CHAPTER

9

Web Penetration Testing and Secure Software Development Lifecycle

This topic deserves a chapter by itself because of its importance. These days, most companies have a website or a web application portal that brings in profits. In this chapter, you will mainly learn about the methodology of web application penetration testing and how to use Burp Suite Pro edition.

In the previous chapter, you learned about the most common web vulnerabilities that you will encounter in your engagements. I encourage you to delve deep into the subject by exploring other references (application security books, online courses, and the OWASP website) to understand the rest of the flaws (e.g., server-side request forgery, open redirect, and much more).

This chapter covers the following topics:

- Web pentesting using Burp Suite Pro
- Web application enumeration tools
- Web application manual pentest checklist
- Secure software development life cycle

Web Enumeration and Exploitation

Burp Suite is an excellent tool to have in your repertoire! It allows you to find tons of web application vulnerabilities, and if you want to be a web penetration tester/

bug bounty hunter, then this tool is a must. This section covers the professional edition of Burp Suite, which is not free.

Burp Suite Pro

To summarize this tool in one simple phrase, Burp Suite allows you to use the proxy to intercept and modify the web requests and responses. This tool can scan for web application–based vulnerabilities and much more (it's not just a tool; it's a beast). Burp Suite has different tabs and functionalities, so let's see what you can do on each one.

NOTE **The company that created Burp Suite is called PortSwigger. Burp Suite comes in three flavors:**

- **Community edition, which is pre-installed on Kali**
- **Professional edition, which supports all the manual web pentesting features and costs $399 USD per year**
- **Enterprise edition, an automated scanner that can integrate into the pipeline of CI/CD (DevOps) and costs $3,999 USD per year**

Web Pentest Using Burp Suite

There is nothing better than a practical penetration test example to show you all the functionalities inside Burp Suite Pro. In this example, you will learn the following:

- How to run Burp Suite Pro
- Understand the proxy
- How to use the target tab
- Understand the repeater
- How to use the intruder
- Utilize the Extender app store

Loading Burp Suite Pro

For this example, we will be using the Mutillidae web application and scanning it with Burp. To start this tool, download the JAR file for the Pro edition from your account at `portswigger.net/`. To run the tool, use the following command:

```
$java - jar [burp suite file name.jar]
```

Once Burp has started, you will encounter two windows that allow you to do the following:

- Save the project on disk before you start
- Create a temporary project without saving it on disk
- Open an existing saved project
- Load a custom saved configuration file of Burp Suite

Before you switch to a browser, click the Proxy tab and click the Intercept Is On button to disable the interceptor. Note that Burp will continue listening to web requests/responses on the back end.

In your web browser, make sure to change the following items:

1. Change the browser settings and set the proxy to port 8080 (we covered how to do this in Chapter 6,"Advanced Enumeration Phase.).
2. Make sure that the browser contains the Burp Suite HTTPS certificate. It needs to trust this certificate to intercept HTTPS communication. To do this, enter the URL **http://burp** in your browser and click the CA Certificate button to download it (see Figure 9.1).

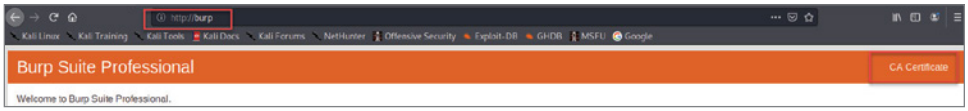

Figure 9.1: Burp Suite Certificate

Once you've downloaded the certificate, open your browser and select Preferences ➪ Privacy & Security ➪ Certificates ➪ View Certificates ➪ Import.

When you click Import, it will ask you to choose the certificate file (which is in your Downloads folder). Next, make sure to select the two check boxes (see Figure 9.2) to trust the uploaded CA certificate; finally, click OK.

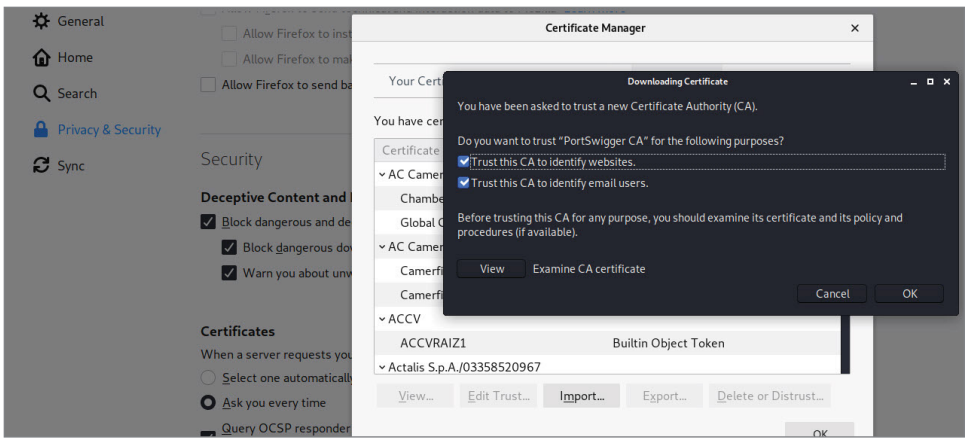

Figure 9.2: Importing the Burp Suite Certificate

Burp Proxy

When you start Burp Suite, consider the following:

- The intercept button is selected by default, so you will need to turn it off unless you want to stop every request/response to the web server. Note that the tool will keep saving all the communications in the back end (there is a HTTP History subtab available on the Proxy tab).
- In the Options section (see Figure 9.3), usually you will be changing three settings.
  - Changing the listening port number (by default, it's 8080)
  - Intercepting the web requests (it's enabled by default)
  - Intercepting the web responses coming back from the server (by default, it's disabled)

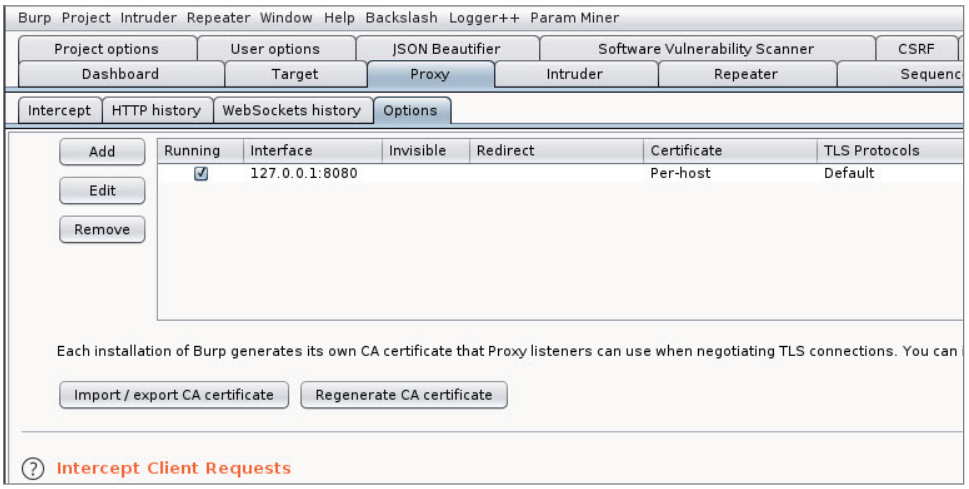

Figure 9.3: Burp Suite Proxy Tab, Options Section

Target Tab

Now it's time to start browsing all the web pages in the Mutillidae application while the intercept button is disabled. Once you finish, click the Target tab to visualize the domain name that you're trying to test.

