Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

2023-06-02 Fri, IT-S NOW - 2023 Conference,
University of Applied Sciences, FH Campus Wien

Outline

$ whoami

Daniel Neagaru

- Working at Akkodis (prev. Modis) for 1.5 year
- Penetration Tester for 5+ years
- 5+ years in IT mostly as a sysadmin
- Started building Raider early 2021
- Became an OWASP project leader August 2021

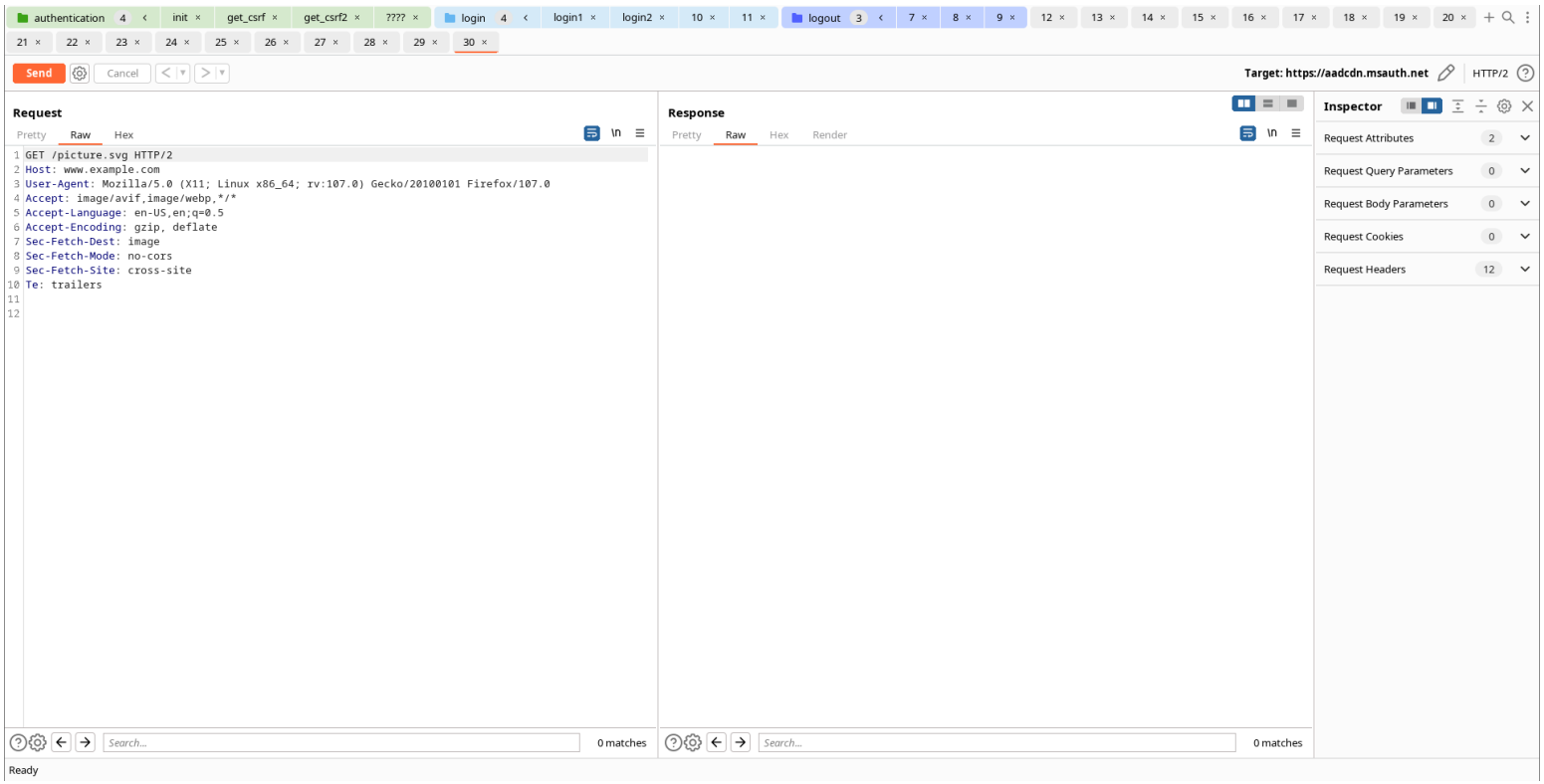
$ whatis raider

Raider was incepted with the goal to help test web authentication systems but has evolved and now can be used for HTTP processes of arbitrary complexity.

