



# Burp Suite: Dashboard

[Introduction](#)

[Event log](#)

[Scans](#)

[Scan details](#)

[Scan types - Crawl](#)

[Scan types - Crawl and audit](#)

[Scan types - Audit selected items](#)

[Protocol settings](#)

[Scan configurations](#)

[Application login](#)

[Resource pool](#)

[Live task](#)

[Issue activity + Advisory](#)

## Introduction

The screenshot displays the Burp Suite dashboard with the following components:

- Tasks Panel:** Shows two tasks. Task 1, 'Live passive crawl from Proxy (all traffic)', is running and has added 428 items to the site map and processed 660 responses. Task 2, 'Live audit from Proxy (all traffic)', is paused and has 4 requests with 0 errors.
- Issue activity Panel:** A table listing security issues found during the scan. The table has columns for #, Task, Time, Action, Issue type, and Host. Issues include TLS cookie without secure flag set, HTML does not specify charset, Vulnerable JavaScript dependency, Cross-domain script include, Mixed content, File upload functionality, Private IP addresses disclosed, Strict transport security not enforced, and Cacheable HTTPS response.
- Event log Panel:** Shows a log of events. The first entry is 'Proxy service started on 127.0.0.1:8080' at 00:17:48 on 14 Mar 2021.
- Advisory Panel:** Currently empty.

At the bottom right, system resources are shown: Memory: 101.7MB and Disk: 20.5MB.

Burp suite has many useful features in store for us, even right after starting up. As a user of the community version of burp, your options here will be somewhat limited but still useful in debugging our project. We will start with one of the few free options available to us.

## Event log

This will display any event occurring in burp suite while we run it, this can range from errors to information to things we print out with our custom written extensions.

If you ever have issues intercepting traffic or decoding it, this is the first place i would look.

## Scans

We can differentiate between 3 different types of scans here, all 3 have their unique properties which we will go over in detail.

### Scan details

#### Scan types - Crawl

Crawling a website allows burp to automatically look for any URL or link it finds on the webpage that you give it and then attempt to surf there and it will repeat the same actions. This can take one or multiple URLs as a starting point and go from there. Please note that this does not serve the purpose of auditing those found items.

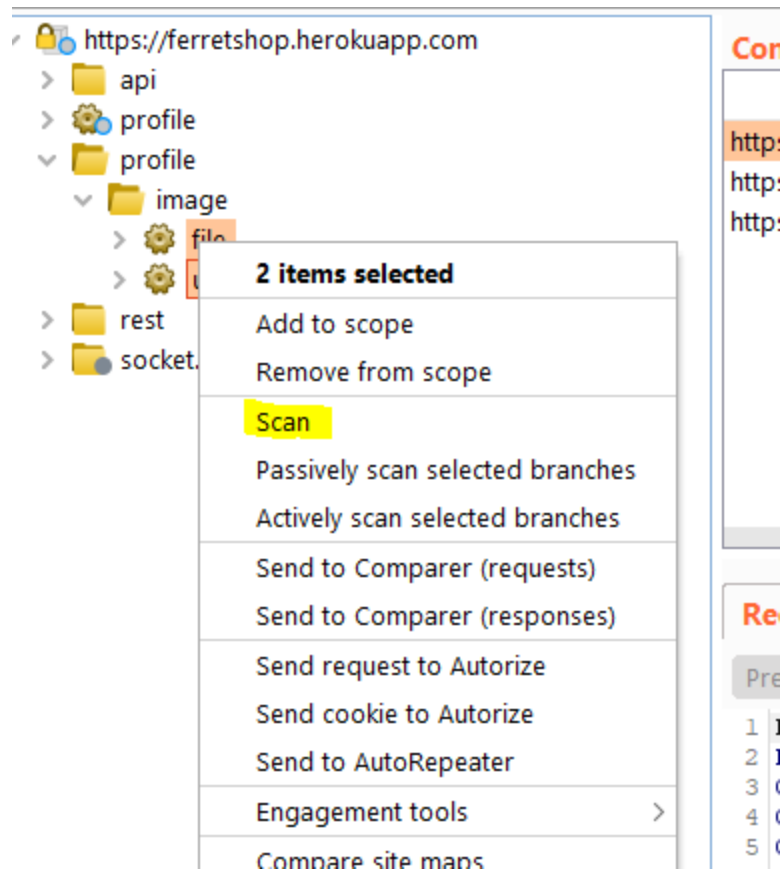
The depth of this crawling and any other options can be set in the 'Scan configurations' which we will go over in the next section.

