ANALELE ŞTIINŢIFICE ALE UNIVERSITĂŢII „ALEXANDRU IOAN CUZA" DIN IAŞI
Tomul LV Ştiinţe Economice 2008

SECURITY TOOLS SOFTWARE IN AN OPEN SOURCE ENVIRONMENT

Napoleon Alexandru SIRETEANU*

Abstract

In a penetration test, once we establish what targets are, and how many may or may not be vulnerable, the next step is to use our tool selection and exploitation methods. The purpose of this article is to help you understand the need for identifying specific services and scanning activities as a start point of penetration test, and help you learn how to best perform these activities with specific tools like nmap or p0f. The main target is to describe specific tools that help reveal the characteristics of your targets, services, versions, and types of resources they offer.

Keywords: scanning, fingerprinting, rootkit, nmap, p0f, Nessus
JEL classification: L86

1. Foreword

"Open source" is one of the most misunderstood concepts in the computing industry today. Most people in the industry, and even a number of people outside the industry, have heard of open source software. They may be able to name some of the most prominent open source projects, like Linux, Apache, and Firefox. But what most people don't understand about open source software is why it works. Software released as open source costs money and time to make and it has value.

Even if source licensing and source code availability are not indicators of the security of a software application, there is a significant benefit of open source to some users alarmed about security. Open source allows experts to inspect their software options before making a choice and also in some cases to make improvements without waiting for fixes from the vendor or source code maintainer.

Open source security tools abound, so we can take advantage of them and avoid paying for commercial products if open source fits your needs. Freshmeat.net and Sourceforge.net are two central repositories to find open source software and information about scanners, authentication software, penetration testing tools, antispam, intrusion-detection systems and more that exist as open source or freeware.

Malicious attackers are sophisticated enough to understand that the real power of the most popular network security tools does not stay in their functionality, but in the framework that allows us to extend and adjust their functionality. These sophisticated

* Napoleon Alexandru SIRETEANU (salexis_ro@yahoo.ro) is assistant professor of Economics, Quantitative Analysis and Business Information Systems Department at "Alexandru Ioan Cuza" University of Iasi, Faculty of Economics and Business Administration. He received his PhD in: 2007. His research interests include: web technologies, security and operating systems. His teaching interests include: web engineering, geographic information systems.

attackers also know how to quickly write their own tools to break into remote networks [Clarke, 2005, 5-20].

The aim of this article is to present how to use existing and powerful open source tools to protect your networks and data from the most experienced attackers. The first thing that must perform, what kind of system we are testing, is identifying specific services and scanning and after that we can begin exploitation and increase the depth of our activities.

2. Identifying specific services

Identifying the specific services and resources that are offered by a target begins with a set of parameters, like an IP address range, or a specific Domain Name Service (DNS) entry, and the open ports on the system. A list of common ports can be found on services file from Unix and Linux (/etc/services) and there are: FTP-data (20), FTP (21), SSH (22), Telnet (23), SMTP (25), Whois (43), DNS (53), TFTP (69), HTTP (80), POP3 (110), NTP (123), Samba (137–139), SNMP (161), LDAP (389), HTTPS (443), MySQL (3306) and X11 (6000–6063) [Dhanjani, 2003, 30-32]. From those services, we can go into deeper scanning, like security scanning and testing, the essence of penetration testing.

For example, banner grabbing is an option implemented into nmap for scanning a network and seeing which services are give up too much information. The most common tools used in this stage include nmap and amap [Openxtra, 2008].

During this phase, is very important to make notes. If the tool we are using cannot output a log file, we must use tools like tee, which will allow us to send the output of a command to your terminal and to a log file.

We can complete this phase using active or passive methods. Proxy methods are passive, because the information we gather will be from a third source and is not intercepted from the target itself. Active methods are truly efficient because we send predetermined types of packets, and then receive packets in return.

3. Scanning

The scanning phase consists in gathering information about the target's purpose; what ports and what services it offers. Information collected during this phase is also used to determine the operating system or firmware version (in the case of routers) of the target devices. The list of active targets accumulated from this phase will be used as a possible target list.