- A framework for manipulating HTTP processes
- Defines a DSL to describe the client-server information exchange
- Modular architecture with flexibility in its DNA
- Main code written in Python
- Configuration files written in hylang (LISP)

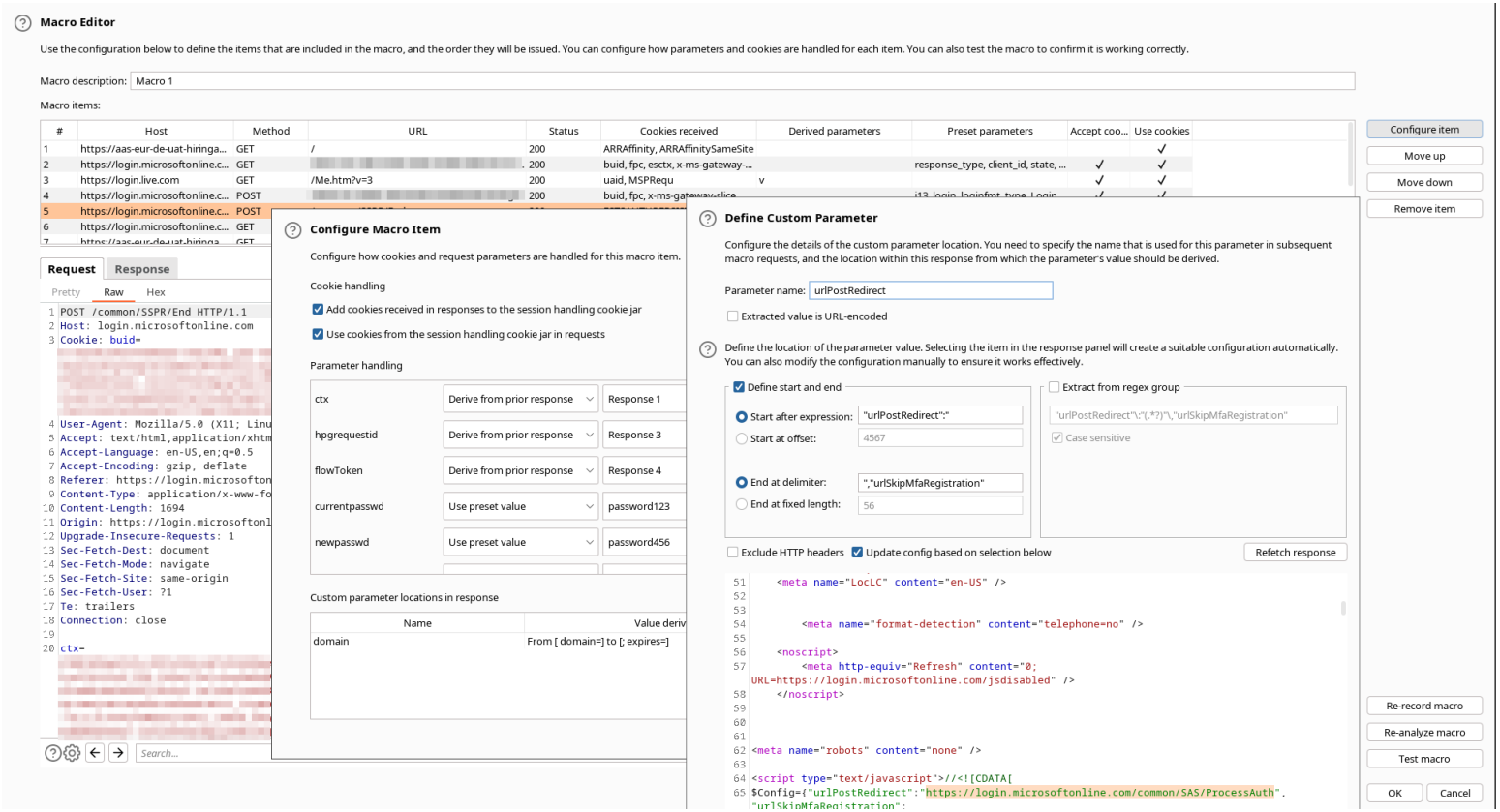
