Лабораторная работа №4. Утилита для исследования сети и сканер портов Nmap

Никитина Анна

3 апреля 2016 г.

Оглавление

1 Цель работы 2
2 Ход работы 2
2.1 Поиск активных хостов 2
2.2 Определение открытых портов 2
2.3 Определение версии сервисов 3
2.4 Изучение файлов nmap-services, nmap-os-db, nmap-service-probes 4
2.5 Добавление собственной сигнатуры в файл nmap-service-probes 5
2.6 Сохранение вывода утилиты в формате XML 7
2.7 Исследование с использованием утилиты WireShark . . 8
2.8 Использвоание db_nmap из состава metasploit-framewor 9
2.9 Анализ записей из файла nmap-service-probes 10
2.10 Описание работы скрипта из Nmap 10
3 Вывод 12

1 Цель работы

Изучить возможности утилиты Nmap на различных примерах.

2 Ход работы

Для выполнения лабораторной работы понадобятся две виртуальные машины. Metasploitable2 -виртуальная машина, содержащая различные уязвимости. Kali Linux - виртуальная машина, способная сканировать и находить уязвимости. Вышеуказанные машины необходимо объединить в общую сеть. Определим IP-адрес на машине Kali Linux

```
root@kali:~# ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.102  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::a00:27ff:fe9b:2f3f  prefixlen 64  scopeid 0x20<link>
```

И на машине Metasploitable2

```
root@kali:~# ifconfig -a
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.103  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::a00:27ff:fe9b:2f3f  prefixlen 64  scopeid 0x20<link>
```

2.1 Поиск активных хостов

Для поиска активных хостов указываем флаг -sP и адрес подсети 192.168.0.*

```
root@kali:~# nmap -sP 192.168.0.*

Starting Nmap 7.01 ( https://nmap.org ) at 2016-03-27 14:48 EDT
Nmap scan report for 192.168.0.1
Host is up (0.0057s latency).
MAC Address: 1C:7E:E5:3E:AF:10 (D-Link International)
Nmap scan report for 192.168.0.100
Host is up (0.00027s latency).
MAC Address: E0:B9:A5:1C:8A:33 (AzureWave Technology)
Nmap scan report for 192.168.0.103
Host is up (0.00081s latency).
MAC Address: 08:00:27:94:82:93 (Oracle VirtualBox virtual NIC)
Nmap scan report for 192.168.0.102
Host is up.
Nmap done: 256 IP addresses (4 hosts up) scanned in 28.20 seconds
```

2.2 Определение открытых портов

Для определения открытых портов утилите достаточно передать адрес хоста.

```
root@kali:~# nmap  192.168.0.103
```

```
Starting Nmap 7.01 ( https://nmap.org ) at 2016-03-27 15:23 EDT
Nmap scan report for 192.168.0.103
Host is up (0.00045s 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  shell
1099/tcp open  rmiregistry
1524/tcp open  ingreslock
2049/tcp open  nfs
2121/tcp open  ccproxy-ftp
3306/tcp open  mysql
5432/tcp open  postgresql
5900/tcp open  vnc
6000/tcp open  X11
6667/tcp open  irc
8009/tcp open  ajp13
8180/tcp open  unknown
MAC Address: 08:00:27:94:82:93 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.33 seconds
```

Также существуют флаги -sT для определения открытх TCP-портов, -sU для определения UDP-портов.

2.3 Определение версии сервисов

Для определения версий сервисов необходимо указать флаг -sV.

```
root@kali:~# nmap -sV 192.168.0.103

Starting Nmap 7.01 ( https://nmap.org ) at 2016-03-27 15:32 EDT
Nmap scan report for 192.168.0.103
Host is up (0.00024s 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
513/tcp  open  login?
514/tcp  open  tcpwrapped
1099/tcp open  rmiregistry GNU Classpath grmiregistry
1524/tcp open  shell       Metasploitable root shell
2049/tcp open  nfs         2-4 (RPC #100003)
2121/tcp open  ftp         ProFTPD 1.3.1
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open  vnc         VNC (protocol 3.3)
6000/tcp open  X11         (access denied)
6667/tcp open  irc         Unreal ircd
8009/tcp open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
MAC Address: 08:00:27:94:82:93 (Oracle VirtualBox virtual NIC)
Service Info: Hosts:  metasploitable.localdomain, localhost, irc.Metasploitable.LAN; OSs:

Service detection performed. Please report any incorrect results at https://nmap.org/submi
Nmap done: 1 IP address (1 host up) scanned in 31.98 seconds
```

