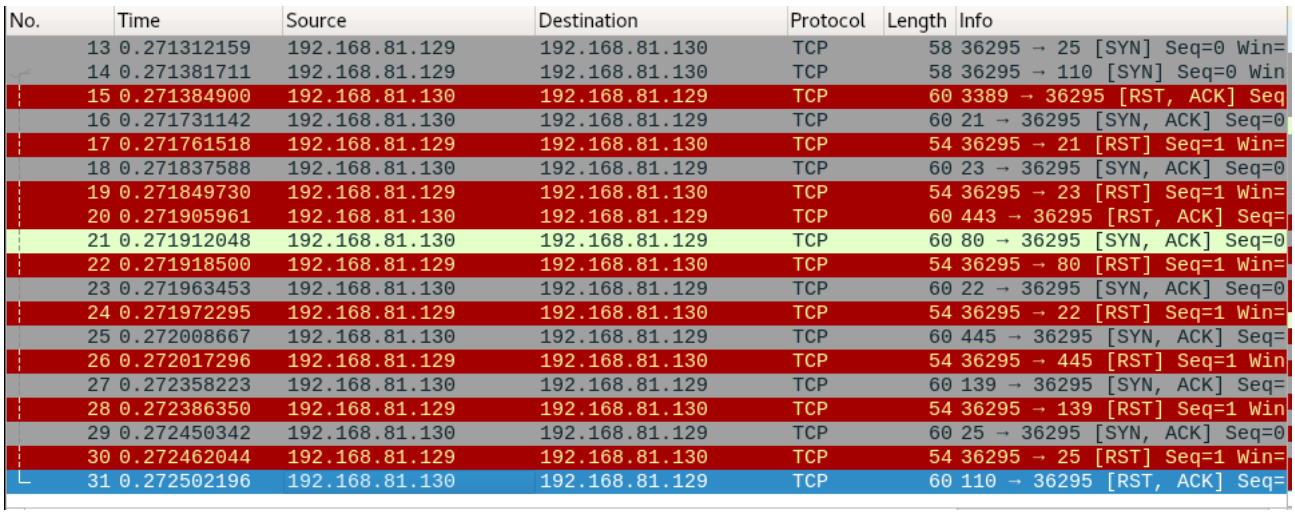


Figure 2.6: Wireshark packets

The screenshot highlights that 10 requests were sent to the most popular ports.

