Automation tricks for Burp Suite Pro

Nicolas Grégoire aka @Agarri_FR

NahamCon 2023

Intro

Who am I?

Nicolas Grégoire 🇫🇷

Twitter → `@Agarri_FR` and `@MasteringBurp`

Email → `nicolas.gregoire@agarri.fr`

Founder & owner of Agarri

Pentest, training and research

Official Burp Suite training partner

Mostly for Europe 🇪🇺

100+ trainees per year (either on-site and online)

What is the plan?

Running the testbed

Fetch and run the containers

Handling CSRF tokens in Intruder

Recursive Grep + Pitchfork

Intro to session handling rules

Terminology and default behavior

Session management for Web apps

Handling CSRF tokens in a generic way

Session management for Web APIs

Using static, dynamic or cached authz headers

Northsec 2023

A 30-minute talk published last month

Slides

https://www.agarri.fr/docs/nsec23-burp_tips_n_tricks.pdf

Video

https://www.youtube.com/watch?v=hslR6hE7fS8&t=25247s

NahamCon 2023

These slides are already online

https://www.agarri.fr/docs/nahamcon23-burp_automation.pdf

Running the testbed

Running the testbed

Get the containers

```
docker pull agarri/miniapp
docker pull agarri/juiceshop
```

Run the containers

```
docker run -d -p 9081:80 agarri/miniapp
docker run -d -p 9082:3000 agarri/juiceshop
```

Now you can access the apps

MiniApp → http://127.0.0.1:9081/

JuiceShop → http://127.0.0.1:9082/

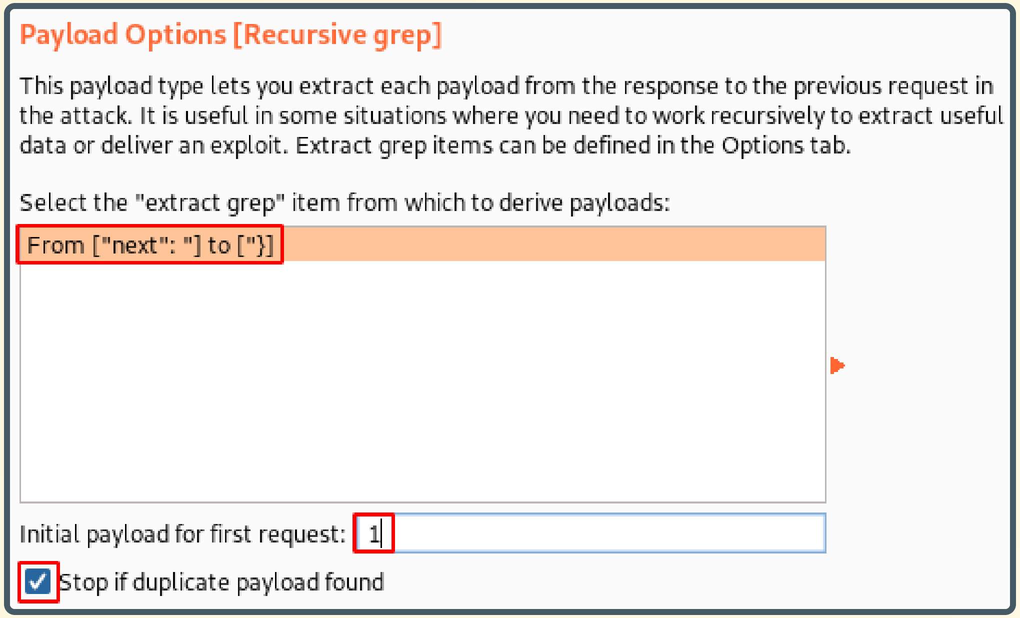
Handling CSRF tokens in Intruder

Recursive Grep

In "Intruder > # > Payloads > Payload type"

Required configuration

- How to start (initial payload)
- How to progress (based on "Grep Extract")
- When to stop (optional)



Payload Options [Recursive grep]

This payload type lets you extract each payload from the response to the previous request in the attack. It is useful in some situations where you need to work recursively to extract useful data or deliver an exploit. Extract grep items can be defined in the Options tab.