#### Scan types - Crawl and audit

We can also audit the crawled items directly which will analyse the results and do things like static or dynamic code analysis (based on our settings in the Scan configurations).

## Scan types - Audit selected items

This option is only available if we select several urls from the sitemap and right click them.



New scan

Scan details

Scan configuration

Application login

Resource pool

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.

https://ferretshop.herokuapp.com/profile/image/file

https://ferretshop.herokuapp.com/profile/image/url

Protocol settings

☒ Scan using HTTP & HTTPS

☐ Scan using my specified protocols

> Detailed scope configuration

?

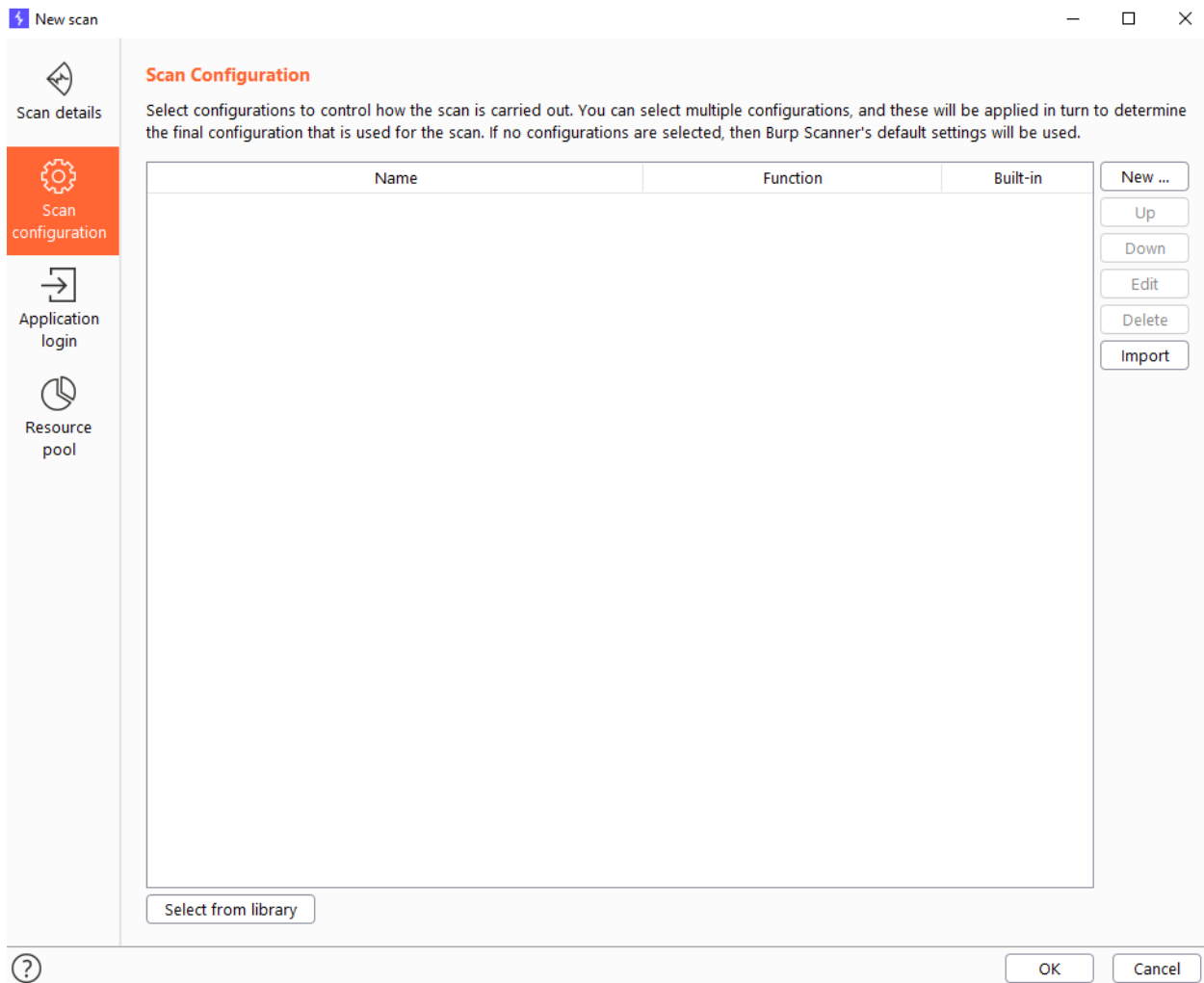
OK

Cancel

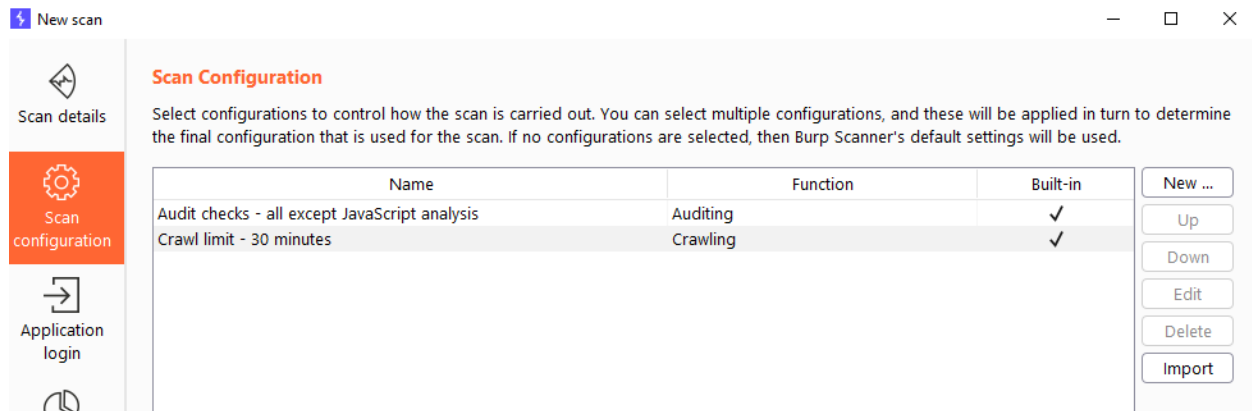
## Protocol settings

Here we can define wheter we want to test for both HTTPS and HTTP or if we want to define which protocols to test for.

## Scan configurations



In here we define rules for our scan. Every line in here can consist of one or several rules and can be combined. For example we can choose to do all the audits except for the javascript analysis and also we might want to limit or crawling to 30 minutes max. This can easily be done by adding some built-in rules.



But we can also build our own rules here by pressing the "New..." button. This will give us a choice between crawling and auditing. If you set audit rules on a crawler only jobs, that's ofcourse not going to have any effect.

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

It really pays off to name your rules properly if you ever want to reuse them. This is a very confusing part and having ambiguous names for rulesets can be very confusing.

Next we want to set up the different options we have. If we want to enable a rule, we need to click open the respective section and the rule will automatically be enabled if it is filled in.

New scanning configuration

Configuration name:

Expand the areas that you want to define in this configuration.

**Audit Optimization**

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

Audit speed:

Audit accuracy:

- ☒ Skip checks unlikely to be effective due to insertion point's base value
- ☒ Consolidate frequently occurring passive issues
- ☒ Automatically maintain session (only applies to crawl-driven audits)
- ☒ Follow redirections where necessary

