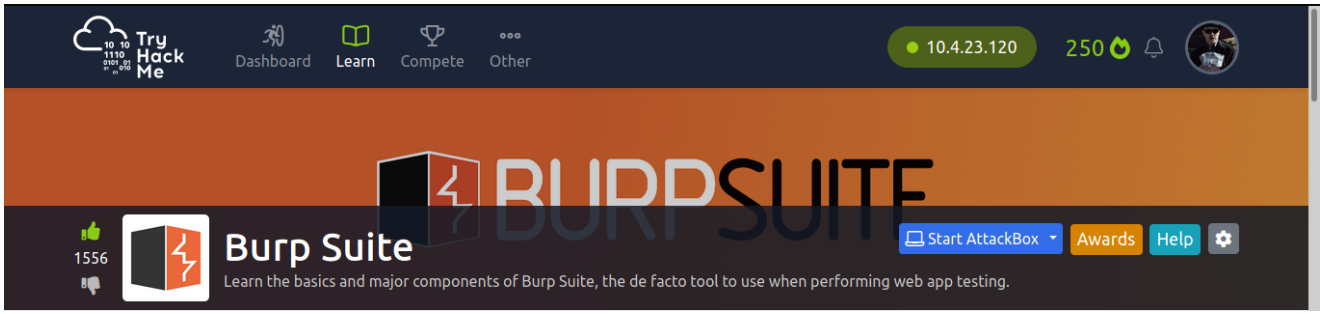


Features

- **Proxy** - What allows us to funnel traffic through Burp Suite for further analysis
- **Target** - How we set the scope of our project. We can also use this to effectively create a site map of the application we are testing.
- **Intruder** - Incredibly powerful tool for everything from field fuzzing to credential stuffing and more
- **Repeater** - Allows us to 'repeat' requests that have previously been made with or without modification. Often used in a precursor step to fuzzing with the aforementioned Intruder
- **Sequencer** - Analyzes the 'randomness' present in parts of the web app which are intended to be unpredictable. This is commonly used for testing session cookies
- **Decoder** - As the name suggests, Decoder is a tool that allows us to perform various transforms on pieces of data. These transforms vary from decoding/encoding to various bases or URL encoding.
- **Comparer** - Comparer as you might have guessed is a tool we can use to compare different responses or other pieces of data such as site maps or proxy histories (awesome for access control issue testing). This is very similar to the Linux tool diff.
- **Extender** - Similar to adding mods to a game like Minecraft, Extender allows us to add components such as tool integrations, additional scan definitions, and more!
- **Scanner** - Automated web vulnerability scanner that can highlight areas of the application for further manual investigation or possible exploitation with another section of Burp. This

feature, while not in the community edition of Burp Suite, is still a key facet of performing a web application test.

Proxy

Generally speaking, proxy servers by definition allow us to relay our traffic through an alternative route to the internet. This can be done for a variety of reasons ranging from educational filtering (common in schools where restricted content must be blocked) to accessing content that may be otherwise unavailable due to region locking or a ban. Using a proxy, however, for web application testing allows us to view and modify traffic inline at a granular level.

Target tab

Lock on Target by Alexei Vella on Dribbble

The Target tab in Burp allows us to perform arguably some of the most important parts of a web application penetration test: defining our scope, viewing a site map, and specifying our issue definitions (although this is more useful within report generation and scanning).

When starting a web application test you'll very likely be provided a few things:

- The application URL (hopefully for dev/test and not prod)
- A list of the different user roles within the application
- Various test accounts and associated credentials for those accounts
- A list of pieces/forms in the application which are out-of-scope for testing and should be avoided

From this information, we can now start to build our scope within Burp, something which is incredibly important in the case we are planning on performing any automated testing. Typically this is done in a tiered approach wherein we work our way up from the lowest privileged account (this includes unauthenticated access), browsing the site as a normal user would. Browsing like this to discover the full extent of the site is commonly referenced as the 'happy path'. Following the creation of a site map via browsing the happy path, we can go through and start removing various items from the scope. These items typically fit one of these criteria:

- The item (page, form, etc) has been designated as out of scope in the provided documentation from the client
- Automated exploitation of the item (especially in a credentialed manner) would cause a huge mess (like sending hundreds of password reset emails - If you've done a web app professionally you've probably done this at one point)
- Automated exploitation of the item (especially in a credentialed manner) would lead to damaging and potentially crashing the web app

Intruder

Arguably the most powerful tool in Burp Suite, Intruder can be used for many things ranging from fuzzing to brute-forcing. At its core, Intruder serves one purpose: automation.