Penetration testing often becomes a test of the client's existing security controls and configurations. If we research network attacks, it is known that most attackers will spend as much time as they can need gathering information on their target before they attack.

The collected list of potential targets from the first phase (identifying specific services) can be very comprehensive. To begin the scanning process, it is necessary to determine if the systems are up and responsive. Several methods can be used, but the most common technique uses Internet Control Message Protocol (ICMP) packets.

Request for Comments (RFC) 1122 define the ICMP echo request packet like a basic one that every host needs to implement and respond to. In reality, many networks, internally and externally, block ICMP echo requests to defend against a denial-of-service (DoS) attack called ping flood and to prevent scanning from the outside. If ICMP packets are blocked, TCP ACK (TCP acknowledge) packets can also be used by ping command. The unexpected ACK packets should return a TCP RST (TCP reset). For that reason, sending this type of packet to a port that is allowed through a firewall, such as port 22 (sshd service), the target should respond with an RST indicating that the target is active.

Ping sweeping is the process of pinging numerous hosts. In the case of a large set of target IP addresses, one must perform a ping sweep to determine alive hosts that respond to ICMP echo requests [Dhanjani, 2003, 78]. This sweep should be done and held a log file that specifies active machines that we can later input into a scanner.

4. Port Scanning

The most common of TCP port scans is SYN scan, used for the TCP SYN flag, which appears in the TCP connection sequence. This type of scan sends a SYN packet to a destination port, the return is analyzed and a SYN/ACK response is sent if the port is open or an RST if the port is closed. Because the TCP connection did not really occur, the service on the target does not see a connection, and does not create an entry in a log file [Insecure.Org, 2008].

Different flags, like FIN, PUSH and URG, are useful for OS detection and to avoid access controls that stand on connections initiated with specific TCP flags set. Nmap support a lot of port scan techniques listed in Table 1.

Table no. 1 - *Port Scanning Techniques*

Nmap switch	Type of packet send	Response if open	Response if close	Details
-sS	TCP SYN scan	SYN/ACK	RST	stealthy scan with root privileges
-sT	TCP connect scan	connect() system call	connection refused	target machines log the connection to syslog
-sU	UDP scan	port is open	port is closed or filtered	UDP scan works by sending an empty UDP header
-sN	tcp flag header is 0	connection timeout	RST	can go through firewalls and packet filtering routers
-sX	sets the FIN, PSH, and URG flags	connection timeout	RST	can go through firewalls and packet filtering routers
-sF	sets only the TCP FIN bit	connection timeout	RST	can go through firewalls and packet filtering routers
-sA	TCP ACK scan	RST	RST	mapping firewall rulesets
-sW	TCP Window scan	RST	RST	uses TCP Window scan value to detect if an filtered port is open or closed
-sM	TCP Maimon scan with FIN/ACK packet	connection timeout	RST	works on BSD systems
--scanflags	arbitrary TCP flags (URG, ACK, PSH, RST, SYN, and FIN)	connection timeout	RST	creative design of your own scan
-sI	TCP SYN scan	SYN/ACK	RST	truly blind TCP port scan of the target (uses a "zombie" host as scan source)
-sO	IP scan	response for all protocols	ICMP protocol unreachable	used to map IP protocols that are used by host
-b	TCP connect scan	connection made	connection refused	FTP bounce scan
-sV	subprotocol check (SMTP, HTTP, HTTPS ...)	N/A	N/A	Used to see what services are running on an open port
-O	TCP and UDP packet checks	N/A	N/A	Useful to establish OS/firmware version

5. Service Identification

On port scanning action we catch the open ports and now we need to be able to verify what the running ports are. Normally we think that HTTP is running on TCP 80, but what if the administrator of the system is trying to obfuscate the service, and is running telnet instead? The easiest way to check the status of a port is a banner grab. Upon connecting to a service, the target's response is captured and compared to a list of known services, such as the response when connecting to an OpenSSH server. The banner in this case is the version of the service, OpenSSH version 4.6p1.

