

Лабораторная работа №4.
Утилита Nmap

Кенть Никита

25 апреля 2016 г.

Оглавление

0.1	Цель работы	2
0.2	Ход работы	2
0.3	Поиск активных хостов	2
0.4	Определение открытых портов	3
0.5	Определение версии сервисов	3
0.6	Изучение nmap-services, nmap-os-db, nmap-service-probes	3
0.6.1	nmap-services	3
0.6.2	nmap-os-db	4
0.6.3	nmap-service-probes	4
0.7	Добавление собственной сигнатуры в файл nmap-service-probes	4
0.8	Сохранение вывода утилиты nmap в формате XML	6
0.9	Исследование работы утилиты nmap при помощи wireshark	7
0.10	Использование db_nmap из состава metasploit-framework	7
0.11	Анализ записей из файла nmap-service-probes	7
0.12	Описание работы скрипта из Nmap	8
0.13	Вывод	11

0.1 Цель работы

Изучить возможности утилиты Nmap на различных примерах.

0.2 Ход работы

Для выполнения данной лабораторной работы нам понадобятся две виртуальные машины: Metasploitable2 и Kali Linux, имеющие адреса 192.168.32.128 и 192.168.32.129 соответственно. Конфигурации представлены на рис. 1 и рис. 2

```
nsfadmin@metasploitable:~$ cd vulnerable/
nsfadmin@metasploitable:~/vulnerable$ ls
mysql-ssl samba tikiwiki twiki20030201
nsfadmin@metasploitable:~/vulnerable$ cd ..
nsfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:48:ea:b0
          inet addr:192.168.32.128  Bcast:192.168.32.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe48:eab0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:521 errors:0 dropped:0 overruns:0 frame:0
          TX packets:89 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:49433 (48.2 KB)  TX bytes:11294 (11.0 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:65536  Metric:1
          RX packets:213 errors:0 dropped:0 overruns:0 frame:0
          TX packets:213 errors:0 dropped:0 overruns:0 carrier:0
```

Рис. 1: Конфигурация Metasploitable2

```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.32.129  netmask 255.255.255.0  broadcast 192.168.32.255
        inet6 fe80::20c:29ff:fe48:571  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:e8:05:71  txqueuelen 1000  (Ethernet)
        RX packets 380  bytes 36570 (35.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 28  bytes 2690 (2.6 KiB)
        TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
        device interrupt 19  base 0x2000

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 0  (Local Loopback)
        RX packets 20  bytes 1200 (1.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 20  bytes 1200 (1.1 KiB)
        TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

Рис. 2: Конфигурация Kali Linux

0.3 Поиск активных хостов

Для поиска активных хостов запустим утилиту с флагом -sP и укажем адрес подсети 192.168.32.* Вывод утилиты приведет на рис. 3

```

root@kali:~# nmap -sP 192.168.32.*

Starting Nmap 7.01 ( https://nmap.org ) at 2016-04-24 18:24 UTC
Nmap scan report for 192.168.32.1
Host is up (0.0014s latency).
MAC Address: 00:50:56:C0:00:00 (VMware)
Nmap scan report for 192.168.32.2
Host is up (0.000095s latency).
MAC Address: 00:50:56:FB:CC:83 (VMware)
Nmap scan report for 192.168.32.128
Host is up (0.00034s latency).
MAC Address: 00:0C:29:48:EA:B0 (VMware)
Nmap scan report for 192.168.32.254
Host is up (0.00019s latency).
MAC Address: 00:50:56:FE:C8:40 (VMware)
Nmap scan report for 192.168.32.129
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 1.73 seconds

```

Рис. 3: Поиск активных хостов

0.4 Определение открытых портов

Определим открытые порты, просто передав адрес хоста. (Рис. 4)

```

root@kali:~# nmap 192.168.32.128

Starting Nmap 7.01 ( https://nmap.org ) at 2016-04-24 18:28 UTC
Nmap scan report for 192.168.32.128
Host is up (0.00043s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  x11

```

Рис. 4: Поиск открытых портов

0.5 Определение версии сервисов

Определим версии сервисов, командой с ключем -sV (Рис. 5)

0.6 Изучение nmap-services, nmap-os-db, nmap-service-probes

0.6.1 nmap-services

Файл nmap-services содержит в себе описание назначения стандартных портов. Файл имеет структуру: имя_сервиса, номер_порта/название_протокола, частота, комментарий. (Рис. 6)