While Repeater best handles experimentation or one-off testing, Intruder is meant for repeat testing once a proof of concept has been established. Per the Burp Suite documentation, some common uses are as follows:

- Enumerating identifiers such as usernames, cycling through predictable session/password recovery tokens, and attempting simple password guessing
- Harvesting useful data from user profiles or other pages of interest via grepping our responses
- Fuzzing for vulnerabilities such as SQL injection, cross-site scripting (XSS), and file path traversal

The Overcoat by Chill Desk on Dribbble

To accomplish these various use cases, Intruder has four different attack types:

1. *Sniper* - The most popular attack type, this cycles through our selected positions, putting the next available payload (item from our wordlist) in each position in turn. This uses only one set of payloads (one wordlist).

2. *Battering Ram* - Similar to Sniper, Battering Ram uses only one set of payloads. Unlike Sniper, Battering Ram puts every payload into every selected position. Think about how a battering ram makes contact across a large surface with a single surface, hence the name battering ram for this attack type.

3. *Pitchfork* - The Pitchfork attack type allows us to use multiple payload sets (one per position selected) and iterate through both payload sets *simultaneously*. For example, if we selected two positions (say a username field and a password field), we can provide a username and password payload list. Intruder will then cycle through the combinations of usernames and passwords, resulting in a total number of combinations equalling the smallest payload set provided.

4. *Cluster Bomb* - The Cluster Bomb attack type allows us to use multiple payload sets (one per position selected) and iterate through all combinations of the payload lists we provide. For example, if we selected two positions (say a username field and a password field), we can provide a username and password payload list. Intruder will then cycle through the combinations of usernames and passwords, resulting in a total number of combinations equalling usernames x passwords. *Do note, this can get pretty lengthy if you are using the community edition of Burp.*

Sequencer

While not as commonly used in a practice environment, Sequencer represents a core tool in a proper web application pentest. Burp's Sequencer, per the Burp documentation, is a tool for analyzing the quality of randomness in an application's sessions tokens and other important data items that are otherwise intended to be unpredictable. Some commonly analyzed items include:

- Session tokens
- Anti-CSRF (Cross-Site Request Forgery) tokens
- Password reset tokens (sent with password resets that in theory uniquely tie users with their password reset requests)

