

Abstract

This report presents implementation and analysis of network scanner. This paper describes consecutive iterations of scanning software and some basics analysis. For a selected internal network I found several servers with outdated and insecure software installed. However, the goal of this task did not consisted of vulnerability testing. Few open SMTP servers were found, although none of the four were actually vulnerable.

Introduction

Our task was to scan given network range and investigate found services. Investigation require us to gather detailed information about each opened service (such as version, name). Having those information we are able to determine if given software might be harmful for the server. In addition we were asked to run additional test for mail servers, in order to find out, if "open mail relay" were found. An open mail relay is a name for an SMPT server configured in such a way that it allows anyone on the Internet to send e-mail through it.

Network scanning

The lab room local network (in my case IP range 156.17.40.1-254) was scanned for all services such as SSH, SMTP, FTP, HTTP, HTTPS. Due to the excessively long scanning time, I decided to limit the range of ports to 22-443 (range of SSH to the HTTPS). I'm aware of the fact, that redirected ports (such as 22 → 2222) will not be found due to enforced limitation. However, in order to fit in given time I had to limit that range.

I decided to use "Network Mapper" (nmap) to scan network. It's open source software, that uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics.

First application (Listing 1) supposed to list all open ports alongside with version and others available information. Software was written in python 2.7 using nmap wrapper. It's just class wrapper for a nmap build-in package. Another package I used was 'PrettyTable', it's a simple Python library, that allows to create ASCII tables.

Figure 1 shows a fragment of received data. I was looking for UDP and TCP protocols, however I didn't found single UDP instance. Almost all services were running on default port, with only exception, which caught my attention. On machine 156.17.40.227 Apache is deployed on both ports 80 and 90. I will describe tests performed on 156.17.40.227 in next section.

```

import nmap
from prettytable import PrettyTable

# scan network - display all opened ports in given range
nm = nmap.PortScanner()
nm.scan('156.17.40.1-255', '22-443')

tab = PrettyTable(["IP address", "Protocol", "Port", "Product name",
                  "Version", "Extra info"])

for host in nm.all_hosts():
    for proto in nm[host].all_protocols():
        lport = nm[host][proto].keys()
        lport.sort()
        for port in lport:
            # incompatible with installed nmap version
            if not isinstance(port, int):
                continue
            item = nm[host][proto][port]
            # skip closed
            if not item['state'] == "open":
                continue

            tab.add_row([host, proto, port, item['product'], item['version'],
                        item['extrainfo']])

print tab

```

Listing 1: Network Scanner - first implementation.

IP address	Protocol	Port	Product name	Version	Extra info
156.17.40.11	tcp	22	OpenSSH	5.3p1 Debian 3ubuntu7	protocol 2.0 (Ubuntu)
156.17.40.11	tcp	80	Apache httpd	2.2.14	
156.17.40.11	tcp	111			
156.17.40.11	tcp	389	OpenLDAP	2.2.X	
156.17.40.12	tcp	22	OpenSSH	5.3p1 Debian 3ubuntu7	protocol 2.0 (Ubuntu)
156.17.40.12	tcp	80	Apache httpd	2.2.14	
156.17.40.12	tcp	111			
156.17.40.12	tcp	389	OpenLDAP	2.2.X	
156.17.40.13	tcp	22	OpenSSH	5.3p1 Debian 3ubuntu5	protocol 2.0 (Ubuntu)
156.17.40.13	tcp	80	Apache httpd	2.2.14	
156.17.40.13	tcp	111			
156.17.40.13	tcp	139	Samba smbd	3.X	workgroup: WORKGROUP
156.17.40.13	tcp	389	OpenLDAP	2.2.X	
156.17.40.14	tcp	22	OpenSSH	5.3p1 Debian 3ubuntu5	protocol 2.0 (Ubuntu)
156.17.40.14	tcp	80	Apache httpd	2.2.14	
156.17.40.14	tcp	111			
156.17.40.14	tcp	389	OpenLDAP	2.2.X	
156.17.40.15	tcp	80	Apache httpd	2.2.14	(Ubuntu)
156.17.40.15	tcp	111			
156.17.40.15	tcp	389	OpenLDAP	2.2.X	
156.17.40.16	tcp	22	OpenSSH	5.3p1 Debian 3ubuntu5	protocol 2.0 (Ubuntu)
156.17.40.16	tcp	80	Apache httpd	2.2.14	
156.17.40.16	tcp	111			
156.17.40.16	tcp	389	OpenLDAP	2.2.X	
156.17.40.17	tcp	22	OpenSSH	5.3p1 Debian 3ubuntu7	protocol 2.0 (Ubuntu)
156.17.40.17	tcp	80	Apache httpd	2.2.14	
156.17.40.17	tcp	111			
156.17.40.17	tcp	389	OpenLDAP	2.2.X	
156.17.40.18	tcp	22	OpenSSH	5.3p1 Debian 3ubuntu6	protocol 2.0 (Ubuntu)
156.17.40.18	tcp	80	Apache httpd	2.2.14	
156.17.40.18	tcp	111			
156.17.40.18	tcp	389	OpenLDAP	2.2.X	
156.17.40.20	tcp	22	OpenSSH	5.3p1 Debian 3ubuntu5	protocol 2.0 (Ubuntu)
156.17.40.20	tcp	80	Apache httpd	2.2.14	
156.17.40.20	tcp	111			
156.17.40.20	tcp	389	OpenLDAP	2.2.X	
156.17.40.21	tcp	22	OpenSSH	5.3p1 Debian 3ubuntu5	protocol 2.0 (Ubuntu)
156.17.40.21	tcp	80	Apache httpd	2.2.14	
156.17.40.21	tcp	111			
156.17.40.21	tcp	389	OpenLDAP	2.2.X	

