Chapter 17

Security scanners

There are more and more applications for network administrators on the market, both commercial and free, which serve to verify the system security. Today's administrator has to be highly motivated and extremely patient, as he has to become familiar with a large amount of new software to help him in his difficult work.

There has been a flood of network monitors, network configurators, and other programs to improve network function, or that relieve the administrator of at least part of his responsibilities. However, it is the administrator's duty to deal with the network. Full automation of the network combined with a superficial status check are a recipe for trouble. This is especially true of huge corporate networks, where security is often neglected. Instead of surveying their systems at least once a day, administrators often hand off this responsibility to various applications. In fact, these applications should only help the administrator in detecting possible irregularities or unauthorized access, and not, as many seem to think, completely take over this task. Software can easily be deceived.

This chapter will show how administrator should monitor network security. We will demonstrate that it is worth dedicating one's time to analyzing and choosing the correct settings instead of automating the monitoring function of the data being sent.

This chapter can also be understood from "the other side" – the hacker's point of view. The majority of network applications, which on the one hand help protect a network, can also be used to manipulate a network for one's own purposes. In particular, we mean scanners, one of the most popular

network kinds of programs in recent years. We should bear in mind that the best method to get to know the network security level is to attempt to break down all the barriers that normally protect our systems. Just as real-life detectives do, to learn the details or the motive of an offense we have to put ourselves in the criminal's place, so as administrators we should become hackers and carry out an attack on our own network. It is also good to try to obtain as much information as possible about it. Just as an intruder would do before cracking.

What are scanners?

Scanners, the subject of this chapter, are "neutral" network applications. This means that they can help both a hacker and an administrator. Their task is to collect information about network devices. As it turns out, this information can be quite varied. We are able to discover which software is used in the system, to check how long it has been running, and to find out about the available ports. Of course the scanners are written in such a way that their activity won't leave unwanted footprints on the target machine. It happens often that scanning is performed using undocumented protocols, the monitoring of which is usually ignored.

The advantages this presents may seem to be useful only to a hacker, but they are also important to an administrator. They allow us to make appropriate changes to the settings and improve the system security level.

We will now describe three popular scanners and explain how to use them. They are: Nmap, Nessus, and Nikto. Each of these applications provides different functions, and they complement each other perfectly.

Nmap

Nmap is a scanner that we will find in almost every distribution of the Linux system. It is most often used from the text console, although a graphical user interface exists, so even novices can use it.

We'll now install the nmap application. The first step is to download the software from the page:

```
http://www.nmap.org/download.html
```

We only need to download the program if it is not already installed in the system.

We download the newest version and start the installation (at the administrator level):

```
[user@localhost]$ su
[root@localhost]# tar xfj nmap-VERSION_NUMBER.tar.bz2
[root@localhost]# cd nmap-VERSION_NUMBER
[root@localhost]# ./configure
[root@localhost]# make
[root@localhost]# make install
```

Of course we substitute VERSION_NUMBER with the packet we currently possess. At the time of writing, version 5.21 was the latest version. The package contains also a graphical front end, but we will describe the command-line version.

To perform most standard scan types we can start up nmap from a normal user account; however, for full functionality the administrator shell is required.

Scanning techniques

Scanning using connect()

The first of the available scanning techniques is scanning by standard calling of the connect() system function, which uses the TCP protocol. This is the most basic scanning form. The connect() function, which downloads as parameters – among others – the target host name and the port to which we want to connect, ends successfully if the port is open. If the port is not available, the function will return an error and will state that the specific service is inaccessible. To perform this kind of scanning, administrator

privileges are not required. Unfortunately, this technique is very easy to detect. If the administrator is monitoring the data flow, he will receive information during scanning about numerous unsuccessful connections, on all possible ports, which allows this method to be detected easily. We run it using the –sT parameter.

TCP SYN scanning

TCP SYN scanning is a more advanced technique that requires administrator privileges. It is often called half-open scanning. It works in the following way: The scanner sends a SYN packet to the target computer. If it receives a SYN packet with the ACK flag set in reply, which means the connection has been successful, it states that the port is open and the service is working. At this moment the scanner sends a packet with the RST flag to break off the connection. An advantage of this technique is the fact that every server will detect the TCP SYN scanning. We run it using the –sS parameter.