2.4 Изучение файлов nmap-services, nmap-os-db, nmap-service-probes

Файл **nmap-services** представляет собой список, каждая строка которого состоит из имени порта, соответствующего ему номера и протокола. Дпалее число, представляющее собой вероятность того, что порт является открытым. Большинство строк имеют также комментарии.

```
mit-ml-dev 83/tcp 0.000539 # MIT ML Device
mit-ml-dev 83/udp 0.001203 # MIT ML Device
ctf 84/tcp 0.000276 # Common Trace Facility
ctf 84/udp 0.000610 # Common Trace Facility
mit-ml-dev 85/tcp 0.000690 # MIT ML Device
mit-ml-dev 85/udp 0.000610 # MIT ML Device
mfcobol 86/tcp 0.000138 # Micro Focus Cobol
mfcobol 86/udp 0.000824 # Micro Focus Cobol
```

Файл **nmap-os-db** содержит примеры ответов различных операционных систем при сканировании Nmap. Он разделен на блоки, называемые Fingerprint, каждый из которых содержат название операционной системы, ее общую классификацию, и данные от нее.

```
# 3Com OfficeConnect 3CR858-91 Software version V1.13-168 (Mar 21 2008 11:57:49)
Fingerprint 3Com OfficeConnect 3CR858-91 router
Class 3Com | embedded || router
CPE cpe:/h:3com:officeconnect_3cr858-91 auto
SEQ(SP=0-5%GCD=B|16|21|2C|37%ISR=30-3A%TI=I%CI=I%II=I%SS=S%TS=U)
```

```
OPS(O1=M578%O2=M578%O3=M280%O4=M578%O5=M218%O6=M109)
WIN(W1=1770%W2=1770%W3=1770%W4=1770%W5=1770%W6=1770)
ECN(R=Y%DF=Y%T=3B-45%TG=40%W=1770%O=M578%CC=N%Q=)
T1(R=Y%DF=Y%T=3B-45%TG=40%S=O%A=O|S+%F=AS%RD=0%Q=)
T2(R=N)
T3(R=N)
```

Файл **nmap-service-probes** представляет из себя список, содержащий сигнатуры для определения различных сервисов. Используются несколько директив, некоторые из них:

- Probe <protocol> <probename> <probestring>
  Директива Probe содержит строку, необходимую для отправки при распознавании сервиса.
- match <service> <pattern> [<versioninfo>]
  Директива match необходима при распознавании сервиса на основе ответов на строку, отправленную предыдущей директивой Probe.
- ports <portlist>
  Директива ports содержит порты сервиса.
- rarity <value between 1 and 9>
  Директива rarity указывает частоту, с которой от сервиса можно ожидать возвращения корректных результатов.

Пример из файла nmap-service-probes

```
Probe TCP Radmin q|\x01\x00\x00\x00\x01\x00\x00\x00\x08\x08|
ports 4899,9001
rarity 8
match fcgiwrap m|^\x01\x0b\0\0\0\x08\0\0\0\0\0\0\0\0\0$| p/fcgiwrap/
match radmin m|^\x01\x00\x00\x00\x25\x09\x00\x01\x10\x08\x01\x00\x09\x08| p/Famatech Radmi
match radmin m|^\x01\x00\x00\x00\x25\x0a\x00\x01\x10\x08\x01\x00\x0a\x08| p/Famatech Radmi
match radmin m|^\x01\x00\x00\x00\x25\x00\x00\x02\x12\x08\x02\x00\x00\x0a| p/Famatech Radmi
match radmin m|^\x01\x00\x00\x00\x25\x71\x00\x02\x12\x08\x02\x00\x71\x0a| p/Famatech Radmi
```

2.5 Добавление собственной сигнатуры в файл nmap-service-probes

Создадим простейший сервер, который будет работать на 8089 порту.

```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#define DEF_PORT 8089

int main(int argc, char** argv)
{
```

```
char str[100];
char *sendStr="\x48\x65\x6c\x6c\x6f";
struct sockaddr_in listenerInfo;
listenerInfo.sin_family = AF_INET;
listenerInfo.sin_port = htons(DEF_PORT);
listenerInfo.sin_addr.s_addr = htonl(INADDR_ANY);
int listener = socket(AF_INET,SOCK_STREAM,0);
if(listener < 0) {
perror("Can't create socket to listen: ");
exit(1);
}
int res = bind(listener,(struct sockaddr *) &listenerInfo,sizeof(listenerInfo));
if(res < 0) {
perror("Can't bind socket");
exit(1);
}
res = listen(listener,5);
if(res) {
perror("Erro while listening:");
exit(1);
}
int client = accept(listener,NULL,NULL);
while(1)
{
bzero( str, 100);
recv(client,str, 100, 0);
printf("Message from client - %s",str);
send(client, sendStr, (int)strlen(sendStr), 0);
}
return 0;
}
```