Select the "extract grep" item from which to derive payloads:

From ["next": "] to ["}]

Initial payload for first request: 1

☑ Stop if duplicate payload found

Mini App - Chall 01

Goal

Dump only valid users

Strategy

Attack type → Sniper
Payload type → Recursive Grep

CSRF tokens in Intruder

Submit a request, get a fresh token back

Re-use the token via the "Recursive Grep"

Easy to setup

Twice faster than session handling rules

Works only if the response contains a token

Mini App - Chall 02

Goal

Find a value between 1 and 50

While providing valid CSRF tokens

Strategy

Attack type → Pitchfork

Payload type → Recursive Grep

Intro to session handling rules

Terminology

Cookie jar

Place where the cookies are stored (like in a browser)

Macro

Sequence of HTTP requests

Execution is triggered by a session handling rule

Session handling rule

Define when and how sessions are managed

May execute one or several macros (among other things)

Session tracer

Live debugger for macros and rules

Default behavior

Proxy

Cookies are written to the cookie jar

Scanner

Cookies are read from the cookie jar

How to easily improve scan coverage?

Add Extensions to the scope of the default rule

Except when using authorization-testing extensions!

Session management for Web apps

Session management for Web apps

Generic handling of CSRF tokens

Overview

Get a fresh anti-CSRF token via a macro

Parameter `csrf_token` is automatically extracted

Update the token in the original request

All requests read from the cookie jar

Use a single thread

Configure the macro

Select the token-fetching request

Using the macro recorder

Set the cookie to a visually invalid value

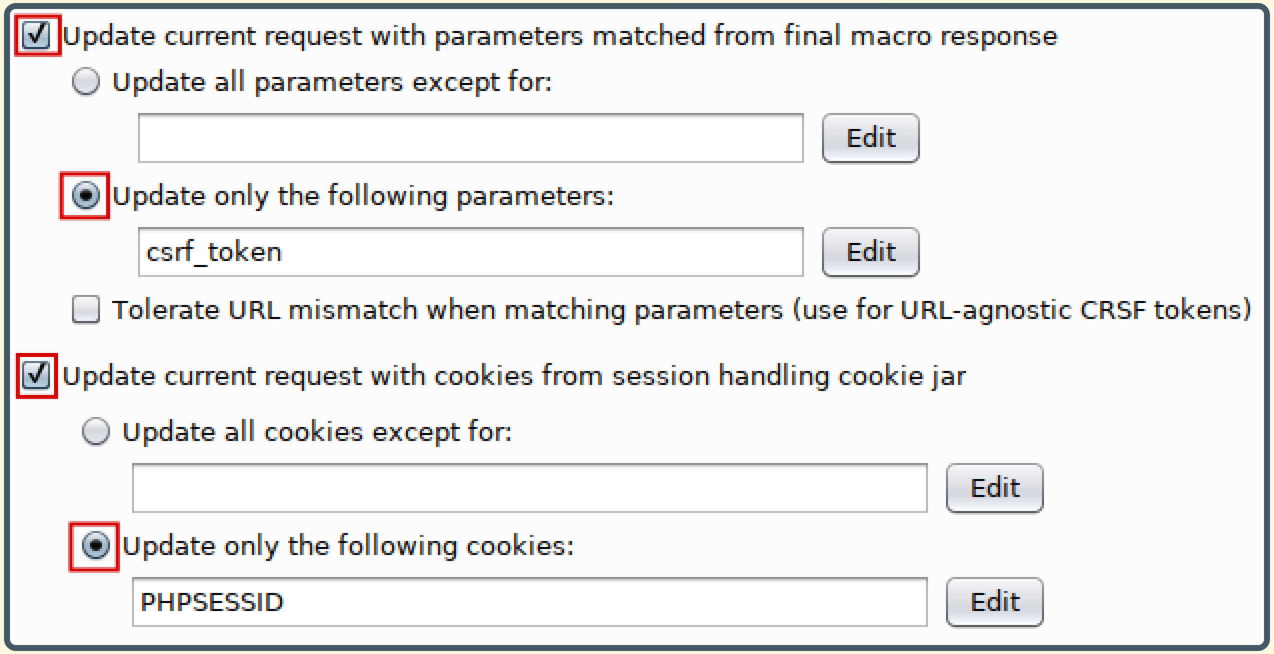
Click on "Configure item"