On the Site Map subtab, you can see that Burp Suite has detected the structure of Mutillidae (refer to Figure 9.4). In this section, you can visualize these site items: HTML pages, images, folders, JavaScript, etc. Next, on the left side of it, you can select the history entries to read the request/response details below it (in the windows where you see the Request and Response tabs). On the far-right side of the screen, Burp Suite will identify passively any security issues,

and at the bottom right, you can see the details of the selected issue (later when you run the vulnerability scanner, the flaws will be listed here as well).

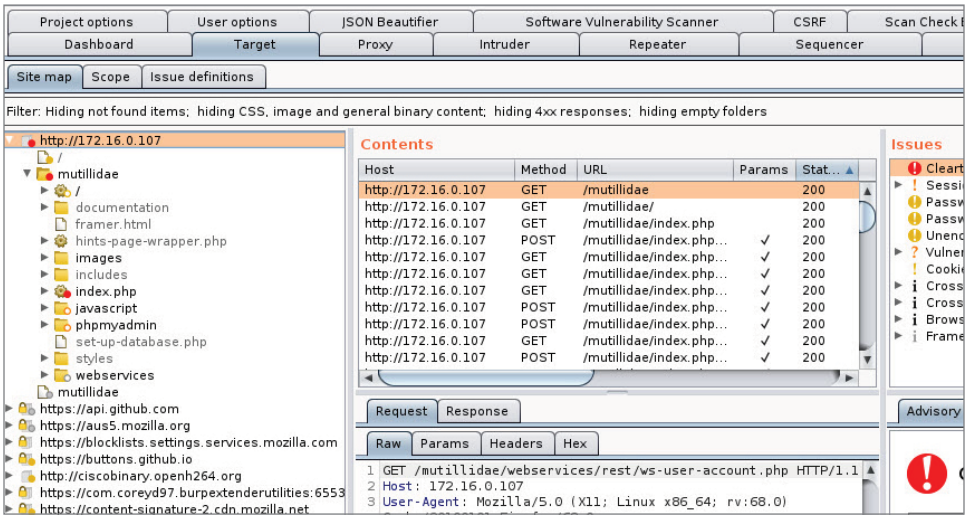

Figure 9.4: Burp Suite Target

At this stage, configure Burp Suite to include Mutillidae in the scope; in other words, you don't want it to scan the other domains previously detected. To add the URL to the scope, right-click the root node of the domain (in my case, the root item is `http://172.16.0.107`) and then click the Add To Scope menu item (see Figure 9.5).

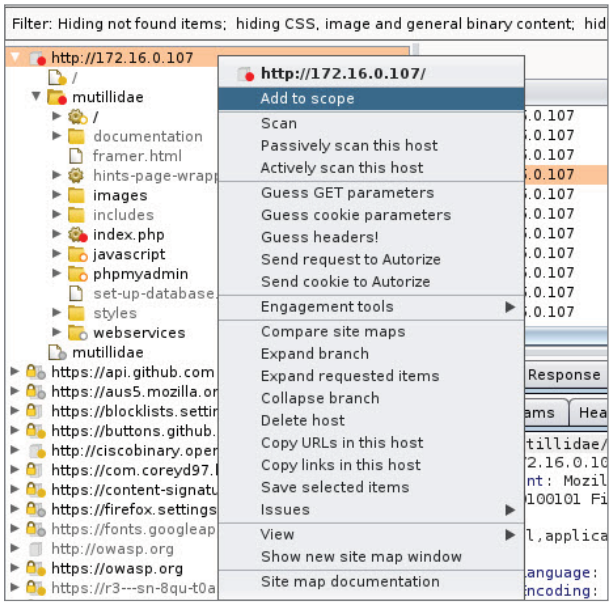

Figure 9.5: Burp Suite Add To Scope Option

Next, click the filter button (where there's the text "Filter: Hiding not found items . . .") and select the check box Show Only In-Scope Items (see Figure 9.6).

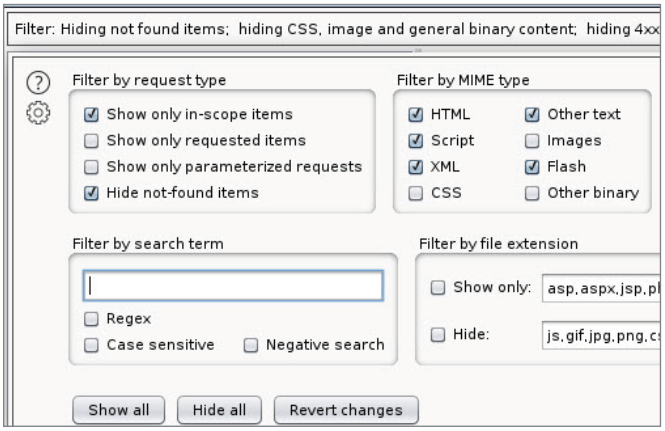

Figure 9.6: Burp Suite In-Scope Filter

To hide this screen, just click anywhere outside of it, and you will see that all the out-of-scope items disappear (see Figure 9.7) from the Site Map tab (only Mutillidae will remain).

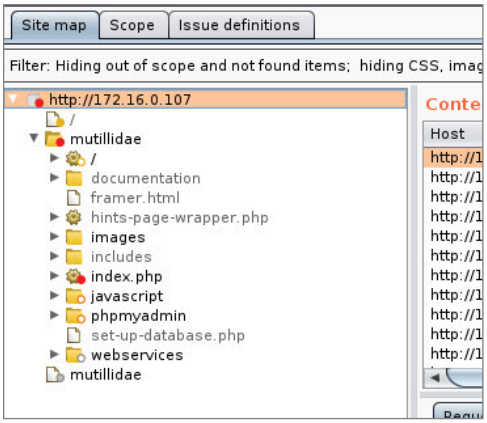

Figure 9.7: Burp Suite In-Scope Filter Applied

Enumerating the Site Items (Spidering/Contents Discovery)

Before starting this section, you need to know that Burp Suite needs a lot of memory to run, or else you will end up with runtime errors in the middle of your engagement.

At this stage, Burp Suite has already found a few items during our manual browsing (they're grayed out). Next, run Discover Content to scan for

more items. Right-click the IP address (root item) of the Mutillidae tree and select Engagement Tools ➪ Discover Content (see Figure 9.8).