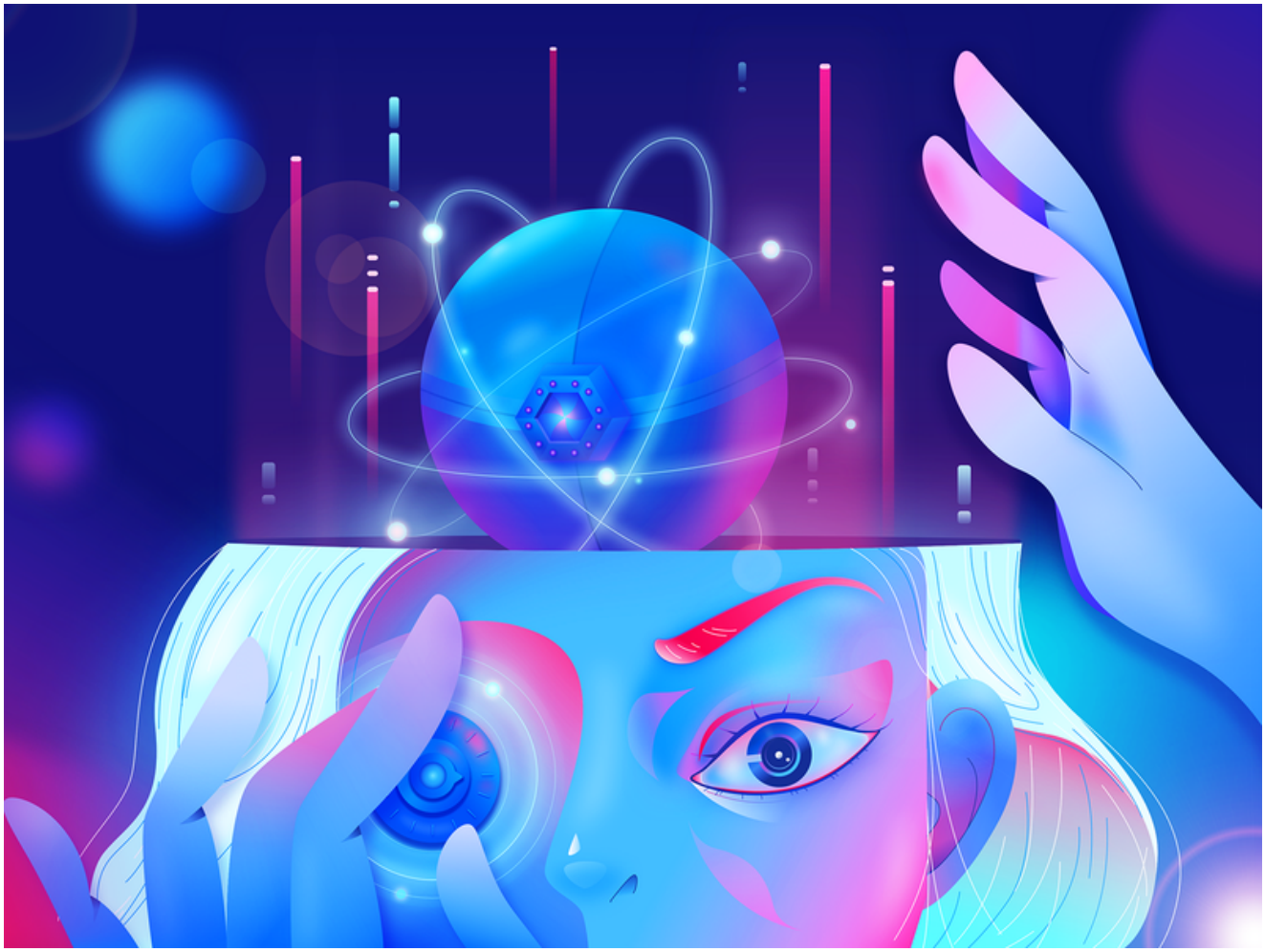
We'll take a quick peek at how we can use Sequencer to examine the session cookies which Juice Shop issues.

SEO Friendly Progressive Web Applications with Angular Universal by Maxime Bourgeois on Dribbble

Decoder and Comparer

Decoder and Comparer, while lesser tools within Burp Suite, are still essential to understand and leverage as part of being a proficient web app tester.

As the name suggests, Decoder is a tool that allows us to perform various transforms on pieces of data. These transforms vary from decoding/encoding to various bases or URL encoding. We chain these transforms together and Decoder will automatically spawn an additional tier each time we select a decoder, encoder, or hash. *This tool ultimately functions very similarly to CyberChef, albeit slightly less powerful.*



Encryption by Muriel on Dribbble

Similarly, Comparer, as you might have guessed is a tool we can use to compare different responses or other pieces of data such as site maps or proxy histories (awesome for access control issue testing). This is very similar to the Linux tool diff.

Per the Burp documentation, some common uses for Comparer are as follows:

- When looking for username enumeration conditions, you can compare responses to failed logins using valid and invalid usernames, looking for subtle differences in responses. *This is also sometimes useful for when enumerating password recovery forms or another similar recovery/account access mechanism.*
- When an Intruder attack has resulted in some very large responses with different lengths than the base response, you can compare these to quickly see where the differences lie.
- When comparing the site maps or Proxy history entries generated by different types of users, you can compare pairs of similar requests to see where the differences lie that give rise to different application behavior. This may reveal possible access

control issues in the application wherein lower privileged users can access pages they really shouldn't be able to.

- When testing for blind SQL injection bugs using Boolean condition injection and other similar tests, you can compare two responses to see whether injecting different conditions has resulted in a relevant difference in responses.