**Issues Reported** Not defined

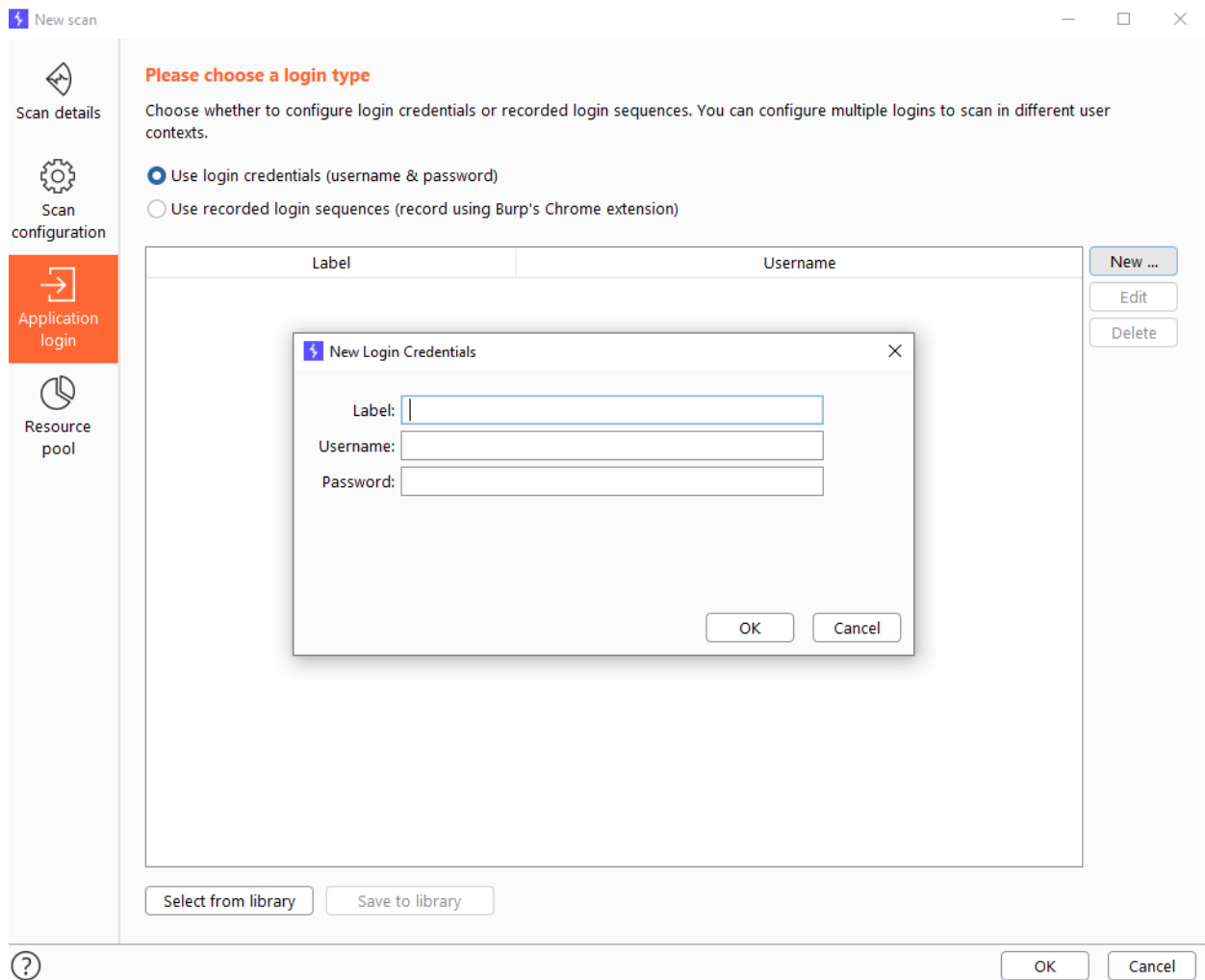
Most of these settings speak for themselves but we will going over each and every one of them in a seperate video so we don't have to interrupt the flow of the course.

We can also save our rules in the library, this will enable us to reuse them later on.

## Application login

Sometimes when crawling, burp will encounter a login screen. If we have credentials available we can enter them here, allowing for an authenticated crawling. This will expose a lot more functionality to us but is not foolproof as burp sometimes doesn't recognize the login pages, however it is still much better than never being able to crawl a website authenticated.





You can give a label to the credentials which are just to make it easier to recognize which credentials you have in front of you.

## Resource pool

This allows us to create "resource pools" which are just variables to which we assign a certain value of maximum concurrent requests, delay between those requests and other options.

We can then use specific resource pools for specific jobs.

New scan

Scan details

Scan configuration

Application login

Resource pool

### Resource Pool

Specify the resource pool in which the scan will be run. Resource pools are used to manage the usage of system resources across multiple tasks.

☒ Use existing resource pool

Selected	Resource pool	Max concurrent requests	Delay between requests
<input checked="" type="radio"/>	Default resource pool	10	

☐ Create new resource pool

Name:

☐ Maximum concurrent requests:

☐ Delay between requests:  milliseconds

☐ Add random variations

?

OK

Cancel

## Live task

Live scanning can perform the same auditing actions as the crawling and auditing can do, but this will happen on all the requests you are executing or that are coming in while you are manually exploring the website.