6. Fingerprinting

The purpose of system fingerprinting is to determine the operating system version and type. There are two common methods of performing system fingerprinting: active and passive scanning.

The active methods use responses sent to TCP or ICMP packets. The TCP fingerprinting process requires setting flags in the header that individual operating systems and versions respond to differently. Different TCP packets are sent and the responses are compared to known fingerprints to determine the remote OS. ICMP-based methods use fewer packets than TCP based methods and are recommended.

Similar to the active method, passive fingerprinting does not send any packets, but is based on sniffing techniques to analyze the information sent in normal network traffic. If our target is running publicly available services, passive fingerprinting may be a good choice for fingerprinting [Insecure.Org, 2008].

7. Period (of time)

Running a port scan of 2000 ports in 30 seconds will be more suspicious than one in which we take ten hours to scan 2000 ports. It is necessary to use data collected from the first phase to amplify the scanning phase. If we found a host through a search engine like Google, you already know that port 80 (httpd) or 443 (https) is open.

Integrated tools such as Nessus, H.E.A.T., or Retina will run some type of portscan, followed by a series of security tests. These functions can be replicated with Google queries, although in most cases the results are near as effective as the results from a well thought out vulnerability scanner or Web assessment tool [Long, 2005, 221-262].

It is stringent to find out when the target passes the most traffic and scans during those peak times. For example, on paydays, or on the first of the month, a bank should have higher traffic than on other days in the month, due to the number of visitors performing transactions.

8. Bandwidth problems

When we are scanning a single target over a business broadband connection, we will not influence the destination network, even running a few scans at the same time. If we are performing a DoS test, an excessive bandwidth usage will be noticed by the system administrator.

Denial of Service (DoS) can come in many forms, from simply swamping the main routers with traffic to actually taking advantage of a weakness in a program to crash that service and therefore the server [Howlett, 2004, 135].

9. Open Source Tools

After we described theories about security, it is time to implement them with the open source tools. We'll present different tools, divided into two categories: scanning and revealing vulnerabilities.

9.1. Scanning tools

These tools will scan a list of targets to determine which hosts are up, and what ports and services are available.

9.1.1. nmap

Port scanners accept a target or a range as input, send a query to specified ports, and then create a list of the responses for each port. The most popular scanner is nmap available at www.insecure.org. Nmap tool has become a standard among testers and network auditors. We will concentrate on a few different scan types and options to make the best use of scanning time and to return the best information.

nmap: Ping Sweep

Before scanning active targets, it is necessary to use ping sweep functionality of nmap with the -sP option. This option will not scan a port on target, but will report which targets are up. When is appealed as root with nmap –sP *ip_address*, nmap will send ICMP echo packets and TCP SYN packets to determine if a host is up.

Ping types, like -P0 and -PS enables a TCP ping sweep, with -P0 indicating "no ICMP ping" and -PS indicating "use TCP SYN method." By detaching the scanning method to just one variant, we increase the speed as well, that is great when scanning multiple /24 networks, or even a /16.

nmap: ICMP Options

If nmap can't see the target, it won't scan it unless the -P0 (do not ping) option is used. We can use the -P option to select another type of ping acting. For example, the -PP option will use ICMP timestamp requests, and the -PM option will use ICMP netmask requests. Before perform a full sweep of a network range, it might be useful to do a few limited tests on known IP addresses, such as Web servers, DNS, to make the ping sweeps and reduce the number of total packets sent and the time taken for the scans.

nmap: Output Options

Capturing the results of the scan is extremely important and depending on our client's requirements it can be submitting them as evidence of vulnerability. The easiest way to capture all the needed information is to use the *–oA* flag, which outputs scan results in three different formats simultaneously: plain text (.nmap), filterable text (.gnmap), and XML (.xml).

nmap: Stealth Scanning

For any scanning we perform, it is not a good idea to use a connect scan (-sT), which fully establishes a connection to a port. Excessive port connections can cause a DoS to older machines. A stealthy port testing method with nmap, such as a SYN scan, is accomplished with -sS flag, which creates a listing of the open ports on the target, and possibly open/filtered ports if the target is behind a firewall.