TCP FIN scanning

The next scanning method using the TCP protocol is even more effective in comparison to SYN. Some firewalls record SYN packets incoming on specified ports, while synlogger programs can detect this type of scanning. However, FIN packets have a better chance of passing through such software unnoticed. In this technique, the closed ports reply to the FIN packet with an RST packet, breaking off the connection. Meanwhile, open ports don't reply to FIN. FIN scanning generates a FIN packet to perform the test. In this group there are some derivative methods. These are Xmas Tree scanning, which sends a packet with set FIN, URG, and PUSH flags and Null, which switches all flags off. Some systems from the Windows family sometimes send a RST reply even if the port is open, which causes erroneous results. We activate FIN scanning using the -sF parameter. The -sX parameter corresponds to the Xmas Tree method, whereas -sN corresponds to Null.

TCP ident scanning

The next method is TCP ident scanning. The identification protocol allows the name of the user who starts any process using the TCP protocol to be revealed, even if it wasn't this process that initiated the connection. Therefore, it is possible, for example, to connect to the HTTP (80) port and to use this protocol to check if the process works with administrator's rights, or to use port 21 (FTP) and to find out who activated this service. We start up the scanning using both the -t and -i parameters, which will show us the owners of the services running on all open ports.

UDP scanning with ICMP

Scanning with the UDP protocol using ICMP packets is a method that distinguishes itself by the use of the UDP protocol. To use it requires administrator privileges. Unfortunately, it's not possible to ascertain whether the service is working by sending only UDP packets of various kinds, as is the case for TCP. This happens because the open ports don't send a reply to an attempted connection, and the closed ports don't send an error message either. So, on this basis alone we are unable to state which ports are open and which are not. However, the majority of computers send an ICMP_PORT_UNREACH error when they receive a packet directed to a closed UDP port. Therefore, we are able to find out which ports are closed. Unfortunately, the UDP packets do not always reach their target, which is one of the main drawbacks of this protocol. We must implement scanner control over packet transmission, or it will produce erroneous results. Unfortunately, this kind of scanning is also characterized by low performance. This is because many systems limit the transmission of ICMP error messages. An example is the Linux system kernel, which limits the number of these packets to twenty per second. We activate scanning using the -sU parameter.

UDP scanning with write() and recvfrom()

A normal user cannot directly read the connection errors deriving from the closure of some ports, while Linux informs him indirectly when they occur.

For example, if we perform the write() function twice on a closed port, the second time it will be terminated with a failure. In addition, the recvfrom() call on the nonblocking sockets of UDP connections usually returns a request for another attempt, EAGAIN, if the error hasn't been received, and if it has, the value ECONNREFUSED (connection refused) is returned. This technique is employed if a user who doesn't have administrator rights uses the -u parameter described above. In activating this kind of scanning, the -l parameter can also be used.

ICMP echo scanning

This method does not return the list of open ports in its results. However, it is useful to determine which computers are switched on and functioning correctly in the network. In order to determine this, the scanner sends several ICMP echo packets to the computer. If it receives an answer it means that the remote computer is working. In order to activate this method we use the -sP option. This is a relatively rapid technique, but to speed it up some more, a variable number of ICMP echo queries can be executed in parallel, using the -L parameter for this purpose.

Fragmentation scanning

Fragmentation scanning is a combination of other techniques. Instead of sending a trial packet, it is divided into several smaller fragments. In this way, the TCP header is divided into several packets, which makes it more difficult for it to be detected by the packet filters. Unfortunately, some programs don't handle packets of this kind, which often leads to the suspension of the service. We activate the method with the -f parameter. We should remember that this only supplements scanning of a different type.

Scanning with version detection

If we already know which TCP or UDP ports are open by using the previous methods, this technique communicates with them, in order to determine which software is running on the target system. Nmap attempts to detect the

service protocol, which application is using which specific port, and which version this application is. We use it with the -sV parameter.

Scanning with the IP protocol

This method allows us to determine which protocols using IP are available on the target computer. Simple IP packets, without headers for different protocols, are sent to each open port of the target computer. If we receive an ICMP error, it means that the protocol is not used. Some systems secured by firewalls cannot send error messages, which can lead to erroneous results – all protocols will look like they are open. This technique is very similar to scanning the UDP ports, therefore, it may turn out that the troublesome limitations of the ICMP packets mentioned earlier will prevent quick scanning. We use the -s0 parameter.

ACK scanning

This is an advanced method, usually used to detect the presence and function of the firewall installed in the target system. It can help us to determine if the firewall is a well-functioning and customized application, or simply a standard program filtering the packets and rejecting SYN. In order to do this, an ACK packet is sent on the given ports. If a signal is returned that informs about breaking off the connection (RST flag), the port is classified as not filtered. If we don't receive an answer or an ICMP error is returned, the port is marked as filtered by the firewall. We use it with the -sA parameter.