Добавим в nmap-service-probes строки о созданном сервисе.
Для распознавания сервиса отправлется строка 'HelloServer!', в ответ ожидается строка 'Hello'. Порт сервиса 8089.

```
Probe TCP MyServer q|HelloServer!|
rarity 1
ports 8089
match MyServer m|\x48\x65\x6c\x6c\x6f| v/0.1/
```

Запустим сервис на машине Metasploitable2 И проверим с помощью nmap наличие сервиса, указав IP-адрес машины 192.168.0.103.

```
root@kali:~# nmap 192.168.0.103 -sV
Starting Nmap 7.01 ( https://nmap.org ) at 2016-04-02 05:57 EDT
Nmap scan report for 192.168.0.103
Host is up (0.00058s latency).
Not shown: 977 closed ports
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
```

```
.....
5900/tcp open  vnc         VNC (protocol 3.3)
6000/tcp open  X11         (access denied)
6667/tcp open  irc         Unreal ircd
8009/tcp open  ajp13       Apache Jserv (Protocol v1.3)
8180/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
8089/tcp open  MyServer    0.1
MAC Address: 08:00:27:94:82:93 (Oracle VirtualBox virtual NIC)
Service Info: Hosts:  metasploitable.localdomain, localhost, irc.Metasploitable.LAN; OSs:

Service detection performed. Please report any incorrect results at https://nmap.org/submi
Nmap done: 1 IP address (1 host up) scanned in 29.93 seconds
```

Созданный сервис успешно распознан, названия и версия при этом указываются.

2.6 Сохранение вывода утилиты в формате XML

Для сохранения вывода утилиты в файле xml необходимо указать флаг -oX, после которого имя файла (в примере ниже example.xml).

```
root@kali:~# nmap 192.168.0.103 -oX example.xml
```

Просмотрим файл example.xml. В него была записана информация о сервисах хоста 192.168.0.103 в формате xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE nmaprun>
<?xml-stylesheet href="file:///usr/bin/../share/nmap/nmap.xsl" type="text/xsl"?>
<!-- Nmap 7.01 scan initiated Sat Apr  2 07:42:49 2016 as: nmap -oX xml.txt 192.168.0.103
<nmaprun scanner="nmap" args="nmap -oX xml.txt 192.168.0.103" start="1459597369" startstr=
<scaninfo type="syn" protocol="tcp" numservices="1000" services="1,3-4,6-7,9,13,17,19-26,3
<verbose level="0"/>
<debugging level="0"/>
<host starttime="1459597369" endtime="1459597382"><status state="up" reason="arp-response"
<address addr="192.168.0.103" addrtype="ipv4"/>
<address addr="08:00:27:94:82:93" addrtype="mac" vendor="Oracle VirtualBox virtual NIC"/>
<hostnames>
</hostnames>
<ports><extraports state="closed" count="977">
<extrareasons reason="resets" count="977"/>
</extraports>
<port protocol="tcp" portid="21"><state state="open" reason="syn-ack"
reason_ttl="64"/><service name="ftp" method="table" conf="3"/></port>
<port protocol="tcp" portid="22"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="ssh" method="table" conf="3"/></port>
<port protocol="tcp" portid="23"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="telnet" method="table" conf="3"/></port>
<port protocol="tcp" portid="25"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="smtp" method="table" conf="3"/></port>
<port protocol="tcp" portid="53"><state state="open"
```