```

root@kali:/usr/share/nmap# nmap -sV 192.168.32.128

Starting Nmap 7.01 ( https://nmap.org ) at 2016-04-24 19:19 UTC
Nmap scan report for 192.168.32.128
Host is up (0.00035s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd

```

Рис. 5: Определение версии сервисов

```

msp      18/udp  0.000610      # Message Send Protocol
chargen  19/tcp  0.002559      # ttytst source Character Generator
chargen  19/udp  0.015865      # ttytst source Character Generator
ftp-data 20/sctp 0.000000      # File Transfer [Default Data]
ftp-data 20/tcp  0.001079      # File Transfer [Default Data]
ftp-data 20/udp  0.001878      # File Transfer [Default Data]
ftp      21/sctp 0.000000      # File Transfer [Control]
ftp      21/tcp  0.197667      # File Transfer [Control]
ftp      21/udp  0.004844      # File Transfer [Control]
ssh      22/sctp 0.000000      # Secure Shell Login
ssh      22/tcp  0.182286      # Secure Shell Login
ssh      22/udp  0.003905      # Secure Shell Login
telnet   23/tcp  0.221265      #
telnet   23/udp  0.006211      #
priv-mail 24/tcp  0.001154      # any private mail system
priv-mail 24/udp  0.000329      # any private mail system
smtp     25/tcp  0.131314      # Simple Mail Transfer
smtp     25/udp  0.001285      # Simple Mail Transfer

```

Рис. 6: nmap-services

0.6.2 nmap-os-db

Файл содержит сигнатуры ответов различных операционных систем при сканировании. Это Пример файла на рис. 7

0.6.3 nmap-service-probes

Файл содержит сигнатуры для определения сервисов, прослушивающих различные порты. Например, SMTP - почтовый сервер. Данные о сервисах задаются при помощи нескольких директив:

- Exclude <port specification>
- Probe <protocol> <probenamе> <probestring>
- match <service> <pattern> [<versioninfo>]

0.7 Добавление собственной сигнатуры в файл nmap-service-probes

Для этого напишем небольшой сервер, который будем идентифицировать утилитой nmap

```
# Alcatel-Lucent OmniPCX Enterprise
# OmniPCX Enterprise R8.0.1-g1.503-12-a-n1-c80s1 (Linux 2.4.17-11-dhs3)
Fingerprint Alcatel-Lucent OmniPCX Enterprise PBX (Linux 2.4.17)
Class Alcatel-Lucent | embedded || PBX
CPE cpe:/h:alcatel-lucent:omnipcx
Class Linux | Linux | 2.4.X | PBX
CPE cpe:/o:linux:linux_kernel:2.4.17 auto
SEQ(SP=C5-D3%GCD=1-6%ISR=CC-D6%TI=Z%II=I%TS=U)
OPS(01=M5B4NNSNW0%02=M5B4NNSNW0%03=M5B4NW0%04=M5B4NNSNW0%05=M5B4NNSNW0%06=M5B4NNS)
WIN(W1=16D0%W2=16D0%W3=16D0%W4=16D0%W5=16D0%W6=16D0)
ECN(R=Y%DF=Y%T=3B-45%TG=40%W=16D0%0=M5B4NNSNW0%CC=N%Q=)
T1(R=Y%DF=Y%T=3B-45%TG=40%S=0%A=S+F=AS%RD=0%Q=)
T2(R=N)
T3(R=Y%DF=Y%T=3B-45%TG=40%W=16D0%S=0%A=S+F=AS%0=M5B4NNSNW0%RD=0%Q=)
T4(R=Y%DF=Y%T=FA-104%TG=FF%W=0%S=A%Z=F=R%0=%RD=0%Q=)
T5(R=Y%DF=Y%T=FA-104%TG=FF%W=0%S=Z%A=S+F=AR%0=%RD=0%Q=)
T6(R=Y%DF=Y%T=FA-104%TG=FF%W=0%S=A%Z=F=R%0=%RD=0%Q=)
```

Рис. 7: nmap-os-db

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>

int main()
{

char str[100];
char *rec_srt="\x48\x65\x6c\x6c\x6f";

int listen_fd, comm_fd;

struct sockaddr_in servaddr;

listen_fd = socket(AF_INET, SOCK_STREAM, 0);

bzero( &servaddr, sizeof(servaddr));

servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
servaddr.sin_port = htons(22000);

bind(listen_fd, (struct sockaddr *) &servaddr, sizeof(servaddr));

listen(listen_fd, 10);

comm_fd = accept(listen_fd, (struct sockaddr*) NULL, NULL);

while(1)
{

bzero( str, 100);
```

```

read(comm_fd,str,100);

printf("echo back - %s",str);

write(comm_fd, rec_srt, strlen(rec_srt)+1);

}
}