Scanning with the TCP window size

This is an advanced scanning technique, and is very similar to the previous one. This method allows us to detect the listening-in ports, because the majority of operating systems modify the window size field. Apart from determining if the connection is monitored by a firewall, information about the port status can be obtained. We activate this with the -sW parameter.

RPC scanning

This technique works in combination with other scanning methods. It consists in sending many packets with NULL (empty) content on all open TCP and UDP ports found, in order to determine if they are RPC (remote procedure calling) ports. If the program determines that they are RPC ports, it returns the service name and version. We activate it with the -sR parameter.

List scanning

This method generates a list of IP addresses or of computers' names without scanning them in order to detect the ports. If we don't add the -n parameter the computers' names will be acquired with help of the DNS servers. We activate this method using the -sL parameter.

Scanning with ping

Sometimes the knowledge that a computer is connected to the network and is running, is all we need. Detecting is simple – nmap sends an ICMP Echo packet to the target computer. If it receives an answer it means that the computer is connected. Unfortunately some computers block these packets. In this situation nmap attempts to send a normal TCP ACK packet on port 80 set by default. If a packet with the RST (reset the connection) flag arrives it means that the computer is running. Another method is to send a SYN packet and to wait for any reply; however, this kind of scanning is available only for a user with administrator rights. Nmap uses both techniques (ICMP + ACK) simultaneously by default. This is also a method used as standard before all other scanning techniques, and only computers that respond to the request are scanned further. We activate it with the -sP parameter.

Idle scan

This is another advanced technique using an indirect "zombie" computer. The IP address of the scanning computer won't be recognized from the vantage point of the scanned computer. A big plus of this technique is the fact that we receive a list of ports that are open as seen from the zombie computer.

We activate the method with the option -sI TARGET_HOST, where TARGET_HOST is the address of the machine we will scan.

The nmap application has many options to enhance testing. Most of them will be useful only with very advanced scanning attempts and will be discussed when the previously described techniques can be practically applied. We would suggest the reader take a closer look at the program documentation and read the system handbook.

Starting up nmap

We will now try to start up nmap without any parameters. Information about using it will be displayed together with the version number. We will now scan our own computer first, performing the commands below and analyzing the results we obtain.

We will choose a standard scan, the first method using the connect() system call, and on this occasion we will check the versions of the applications used.

```
[user@localhost ~]$ nmap -sT -sV 127.0.0.1

Starting nmap 5.00 ( http://www.insecure.org/nmap/ ) at 2010-07-09 16:46 CET
Interesting ports on localhost (127.0.0.1):
(The 1656 ports scanned but not shown below are in state: closed)
PORT    STATE           SERVICE  VERSION
22/tcp  open            ssh              OpenSSH 3.9p1 (protocol 1.99)
25/tcp  open            smtp             Sendmail 8.13.1/8.13.1
111/tcp open            rpcbind          2 (rpc #100000)
631/tcp open            ipp              CUPS 1.1

Nmap run completed -- 1 IP address (1 host up) scanned in 21.849 seconds
```

As we can see, we discovered that 1,660 ports have been scanned, from which 1,656 ports proved to be closed. In the received table four columns are visible; these are, in sequence: port, port scan, service, what uses it, and service version.

We see several ports that could probably be used for an attack. On port 25 the Sendmail server is running, version 8.13.1. This is an application that was widely discussed recently due to its many security errors that allow intruders

to remotely gain control over the system. We also see port 631, on which the CUPS print server is running. Port 111 has the rpcbind service running, which handles remote commands. On port 22 an application has started up on the OpenSSH server.

We will now check the version of the system used on another remote computer:

```
[root@localhost]# nmap -sT -sV -O xxxyyyzzz.com

Starting nmap 5.00 ( http://www.insecure.org/nmap/ ) at 2010-07-09 16:58 CET
Interesting ports on xxxyyyzzz.com (180.180.180.180):
(The 1651 ports scanned but not shown below are in state: closed)
PORT    STATE           SERVICE         VERSION
21/tcp  open            ftp             vsFTPd 1.1.3
22/tcp  open            ssh             OpenSSH 3.5p1 (protocol 1.99)
25/tcp  open            smtp            Postfix smtpd
80/tcp  open            http            Apache httpd 1.3.27 ((Linux/SuSE)
                                        mod_ssl/2.8.12 OpenSSL/0.9.6i
                                        PHP/4.3.1 mod_perl/1.27)
110/tcp open            pop3
135/tcp filtered        msrpc
143/tcp open            imap            UW imapd 2002.332
443/tcp open            http            Apache httpd 1.3.27 ((Linux/SuSE)
                                         mod_ssl/2.8.12 OpenSSL/0.9.6i
                                         PHP/4.3.1 mod_perl/1.27)
445/tcp filtered        microsoft-ds

1 service unrecognized despite returning data. If you know the service/version, please
submit the following fingerprint at http://www.insecure.org/cgi-bin/servicefp-submit.cgi
:
SF-Port110-TCP:V=3.70%D=1/26%Time=41F7C059%P=i386-redhat-linux-gnu%r(NULL,
SF:22,"\+OK\x20ready\x20\x20<28500\.1106760314@04>\r\n")%r(GenericLines,22
SF:,"\+OK\x20ready\x20\x20<28500\.1106760314@04>\r\n");

Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.17 - 2.6.26
Uptime 26.233 days (since Wed Jun 30 11:31:47 2010)

Nmap run completed -- 1 IP address (1 host up) scanned in 28.361 seconds
```