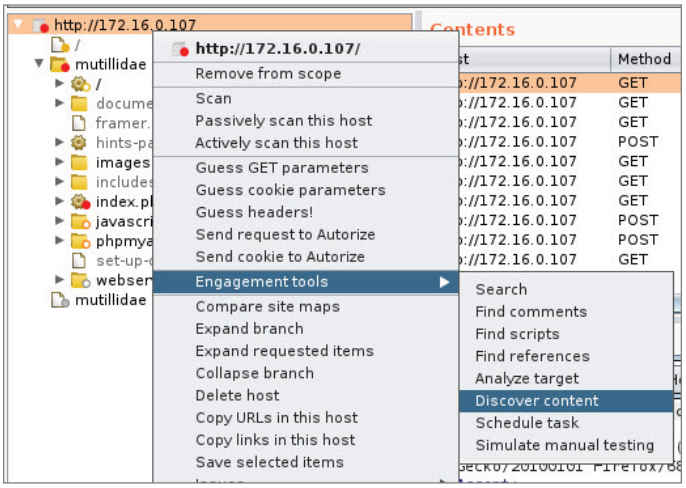

Figure 9.8: Burp Suite Discover Content Menu Item

Once the content discovery window is open, click the Config tab to add the proper configurations. In this section, set the maximum depth of the subdirectories to 2 to finish the task faster. In general, when I run Burp Suite in a real engagement, I choose the following options (depending on the structure of the website):

- Max Depth: 5
- Number Of Discovery Threads: 4
- Number Of Spider Threads: 4

Next, select the Control tab and click Session Is Not Running to execute Burp Suite (see Figure 9.9).

This task will take a few hours to complete (depending on the website's structure). When there are no more queued tasks, click the Session Is Running button to stop the spidering and go back to the Site Map tab to inspect the results.

Automated Vulnerabilities Scan

Now it's time to start finding some vulnerabilities in Mutillidae. Burp Suite Pro has an internal scanner that can find flaws. To get the job done, right-click the root domain name in the Site Map tab and select Actively Scan This Host. Next, select the Dashboard tab to visualize the progress of the scan results (see Figure 9.10).

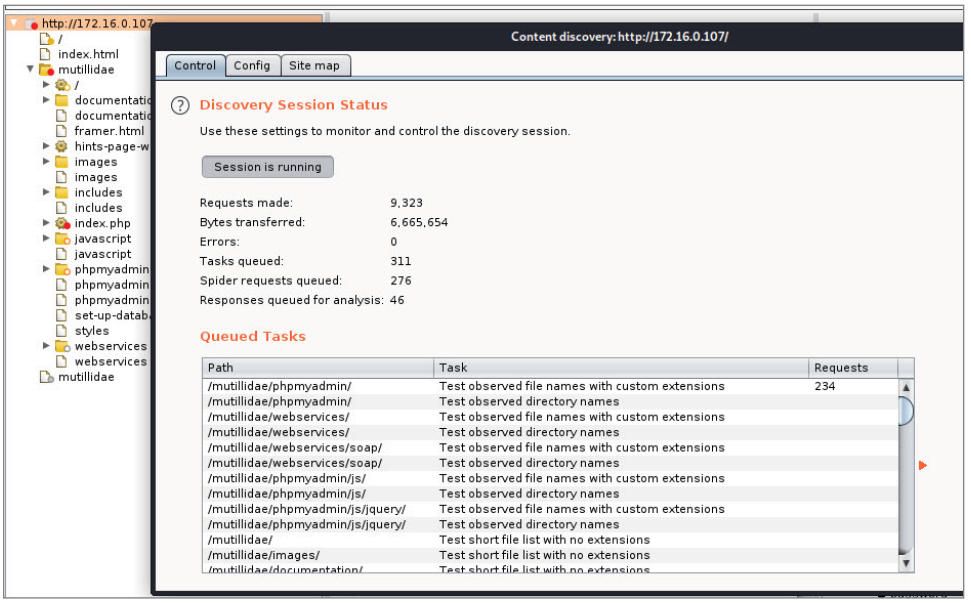

Figure 9.9: Burp Suite Running Discover Content Feature

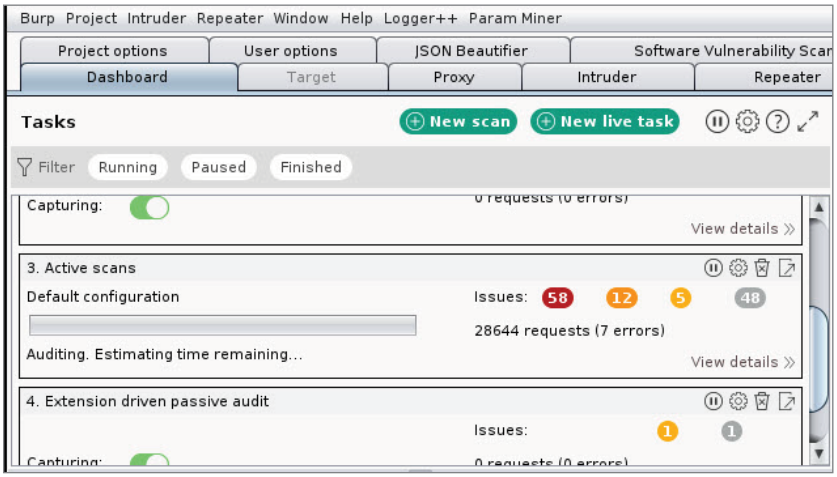

Figure 9.10: Burp Suite Active Scan

The Repeater Tab

This tab is helpful when you want to manually inject a payload before sending it to a web server. The payload could be any type of input you choose, so you can try to verify how the web server will respond. Most of the time, you can enable the interception and then in the intercept window, right-click the request and click Send To Repeater (see Figure 9.11) from the menu.

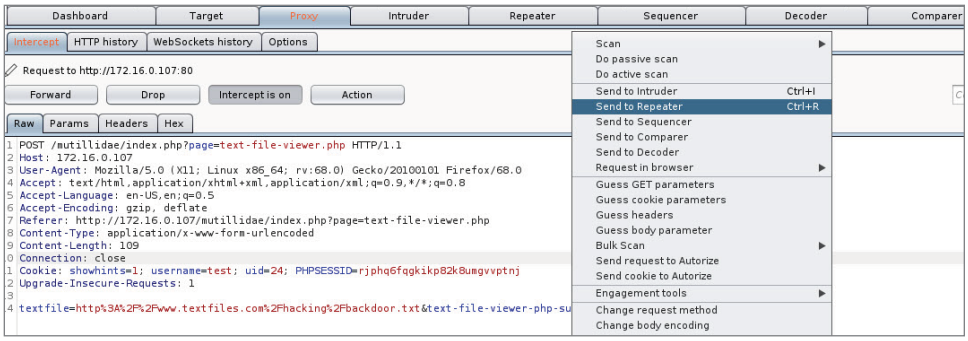

Figure 9.11: Burp Suite Send To Repeater Menu Item

Once the request is on the Repeater tab, you can manipulate it any way you want. In this example, we will change the UID value to 1 in the cookie header and watch for the response. After clicking Send, the response header will indicate that you are logged in as the admin (see Figure 9.12).