```

Для определения данного сервера, в файл nmap-service-probes добавим:

```

Probe TCP Server q|HelloServer|
rarity 1
ports 22000
match testServer m|\x48\x65\x6c\x6c\x6f| v/1.0/

```

Запустим сервис на машине Metasploitable2 и проверим с помощью nmap

```
root@kali:~/test# nmap 192.168.32.128 -p 22000 -sV
```

```

Starting Nmap 7.01 ( https://nmap.org ) at 2016-03-20 19:33 EDT
Nmap scan report for 192.168.202.2
Host is up (0.00048s latency).
PORT      STATE SERVICE      VERSION
22000/tcp  open  Server       1.0

```

0.8 Сохранение вывода утилиты nmap в формате XML

Для сохранения вывода утилиты nmap в формате XML необходимо при запуске указать ключ оX". Часть содержания файла представлена на (Рис.8)

```

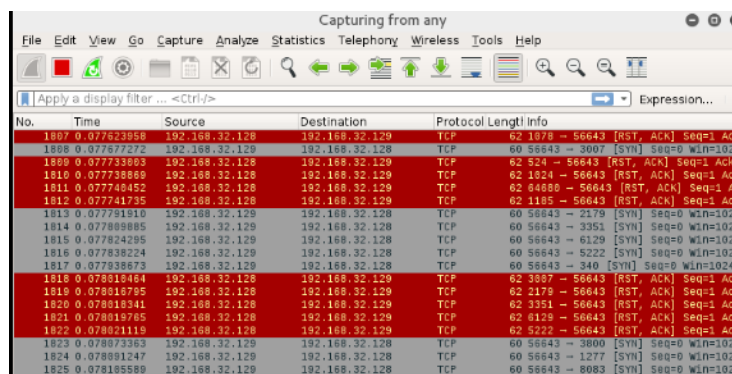
root@kali:~/usr/share/nmap# cat out.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE nmaprun>
<?xml-stylesheet href="file:///usr/bin/.../share/nmap/nmap.xml" type="text/xsl"?>
<!-- Nmap 7.01 scan initiated Sun Apr 24 20:00:58 2016 as: nmap -T4 -A -p 1-5000 -oX -
192.168.32.128 -->
<nmaprun scanner="nmap" args="nmap -T4 -A -p 1-5000 -oX - 192.168.32.128" start="146152
8658" startstr="Sun Apr 24 20:00:58 2016" version="7.01" xmloutputversion="1.04">
<scaninfo type="syn" protocol="tcp" numservices="5000" services="1-5000"/>
<verbose level="0"/>
<debugging level="0"/>
<host starttime="1461528059" endtime="1461528084"><status state="up" reason="arp-respon
se" reason_ttl="0"/>
<address addr="192.168.32.128" addrtype="ipv4"/>
<address addr="00:0C:29:48:EA:B0" addrtype="mac" vendor="VMware"/>
<hostnames>
</hostnames>
<ports><extrareasons reason="resets" count="4982"/>
</extrareasons>

```

Рис. 8: Определение своего сервиса

0.9 Исследование работы утилиты nmap при помощи wireshark

Запустим Wireshark и просканируем хост 192.168.32.128. Результаты на Рис. 9)