The TCP connect() technique has been used, as in the previous example, in this case in checking the active service versions, which is very useful. The -O option has also been used, which causes a packet to be sent that enables us to determine the version of the operating system and the server uptime. An analysis will show us that 1,660 ports have been scanned. We also see the information from the second line of the report, which gives us the real server

name on which the domain is registered, xxxyyyzzz.com, meaning that it is a virtual domain. Let's have a quick look at the services available.

As we can see, the HTTP server, using the Apache application, is running twice – on the standard port 80 and on port 443. We should also notice that the SSH server and the FTP service are activated. Sending mail with the SMTP protocol is also activated with the Postfix service on the port 25. We also see another service, which will show us some information. It turns out to be the UW imapd server, functioning on port 143, whose task is to receive mail. There is also a POP3 server running on port 110.

At the end of the report there is information about the operating system (in this case Linux) and the probable version of its kernel (2.6.17 or 2.6.26). The time of server activity amounts to 26 days.

To scan the whole subnetwork it is necessary to give the value identifying the network together with the mask. The subnet mask determines how deeply the search should take place. However, scanning a chosen network can be defined in another, more accessible way. In the program the * character can also be used, which substitutes any range of a single segment of the IP address from 0 to 255. If we want to choose as our target the B class network, with addresses beginning 192.168.xxx.xxx, we enter the parameter 192.168.*.* as the name of the scanned computer. Yet another method exists that allows us to define any scanning range. Entering 192.168.32.30-69 as the name parameter we will cause only computers from the network 192.168.32.0 with end addresses from 30 to 69 to be subject to our scanning.

Introduction to Nessus

Nessus is another application worth discussing. This program is similar in functionality to nmap, but it distinguishes itself by an extended error database, updated every day, that is very useful for the user. In addition, Nessus is easy to keep up-to-date, using a plugin system for this purpose. The plugins are created with a special NASL script language.

Information about the application can be obtained on the homepage of the project:

```
http://www.nessus.org
```

Configuration

Nessus requires us to create user profiles for it to work correctly. The program is built based on the client-server architecture, which can be difficult for an inexperienced user to understand at first. And so at each system startup, Nessus starts up the nessusd server process. To use the application we log in using the client program, Nessus. As login and password it is necessary to enter the data set during the scanner configuration. The client is a graphic application, which makes things much easier for less advanced users.

Starting up Nessus

Now let's analyze the practical side of this application. During configuration we will use an administrator account. At the very beginning we will choose as the current directory /usr/local/sbin, to reach the place, where some program binaries are stored.

```
[root@localhost]# cd /usr/local/sbin
[root@localhost]# ls *essus*
nessus-adduser nessusd nessus-rmuser  uninstall-nessus
nessus-check-signature  nessus-mkcert  nessus-update-plugins
```

These applications will be useful to us a little later. The nessus-adduser program, as its name suggests, serves for adding new users authorized to access the server from the client program level. In a moment we will use it to create the first user. Nessusd is the application server we already mentioned, and we will add it to the startup scripts to save ourselves the necessity of starting it up each time. Nessus-rmuser gives us the ability to remove users previously created. Uninstall-nessus allows a complete removal of the application, but leaving the configuration files.

To add a new user it is necessary to create a server certificate, as presented below:

```
[root@localhost]# nessus-mkcert
```

The script will be asking us to enter information regarding the certificate, which verifies the Nessus server. The certificate is created in order to log in to our server using the SSL protocol. We will also see the information assuring us that none of the entered information will be sent anywhere, and that everything we enter is written only to the local certificate. But when connecting to our server, everyone who enters a correct login and password will be able to see this certificate. Therefore we enter all the information required and we start creating a new user.