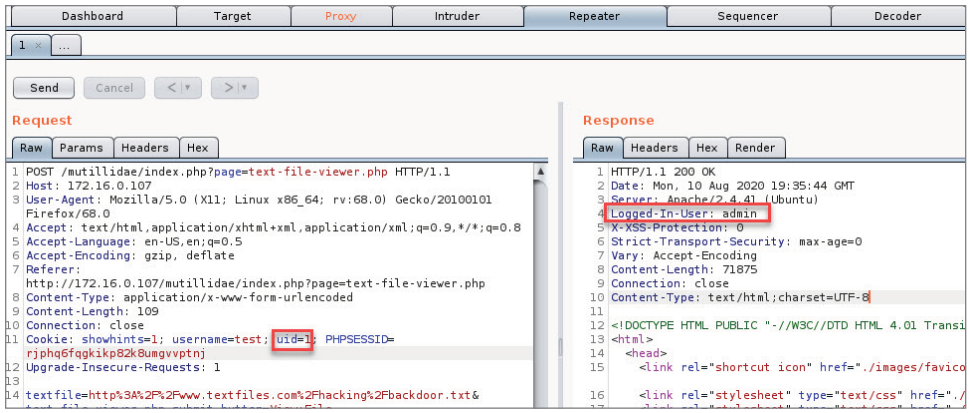

Figure 9.12: Changing the UID Param

The Intruder Tab

The Intruder tab allows you to fuzz and automate web requests sent to the web server. There are a lot of tasks that you can do with the intruder including the following:

- Brute-forcing login pages
- Enumerating usernames
- Enumerating any UID type (e.g., AccountID, ProductID, EmployeeID, etc.)
- Manual crawling (to find web pages, directories, files, etc.)
- Fuzzing to find flaws (e.g., SQLi, XSS, LFI, etc.)

Now that you know what you can do with the Intruder tab, the next challenge is to understand the attack types:

- **Sniper**: This attack type is used with one payload. There is a lot of practical examples that can be used with the sniper method (e.g., enumerating usernames, UID, finding files/directories, etc.).
- **Battering ram**: This attack type will allow you to use one payload. In the battering ram type, you can insert the same payload into multiple areas in the web request (e.g., inserting the same payload into the URL and the web request header as well).
- **Cluster bomb**: This type will allow you to insert multiple *unrelated* payloads into the web request (maximum 20). A good example is the login page brute-force; in this scenario, you will need to insert two different payloads: one for the username and one for the password.
- **Pitchfork**: This one will allow you to insert multiple *related* payloads into the web request (I've rarely used this attack type). A practical example is fuzzing the employee name and its associated UID in another field.

Let's see, together, a practical example to brute-force the login page of Mutillidae using Burp Suite's Intruder tab. To get the job done, submit the login page to Burp Suite (for this example, enter **admin** for the username and **password123** for the password) and intercept it using the Proxy section. Once the web requests are intercepted, right-click and select Send To Intruder from the menu.

On the Intruder tab, make sure the Positions subtab is selected (see Figure 9.13).

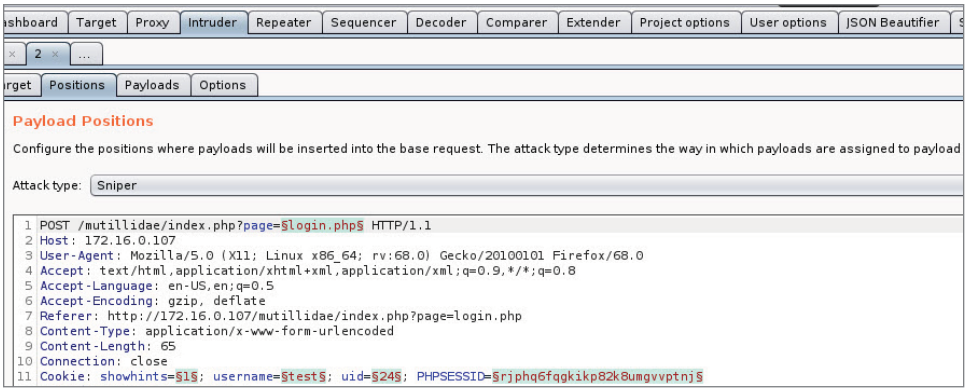

Figure 9.13: Burp Suite Intruder Positions Subtab

By default, the sniper attack type is selected, and Burp Suite already identified the insertion points. For this example, brute-force the admin username

(static) and load the password from a dictionary file (variable payload). In other words, you will need only one payload, which is the password value. Next, choose the Sniper attack type (keep it because it's already selected) and click Clear to remove all the insertion points. To select your payload, highlight the password123 value and click Add to insert your variable (see Figure 9.14).

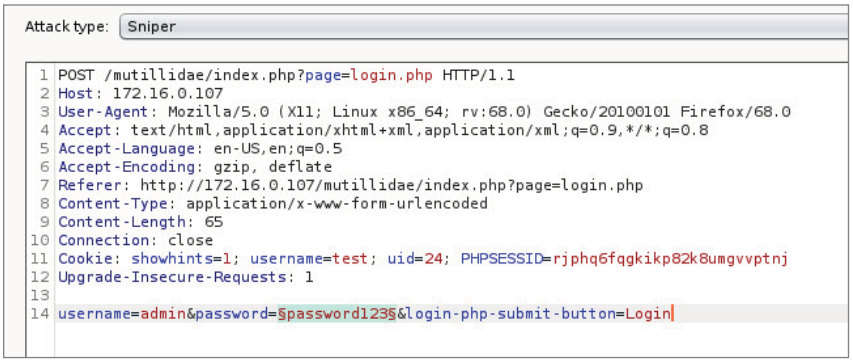

Figure 9.14: Burp Suite Intruder Payload

At this stage, click the Payloads tab and keep the payload type as Simple list. Next, click the Add From List drop-down menu and choose Passwords. You can start with this one, but for this example, we will use a custom dictionary file by clicking Load. (see Figure 9.15). Once the passwords are loaded, click the Start Attack button located on the top-right corner.

When the attack is started, a new window will open to show you the progress of the passwords tested. In our case, the candidate password adminpass (see Figure 9.16) has returned a 302 redirect (note that the previous ones are all 200) and a response length of 373 (different than the others).

Burp Extender

This tab will allow you to add more tools and functionalities to Burp Suite. For some of them, you are required to have the Pro version. For example, one of the modules that you can add is Retire.js, which scans for outdated JavaScript libraries. There is a tool for every occasion: .NET, Java, JavaScript, JSON, XSS, and CSRF, to name a few.

Before using the BApp Store, you will need to download the latest version of the Jython JAR file from `www.jython.org/download.html`.

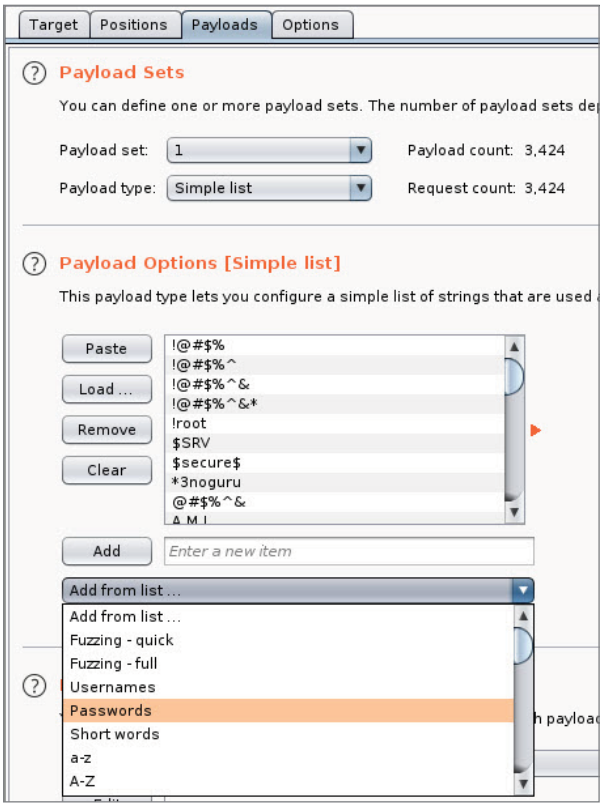

Figure 9.15: Burp Suite Intruder Payload Option

Some apps are required to have Jython, which is a combination library of Python and Java. Once you download the Jython JAR file, head to the Extender tab, select the Options subtab, and locate the path where you saved the JAR file (see Figure 9.17; your path might be different).