nmap: OS Fingerprinting

In fact, ports 135, 137, 139, or 445 often indicate a Windows-based target. If we want to get more specific, we can use nmap -O flag, which appeal nmap fingerprinting mode. On the Linux targets we can see open ports like 21, 22, 53, 80, 443 or 3306.

nmap: Scripting

When we specify our targets for scanning, nmap will accept specific IP addresses and address ranges. If we have a host file, which may have been generated from our ping sweep earlier, we can specify it as well, using the -iL flag. Using scripting we can use *awk* to create a quick hosts file from an nmap ping sweep.

For example, the following command: grep 'online hosts' ping-sweep-home.nmap | awk -F \('{print $2}' | awk -F \('{print $1}' > valid-hosts will output a file that contains a list of online hosts: 192.168.0.100, 192.168.0.106, 192.168.0.120.

nmap: Speed Options

Nmap allows the user to specify the "speed" of the scan and the total of time from probe sent to reply received. On a fast LAN, we can optimize scanning by setting the -T option to 4, or aggressive, usually without dropping any packets during send. If a normal scan is takes very long due to filtering, or a firewall device, we will enable aggressive scanning.

By default, nmap scans 1663 ports for common services, which will catch most open TCP ports. Several sysadmins may run ports on uncommon ports, practicing security through obscurity. If we suspect that a system may be running other services, will run nmap with the -p1-65535 parameter, which will scan all 65000 TCP ports. Even on a fast LAN, this will take from 30 minutes to a few hours. Performing a test like this over the Internet may take even longer, which will also allow more time for the system owners to note the excessive traffic and close de connection.

9.1.2. netenum: Ping Sweep

If we need a very simple ICMP ping sweep program that can be used for scriptable applications, netenum might be useful. Netenum performs a basic ICMP ping and then replies with only the reachable targets and requires a timeout to be specified for the entire test [Vulnerabilityassessment.co.uk, 2008].

9.1.3. unicornscan: Port Scan

Unicornscan is different from a standard port scanning program, and allows specifying more information, such as source port, packets per second sent, and randomization of source IP information. Unicornscan perform a basic SYN port scan and is best for scanning during an IDS test, where the packet forging capabilities could be put to more use [Securityforest.com, 2008].

9.1.4. scanrand: Port Scan

Scanrand offers different options than a typical port scanner and implements two separate scanner processes, one for sending requests and one for receiving those requests. For this reason, processes can run asynchronously, which gives an improvement in terms of speed. The packets are encoded with digital signatures that allow the processes to keep track of the requests and prevent unreal responses [Doxpara.com, 2002].

Scanrand has the ability to specify bandwidth usage for the scan, from bytes to gigabytes. When performing testing over a high-band connection, such as satellite, the ability to use these is very important.

9.2. Tools for revealing vulnerabilities

These tools will scan a list of targets and ports to help determine more information about each target. This phase will reveal program names, version numbers, and other detailed information that will be used to determine vulnerabilities on those systems.

9.2.1. nmap: Banner Grabbing

This version-scanning feature of nmap is called with the -sV flag. Based on a returned banner, or on a specific response, a match is made between the service response and the nmap service fingerprints. For example a scan using nmap –sS –sV –O –v on a Linux server, will perform a SYN-based port scan, a version scan, and the OS fingerprinting function, all with detailed output. Typically, the scanner gathers the version and protocol of OpenSSH, in use, and the Web server name and version (Apache 2.2.4). The FTP server, Mail server, and netbios-ns, show closed, but present. In this case, the firewall rules are open for those services, but they are not currently running. In theory the system is a general server and the open firewall ports could be scanned later to determine if the backend servers are running.

9.2.2. p0f: Passive OS Fingerprinting

If we want to be extremely quiet in initial scan process, and don't wish getting high-level results for OS fingerprinting, p0f is the perfect tool. P0f analyze the responses from our target on safe queries, such as Web traffic, or ping replies. p0f gives the best estimation on an operating system based on those replies [Lcamtuf.coredump.cx, 2006].