**These examples are taken nearly in their entirety from the Burp docs simply to provide a broader set of examples to consider when using Comparer.*

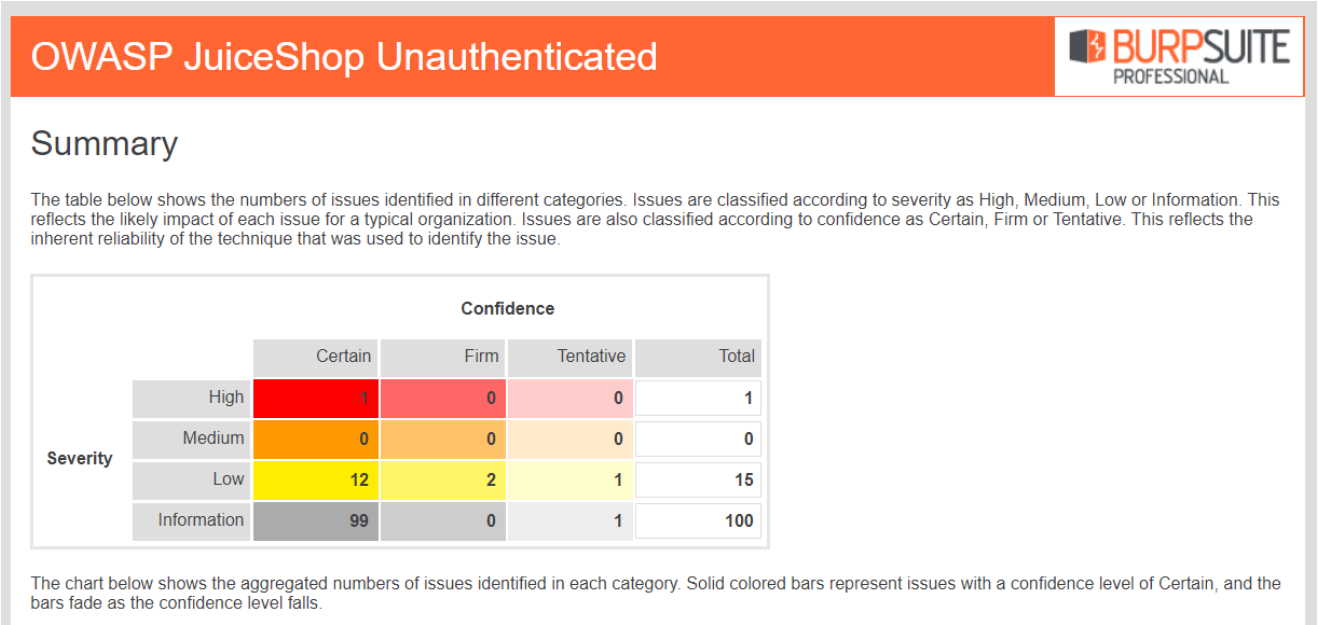
JavaScript Arrays in Depth by Maxime Bourgeois on Dribbble

Burp Suite Scanner and Collaborator Client!

The features that Burp Suite Professional offers: The Burp Suite Scanner and Collaborator Client!

Engage by Todd Zlab on Dribbble

Arguably the most powerful feature in Burp Suite, the Burp Suite Scanner allows us to passively and actively scan and spider the website we are testing for vulnerabilities. In Burp 2.0's task-based model, we can launch these scans (Scanner and Spider) from the dashboard and let them run in the background while we continue to examine the web app. In this case, I've run an unauthenticated scan against Juice Shop and have attached it to this task. These reports can provide a starting place for further enumeration and exploitation via the other tools in Burp Suite.