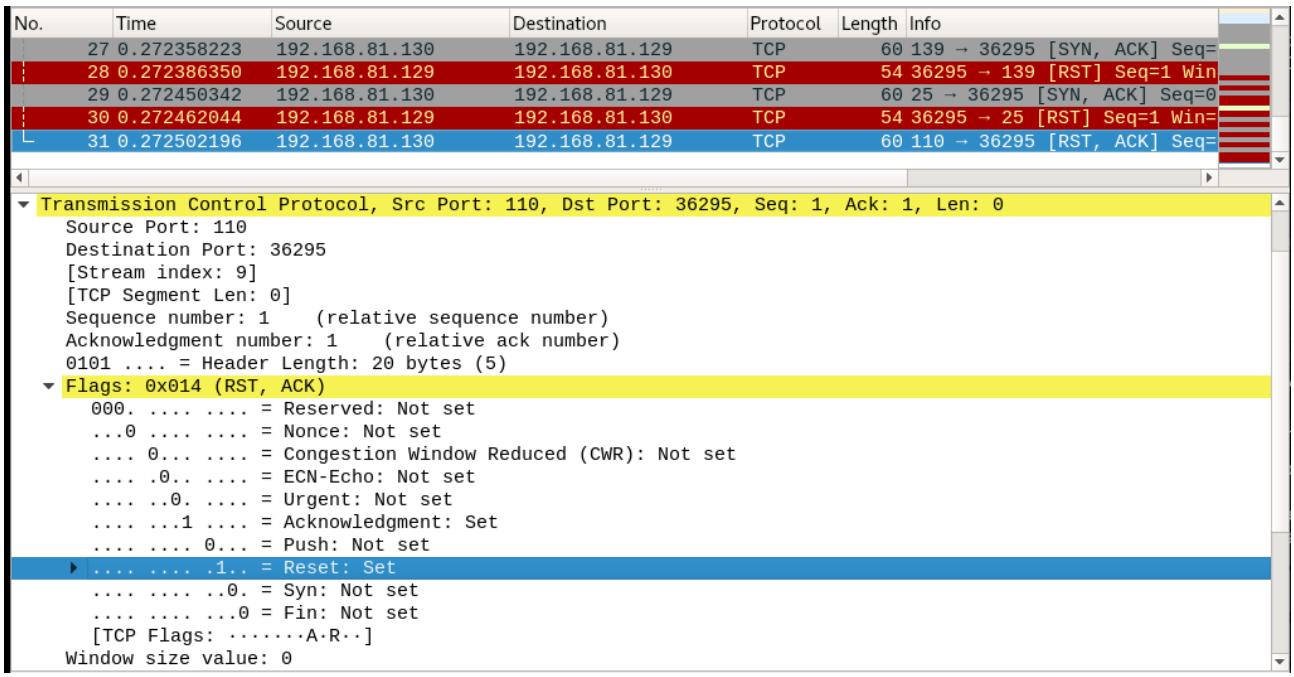


No.	Time	Source	Destination	Protocol	Length	Info
13	0.271312159	192.168.81.129	192.168.81.130	TCP	58	36295 → 25 [SYN] Seq=0 Win=
14	0.271381711	192.168.81.129	192.168.81.130	TCP	58	36295 → 110 [SYN] Seq=0 Win
15	0.271384900	192.168.81.130	192.168.81.129	TCP	60	3389 → 36295 [RST, ACK] Seq
16	0.271731142	192.168.81.130	192.168.81.129	TCP	60	21 → 36295 [SYN, ACK] Seq=0
17	0.271761518	192.168.81.129	192.168.81.130	TCP	54	36295 → 21 [RST] Seq=1 Win=
18	0.271837588	192.168.81.130	192.168.81.129	TCP	60	23 → 36295 [SYN, ACK] Seq=0
19	0.271849730	192.168.81.129	192.168.81.130	TCP	54	36295 → 23 [RST] Seq=1 Win=
20	0.271905961	192.168.81.130	192.168.81.129	TCP	60	443 → 36295 [RST, ACK] Seq=
21	0.271912048	192.168.81.130	192.168.81.129	TCP	60	80 → 36295 [SYN, ACK] Seq=0
22	0.271918500	192.168.81.129	192.168.81.130	TCP	54	36295 → 80 [RST] Seq=1 Win=
23	0.271963453	192.168.81.130	192.168.81.129	TCP	60	22 → 36295 [SYN, ACK] Seq=0
24	0.271972295	192.168.81.129	192.168.81.130	TCP	54	36295 → 22 [RST] Seq=1 Win=
25	0.272008667	192.168.81.130	192.168.81.129	TCP	60	445 → 36295 [SYN, ACK] Seq=
26	0.272017296	192.168.81.129	192.168.81.130	TCP	54	36295 → 445 [RST] Seq=1 Win
27	0.272358223	192.168.81.130	192.168.81.129	TCP	60	139 → 36295 [SYN, ACK] Seq=
28	0.272386350	192.168.81.129	192.168.81.130	TCP	54	36295 → 139 [RST] Seq=1 Win
29	0.272450342	192.168.81.130	192.168.81.129	TCP	60	25 → 36295 [SYN, ACK] Seq=0
30	0.272462044	192.168.81.129	192.168.81.130	TCP	54	36295 → 25 [RST] Seq=1 Win=
31	0.272502196	192.168.81.130	192.168.81.129	TCP	60	110 → 36295 [RST, ACK] Seq=

Figure 2.7: Wireshark packets

As result wireshark got 10 answers. Now let's how nmap understand that port state is open or closed.