9.2.3. Xprobe2: OS Fingerprinting

Even if many operating system designers follow RFCs, the operating system kernel authors sometimes interpret details differently. Tools such as Xprobe2 and Nmap explore such differences in order to fingerprint the target operating systems.

Xprobe2 is an OS fingerprinter, but also has some basic port-scanning functionality built in to identify open or closed ports. We can specify known open or closed ports, to which Xprobe2 performs several tests (TCP, UDP, and ICMP) to determine the remote OS [Arkin, O., 2003].

9.2.4. httprint: Web Fingerprinting

Httprint is designed to run across a Web server and to discover the HTTP daemon running and only fingerprints http servers [Net-Square, 2008].

9.2.5. amap: Application Protocol Detection

Amap sends multiple queries and probes to a specific service, and then analyze the results, including returned banners, to identify what application or service is actually running on a specific port [The Hacker's Choice, 2008].

9.2.6. smbgetserverinfo/smbdumpusers for Windows targets

If TCP ports 135, 137, 139, or 445 are open, this indicates that the target machine is Windows-based or is running a Windows-like service such as Samba. In Windows, if the registry keys RestrictAnonymous and RestrictAnonymousSAM are set to 0, an anonymous user can connect to the system with a null session and dump the list of local user accounts and shared folders for the system. By connecting to a Samba server via a null session, we can get the Samba system name and the operating system version. The Windows XP target does not return any information, but the Linux target returns the listing of all local users. For attacking newer Windows versions, tools such nbtscan and vulnerability scanner software like Nessus are useful.

Smbclient, a file transfer client for SMB networks operates like an FTP client and it permits to specify the username, with -U parameter and target host using -I. Let us see an example of successful connection to a target system using smbclient for the purpose of sending a file that may be later used in the penetration test:

```
Smbclient -I 192.168.1.121 -U Administrator \\\\192.168.1.121\\C$
Password: ***
Domain:[HOME] OS=[Windows 5.0] Server=[Windows 2003 LAN Manager]
smb: \> cd winnt
smb: \winnt> cd system32
smb: \winnt\system32> put /tmp/trojan.exe RUNME.EXE
putting file /tmp/trojan.exe as \winnt\system32\RUNME.EXE (1.0 kb/s)
(average 1.0 kb/s)
smb: \winnt\system32>
```

10. Real tests

On real scenario based on external and internal penetration tests, the first step is to perform a whois lookup, ping, and host queries to make sure the system is the real target. If we running whois hostname.domainname returns NOT FOUND, we do a whois on the domain only, domainname. This returns registration information for DynDNS.domainname, which indicates that the target is probably a dynamic IP address using DynDNS for an externally reachable DNS name.

If we run nmap –sS –v –oA external-nmap hostname.domainname, we will perform a SYN scan and version scan, writing the output to the file external-nmap. This scan returned four TCP ports open: 21, 22, 80, and 8080.

To identify what those open ports are running, we will run amap (amap –Abq hostname.domainname 22 and amap –Abq hostname.domainname 80 and amap –Abq hostname.domainname 8080), revealing that port 22 is running OpenSSH 4.6p1, with protocol version 2.0, port 80 shows as Apache 2.2.4 (Ubuntu), and 8080 appears to be the login page for a Linksys WRT54G Wireless Access Point/Router.

Additional scanning results on the target system, we can obtain by using Nessus port scan. Nessus discover that BIND 9.x is running on UDP 53. The OpenSSH server shows as providing 1 and 2 protocol support.

Each of the discovered software products would be investigated to search for known vulnerabilities, and additional testing would be performed against the software to determine any misconfigurations.

For the internal tests, we will scan the 192.168.0.0/24 network. Internal network firewalls do not exist, but host firewalls are installed. Performing a ping sweep using nmap -sP -PS -oA internal-ping-sweep 192.168.0.0/24 reveals three targets: 192.168.0.100, 192.168.0.106, 192.168.0.120, and 192.168.0.255 (certainly broadcast address). The DNS names laptop, desktop2 and axim were listed.