```
 reason="syn-ack" reason_ttl="64"/><service name="domain" method="table" conf="3"/></port>
<port protocol="tcp" portid="80"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="http" method="table" conf="3"/></port>
...
<port protocol="tcp" portid="514"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="shell" method="table" conf="3"/></port>
<port protocol="tcp" portid="1099"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="rmiregistry" method="table" conf="3"/></
<port protocol="tcp" portid="1524"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="ingreslock" method="table" conf="3"/></p
<port protocol="tcp" portid="2049"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="nfs" method="table" conf="3"/></port>
<port protocol="tcp" portid="2121"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="ccproxy-ftp" method="table" conf="3"/></
<port protocol="tcp" portid="3306"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="mysql" method="table" conf="3"/></port>
<port protocol="tcp" portid="5432"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="postgresql" method="table" conf="3"/></p
<port protocol="tcp" portid="5900"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="vnc" method="table" conf="3"/></port>
<port protocol="tcp" portid="6000"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="X11" method="table" conf="3"/></port>
<port protocol="tcp" portid="6667"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="irc" method="table" conf="3"/></port>
<port protocol="tcp" portid="8009"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="ajp13" method="table" conf="3"/></port>
<port protocol="tcp" portid="8180"><state state="open"
 reason="syn-ack" reason_ttl="64"/><service name="unknown" method="table" conf="3"/></port
</ports>
<times srtt="448" rttvar="260" to="100000"/>
</host>
<runstats><finished time="1459597382" timestr="Sat Apr  2 07:43:02 2016" elapsed="13.42" s
</runstats>
</nmaprun>
```

2.7 Исследование с использованием утилиты WireShark

Запустим утилиту WireShark, после чего просканируем хост 192.168.0.103.

```
root@kali:~# nmap 192.168.0.103
```

Как показано на рисунке 1. Изначально nmap посылат на существующие порты TCP-пакеты с установленным флагом SYN, что означает установление соединения. Если при этом сканируемый порт отправляет ответ с установленными флагами [RST, ACK], значит соединение невозможно - порт закрыт.

Если после отправки TCP-пакета с флагом SYN сканируемый порт отправляет ответ также с установленным флагом SYN, это означает, что заданый порт открыт. Пример показан на рисунке 2 .

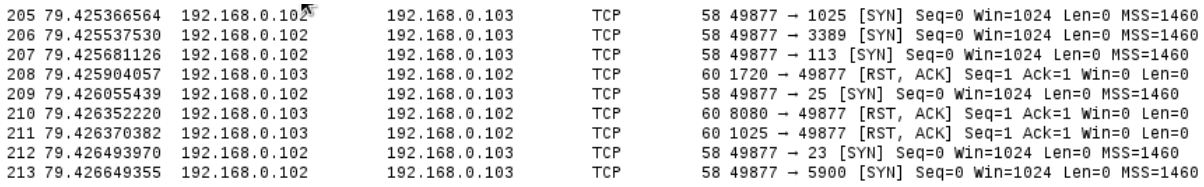

Рис. 1: Вывод утилиты WireShark.

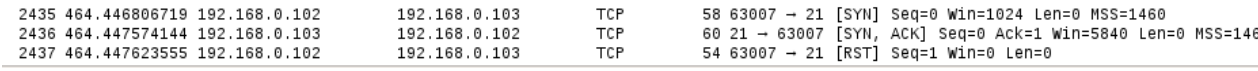

Рис. 2: Вывод утилиты WireShark.

2.8 Использвоание db_nmap из состава metasploit-framewor

Первым шагом надо запустить postgresql сервер. После чего создать и инициализировать базу данных командой msfdb init. И запустить консоль msfconsole.

```
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
root@kali:~# msfconsole
```

После чего доступна команда db_nmap, которая имеет ту же функциональность, что и команда nmap. Однако в этом случае результаты сканирования будут автоматически сохранены в базе данных.

```
msf > db_nmap 192.168.0.103 -p 21
[*] Nmap: Starting Nmap 7.01 ( https://nmap.org ) at 2016-04-02 09:06 EDT
[*] Nmap: Nmap scan report for 192.168.0.103
[*] Nmap: Host is up (0.00095s latency).
[*] Nmap: PORT   STATE SERVICE
[*] Nmap: 21/tcp open  ftp
[*] Nmap: MAC Address: 08:00:27:94:82:93 (Oracle VirtualBox virtual NIC)
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 13.48 seconds
msf > db_nmap 192.168.0.103 -p 21
[*] Nmap: Starting Nmap 7.01 ( https://nmap.org ) at 2016-04-02 09:06 EDT
[*] Nmap: Nmap scan report for 192.168.0.103
[*] Nmap: Host is up (0.00079s latency).
[*] Nmap: PORT   STATE SERVICE
[*] Nmap: 21/tcp open  ftp
[*] Nmap: MAC Address: 08:00:27:94:82:93 (Oracle VirtualBox virtual NIC)
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 13.20 seconds
```

2.9 Анализ записей из файла nmap-service-probes

Рассмотрим следующие строки из файла nmap-service-probes.