For example port 110 in state closed.



No.	Time	Source	Destination	Protocol	Length	Info
27	0.272358223	192.168.81.130	192.168.81.129	TCP	60	139 → 36295 [SYN, ACK] Seq=
28	0.272386350	192.168.81.129	192.168.81.130	TCP	54	36295 → 139 [RST] Seq=1 Win
29	0.272450342	192.168.81.130	192.168.81.129	TCP	60	25 → 36295 [SYN, ACK] Seq=0
30	0.272462044	192.168.81.129	192.168.81.130	TCP	54	36295 → 25 [RST] Seq=1 Win=
31	0.272502196	192.168.81.130	192.168.81.129	TCP	60	110 → 36295 [RST, ACK] Seq=

```
▼ Transmission Control Protocol, Src Port: 110, Dst Port: 36295, Seq: 1, Ack: 1, Len: 0
    Source Port: 110
    Destination Port: 36295
    [Stream index: 9]
    [TCP Segment Len: 0]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 1    (relative ack number)
    0101 .... = Header Length: 20 bytes (5)
  ▼ Flags: 0x014 (RST, ACK)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 0... = Push: Not set
    ▸ .... .... .1.. = Reset: Set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
      [TCP Flags: ·····A·R··]
    Window size value: 0
```

Figure 2.8: TCP flag RST

This porst is closed, because in response messege we got RST flag, which means, according to the tcp methodology, that there is no connection.

Port 25(Open) don't have this flag.