BurpSuite

When testing authentication process in BurpSuite I ended up with many poorly organized Repeater tabs.



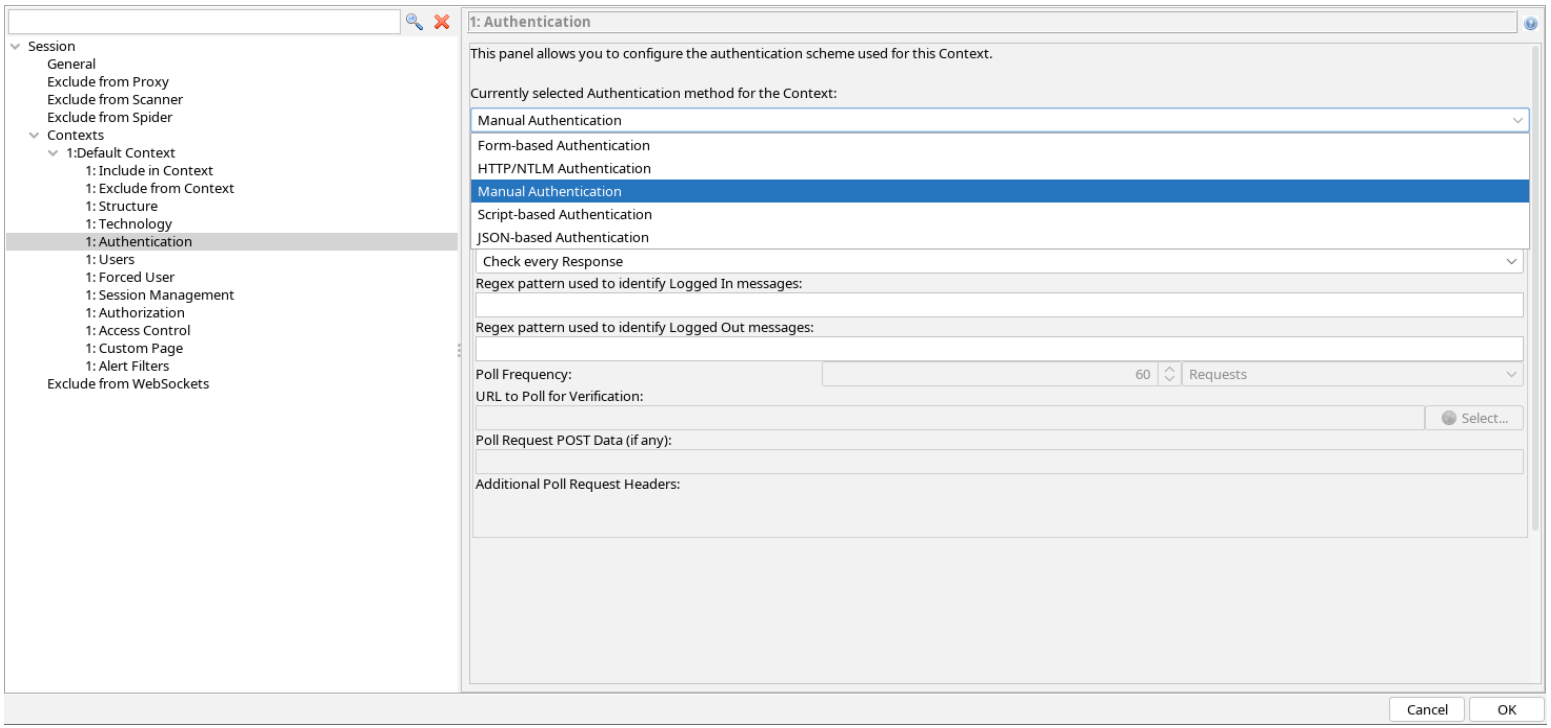
BurpSuite

After figuring out how authentication works, I had to write new BurpSuite Macros.