It is recommended we create only one user for the test purposes. To do this we give the command nessus-adduser.

```
[root@localhost]# nessus-adduser
```

We enter in sequence the user name, authorization method (pass), password, and user rules. We leave the rules field empty, and then we confirm the new settings with the key combination CTRL+D. On the screen a summary will appear and a request to accept the new settings. If all information conforms to our preferences, we accept it.

Creating rules for users

As we have already said, we can define a rule kit for each user at the time of account creation. We enter the principles (rules) in the following way: one principle in one line; after finishing entering the set we press CTRL+D to confirm. Each rule can restrict the scanning of remote computers for a specific user. The syntax is the following:

```
(accept|deny) ip/mask
```

The keyword “default” with the supplement “accept|deny” can also be used at the end, to determine how the server should behave in the event of other

connections. “Deny” causes scanning to be aborted. Here is an example of a set of rules:

```
accept 241.25.52.12/24
accept 192.168.0.1/16
accept 24.57.13.8/24
accept 28.9.32.253/16
default deny
```

For example, if we would like to allow scanning of all computers with the exception of the networks 235.6.11.51/16 and 50.124.15.63/16, we create a similar set. We have to change the default value to accept, in order for each connection to be allowed apart from two forbidden ones:

```
deny 235.6.11.51/16
deny 50.124.15.63/16
default accept
```

There is also a client_ip parameter, which determines the IP address of the computer on which the Nessus program server is running. The scanning can therefore be limited, for example, only to the computer on which nessusd was started up, using the client_ip value instead of the IP address.

Using Nessus

Since we managed to wade through the configuration, it’s time to try the client startup. If, since the configuration, we didn’t restart the computer, it will be necessary to start up the server manually. Thus we will now execute the command:

```
/usr/local/sbin/nessusd -D
```

The next step is to start up the client. From the console level in the graphical environment we give the “nessus” command. We will see a window with an activated tab containing the connection parameters.

If we have started up the client on the same computer on which the server is running, we can leave the Host field set by default to localhost, or else we enter a server address. We should remember that if the firewall blocks access

to port 1241, connections with the Nessus server won't be possible. We leave the port field unchanged, while below we enter the login and the password of the user created a few minutes ago during the server installation. If we don't do this and don't log in, none of the scanning operations will be available. After clicking on log in, the certificate we generated earlier will appear. We accept it by choosing the Yes button. We will see a warning about the possibility of suspension of remote services or even of the whole server, which will be the scanning target. Because of this, some plugins have been switched off. However, it is recommended to start them up to gain complete information about the computer being scanned. Therefore we accept the warning.

Now, let's have a look at the next tab – Plugins. We choose the plugins set appropriate for the machine being scanned. Knowing the target system there is no sense to choose a plugins set that doesn't correspond to it. Therefore, we will now scan the local network. If we have Windows systems in the network, we also select the plugin corresponding to them. We recommend also choosing "CGI Abuses," which detects errors in CGI scripts if they appear. "Denial of service" is a scanning option that detects the susceptibility to DOS (denial of service) attacks, causing service suspension or in extreme cases, the immobilization of the whole system. "SMTP Problems" refers to errors in the SMTP mail transfer protocol. We recommend activating the plugins set from this family. "Gain a Shell" can make the remote terminal available to us, even if we don't have an account on the target server. The "Misc" option allows us to perform an additional analysis of unclassified options – and therefore it's worth taking a look at. "Firewalls" activates checking whether firewall configuration is correct and whether software or hardware errors in the firewall could allow an attack. "Backdoors," as the name suggests, activates the scanning option detecting the presence of a backdoor. "General" means typical common problems. "Netware" is a group of errors that are to be found in the Novell network environment. If the scanning target isn't a system from this family we recommend switching off this option. "Gain root" remotely switches on the analysis of remotely gaining access to the system administrator account. "Remote file access" allows checking the possibility of unauthorized access to important files. Next, we will find some plugin sets regarding specific applications, such as FTP servers. "Peer-to-peer file

sharing" activates scanning for false settings in file sharing applications. "Useless services" scans services that can be exploited for the attack. "RPC" searches for errors in the software of the remote procedure call. "Settings" show errors in the settings of certain software packets. "Finger abuses" concerns errors in server imprints, which normally allow the system to be identified. "Default unix accounts" allows us to check simple but often omitted errors in the configuration of the files /etc/password and /etc/shadow, which by default often grant access to the server to any user with a system or service account. After going through the plugin configuration stage we go to the next tab.