To install a Burp tool, you will need to select the BApp Store tab. In this window, you will see all the available applications to choose from. To install an app, you must select it from the list and then click Install on the right of the window (see Figure 9.18).

The biggest question is how to choose an app from the list. I choose the application based on the following criteria:

- Has a rating of four stars or above
- Has a high popularity score
- The tool functionalities of the app will help during a pentest

Your criteria might be different.

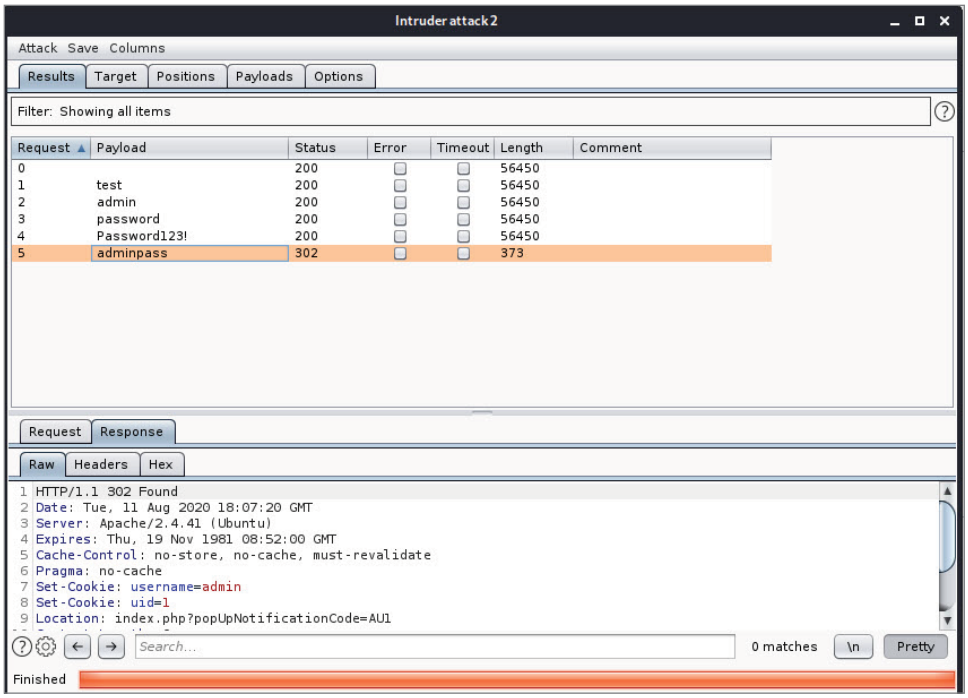

Figure 9.16: Burp Suite Intruder Attack

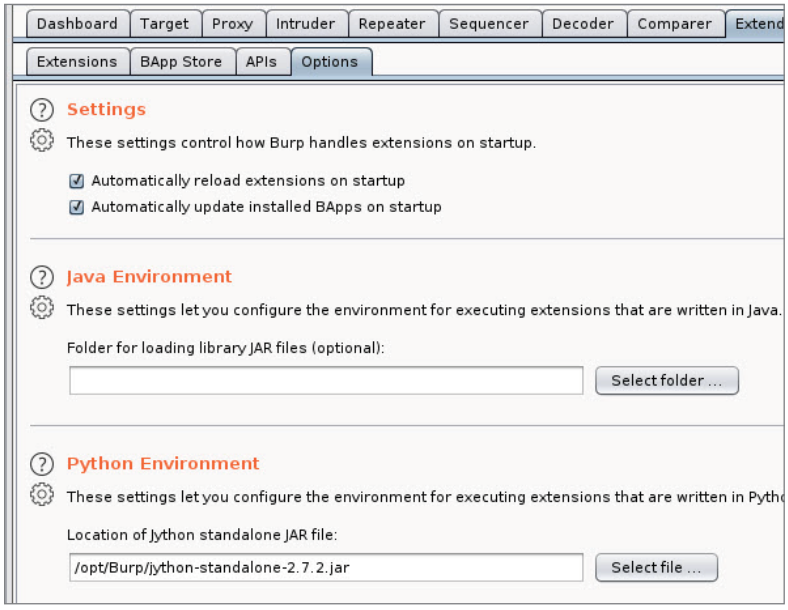

Figure 9.17: Burp Suite Extender Tab

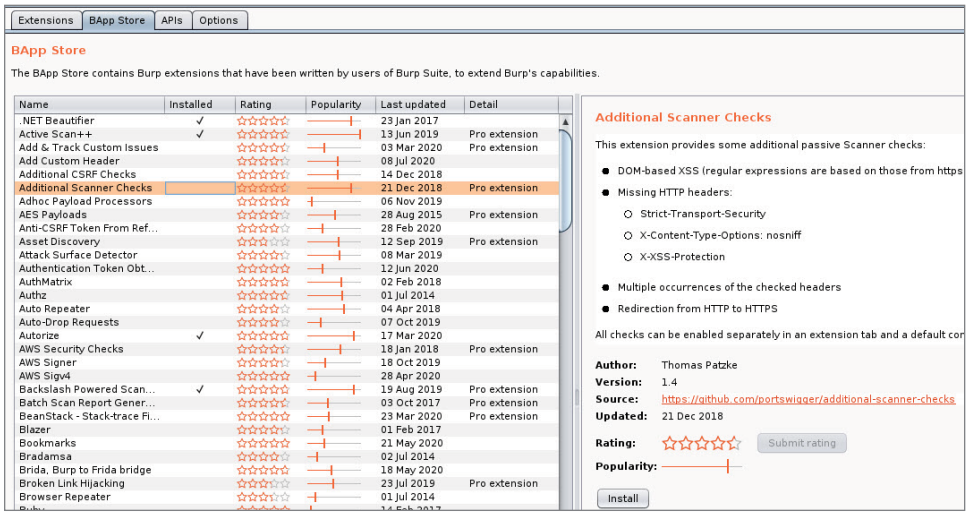

Figure 9.18: BApp Store

Creating a Report in Burp

Once you've finished your pentest in Burp, make sure to save your work and export a report of the flaws found. To create a report in Burp, you will have to be in the Site Map tab and select the root domain item (in my example, it's the Mutillidae server's IP address). Next, right-click, select Issues from the menu (see Figure 9.19), and click Report Issues For This Host.