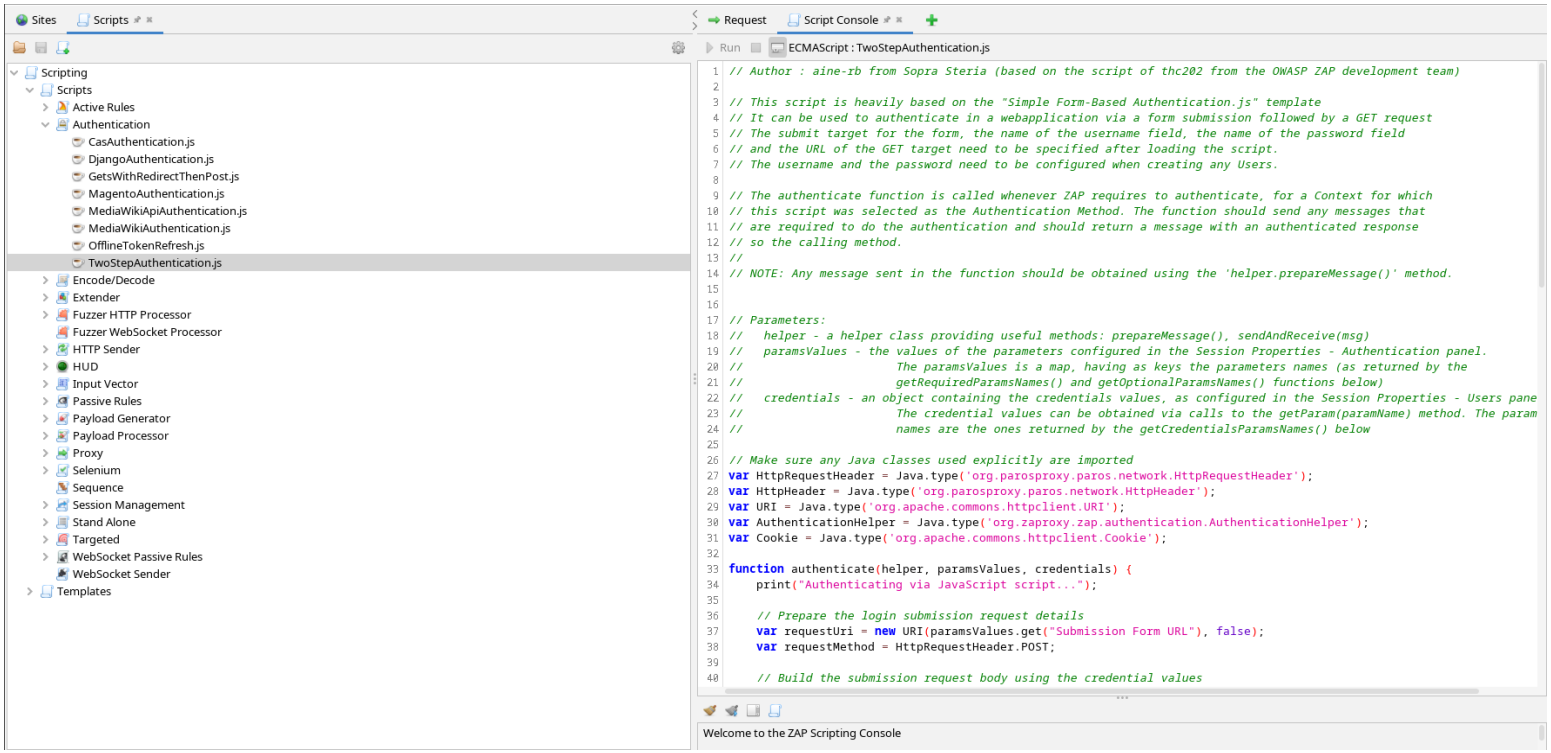
ZAProxy

To automate authentication in ZAProxy you have to set up the context properly.