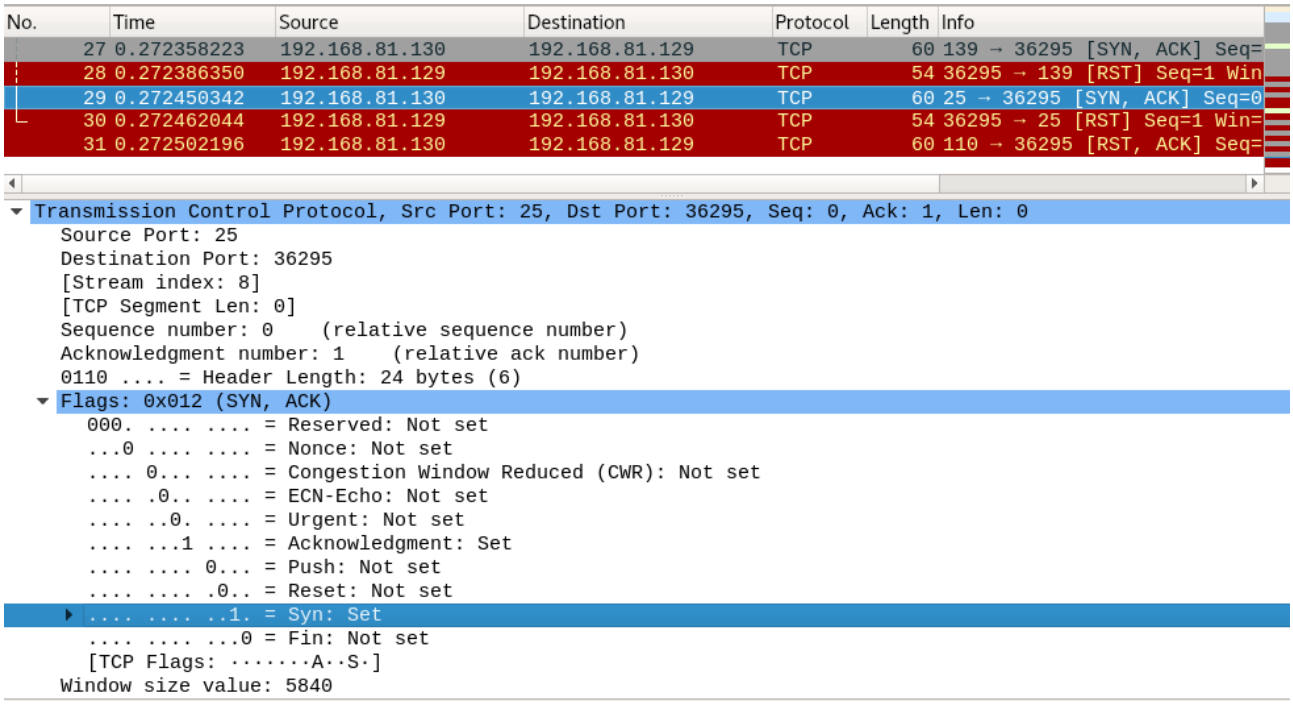


Figure 2.9: RST flag not setted

2.8 Perform VM Metasploitable2 scanning using db_nmap from metasploit framework

To use db_nmap, need to perform steps below:

1. start posgresql server;
2. initialize the database with the msfdb init command;
3. start the console of the msfconsole.