"Prefs." are the program settings. Most of these should be clear, although the most important of them will be discussed later. Nessus uses the nmap application to scan the ports, so its parameters can be set here. We therefore choose the best scanning technique for us. These methods were discussed at the beginning of this chapter. We leave the remaining nmap settings unchanged. Let's move on now to the next tab. These are the scanning options. At the top in the "Port range" field it is necessary to determine the scope of the scanned ports. Below there is an option that treats the unscanned ports as if they were closed. Next, we see a series of options, which we also leave unchanged, namely: the number of computers to scan at the same time, the number of tests at the same time, and the path to the CGI scripts. The next five options introduce the possibility of reverse searching the IP address, test optimization, secure scanning, host identification using the MAC address instead of IP, as well as what is known as "disconnected" scanning, sending the results to a specific email address.

The "Target" tab selection allows us to establish the target of our analysis. The scanning session can also be saved (Save option) and the previously performed tests can bee seen. In the Target field it is possible to include a file with the list of computers to be scanned.

The User tab allows the modification of the user rules.

The next tab allows saving the analysis results to the KB (knowledge base) of the Nessus program for later checking.

After setting all interesting options we click the "Start the scan" button. We will see a window where it is possible to stop the test entirely and preview the results of the analysis performed or to cancel the scanning with the chosen plugin by clicking the Stop button during the test. After terminating the test, we can also familiarize ourselves with the report or save it as HTML.

As the reader will have noticed, Nessus is a program that makes available a vast quantity of information about the analyzed system. Its advantages also include the updates of the plugins, about which we can learn more on the project homepage. Therefore, we should remember to perform frequent updates, using the nessus-update-plugins command.

Attention.
Nessus version attached to the Training Operating System includes a basic database of plugins available under the GNU GPL license. In order to download the current, much more extensive set of plugins, first you must register on the Nessus website.

Nikto

Nikto is a free application that performs a series of WWW server tests. The program detects over 3100 errors in the files of CGI and PHP scripts. The user also receives information on the server version and details regarding possible errors in the installed software. Therefore, we will be notified if an error has been found in the software and an exploit is available that can take advantage of it.

An automatic program update, which updates all data regarding protections, is available. The base is relatively extensive, and the number of plugins available allows the scanner's abilities to be expanded freely.

Nikto enables a relatively quick system analysis. It contains options to enable omitting IDS (intrusion detection system) applications, used to secure servers. As in the case of the previous scanners, versions for different operating systems are available.

Installation

The program is available from the project homepage:

```
http://www.cirt.net/nikto2/
```

We download the packet with the .tar.gz extension and unpack it.

```
[root@localhost]# tar -xzf nikto-current.tar.gz
[root@localhost]# cd nikto-1.34/
[root@localhost]# ls
.  ..  config.txt  docs  nikto.pl  plugins
[root@localhost]#
```

The archive contains a nikto.pl script, which is the main program of the scanner written in the Perl language.

Use and options

A whole series of parameters to help with determining the scanning options is available. Below are descriptions of the most useful ones. A complete description is included in the documentation available under the address:

```
http://www.cirt.net/nikto2-docs/
```

Forced scanning of CGI directories allows forcing of the folder preview in search of the erroneous CGI scripts. This parameter can be defined as "none," which causes the omission of this test, or "all" to scan all CGI directories. It is also possible to enter the value determining the path to the CGI directory, for example "/cgi-bin/." We activate this option with the -Cgidirs or -C parameters.

The option that allows an attempt to omit the IDS system is -evasion. There are nine methods that also call for this parameter.

These should be taken into account during the call. They are:

```
1        Random URI coding (not using Unicode UTF-8 set)
2        Adding to the reference path to the current directory (/./)
3        Premature termination of the URL address
4        Adding a character set generated randomly to the request
5        False filenames in the call
6        Using tab instead of the space character in the file request
7        Random changing of the character size to big or small
8        Using Windows directory separator \ instead of Unix one /
9        Dividing session into fragments
```

To use one of these parameters, or several together, we enter the options together; e.g., "-e 147" when starting up the script, which causes the activation of the functions 1, 4, and 7.

There are two further options, one that will enable the creation of a report output file and another that can set its format. The first is -output (-o), in which the path to the output file should be enclosed. When using the second parameter -Format or –F, it is possible to set the type of the output file, that is HTML, TXT (default), or CSV (file format with values separated by commas used in some databases, although not very popular).

The -generic or -g option gives us the ability to force a full scan. By default the target server adds the Server variable, in which it gives its name and the software version, to the HTTP packets it sends. This is often used by administrators to mislead attackers. On the other hand, switching on the -generic option allows the detection the true identity of the server.

It is also useful to perform analysis with the -mutate or -m option, which will cause the use of different methods in order to detect any possible errors. This enables more information to be obtained, although it significantly slows down script function.