OWASP JuiceShop Unauthenticated

BURPSUITE PROFESSIONAL

Summary

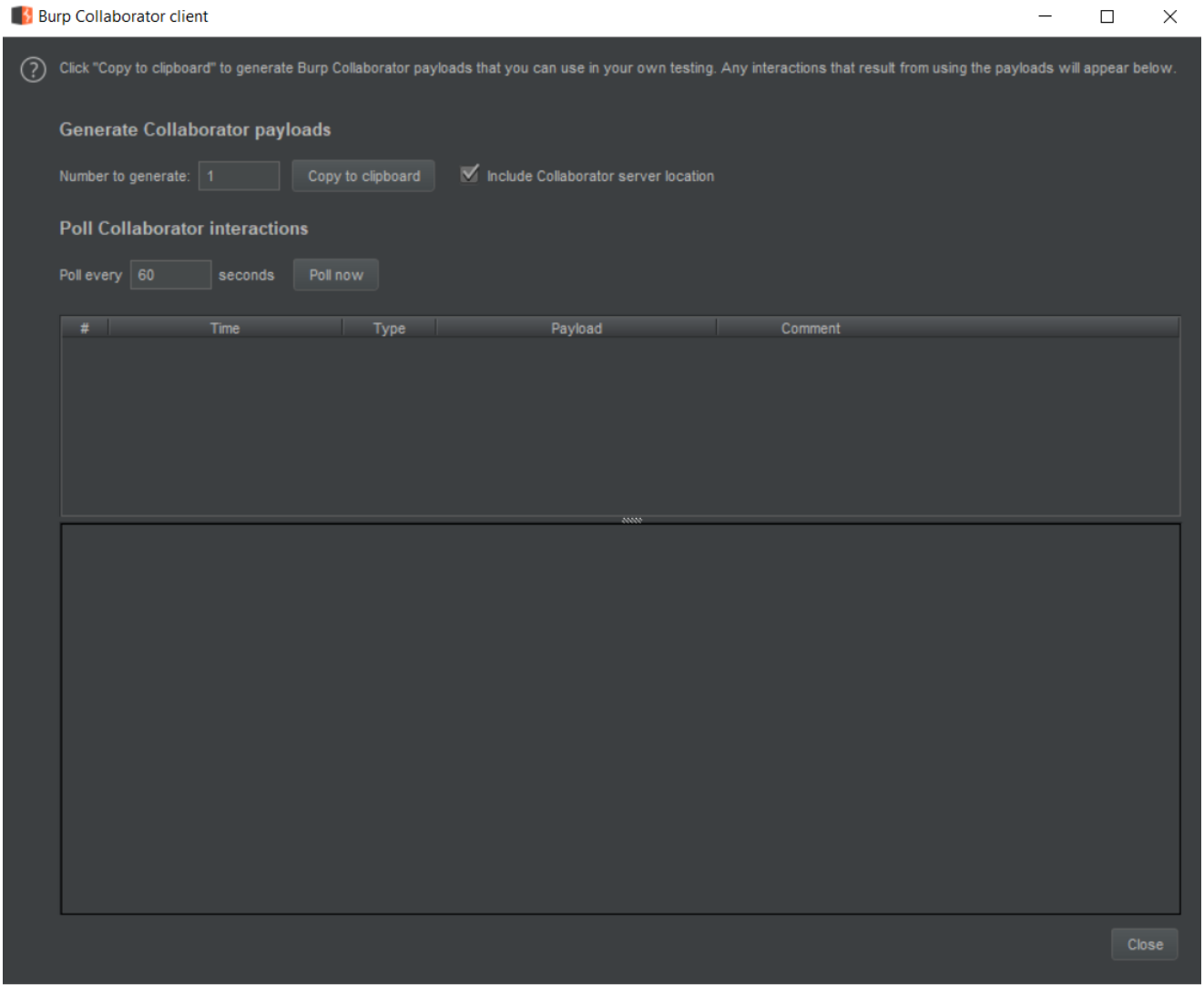
The table below shows the numbers of issues identified in different categories. Issues are classified according to severity as High, Medium, Low or Information. This reflects the likely impact of each issue for a typical organization. Issues are also classified according to confidence as Certain, Firm or Tentative. This reflects the inherent reliability of the technique that was used to identify the issue.

		Confidence			
		Certain	Firm	Tentative	Total
Severity	High	1	0	0	1
	Medium	0	0	0	0
	Low	12	2	1	15
	Information	99	0	1	100

The chart below shows the aggregated numbers of issues identified in each category. Solid colored bars represent issues with a confidence level of Certain, and the bars fade as the confidence level falls.

A Preview of the Report Attached to this Task Created with Burp Professional

Commonly used in manual tests, Burp Collaborator Client allows us to gain insight into issues that may otherwise seem to produce no output. Often during testing, we may come across items which, either due to timing/slowness of the web app or a lack of any reaction, are likely vulnerable but don't produce any sure-fire indicators. With Burp Collaborator, however, we can produce out-of-band alerts via generating payloads that reach back to Burp Suite's servers for us.



Burp Suite reference documentation for Scanner and Collaborator Client