Then use db_nmap, which has same functionality as nmap, but all results will be stored in the database.

```
msf > db_nmap -v -sV 192.168.81.130
[*] Nmap: Starting Nmap 7.60 ( https://nmap.org ) at 2017-11-04
    ↪ 22:24 EDT
[*] Nmap: NSE: Loaded 42 scripts for scanning.
[*] Nmap: Initiating ARP Ping Scan at 22:24
[*] Nmap: Scanning 192.168.81.130 [1 port]
[*] Nmap: Completed ARP Ping Scan at 22:24, 0.26s elapsed (1 total
    ↪ hosts)
[*] Nmap: Initiating Parallel DNS resolution of 1 host. at 22:24
[*] Nmap: Completed Parallel DNS resolution of 1 host. at 22:24,
    ↪ 0.00s elapsed
[*] Nmap: Initiating SYN Stealth Scan at 22:24
[*] Nmap: Scanning 192.168.119.128 [1000 ports]
[*] Nmap: Discovered open port 25/tcp on 192.168.81.130
[*] Nmap: Discovered open port 53/tcp on 192.168.81.130
[*] Nmap: Discovered open port 139/tcp on 192.168.81.130
```

```
14 [*] Nmap: Discovered open port 23/tcp on 192.168.81.130
15 [*] Nmap: Discovered open port 21/tcp on 192.168.81.130
16 [*] Nmap: Discovered open port 445/tcp on 192.168.81.130
17 [*] Nmap: Discovered open port 22/tcp on 192.168.81.130
18 [*] Nmap: Discovered open port 5900/tcp on 192.168.81.130
19 [*] Nmap: Discovered open port 111/tcp on 192.168.81.130
20 [*] Nmap: Discovered open port 3306/tcp on 192.168.81.130
21 [*] Nmap: Discovered open port 80/tcp on 192.168.81.130
22 [*] Nmap: Discovered open port 8180/tcp on 192.168.81.130
23 [*] Nmap: Discovered open port 6667/tcp on 192.168.81.130
24 [*] Nmap: Discovered open port 2049/tcp on 192.168.81.130
25 [*] Nmap: Discovered open port 8009/tcp on 192.168.81.130
26 [*] Nmap: Discovered open port 514/tcp on 192.168.81.130
27 [*] Nmap: Discovered open port 6000/tcp on 192.168.81.130
28 [*] Nmap: Discovered open port 513/tcp on 192.168.81.130
29 [*] Nmap: Discovered open port 1524/tcp on 192.168.81.130
30 [*] Nmap: Discovered open port 512/tcp on 192.168.81.130
31 [*] Nmap: Discovered open port 2121/tcp on 192.168.81.130
32 [*] Nmap: Discovered open port 5432/tcp on 192.168.81.130
33 [*] Nmap: Discovered open port 1099/tcp on 192.168.81.130
34 [*] Nmap: Completed SYN Stealth Scan at 22:24, 9.45s elapsed (1000
     ↪ total ports)
35 [*] Nmap: Initiating Service scan at 22:24
36 [*] Nmap: Scanning 23 services on 192.168.81.130
37 [*] Nmap: Completed Service scan at 22:24, 14.22s elapsed (23
     ↪ services on 1 host)
38 [*] Nmap: NSE: Script scanning 192.168.81.130.
39 [*] Nmap: Initiating NSE at 22:24
40 [*] Nmap: Completed NSE at 22:24, 0.92s elapsed
41 [*] Nmap: Initiating NSE at 22:24
42 [*] Nmap: Completed NSE at 22:24, 0.09s elapsed
43 [*] Nmap: Nmap scan report for 192.168.81.130
44 [*] Nmap: Host is up (0.00089s latency).
45 [*] Nmap: Not shown: 977 closed ports
46 [*] Nmap: PORT STATE SERVICE VERSION
47 [*] Nmap: 21/tcp open ftp vsftpd 2.3.4
48 [*] Nmap: 22/tcp open ssh OpenSSH 4.7p1 Debian 8ubuntu1 (protocol
     ↪ 2.0)
49 [*] Nmap: 23/tcp open telnet Linux telnetd
50 [*] Nmap: 25/tcp open smtp Postfix smtpd
51 [*] Nmap: 53/tcp open domain ISC BIND 9.4.2
52 [*] Nmap: 80/tcp open http Apache httpd 2.2.8 ((Ubuntu) DAV/2)
53 [*] Nmap: 111/tcp open rpcbind 2 (RPC #100000)
54 [*] Nmap: 139/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup
     ↪ : WORKGROUP)
55 [*] Nmap: 445/tcp open netbios-ssn Samba smbd 3.X - 4.X (workgroup
     ↪ : WORKGROUP)
56 [*] Nmap: 512/tcp open exec netkit-rsh rexecd
57 [*] Nmap: 513/tcp open login OpenBSD or Solaris rlogind
58 [*] Nmap: 514/tcp open tcpwrapped
```

```
59 [*] Nmap: 1099/tcp open rmiregistry GNU Classpath grmiregistry
60 [*] Nmap: 1524/tcp open shell Metasploitable root shell
61 [*] Nmap: 2049/tcp open nfs 2-4 (RPC #100003)
62 [*] Nmap: 2121/tcp open ftp ProFTPD 1.3.1
63 [*] Nmap: 3306/tcp open mysql MySQL 5.0.51a-3ubuntu5
64 [*] Nmap: 5432/tcp open postgresql PostgreSQL DB 8.3.0 - 8.3.7
65 [*] Nmap: 5900/tcp open vnc VNC (protocol 3.3)
66 [*] Nmap: 6000/tcp open X11 (access denied)
67 [*] Nmap: 6667/tcp open irc UnrealIRCd
68 [*] Nmap: 8009/tcp open ajp13 Apache Jserv (Protocol v1.3)
69 [*] Nmap: 8180/tcp open http Apache Tomcat/Coyote JSP engine 1.1
70 [*] Nmap: MAC Address: 00:0C:29:88:7b:e8 (VMware)
71 [*] Nmap: Service Info: Hosts: metasploitable.localdomain,
     ↪ localhost, irc.Metasploitable.LAN; OSs: Unix, Linux;
72 CPE: cpe:/o:linux:linux_kernel
73 [*] Nmap: Read data files from: /usr/bin/../share/nmap
74 [*] Nmap: Service detection performed. Please report any incorrect
     ↪ results at https://nmap.org/submit/ .
75 [*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 32.19
     ↪ seconds
76 [*] Nmap: Raw packets sent: 1411 (61.821KB) | Rcvd: 1411 (55.542KB
     ↪ )
```

Listing 2.17: db_nmap output

2.9 Get 5 records from nmap-service-probes and describe them.

Let's analyze a Listing 2.8. It describes the behavior of various services that work with the SIP protocol.

Line **12975** separates one set of rules from another.

Line **12979** contains the probe directive. It is used to indicate which data is sent during the service definition process. The command already analyzed at page 10.

In the line **12980**, the rarity parameter is set to 1. The higher its value (maximum 9), the fewer chances to expect results from this test.