Add received cookies to the cookie jar

Use cookies from the cookie jar

No need to configure any data extraction

Configure the rule



☑ Update current request with parameters matched from final macro response

○ Update all parameters except for:

● Update only the following parameters:

csrf_token

□ Tolerate URL mismatch when matching parameters (use for URL-agnostic CRSF tokens)

☑ Update current request with cookies from session handling cookie jar

○ Update all cookies except for:

● Update only the following cookies:

PHPSESSID

Mini App - Chall 02

Using Intruder

Attack type: "Sniper"

Make sure the rule's scope includes Intruder

Session management for Web APIs

Overview of Juice Shop

Log into the API

URL → `/rest/user/login`

Extract the JWT from the response

Non-greedy regex → `token":"(.*?)"`

Validate the JWT

URL → `/api/users`

Header → `Authorization: Bearer [value]`

Known account

jim@juice-sh.op / ncc-1701 (feel free to create your own)

Session management for Web APIs

Static authz header

Static authz header

Pick a token from your Proxy History

Create a session handling rule

- Action "Set a specific header value"
- Make sure to include the `Bearer[space]` prefix

Access the API

Session management for Web APIs

Dynamic authz header

Dynamic authz header

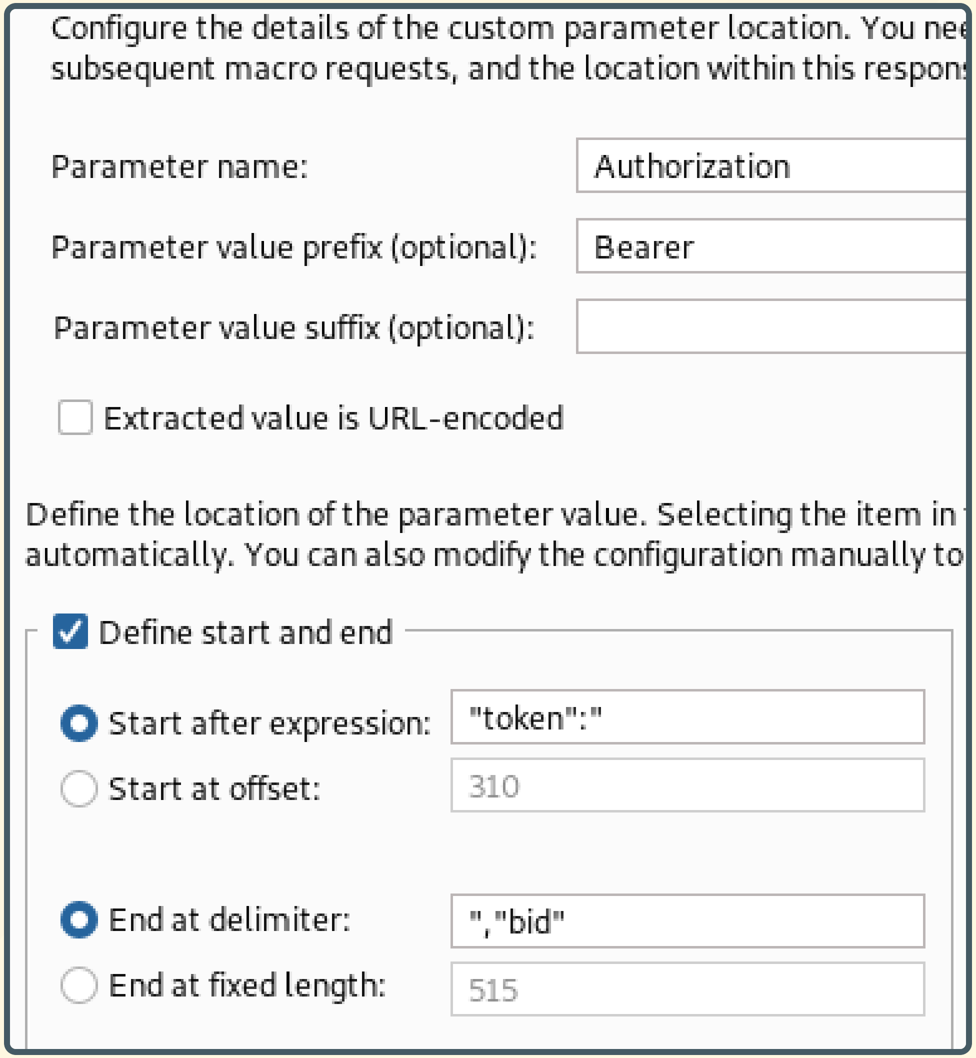
Create a login macro

Configure data extraction

In "Configure item > Custom parameter locations"

- Select the authentication token
- Set the parameter name to `authorization`
- Set the parameter prefix to `Bearer[space]`

Dynamic authz header



Configure the details of the custom parameter location. You nee
subsequent macro requests, and the location within this respon

Parameter name: Authorization

Parameter value prefix (optional): Bearer

Parameter value suffix (optional):

□ Extracted value is URL-encoded

Define the location of the parameter value. Selecting the item in
automatically. You can also modify the configuration manually to

☑ Define start and end

● Start after expression: "token":"

○ Start at offset: 310

● End at delimiter: ","bid"

○ End at fixed length: 515

Dynamic authz header

Create a session handling rule

Configure its actions

Add action "Set a specific header value"