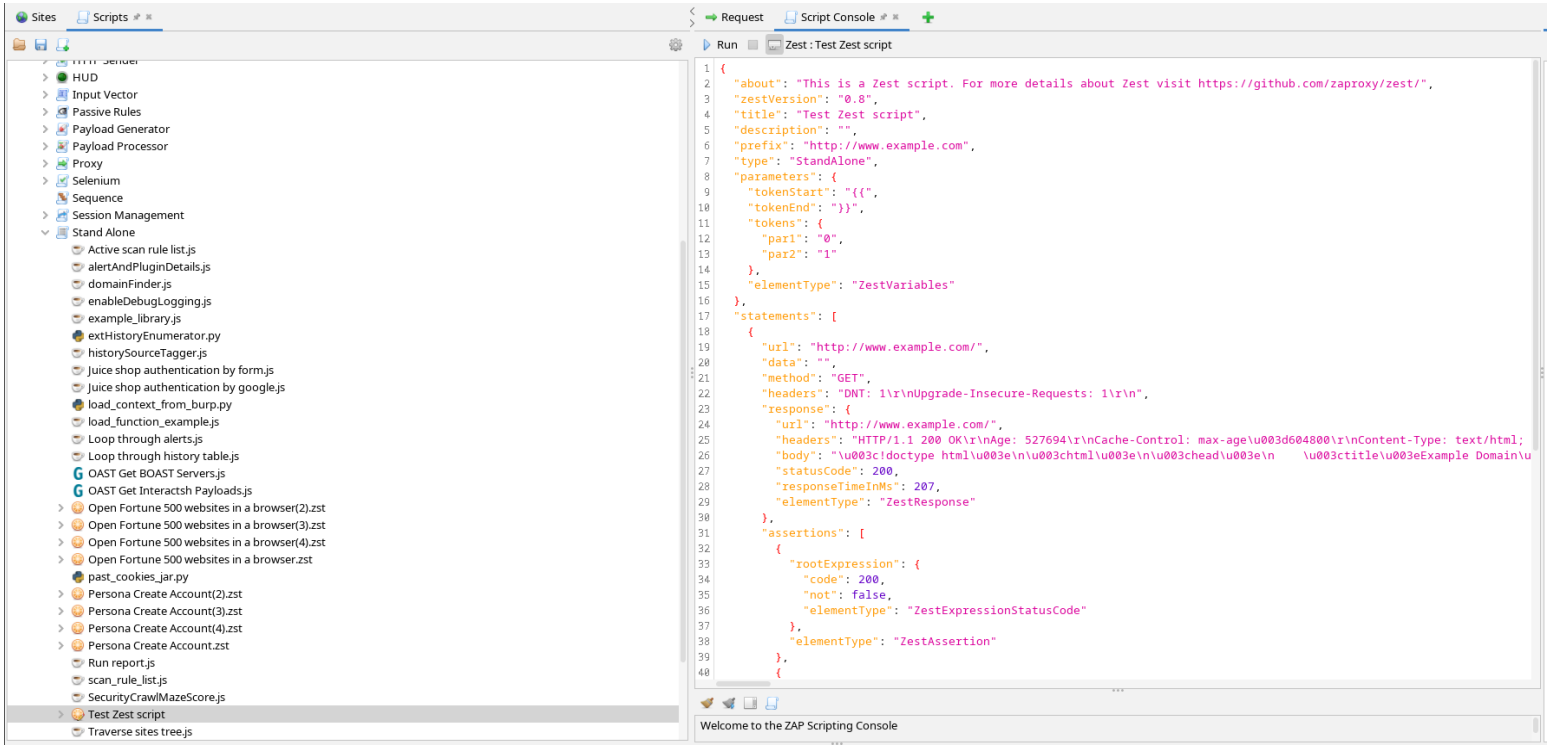
ZAProxy

ZAProxy provides some authentication scripts.