To provide a complete scan, we ran nmap -sS -sV -O -iL valid-hosts -oA fullinternal-sweep where valid-hosts was created by the use of the previous awk command. We record from this scan a ProFTPD and Samba server on 192.168.0.100 (a Linux system).The 192.168.0.106 machine seems to be a Windows 2003 Server system that is additionally running Apache Web Server. Information like this will be useful for future attack scenarios.

If we run Nessus again with standard options to determine any suspicious ports or vulnerabilities, we can find an FTP server on 192.168.0.100 that is configured to allow remote connections and may be vulnerable to a DoS or remote code execution flaw from a user. Because this service allows anonymous connections, it would be an easy to exploit system.

If we suspect that you have a compromised system, it is a good idea to check for a rootkit, which is a collection of programs which intruders often install after they have compromised the root account of a system. These programs help the intruders clean up their tracks and provide access back into the system. Rootkits sometimes leave processes running to allow the intruder to return easily and without the system administrator's knowledge. A script like chkrootkit will detect over 50 different rootkits, network interfaces that are in promiscuous mode, altered lastlog files, and altered wtmp files.

In real world, after an intruder obtains root permissions, he will have total control of the victim host. Next, the intruder must hide his traces, and plant Trojans and backdoors in order to ensure continued access. From cleaning log files to installing backdoors and rootkits, it is often difficult to detect the presence of refined hackers.

11. Conclusions

There will be periods during the penetration test when attack technique may not work. IP addresses may change, routes may change or drop, or tools may stop working without any warning. Even negative results may produce positive information, such as the fact that the firewall imitates open ports for closed ports.

Open source security tools offer numerous benefits to enterprise security, but they can also come with their own vulnerabilities. In the fast-moving world of computers, things are always changing. Network security techniques and tools have evolved rapidly to meet new and more sophisticated threats that occur with alarming regularity.

Although many peoples sustain that open source projects are going to have less vulnerabilities than closed source projects, that's not really true; the number of vulnerabilities present in a given system can't be simply associated with the openness of its source code. Finally, it's about the way the project and its developers handle and integrate security.

References

Arkin, O., Yarochkin, F., *The Present and Future of Xprobe2; The Next Generation of Active Operating System Fingerprinting*, http://www.sys-security.com/archive/papers/Present_and_Future_Xprobe2-v1.0.pdf, accessed on September 9, 2008

Clarke, J., Dhanjani, N., *Network Security Tools*, O'Reilly, 2005

Dhanjani, N., HackNotes™ Linux and Unix Security Portable Reference, McGraw-Hill, 2003

Howlett, T., Open Source Security Tools: A Practical Guide to Security Applications, Prentice Hall, 2004

Long, J., Google Hacking for Penetration Testers, Syngress Publishing, 2005

*** *Scanning modes available in Nmap*, at http://www.openxtra.co.uk/support/howto/nmap-scan-modes.php, accessed on September 10, 2008

*** *Port Scanning Techniques*, at http://insecure.org/nmap/man/man-port-scanning-techniques.html, accessed on September 10, 2008

*** *Fingerprinting Methods Avoided by Nmap*, at http://insecure.org/nmap/osdetect/osdetect-other-methods.html, accessed on September 10, 2008

*** *Netenum*, at http://www.vulnerabilityassessment.co.uk/netenum.htm, accessed on September 15, 2008

*** *Unicornscan – SecurityForest*, at http://securityforest.com/wiki/index.php/Unicornscan, accessed on September 15, 2008

*** *Scanrand 1.0: A Demonstration*, at http://www.doxpara.com/read.php/docs/scanrand_logs.html, accessed on September 15, 2008

*** *The new p0f*, at http://lcamtuf.coredump.cx/p0f.shtml, accessed on September 15, 2008

*** *Net-Square: httprint*, at http://net-square.com/httprint/, accessed on September 20, 2008

*** *THC-AMAP - fast and reliable application fingerprint mapper*, at http://freeworld.thc.org/thc-amap/, accessed on September 20, 2008