The parameter -port or -p determines the number of the port on which the HTTP server is running. It can also be a port range or a list of ports, separated, in the case of a range, with a dash (e.g., 12-80), and in the case of a list with a comma (e.g., 12,52,80,143). If on a specific port a working WWW server is found, a full scan is performed, unless the -f option is switched off.

By default this parameter assumes the value 80, which is the number of the standard port for an HTTP server.

The -root or -r option determines the path, in which the files of the Internet page that is the target of our analysis are located.

The -Version or -V option prints on the screen the numbers of the program version, the installed plugins, and the databases.

There exist also two parameters that provide much more information, although sometimes can prove useless due to unintelligible results. These are -verbose and -debug. They are useful if, for some reason, the scanning didn't succeed and we want to discover why. This will enable a precise determination of the cause of the error through reporting every performed operation on the screen.

The -update option will be useful, as it causes the newest database and plugins to be downloaded. We can use this option only when we don't enter any other parameter.

-hostname or -h is a required parameter. We use it to set the name of the target of our analysis. Here we can also enter the path to the file containing the list of computers to be scanned. The file must, however, fulfill certain conditions. Each address must be entered in a separate line. If the port isn't entered, a default port (80) will be used. To define the complete information we use the pattern:

```
host:port
```

Whereas if we simultaneously want to check if the server is present on the ports, e.g., 80, 120, and 443, we enter the record below:

```
host:80:120:443
```

Of course, instead of the domain name the IP address can be entered.

Nikto configuration

We configure Nikto by modifying the config.txt file that is located in the main directory. It is made up of pairs of settings of the type name=value. The commentary is marked with the # character before a specific line. Options that assume several parameters as values are separated using spaces. All values apart from predefined ones (preceded by the @ character) are not required and can be set optionally.

Here is the list of the options available in the program:

CLIOPTS – determines the options that will be added to each startup of Nikto as if they were entered manually, for example "CLIOPTS=-f –m." The only limitation is the lack of ability to add options requiring an additional parameter.

NMAP – determines the path to nmap. This causes an increase in scanning speed.

SKIPPORTS – defines ports that shouldn't be scanned by the program.

PROXYHOST – defines the server mediating in the transfer of the packets. Only a host name should be entered, without the beginning of the protocol http://; a specific IP address can also be entered.

PROXYPORT – defines the number of the port used by PROXYHOST to perform the mediation service.

PROXYUSER – we enter the user name, which is available in case the proxy server requires authorization. If we don't enter it, and the proxy server asks for this parameter, the program will show this on the screen and it will be necessary to enter this parameter. Additionally, if the server also requires a password, it is entered in the parameter PROXYPASS.

PLUGINDIR – defines the path to the directory with plugins. Defining this parameter is necessary if the program doesn't find a default folder.

DEFAULTHTTPVER – the number of the HTTP protocol version is entered here that we want to use. The application usually detects it, so setting this parameter can be superfluous.

UPDATES – switches on sending data to the cirt.net server. If we set this value to "no," the program won't send any data to cirt.net, whereas the setting to "auto" will cause an automatic data transfer using the HTTP protocol. If we set this value to "yes," the program will ask each time if the data should be sent. The only information that is sent is the number of the updated version.

MAX_WARN – if the number of the message OK or MOVED is bigger than the set value, the application will inform us about it.

PROMPTS – value that determines the reply to possible queries of the application. If we set the value to "no," we won't receive any questions during the scanner startup, which helps in the automation of the process, but makes the interaction between the user and the program more difficult and reduces control over the actions that are undertaken.

STATIC-COOKIE – defines the content of the cookie that is sent at each startup. It is useful if the target side requires authorization using cookies. We enter here the pair cookie_name=cookie_content.

Here is a list of the predefined values:

@CGIDIRS - defines CGI directories searched for by the program on the target server.

@MUTATEDIRS – here additional folders are defined that can be used if the mode -mutate is activated (excluding folders defined in the database files .db).

@MUTATEFILES – additional files are entered that will be used by the program operating in the -mutate mode (excluding those defined in the database .db).

@ADMINDIRS – contains paths to the directories in which the server administration files are typically located.

@USERS – typical user names for plugins that try to guess the authorization name on the target server.

Practical application of Nikto

We will now turn to the practical applicatios and will perform an analysis of the xxyyzz.com server. Therefore, we start up Nikto and analyze the report without using any additional options.