Figure 1: Selected scanned IP addresses

Looking for insecure software

Another part of the laboratory was to perform manual scan of obsolete or insecure version of software installed on scanned machines. We had all necessary informations already, therefore this part was rather simple. There are several websites that aggregates information about specific application vulnerabilities, one of them have an API, so entire process could be automatized.

List of machines that have vulnerable software installed (in case of multiple instances running the same version of software - only one was listed)

IP Address	Port	Software name	Software version	CVSS Score
156.17.40.105	80	Microsoft IIS webserver	7.5	10.0/10
156.17.40.148	25	Exim smtpd	4.72	7.5/10
156.17.40.12	389	OpenLDAP	2.2.X	7.1/10
156.17.40.220	22	OpenSSH	3.8.1p1	9.3/10
156.17.40.18	22	OpenSSH	5.3p1	7.5/10
156.17.40.85	25	Sendmail	8.14.3	7.5/10

I'm not going to explain in details each security vulnerabilities, instead I will focus on few of them. Detailed description of each listed applications and their versions can be found on cvedetails.com.

First worth describing vulnerability is ability to bypass intended access restrictions via a crafted client certificate. Sendmail version 8.14.3 is the last release with bug. That's the only major issue with Sendmail. Another bypass attack is available on Openssh version 5.3, LDAP user can be successfully authenticated without knowledge of the shared secret. Installed version of Openssh is also vulnerable to DoS attack, that will cause memory corruption. Last but not least, Microsoft IIS webserver with multiple flaws. IIS 7.5 allows remote attackers to execute arbitrary code or cause a denial of service.

I also did some private analysis on previously mentioned PHP script on machine 156.17.40.227 (port 90), although there is no major issue I decided to spent some time to describe it. Site is vulnerable to XSS attack, limited (not harmful) SQL Injection and other less important flaws. Another worth pointing out issue is invalid PHP configuration, when invalid arguments will be provided to the script, PHP error log will be returned (with server paths, internal file names). I did use sqlmap to find out whenever search parameter is injectable and it's vulnerable to SQL inject, although percent sign is not filtered - parameter value is injected to the query in insecure way. As presented in Figure 4 I did filter city name by starting with letter "B". Suppose, that the same query builder was used for authentication. We would be able to login using `username=%` and `password=%`.

```

<br />
<b>Warning</b>: include(../../eka_lib.php) [

```

Figure 2: PHP configuration flaw

156.17.40.227:90/katalog/kk_opis.php?rok=<span%20style=color:red>sample&frm=ete8305.frm

SCHOOL OF ELECTRONICS
POSLAW UNIVERSITY OF TECHNOLOGY



pisów	Teaching	Arc
-------	----------	-----

Course catalog sample

008/2009

Brak danych

version •

Figure 3: XSS code injection

Nr	Nazwa firmy	Miasto	Spec	Dodano
1	Alfa Poland	Bielsko-Biała	AIR	2008-02-14
2	BOT Elektrownia Opole S.A.	Brzeziny k. Opola	EIT, AIR	2008-02-13
3	BOT KWB Bełchatów Spółka Akcyjna	Bełchatów	EIT	2008-02-26
4	BOT KWB Turów	Bogatynia	AIR	2008-02-18
5	Cumulus Jacek Mazoń	Bielany Wrocławskie	EIT	2008-03-13
6	FCPK-BYTÓW Sp. z o.o.	Bytów	AIR	2008-02-14
7	Fiat Auto Poland S.A.	Bielsko-Biała	AIR	2008-03-04
8	KFKI RESEARCH INSTITUTE FOR PARTICLE AND NUCLEAR PHYSICS Department of Space Technology	Budapest Hungary	INF	2008-02-18
9	KOMES	Bielsko-Biała	EIT	2008-03-13
10	PCC ROKITA	Brzeg Dolny	EIT	2008-02-27
11	Partner J&M Grabarz	Bolesławiec	EIT	2008-03-18
12	Security System Integration Sp. z o.o.	Bielany Wrocławskie Kobierzyce	EIT	2008-03-13
13	TELE VIDEO MEDIA Sp. z o.o.	Bielany Wrocławskie	EIT	2008-03-18
14	Z.P.B. Bielbaw S.A.	Bielawa	EIT	2008-03-04