ZAProxy

HTTP processes can also be automated using Zest scripts.



ZAProxy

Using authentication auto-detection ZAProxy helps the user test and diagnose authentication issues.

ZAP can now automatically handle many types of authentication as detailed in the Authentication Auto-Detection blog post.

However it did mean you had to create an Automation Framework plan via the command line, which is not ideal.

It is now *much* easier to test if ZAP can handle your app's authentication with a brand new dialog!

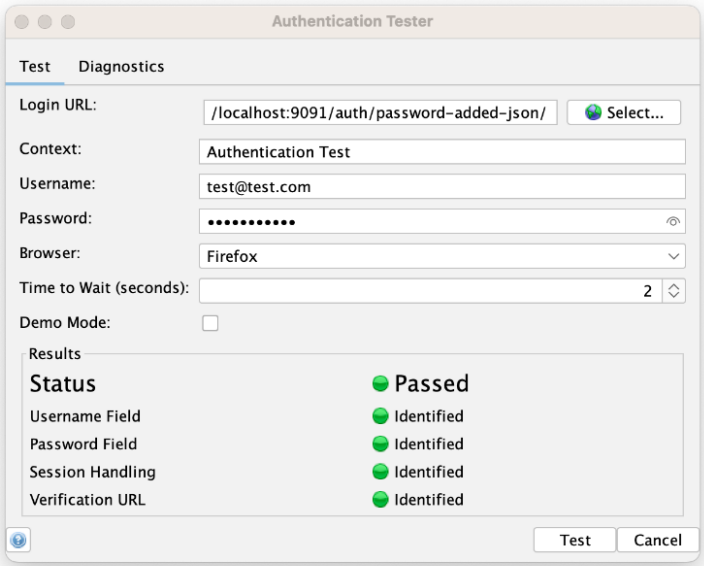
Authentication Tester Dialog

The Authentication Helper add-on now adds a new Authentication Tester dialog which can be accessed via the Tools menu item or the key combination: T.

Table of Contents

Tags

- blog
- authentication



ZAProxy

Authentication makes your life hard.

Authentication - Make your Life Easier

DOCUMENTATION > ZAP AUTHENTICATION > AUTHENTICATION - MAKE YOUR LIFE EASIER

Authentication is a key way of restricting access to an app. Some authentication mechanisms also make it significantly harder to use tools like ZAP, even for those people who have permission to use them.

Test in a Safe Environment

Testing with valid credentials in a production environment is a really bad idea. You will pollute data stores with invalid data and you always run the risk of taking the service down or impacting valid users in some other way.

Disable Security Controls

You are in a safe environment and you want to test the app not the security controls, so disable any firewalls or other security features that you use in production.

Disable or Simplify Authentication

If your app can be run with full functionality and without authentication then just do that - in this case you are testing the app, not the authentication controls.

Single Sign On systems can be especially hard to work with. If you can use a simpler authentication mechanism like HTTP auth or a simple POST form then do that - these options will be much easier to set up and much less likely to break your testing. Some SSO providers do document ways to authenticate automatically - see the next page.

Two factor authentication (2AF) can be even harder to work with. ZAP does not work by magic - if you want to perform automated scanning but need a 2FA token then you are going to need to be able to get that token to ZAP. If you cannot do that then you will not be able to automate your authentication.

If you are testing your own app then seriously consider what options you have you making it easier for you to test it using automation.