Below is the information collected by Nikto:

```
[root@localhost]# ./nikto.pl -h xxyyzz.com
-***** SSL support not available (see docs for SSL install instructions) *****
---------------------------------------------------------------------------
- Nikto 2.03     -     www.cirt.net
+ Target IP:       180.180.180.180
+ Target Hostname: xxyyzz.com
+ Target Port:     80
+ Start Time:      Wed Jun 30 14:08:08 2010
---------------------------------------------------------------------------
- Scan is dependent on "Server" string which can be faked, use -g to override
+ Server: Apache/1.3.27 (Linux/SuSE) mod_ssl/2.8.12 OpenSSL/0.9.6i PHP/4.3.1
mod_perl/1.27
- Server did not understand HTTP 1.1, switching to HTTP 1.0
+ Server does not respond with '404' for error messages (uses '400').
+     This may increase false-positives.
- Retrieved X-Powered-By header: PHP/4.3.1
+ PHP/4.3.1 appears to be outdated (current is at least 5.0.1)
+ Apache/1.3.27 appears to be outdated (current is at least Apache/2.0.50). Apache
1.3.31 is still maintained and considered secure.
+ mod_ssl/2.8.12 appears to be outdated (current is at least 2.8.19) (may depend on
server version)
+ OpenSSL/0.9.6i appears to be outdated (current is at least 0.9.7d) (may depend on
server version)
+ PHP/4.3.1 appears to be outdated (current is at least 5.0.1)
```

```
+ mod_perl/1.27 appears to be outdated (current is at least 1.99_14)
+ 2.8.12 OpenSSL/0.9.6i PHP/4.3.1 mod_perl/1.27 - TelCondex Simpleserver 2.13.31027
Build 3289 and below allow directory traversal with '/.../' entries.
+ mod_ssl/2.8.1 - mod_ssl 2.8.7 and lower are vulnerable to a remote buffer overflow
which may allow a remote shell (difficult to exploit). CAN-2002-0082.
+ PHP/4.3.1 - PHP below 4.3.3 may allow local attackers to safe mode and gain access to
unauthorized files. BID-8203.
+ Apache/1.3.27 - Windows and OS/2 version vulnerable to remote exploit. CAN-2003-0460
+ Apache/1.3.27 - Apache 1.3 below 1.3.29 are vulnerable to overflows in mod_rewrite and
mod_cgi. CAN-2003-0542.
+ /~root - Enumeration of users is possible by requesting ~username (responds with
Forbidden for real users, not found fornon-existent users) (GET).
+ /icons/ - Directory indexing is enabled, it should only be enabled for specific
directories (if required). If indexing is not used all, the /icons directory should be
removed. (GET)
+ / - TRACE option appears to allow XSS or credential theft. See
http://www.cgisecurity.com/whitehat-mirror/WhitePaper_screen.pdf for details (TRACE)
+ /index.php?module=My_eGallery - My_eGallery prior to 3.1.1.g are vulnerable to a
remote execution bug via SQL command injection. (GET)
+ /index.php?top_message=&lt;script&gt;alert(document.cookie)&lt;/script&gt;  - Led-
Forums allows any user to change the welcome message, and it is vulnerable to Cross Site
Scripting (XSS). CA-2000-02. (GET)
```

The report, as we can see, is quite long. What can we conclude from it? Let's start our analysis from the very beginning. A version of the Apache server is shown together with the software versions it uses: the module of the SSL encrypted connections, version of the interpreter of the PHP script language, and the Perl language module. Below potential errors detected on the server are shown. We see that the software is obsolete, as newer versions are available. It is evident from this that the administrator doesn't care about the updates or he thinks that the old software offers a good security level.

However, let's have a look at the number of serious errors detected in the application. Among others the mod_ssl module contains a buffer overflow error, which can be used to gain remote access to the server. Information about the difficulty of using this error is given in brackets. The reference to the source from which we can obtain more information on this subject – CAN-2002-0082 is also given. Furthermore, the Apache server, in versions below 1.3.29, is susceptible to an attack on the functions mod_rewrite and mod_cgi, which enable a consecutive buffer overflow and therefore the injection of the remote code. Information on this subject can be very easily found in the network. What else? We have three final errors, which are of less importance than the previous ones, but can also be exploited to take control over the site. First one, thanks to the TRACE option, allows cross-site

scripting (injecting your own code to the site through manipulating transfer parameters), which causes a data outflow. Here an address is also given where details on this subject can be seen. The software My_eGallery is also susceptible to attack, this time through remote call of the commands using the SQL injection method.

We see that even a simple startup of this scanner delivers a huge quantity of information. However, it is necessary to be careful, because some options can cause as many as 70,000 HTTP packets to be generated, resulting in a considerable load on the scanned computer, or in the worst case, causing the service or the whole server to crash.

As always, we would advise the reader to deepen his knowledge of the tools we have discussed.