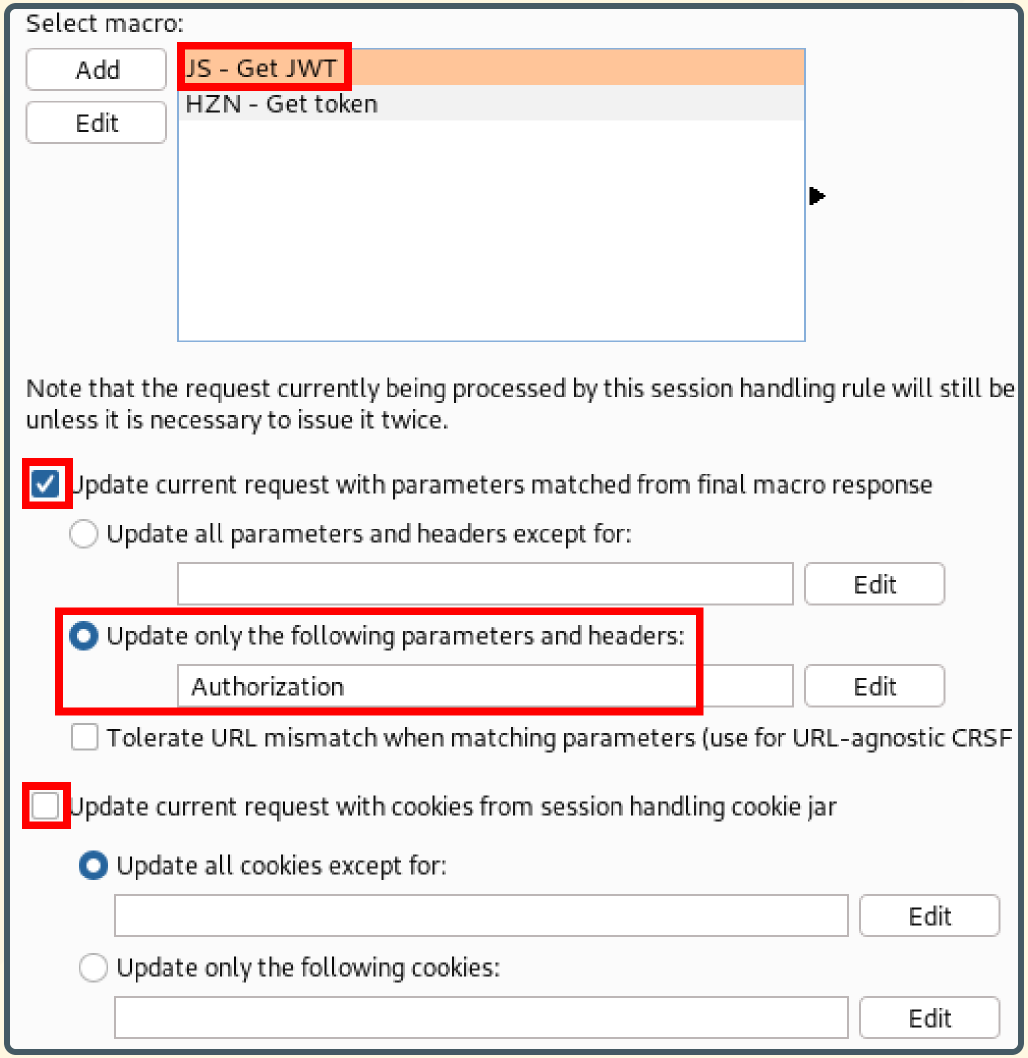
- Make sure to check "Add if not present"

Add action "Run macro"

- Select the login macro
- Update the `Authorization` header

Access the API

Dynamic authz header



Select macro:

JS - Get JWT

HZN - Get token

Note that the request currently being processed by this session handling rule will still be unless it is necessary to issue it twice.

☑ Update current request with parameters matched from final macro response

○ Update all parameters and headers except for:

● Update only the following parameters and headers:

Authorization

□ Tolerate URL mismatch when matching parameters (use for URL-agnostic CRSF

□ Update current request with cookies from session handling cookie jar

● Update all cookies except for:

○ Update only the following cookies:

Session management for Web APIs

Cached authz header

Cached authz header

Download extension "JWT ReAuth" (v1.0.1)
https://github.com/nccgroup/jwt-reauth/releases

Load it in Burp Suite

Cached authz header

From "Proxy History"

Send the login request to the extension

- Using action "Set auth request"

Set the extension's scope

- Using action "Add to scope"