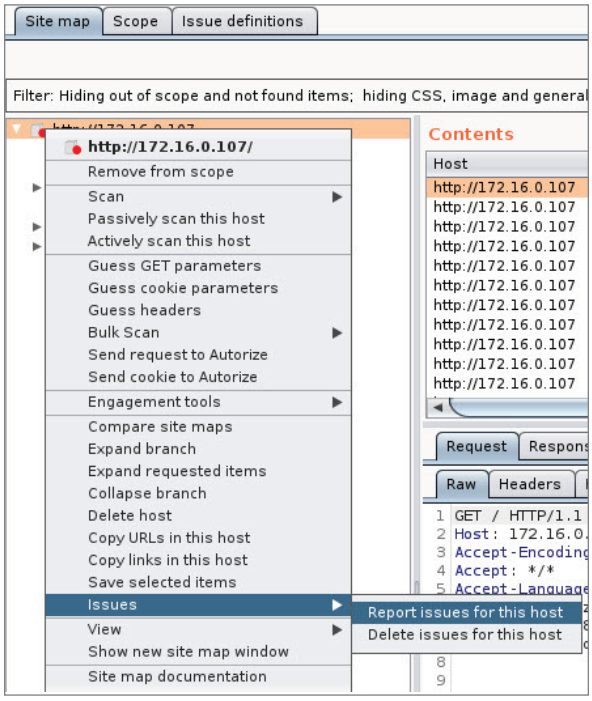

Figure 9.19: Creating Reports in Burp Suite

Once you click the menu item, a new pop-up window will appear. At this stage, you will need to follow up on the steps in the wizard. Once you're done, Burp will generate a nice-looking HTML report (see Figure 9.20).

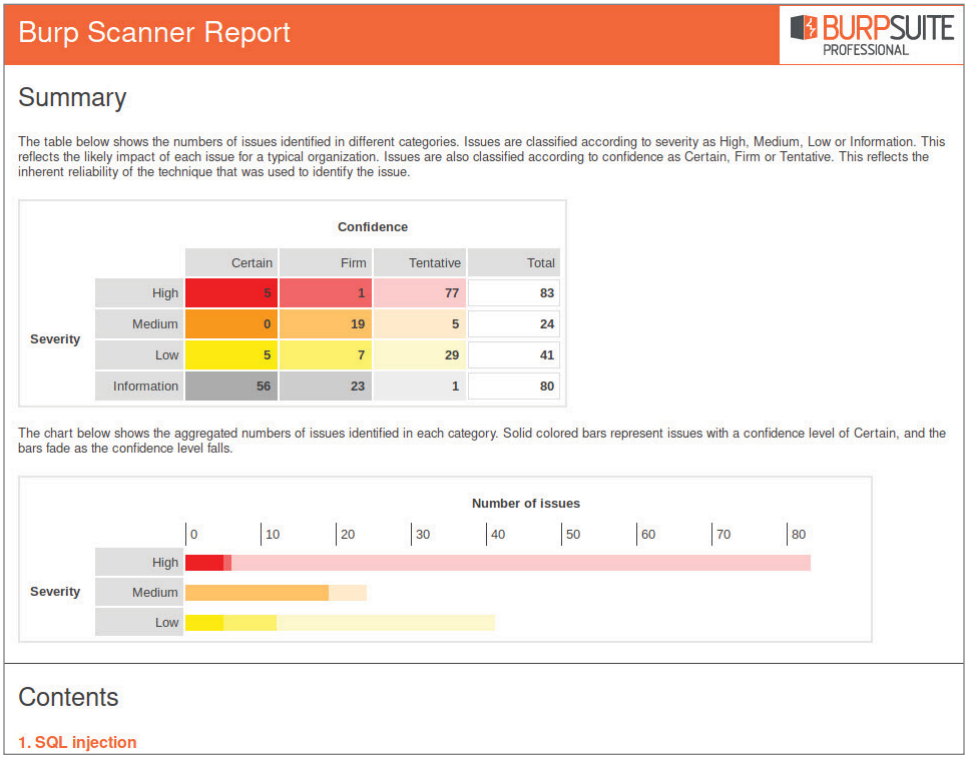

Figure 9.20: Report Sample in Burp Suite

Burp classifies the severity of the flaw into High, Medium, Low, and Information. What is great about this tool is that it gives you the confidence level (false positive probability) of each one. At this stage, your role is to test each vulnerability and make sure you don't report a false positive. In other words, don't just copy the report and send it to your employer/client without verification. (To verify the false positive, you can use the Repeater tab in Burp to reproduce the web requests and test the injected payloads.)

More Enumeration

When it comes to a web application, the enumeration phase is a little bit different than other TCP protocols. In this section, you will get a summary of the tools that you can use to enumerate a web application. At this stage, we're looking to achieve the following tasks:

- Crawl (a second time) the files on the web server

- Identify if the web server version is old and has a critical vulnerability
- Detect any missing configuration that can lead you to unauthorized resources

Nmap

It's a good idea to start with Nmap when it comes to enumeration. We can use the scripting engine that we saw earlier in this book:

```
$nmap -sV -p 80 -sC [IP address]
```

Crawling

Crawling is an important step to identify hidden contents inside the web server. People often deploy unnecessary configuration files and secrets into the production environment. An excellent tool for this purpose is called GoBuster (you can use this tool along with Burp Crawler to double-check if you missed any hidden spots):

```
root@kali:~# gobuster  dir -u http://172.16.0.107/mutillidae/ -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -e -t 25
===============================================================
Gobuster v3.0.1
[...]
===============================================================
2020/08/12 08:04:37 Starting gobuster
===============================================================
http://172.16.0.107/mutillidae/documentation (Status: 301)
http://172.16.0.107/mutillidae/ajax (Status: 301)
http://172.16.0.107/mutillidae/test (Status: 301)
http://172.16.0.107/mutillidae/includes (Status: 301)
http://172.16.0.107/mutillidae/javascript (Status: 301)
http://172.16.0.107/mutillidae/classes (Status: 301)
http://172.16.0.107/mutillidae/styles (Status: 301)
http://172.16.0.107/mutillidae/webservices (Status: 301)
http://172.16.0.107/mutillidae/images (Status: 301)
http://172.16.0.107/mutillidae/passwords (Status: 301)
http://172.16.0.107/mutillidae/configuration (Status: 301)
http://172.16.0.107/mutillidae/phpmyadmin (Status: 301)
===============================================================
2020/08/12 08:04:51 Finished
```

Note that GoBuster is not installed by default on Kali Linux. To install it, you will need to execute the `apt` command:

```
$apt install gobuster -y
```

Vulnerability Assessment

To find vulnerabilities in the middleware (web server), you will need to run a vulnerability scanner. You learned in Chapter 7, "Exploitation Phase," how to use OpenVAS to scan for vulnerabilities. In this section, you will learn about another quick tool that can scan for flaws in the web application, and it's called Nikto:

```
root@kali:~# nikto -host http://172.16.0.107/mutillidae/
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          172.16.0.107
+ Target Hostname:    172.16.0.107
+ Target Port:        80
+ Start Time:         2020-08-12 08:15:56 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.41 (Ubuntu)
+ Cookie PHPSESSID created without the httponly flag
+ Cookie showhints created without the httponly flag
+ The anti-clickjacking X-Frame-Options header is not present.
+ X-XSS-Protection header has been set to disable XSS Protection. There is
unlikely to be a good reason for this.
+ Uncommon header 'logged-in-user' found, with contents:
+ The X-Content-Type-Options header is not set. This could allow the user agent
to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ "robots.txt" contains 8 entries which should be manually viewed.
+ OSVDB-630: The web server may reveal its internal or real IP in the Location
header via a request to /images over HTTP/1.0. The value is "127.0.1.1".
+ Allowed HTTP Methods: GET, POST, OPTIONS, HEAD
+ Web Server returns a valid response with junk HTTP methods, this may cause
false positives.
+ DEBUG HTTP verb may show server debugging information. See
http://msdn.microsoft.com/en-us/library/e8z01xdh%28VS.80%29.aspx for details.
+ /mutillidae/index.php?page=../../../../../../../../../../etc/passwd: The
PHP-Nuke Rocket add-in is vulnerable to file traversal, allowing an attacker to
view any file on the host. (probably Rocket, but could be any index.php)
[...]
```