## Issue activity + Advisory

When scanning, burp will come across some issues. Whenever it does, it will display them here.

Everything here pretty much speaks for itself and burp will explain things in the "Advisory" section when we click on an issue in the "Issue activity" window.

**Issue activity** ? ↗

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

#	Task	Time	Action	Issue type	Host
21	2	00:58:29 11 Mar 2021	Issue found	ⓘ TLS cookie without secure flag set	https://ferretshop.herokuapp.com /socket.io
20	2	22:22:23 10 Mar 2021	Issue found	ⓘ HTML does not specify charset	https://ferretshop.herokuapp.com /socket.io/
19	2	21:11:52 10 Mar 2021	Issue found	ⓘ Vulnerable JavaScript dependency	https://ferretshop.herokuapp.com /favicon.ico
18	2	21:11:52 10 Mar 2021	Issue found	ⓘ Cross-domain script include	https://ferretshop.herokuapp.com /favicon.ico
17	2	21:03:28 10 Mar 2021	Issue found	ⓘ Cross-domain script include	https://ferretshop.herokuapp.com /profile
16	2	21:03:28 10 Mar 2021	Issue found	ⓘ Vulnerable JavaScript dependency	https://ferretshop.herokuapp.com /profile
15	2	21:03:28 10 Mar 2021	Issue found	ⓘ Mixed content	https://ferretshop.herokuapp.com /profile
14	2	21:03:28 10 Mar 2021	Issue found	ⓘ File upload functionality	https://ferretshop.herokuapp.com /profile
13	2	20:51:34 10 Mar 2021	Issue found	ⓘ Private IP addresses disclosed	https://ferretshop.herokuapp.com /rest/admin/
12	2	20:51:34 10 Mar 2021	Issue found	ⓘ TLS cookie without secure flag set	https://ferretshop.herokuapp.com /socket.io/
11	2	20:51:34 10 Mar 2021	Issue found	ⓘ Cacheable HTTPS response	https://ferretshop.herokuapp.com /
10	2	20:51:34 10 Mar 2021	Issue deleted	ⓘ Cacheable HTTPS response	https://ferretshop.herokuapp.com /
9	2	20:51:34 10 Mar 2021	Issue found	ⓘ Strict transport security not enforced	https://ferretshop.herokuapp.com /
8	2	20:51:34 10 Mar 2021	Issue deleted	ⓘ Strict transport security not enforced	https://ferretshop.herokuapp.com /
7	2	20:51:32 10 Mar 2021	Issue found	ⓘ Strict transport security not enforced	https://ferretshop.herokuapp.com /polyfills-es2
6	2	20:51:31 10 Mar 2021	Issue found	ⓘ Strict transport security not enforced	https://ferretshop.herokuapp.com /styles.css
5	2	20:51:31 10 Mar 2021	Issue found	ⓘ Vulnerable JavaScript dependency	https://ferretshop.herokuapp.com /
4	2	20:51:31 10 Mar 2021	Issue found	ⓘ Cacheable HTTPS response	https://ferretshop.herokuapp.com /

Advisory Request Response

### ? Vulnerable JavaScript dependency

Issue: **Vulnerable JavaScript dependency**  
 Severity: **Low**  
 Confidence: **Tentative**  
 Host: **https://ferretshop.herokuapp.com**  
 Path: **/favicon.ico**

**Issue detail**

We observed a vulnerable JavaScript library.

We detected **jquery** version **2.2.4**, which has the following vulnerabilities:

- [CVE-2015-9251](#): 3rd party CORS request may execute
- [CVE-2015-9251](#): parseHTML() executes scripts in event handlers
- [CVE-2019-11358](#): jQuery before 3.4.0, as used in Drupal, Backdrop CMS, and other products, mishandles jQuery.extend(true, {}, ...) because of Object.prototype pollution
- [CVE-2020-11022](#): Regex in its jQuery.htmlPrefilter sometimes may introduce XSS
- [CVE-2020-11022](#): Regex in its jQuery.htmlPrefilter sometimes may introduce XSS

Memory: 122.7MB  Disk: 20.5MB

The most useful part here are the filters, they allow us to show specific vulnerability types, making it easier to distinguish between the ones we care about and the ones we do not. I've found that almost none of the issues in here are worth it to investigate when i am doing bug bounties.