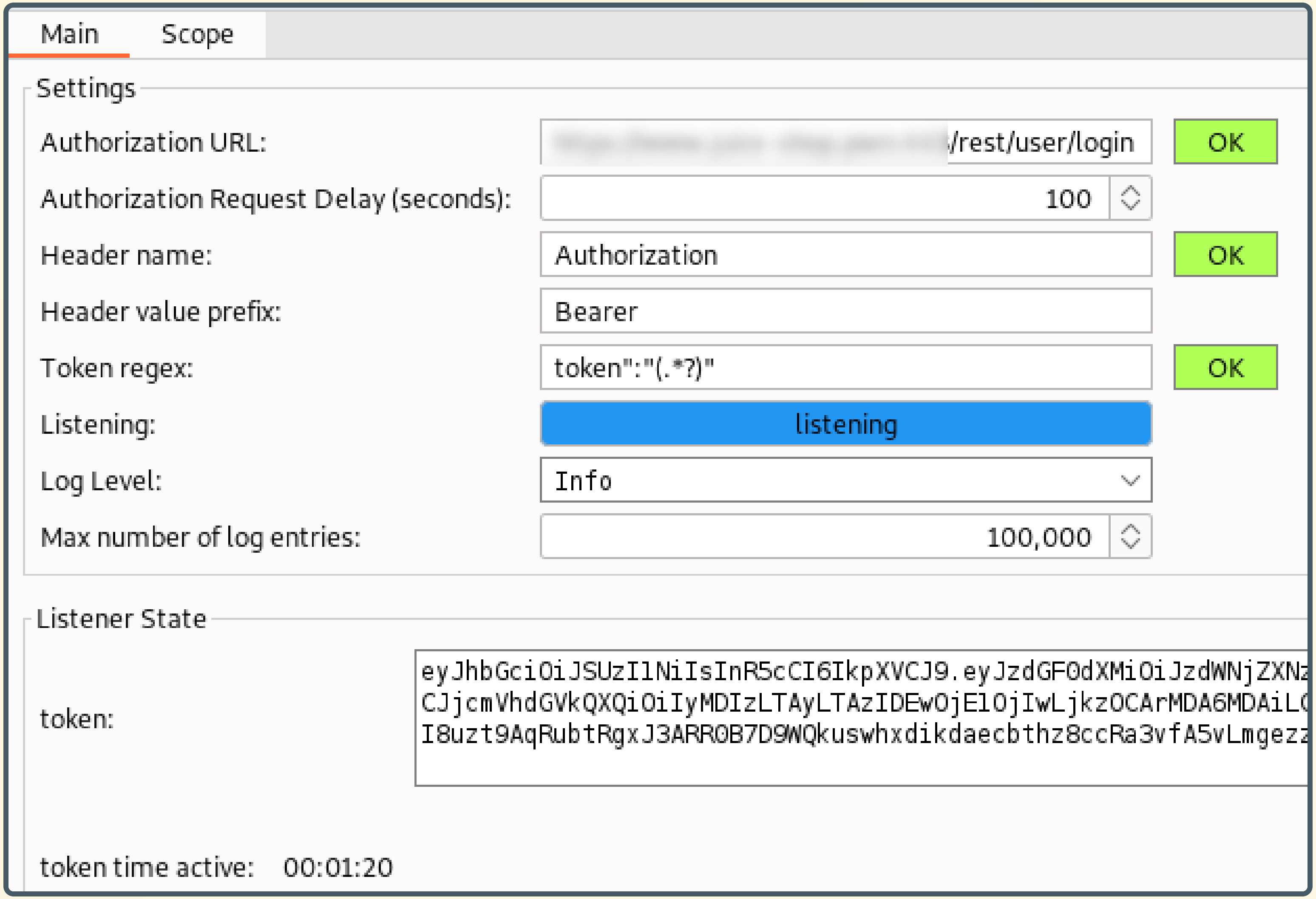
Cached authz header

From the extension's tab

- Edit the scope (should end with `/api/`)
- Change the "Token regex" field to `token":"(.*?)"`
- Change the "Authorization Request Delay" field to `100`
- Enable processing (click on `Not listening`)

Access the API

Cached authz header



Settings

Authorization URL: [redacted]/rest/user/login OK

Authorization Request Delay (seconds): 100

Header name: Authorization OK

Header value prefix: Bearer

Token regex: token":"(.*?)" OK

Listening: listening

Log Level: Info

Max number of log entries: 100,000

Listener State

token:

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdGF0dXMiOiJzdWNjZXN
CJjcmVhdGVkQXQiOiIyMDIzLTAyLTAzIDEwOjE1OjIwLjkzOCArMDA6MDAiL
I8uzt9AqRubtRgxJ3ARR0B7D9WQkuswhxdikdaecbthz8ccRa3vfA5vLmgez
```

token time active: 00:01:20

Thanks for listening!

I hope you learnt a few things...