Manual Web Penetration Testing Checklist

In your manual inspection, you will encounter a lot of repetitive tasks for each web page. Some special pages (e.g., login page, register page, etc.) have additional

checks. Let's divide this checklist into two parts, one for the common scenarios and one for the exceptional ones. In the engagement for our example, we'll ask for two types of accounts: one with low-privilege and one with admin access.

Common Checklist

You'll apply this list to any type of web page (even the exceptional ones). Let's get started:

1. Stop each web request on your Proxy Intercept tab and send it to the repeater tab for inspection.
2. Identify the entry points to the backend server, as shown here:
   - URL (look for query string)
   - Request header (inspect the cookie, etc.)
3. Inject special characters into the entry points and inspect the web response (you can use Burp's repeater or intruder to get the job done). Also, read the error message to see if it displays juicy information.
   - **For SQLi**: Insert a single quote
   - **For XSS**: Try to insert a script tag <script> alert(1) </script>
4. Manipulate the behavior of the web page (e.g., enter a negative number in the shopping cart field, etc.).
5. Use Burp Target/Site map to locate it:
   - Hidden files (e.g., Robots.txt, backup files, text/PDF files, debugging leftover, admin portal)
   - Unidentified scope targets
   - Web API/SOAP calls
   - CMS platform (e.g., WordPress)
6. Inspect the HTML logic to find any JavaScript/HTML flaw.
7. Try to call admin resources using a low privilege account.
8. Try to call other user resources (e.g., an image of another user).
9. Try to call a protected resource without authentication.
10. Use the Intruder tab to fuzz any input type (e.g., UID).

Special Pages Checklist

In some pages, you will need to execute additional tests to the ones listed before. The nature of these pages is different; thus, we need to run additional tests.

Upload Page

When you encounter an upload page, you should test these items:

- Make sure to send the web request to the Repeater tab in Burp.
- Can you bypass the validation (check the previous chapter for more information)?
  - Change the file extension.
  - Change the content type.
  - Change the binary contents.

Login Page

The login page, if bypassed, will allow accessing unrestricted resources. Let's see how we can test it:

- Try the default credentials (e.g., username=admin and password=admin).
- Brute-force it.
- Use SQL injection to bypass the login page.
- Can you enumerate usernames by logging in with an unregistered user?

User Registration

Registering a user will allow you to log in to the portal and explore what's inside.

- Register with an existing user and check the error message (this will allow you to enumerate users).
- Register with a weak password.
- Test for SQL injection (remember that this form is creating a new user record in the database).

Reset/Change Password

Here is another interesting form that interacts with the user's records in the back end. Check the following items manually:

- Try to reset an existing user to understand the workflow of the password reset.
- Can you change someone else's password?
- If the system generates a random temporary password, check for the following:
  - The complexity of the password.

- Its lifetime (timeout expiry).
- Is the user enforced to change it after the first login?

Secure Software Development Lifecycle

Every web/mobile application goes through different phases before being deployed into the production environment. This is an important topic in application security and penetration testing at the same time. Big companies do not manage just one application; sometimes it's more than 100. This is not an exaggeration; in fact, it's normal in an enterprise environment. Note that this section is for full-time penetration testers who work at a company (either an employee or a consultant) and supervise the projects from start to finish.

Figure 9.21 shows the steps that a typical project goes through in each phase of a software development lifecycle (SDLC).

Figure 9.21: Software Development Lifecycle

In this section, you will learn about security analyst roles and duties in each phase of SDL. The purpose of this process is to turn it into a secure software development life cycle (SSDL). A picture worth a million words, so Figure 9.22 shows the summary of SSDL.

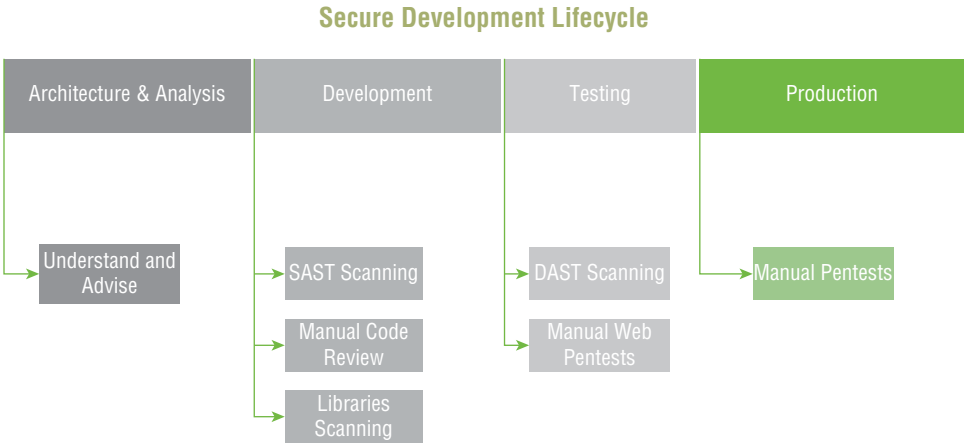

Figure 9.22: Secure Development Lifecycle

Analysis/Architecture Phase

At this stage, the project is still fresh, and everyone is laying out the foundation of the business cases. As a security professional, you will be attending kick-off meetings to understand and provide the necessary advice (the project team members) in advance before developing the product. This is where you start planning for later tests and prepare ahead for implementing the security tools in later stages. Speaking about planning, it would be good to prepare a security architecture document (e.g., threat modeling) ahead of time (see the next section for how to prepare an application threat modeling document).

Application Threat Modeling

Application threat modeling will allow you to analyze the posture of an application and help you to design the attack scenarios before starting on the security tests later in the project. An application threat modeling document will mainly contain the following sections:

- Assets
- Entry points
- Third parties
- Trust levels
- Data flow diagram (DFD)

Before you proceed with the upcoming sections, you will need to ask yourself the following questions (to properly evaluate the project):

- Is the application visible to the internet or just the intranet?
- Does the application store confidential data (e.g., personal identification information, aka PII)?
- How is the application consumed (e.g., is it only a web app?)?
- Are there any third-party entities (e.g., cloud services or external providers)?
- Do you have the infrastructure diagram? (If not, ask for it.)

Assets

Assets are the items that attackers (hackers) are looking to steal. The most obvious example is confidential data, such as credit cards, insurance numbers, client personal information, etc. (PII). In summary, here's the list of infrastructure assets a hacker can exploit and you should consider when working on this section:

- Networking devices
- Host servers

- Middleware
- Application level (web/mobile)

Entry Points

Like the name says, entry points are the entryways from which an attacker can interact with an asset. Remember that an asset is not only the web application, but it could also be a database or the host itself (or VMs/containers, etc.).