```
Probe TCP HTTPOptions q|OPTIONS / HTTP/1.0\r\n\r\n|
rarity 4
ports 80-85,2301,631,641,3128,5232,6000,8080,8888,9999,10000,10031,37435,49400
sslports 443,4443,8443
fallback GetRequest
match caldav m|^HTTP/1\.1 200 OK\r\nServer: DavMail Gateway ([\w._-]+)\r\nDAV: 1, calendar
```

Директива Probe указывает на то, какое сообщение необходимо отправить для индентификации сервиса. В данном случае, сервис - HTTPOptions, используемый протокол - TCP, отправляется следующая строка:

```
OPTIONS / HTTP/1.0\r\n\r\n.
```

Строка с директивой rarity указывает частоту, с которой от сервиса можно ожидать возвращения корректных результатов. В данном случае - 4.
Директивы ports и sslports указывают на порты, используемые данным сервисом.
Директива 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. При этом выражение в скобках

```
([\w._-]+)
```

распознается как аргумент, к котоому далее можно обратиться слеудющим образом - $1. Флаг v указывает на версию сервиса, флаг d - на тип устройства, на котором сервис работает.

2.10 Описание работы скрипта из Nmap

Ресурс Nmap.org описывает систему поддержки сценариев (Nmap Scripting Engine (NSE)) как одну из самых мощных и гибких возможностей программы. Она позволяет разрабатывать и распространять простые сценарии на языке программирования Lua, предназначенные для автоматизации ряда задач, связанных с исследованием сети.
Рассмотрим скрипт http-errors, чтобы использовать скрипт необходимо указать флаг script и название используемого скрипта http-errors. Этот скрипт сканирует веб-сайты на поиск страниц, которые возвращают коды ошибок. Скрипт возвращает страницы (отсортированных по коду ошибки), отвечающие кодом HTTP, равным или большем 400. Ниже приведен листинг скрипта. По коду видно, что возвращаемый код сравнивается со значением 400, при превышении этого значения код считается ошибочным.

```
...
categories = {"discovery", "intrusive"}
author = "George Chatzisofroniou"
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
```

```
local shortport = require "shortport"
local stdnse = require "stdnse"
local table = require "table"
local httpspider = require "httpspider"

portrule = shortport.port_or_service( {80, 443}, {"http", "https"}, "tcp", "open")

local function compare(a, b)
  return a[1] < b[1]
end

local function inTable(tbl, item)

  item = tostring(item)
  for key, value in pairs(tbl) do
    if value == tostring(item) then
      return true
    end
  end
  return nil

end

action = function(host, port)

  local errcodes = stdnse.get_script_args("http-errors.errcodes") or nil

  local crawler = httpspider.Crawler:new(host, port, '/', { scriptname = SCRIPT_NAME,
    maxpagecount = 40,
    maxdepth = -1,
    withinhost = 1
  })

  crawler.options.doscraping = function(url)
    if crawler:iswithinhost(url)
      and not crawler:isresource(url, "js")
      and not crawler:isresource(url, "css") then
      return true
    end
  end

  crawler:set_timeout(10000)

  local errors = {}

  while (true) do

    local response, path
```

```
    local status, r = crawler:crawl()
    -- if the crawler fails it can be due to a number of different reasons
    -- most of them are "legitimate" and should not be reason to abort
    if (not(status)) then
      if (r.err) then
        return stdnse.format_output(false, r.reason)
      else
        break
      end
    end

    response = r.response
    path = tostring(r.url)

    if (response.status >= 400 and not errcodes) or
      ( errcodes and type(errcodes) == "table" and inTable(errcodes, response.status) ) th
      table.insert(errors, { tostring(response.status), path })
    end

  end

  -- If the table is empty.
  if next(errors) == nil then
    return "Couldn't find any error pages."
  end

  table.sort(errors, compare)
...
```

3 Вывод

В ходе лабораторной работы была изучена утилита для исследования сети и сканер портов - Nmap. Были протестирвоаны некорые возможности утилиты: поиск активных хостов, открытых портов и версий сервисов, сохранение вывода в формате xml. Изучены служебные файлы nmap-services, nmap-os-db, nmap-service-probe. Файл nmap-service-probe также был изменен, путем добавления записей о собственном сервисе, котрый после был успешно распознан. Также исследована работа утилиты nmap с помощью wireshark.