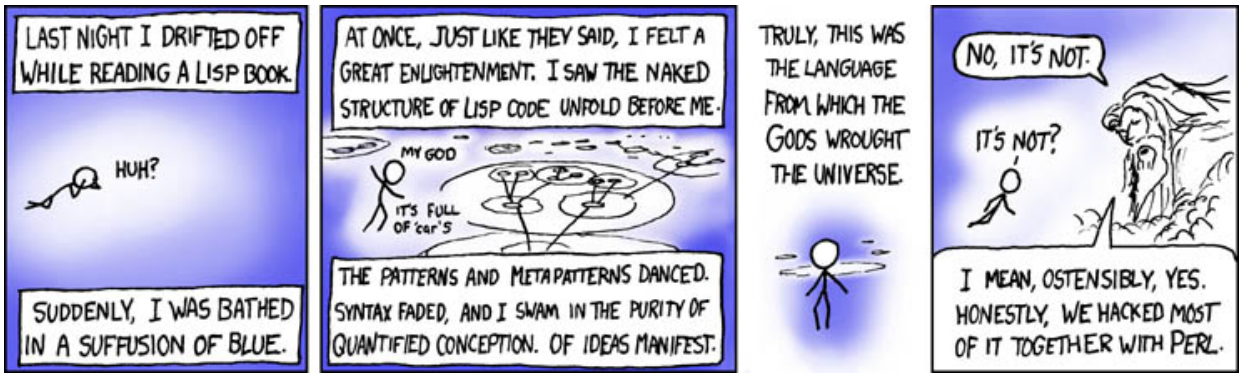
Test with the ZAP Desktop

Why not JSON?

- JSON gets complicated fast
- File gets long and editing it manually is painful
- Referencing previously defined items is ugly
- Complex syntax needed to process items (encode/decode/split/etc...)
- Can't reuse previously defined parts of json
- No user access to real code
- Not easily extendable
- And many more issues...

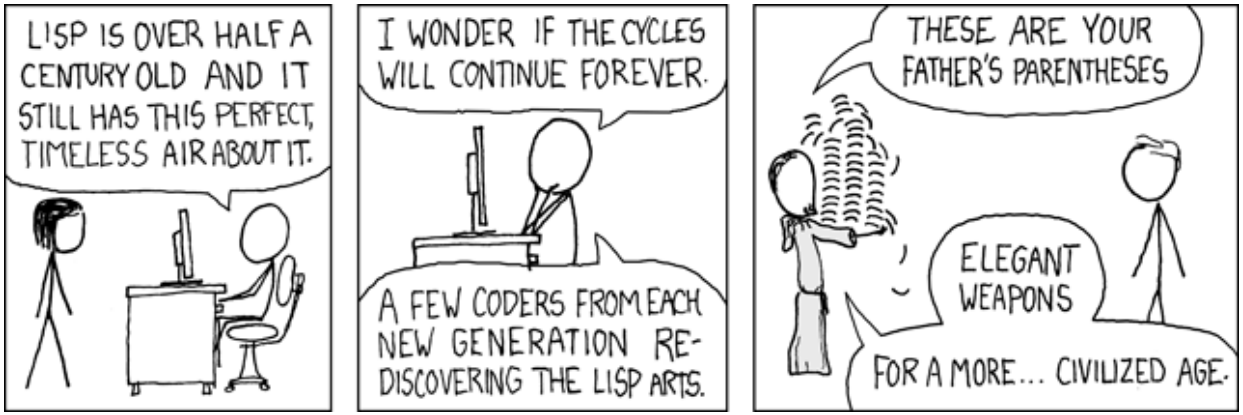
(Why LISP?)

- Need to create a language to describe the information exchange
- LISP languages are ideal for creating a custom DSL
- Ability to define own syntax
- Homoiconicity, i.e. code is data, data is code concept
- Metaprogramming, LISP macros can be defined to generate pieces of code



(Why hylang?)

- LISP-stick on a Python
- Compiles into Python code
- Access all the Python libraries
- Combines LISP flexibility with Python power and simplicity
- Easy to learn if you know Python