Figure 4: Invalid input filtering

Open mail relay scan

All previously found mail servers should be tested whether they are "closed" or "open relay". Following code (Listing 3) was used to obtain list of servers with available SMTP. It's slight modified version, looking only for port 25. List of found services is shown in Figure 5.

```
import nmap
from prettytable import PrettyTable

# scan network - display all opened ports in given range
nm = nmap.PortScanner()
nm.scan('156.17.40.1-255', '25')

tab = PrettyTable(["IP address", "Protocol", "Port", "Product name",
                  "Version", "Extra info"])

for host in nm.all_hosts():
    for proto in nm[host].all_protocols():
        lport = nm[host][proto].keys()
        lport.sort()
        for port in lport:
            # incompatible with installed nmap version
            if not isinstance(port, int):
                continue
            item = nm[host][proto][port]
            # skip closed
            if not item['state'] == "open":
                continue

            tab.add_row([host, proto, port, item['product'], item['version'],
                        item['extrainfo']])

print tab
```

Listing 2: Network Scanner - just mail services.

Using found IP addresses of a SMTP servers we can proceed to "open relay" test. I used the data given in task description ("xx@plonk.ict.pwr.wroc.pl" and "zxcvbnm@heaven.org"). I created simple Python script, whose job is to determine whatever server is vulnerable or not.

IP address	Protocol	Port	Product name	Version	Extra info
156.17.40.148	tcp	25	Exim smtpd	4.72	
156.17.40.162	tcp	25	Exim smtpd	4.80	
156.17.40.46	tcp	25	Brother printer smtpd		
156.17.40.85	tcp	25	Sendmail	8.14.3/8.14.3/Debian-9.1ubuntu1+bspm1.13+rchk1.22	

Figure 5: Mail services

```

import smtplib

SERVERS = ["156.17.40.148", "156.17.40.162", "156.17.40.46", "156.17.40.85"]

for SERVER in SERVERS:
    FROM = "test@pwr.wroc.pl"
    TO = ["xx@plonk.ict.pwr.wroc.pl", "zxcvbnm@heaven.org"]

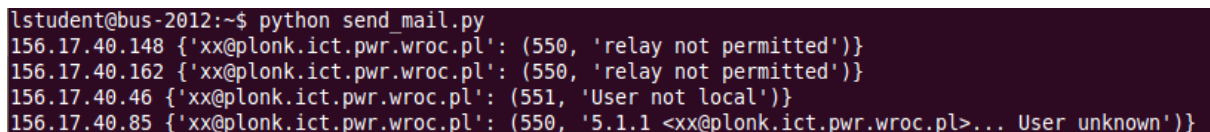
    message = """\
    From: %s
    To: %s
    Subject: %s

    %s
    """ % (FROM, ", ".join(TO), "test", "test")

    server = smtplib.SMTP(SERVER)
    #server.set_debuglevel(True)
    try:
        server.sendmail(FROM, TO, message)
    except smtplib.SMTPRecipientsRefused, e:
        print SERVER, e
    else:
        print SERVER, "ok"
    server.quit()

```

Listing 3: Open relay testing code



```

lstudent@bus-2012:~$ python send_mail.py
156.17.40.148 {'xx@plonk.ict.pwr.wroc.pl': (550, 'relay not permitted')}
156.17.40.162 {'xx@plonk.ict.pwr.wroc.pl': (550, 'relay not permitted')}
156.17.40.46 {'xx@plonk.ict.pwr.wroc.pl': (551, 'User not local')}
156.17.40.85 {'xx@plonk.ict.pwr.wroc.pl': (550, '5.1.1 <xx@plonk.ict.pwr.wroc.pl>... User unknown')}

```

Figure 6: Scanning result

None of found services is vulnerable to "Open mail relay", all of them returned 55X error message (as shown in Figure 6).

References

- [1] <http://www.computerhope.com/unix/nmap.htm>
- [2] <http://www.cvedetails.com/>
- [3] <http://dream.ict.pwr.wroc.pl/ssn/index.en.html>