The line **12981** indicates the port number to which data will be sent from the probe directive. In our case, we use array of ports (then they are separated by commas), but in general there can be one single port. Also, if needed to install an encrypted connection over SSL (then the sslports directive is used instead of the ports).

Now let's analyze **12988, 12991, 12993** lines. All theese lines had the **match** directive that tells nmap how to accurately determine the service using the received response to the request sent by the previous probe directive. This directive is used in the case when the received response completely coincides with the template. In this case, the testing of the port is considered complete, and with the help of additional specifiers, nmap builds a report on the name of the application, the version number and additional information received during the test.

The syntax of the match:

match <service> <pattern> [<versioninfo>]

where

- **service** - This is simply the service name that the pattern matches. Examples would be ssh, smtp, http, or snmp. As a special case, you can prefix the service name with ssl/, as in ssl/vmware-auth. In that case, the service would be stored as vmware-auth tunneled by SSL.
- **pattern** - This pattern is used to determine whether the response received matches the service given in the previous parameter. The format is like Perl, with the syntax being m/[regex]/[opts]. The "m" tells Nmap that a match string is beginning. The forward slash (/) is a delimiter, which can be substituted by almost any printable character as long as the second slash is also replaced to match.
- **versioninfo** - The <versioninfo> section actually contains several optional fields. Each field begins with an identifying letter (such as h for "hostname"). Next comes a delimiter character which the signature writer chooses. The preferred delimiter is slash ('/') unless that is used in the field itself. Next comes the field value, followed by the delimiter character.

2.10 Choose one Nmap Script and describe it

Let's describe script named as **unittest.nse**.

```
1  local stdnse = require "stdnse"
2  local unittest = require "unittest"
3
4  description = [[
5  Runs unit tests on all NSE libraries.
6  ]]
7
8  ---
9  -- @args unittest.run Run tests. Causes <code>unittest.testing()</
   ↪ code> to
10 --                   return true.
11 --
12 -- @args unittest.tests Run tests from only these libraries (
   ↪ defaults to all)
13 --
14 -- @usage
15 -- nmap --script unittest --script-args unittest.run
16 --
17 -- @output
18 -- Pre-scan script results:
19 -- | unittest:
20 -- |_ All tests passed
21
22 author = "Daniel Miller"
23
24 license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
25
26 categories = {"safe"}
27
```

```
28
29 prerule = unittest.testing
30
31 action = function()
32   local libs = stdnse.get_script_args("unittest.tests")
33   local result
34   if libs then
35     result = unittest.run_tests(libs)
36   else
37     result = unittest.run_tests()
38   end
39   if #result == 0 then
40     return "All tests passed"
41   else
42     return result
43   end
44 end
```

Listing 2.18: unittest.nse

In first two lines, variables **stdnse** and **unittest** are declared for later use in the script.

In line 4 describes the purpose of this module - run unit tests on all NSE libraries.

In line's from 9 to 20 we see comment's, what arguments must be passed, example of usage and output of result.

In the line 22 the author is indicated, in the 24th line the type of license.

Line 26 defines the categories of the script. There are 10 categories in total. The safe category says that the script is safe, and his work will not lead to incorrect operation or stopping of any service.

In the remaining lines, the main logic of the script is presented. In the parameter, you can specify libraries and only for them will be held a unittest's. If you do not specify a parameter, the unittests will be run for all libraries. To perform tests, the **unittest.run_tests()** function is calling, and if its return result 0 then this means that all tests were successful.

Conclusion

As result in this report i learned how to use nmap tool - a powerful tool for researching a new network or studying the effects of external penetration.

Nmap features include:

- Host discovery – Identifying hosts on a network. For example, listing the hosts that respond to TCP and/or ICMP requests or have a particular port open.
- Port scanning – Enumerating the open ports on target hosts.
- Version detection – Interrogating network services on remote devices to determine application name and version number.
- OS detection – Determining the operating system and hardware characteristics of network devices.

Also results can be saved in external XML file or into database, using db_nmap.