No.	Time	Source	Destination	Protocol	Length	Info
1887	0.077623958	192.168.32.128	192.168.32.129	TCP	62	1878 → 56643 [RST, ACK] Seq=1 ACK=56643
1888	0.077672372	192.168.32.129	192.168.32.128	TCP	60	56643 → 2007 [SYN] Seq=0 Win=1024
1889	0.077733603	192.168.32.128	192.168.32.129	TCP	62	531 → 56643 [RST, ACK] Seq=1 ACK=56643
1890	0.077738869	192.168.32.128	192.168.32.129	TCP	62	1824 → 56643 [RST, ACK] Seq=1 ACK=56643
1891	0.077748452	192.168.32.128	192.168.32.129	TCP	62	64880 → 56643 [RST, ACK] Seq=1 ACK=56643
1892	0.077741735	192.168.32.128	192.168.32.129	TCP	62	1185 → 56643 [RST, ACK] Seq=1 ACK=56643
1893	0.077791910	192.168.32.129	192.168.32.128	TCP	60	56643 → 2179 [SYN] Seq=0 Win=1024
1894	0.077869885	192.168.32.129	192.168.32.128	TCP	60	56643 → 3351 [SYN] Seq=0 Win=1024
1895	0.077824295	192.168.32.129	192.168.32.128	TCP	60	56643 → 6129 [SYN] Seq=0 Win=1024
1896	0.077838224	192.168.32.129	192.168.32.128	TCP	60	56643 → 5222 [SYN] Seq=0 Win=1024
1897	0.077838673	192.168.32.129	192.168.32.128	TCP	60	56643 → 340 [SYN] Seq=0 Win=1024
1898	0.078018464	192.168.32.128	192.168.32.129	TCP	62	3807 → 56643 [RST, ACK] Seq=1 ACK=56643
1899	0.078018795	192.168.32.128	192.168.32.129	TCP	62	2179 → 56643 [RST, ACK] Seq=1 ACK=56643
1900	0.078018341	192.168.32.128	192.168.32.129	TCP	62	3351 → 56643 [RST, ACK] Seq=1 ACK=56643
1901	0.078019765	192.168.32.128	192.168.32.129	TCP	62	6129 → 56643 [RST, ACK] Seq=1 ACK=56643
1902	0.078021119	192.168.32.128	192.168.32.129	TCP	62	5222 → 56643 [RST, ACK] Seq=1 ACK=56643
1903	0.078073363	192.168.32.129	192.168.32.128	TCP	60	56643 → 3800 [SYN] Seq=0 Win=1024
1904	0.078091247	192.168.32.129	192.168.32.128	TCP	60	56643 → 1277 [SYN] Seq=0 Win=1024
1905	0.078185589	192.168.32.129	192.168.32.128	TCP	60	56643 → 8083 [SYN] Seq=0 Win=1024

Рис. 9: WireShark

0.10 Использование db_nmap из состава metasploit-framework

```
root@kali:~# service postgresql start
root@kali:~# msfdb init
Creating database user 'msf'
Enter password for new role:
Enter it again:
Creating databases 'msf' and 'msf_test'
Creating configuration file in /usr/share/metasploit-framework/config/database.yml
Creating initial database schema
```

После чего необходимо запустить msfconsole и можно использовать любую из команд, описанных выше, но вместо nmap можно использовать db_nmap. Результаты работы будут записаны в базу данных, тем самым обеспечивая экономию времени при сканировании портов.

0.11 Анализ записей из файла nmap-service-probes

Рассмотрим строки, указанные на рис. 10)

Директива Probe указывает, какое сообщение необходимо отправить для идентификации сервиса. В данном случае, сервис - RTSPRequest, протокол - TCP, отправляется следующая строка:

```
OPTIONS / RTSP/1.0\r\n\r\n
```



```

-----
Probe TCP RTSPRequest q|OPTIONS / RTSP/1.0\r\n\r\n|
rarity 5
ports 80,554,3052,3372,5000,7070,8080,10000
fallback GetRequest

match raop m|^RTSP/1\..0 401 Unauthorized\r\nWWW-Authenticate: Digest realm=\"raop\",
nonce=\"[0-9A-F]{40}\"\\r\nContent-Length: 0\\r\n\r\n$| p/Remote Audio Output Protocol/ i/
Rogue Amoeba Airfoil speakers/ d/media device/
-----

```

Рис. 10: Пример nmap-service-probes

rarity указывает частоту, с которой от сервиса можно ожидать возвращения корректных результатов. В данном случае - 5.

Директивы ports и sslports указывают на порты, используемые данным сервисом.

fallback указывает на то, какие Probe необходимо использовать при отсутствии совпадений в текущей секции Probe.