Third Parties

In this section, you list the third-party items that the application will interact with. A common example is the cloud such as Microsoft Azure or Amazon Web Services (AWS), etc.

Trust Levels

After identifying all the assets components including the third-party items, then you must go over the authentication/authorization of each one of them. An example is a rogue administrator (insider) that reads clients' confidential data in the case where some records are not encrypted.

Data Flow Diagram

The network diagram will show, visually, everything you have gathered so far. For example (see the network diagram in Figure 9.23), our company has a website and a mobile app. Data is consumed by a separate web API server and finally stored inside the database.

You can simplify the network diagram above by using a data flow diagram (DFD) to visualize the components all together (see Figure 9.24).

Development Phase

At this stage, the project is already approved (by the architecture board) and moved to the development phase. This is where the website or mobile app features are developed using programming languages like C# .NET, Java, Angular, Swift, etc.

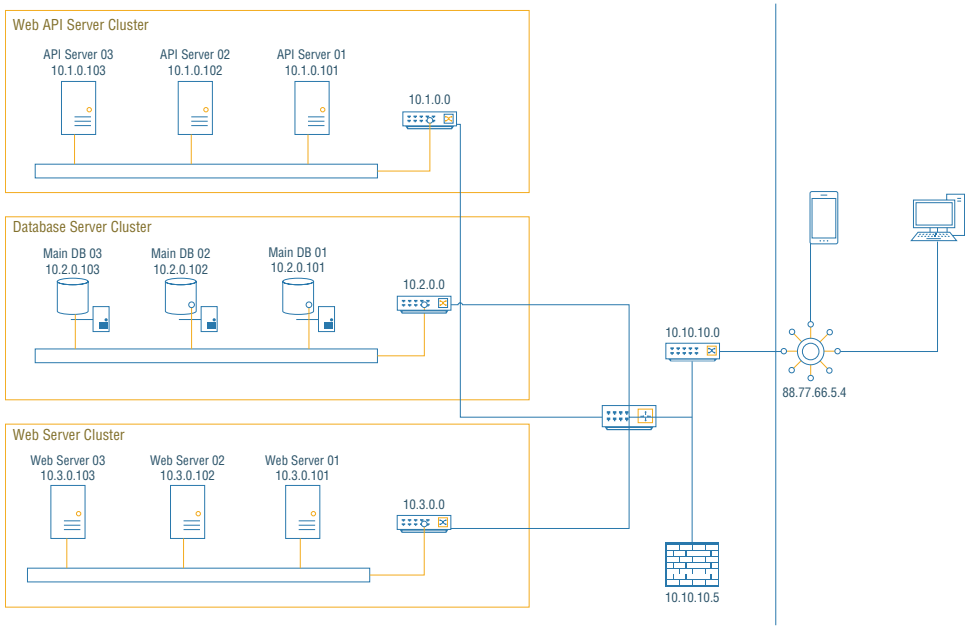

Figure 9.23: Network Diagram

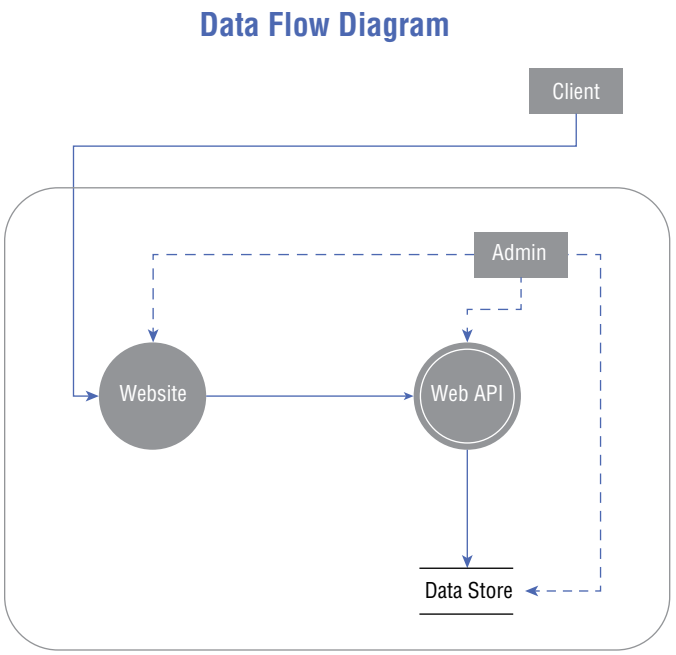

Figure 9.24: Data Flow Diagram

During the development phase, the team will use a build server, like Jenkins or Team Foundation Server (nowadays TFS is called Azure DevOps). This build server will orchestrate the project source code; hence, we have continuous integration (CI) and continuous deployment (CD). In the development phase, you must ensure the following:

- The source code is scanned regularly using a static application security testing (SAST) automated tool (e.g., Veracode, Checkmarx, etc.). The SAST scanner will spot security flaws inside the source code once it's submitted to the build server (Jenkins, TFS, etc.). You must enforce this good habit to the project so they can spot issues at the beginning during the development of the product (instead of doing this scan at the last minute before deployment into the production server).
- Manual code review is essential because an automated tool is not enough to cover 100 percent of the source code. After finishing a new feature (by the dev), you must go through this step.
- Most of the time, the source code uses open source third-party libraries (the goal is to add more functionalities to the application). In this case, it's your duty to check for these three items:
  - Is the library containing any vulnerabilities?
  - Is it up-to-date?
  - Is it legal to use it for commercial use?

To get the job done, there are automated scanners (e.g., Sonatype Lifecycle) that can check these three items.

Testing Phase

After finishing a few product features, the project is ready to deploy into the test (QA) server. The deployment goal is to have a separate stable server where testers can go and test the web/mobile application. At this stage, you, as a security professional, should check the following items:

- It's your chance to go and learn the UI new functionalities.
- Use Burp Professional edition to manually test for vulnerabilities (web pentests) of the new features.
- Use a Dynamic Application Security Testing (DAST) tool (e.g., Burp Suite Enterprise Edition, Acunetix, etc.) to automate the web pentests. This task is implemented at the testing phase because you need a running website (so it must be deployed to a stable environment, prior to running the tool). Note that you can automate this task using an orchestrator (Jenkins or TFS) to trigger the scan once the application is deployed to a test environment.

Production Environment (Final Deployment)

Once the code is deployed into the production environment, the website/mobile app will be under the responsibility of the operation security (OpSec) team. OpSec will take care of monitoring and doing regular pentests (generally it's once per year) on the infrastructure and web pentests as well on the application itself. Lately, bug bounty programs came into place, and companies started to hire external bug bounty hunters to look for any missing flaws that were not spotted during all the prior phases.

Summary

More than 80 percent of a cybersecurity consultant's time is likely spent on web application pentests. You are highly encouraged to master this topic before you proceed further in this book. The simple formula to master any skill is based on two factors: knowledge and practice (the inherited skill is a minor factor as well). The information shared with you in this chapter should help you to start testing web applications like a boss.

What's next? In the upcoming chapter, we will dive deep into the Linux privilege escalation concept.