Директива match необходима при распознавании сервиса на основе ответов на строку, отправленную предыдущей директивой Probe.

0.12 Описание работы скрипта из Nmap

Скриптовый движок Nmap (NSE) это одна из наиболее мощных и гибких возможностей Nmap. Он позволяет пользователям писать (и делиться ими) скрипты для автоматизации широкого круга сетевых задач. Рассмотрим скрипт dhcp-discover, который Обнаруживает сервера DHCP в сети. Запрос отправляется с UDP-порта 67, ответ принимается на порт 68.

```
nmap -sU -p 67 --script=dhcp-discover <target>
```

Возможный вывод скрипта:

```

Interesting ports on 192.168.1.1:
PORT      STATE SERVICE
67/udp    open  dhcps
| dhcp-discover:
|   DHCP Message Type: DHCPACK
|   Server Identifier: 192.168.1.1
|   IP Address Lease Time: 1 day, 0:00:00
|   Subnet Mask: 255.255.255.0
|   Router: 192.168.1.1
|_  Domain Name Server: 208.81.7.10, 208.81.7.14

```

Приведем листинг скрипта:

```

author = "Ron Bowes"

license = "Same as Nmap--See https://nmap.org/book/man-legal.html"

categories = {"discovery", "safe"}

```

```

-- We want to run against a specific host if UDP/67 is open
function portrule(host, port)
    if nmap.address_family() ~= 'inet' then
        stdnse.debug1("is IPv4 compatible only.")
        return false
    end

    return shortport.portnumber(67, "udp")(host, port)
end

local function go(host, port)

    -- Build and send a DHCP request using the specified request type, or DHCPINFORM
    local requests = tonumber(nmap.registry.args.requests or 1)
    local results = {}
    for i = 1, requests, 1 do
        -- Decide which type of request to make
        local request_type = dhcp.request_types[nmap.registry.args.dhcptype or "DHCPINFORM"]
        if(request_type == nil) then
            return false, "Valid request types: " .. stdnse.strjoin(", ", dhcp.request_types_str)
        end

        -- Generate the MAC address, if it's random
        local mac_addr = host.mac_addr_src
        if(nmap.registry.args.randomize_mac == 'true' or nmap.registry.args.randomize_mac == ' ')
            stdnse.debug2("Generating a random MAC address")
            mac_addr = {}
            for j=1, 6, 1 do
                mac_addr[j] = string.char(math.random(1, 255))
            end
            mac_addr = table.concat(mac_addr)
        end

        local iface, err = nmap.get_interface_info(host.interface)
        if ( not(iface) or not(iface.address) ) then
            return false, "Couldn't determine local ip for interface: " .. host.interface
        end

        local status, result = dhcp.make_request(host.ip, request_type, iface.address, mac_addr)
        if( not(status) ) then
            stdnse.debug1("Couldn't send DHCP request: %s", result)
            return false, result
        end

        table.insert(results, result)
    end

    -- Done!
    return true, results
end

```

```

local commasep = {
  __tostring = function (t)
    return table.concat(t, ", ")
  end
}

action = function(host, port)
  local status, results = go(host, port)

  if(not(status)) then
    return stdnse.format_output(false, results)
  end

  if(not(results)) then
    return nil
  end

  -- Set the port state to open
  if(host) then
    nmap.set_port_state(host, port, "open")
  end

  local response = stdnse.output_table()

  -- Display the results
  for i, result in ipairs(results) do
    local result_table = stdnse.output_table()

    if ( nmap.registry.args.dhcptype and
        "DHCPINFORM" ~= nmap.registry.args.dhcptype ) then
      result_table["IP Offered"] = result.yiaddr_str
    end
    for _, v in ipairs(result.options) do
      if(type(v.value) == 'table') then
        setmetatable(v.value, commasep)
      end
      result_table[ v.name ] = v.value
    end

    if(#results == 1) then
      response = result_table
    else
      response[string.format("Response %d of %d", i, #results)] = result_table
    end
  end

  return response
end

```

0.13 Вывод

В ходе лабораторной работы была изучена утилита Nmap. Попробовали некоторые её возможности, такие как: поиск активных хостов, открытых портов, версий сервисов и др. Изучены файлы `nmap-services`, `nmap-os-db`, `nmap-service-probe`. Идентифицировали собственный сервис.