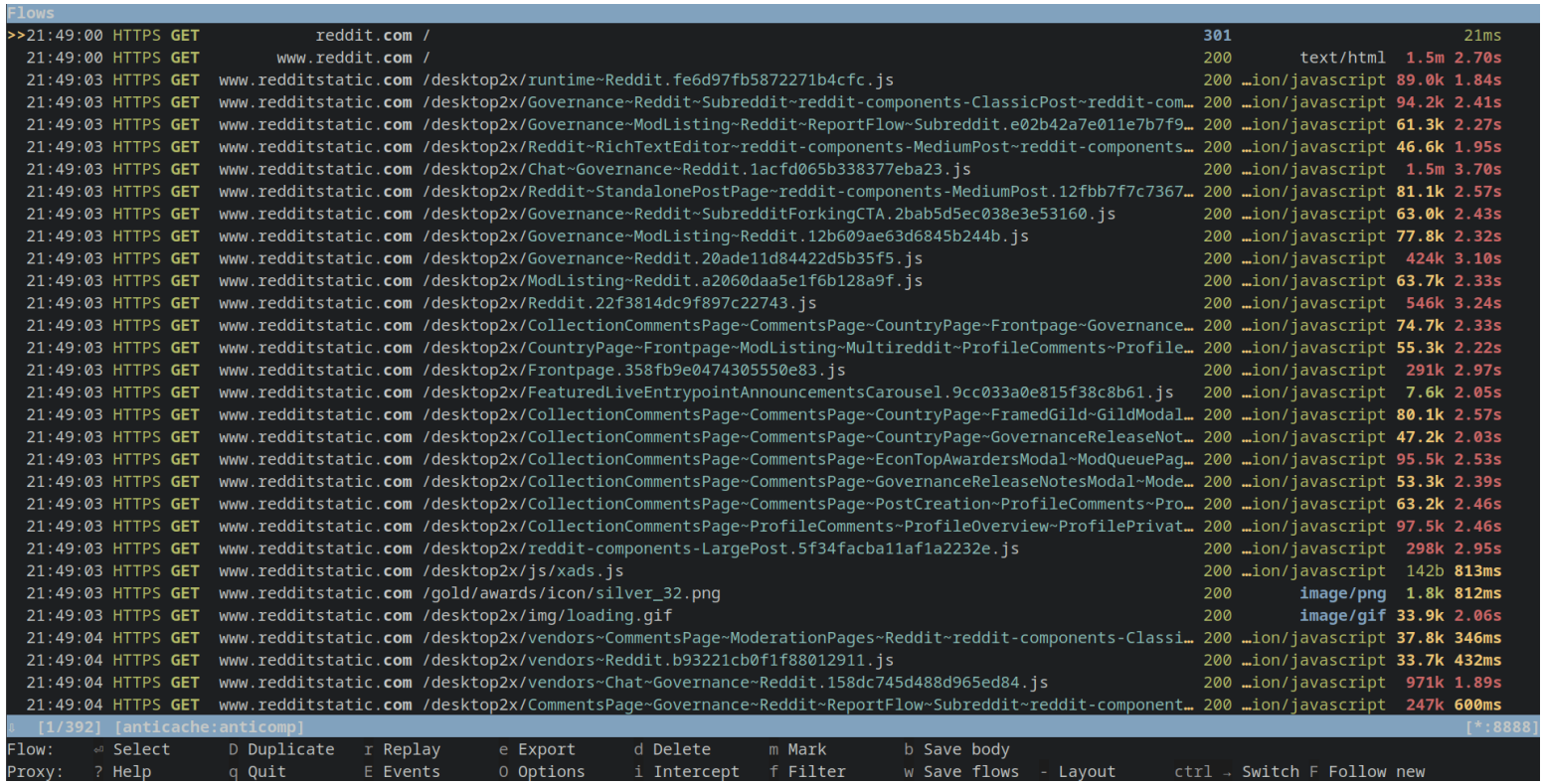


(Understanding authentication)

- Often seen as a black box by pentesters
- Many bugs are overlooked
- Raider aims to make it easier to test and understand complex HTTP processess, like authentication

(Understanding authentication)

392 requests captured during Reddit authentication. Most are irrelevant.



(Understanding authentication)

Removing obviously irrelevant requests makes it easier to analyse.

```
Flows
>>21:49:00 HTTPS GET       reddit.com /                                                    301                                21ms
  21:49:00 HTTPS GET   www.reddit.com /                                                    200       text/html  1.5m 2.70s
  21:49:09 HTTPS POST  www.reddit.com /errors                                             204                           233ms
  21:49:11 HTTPS POST …way.reddit.com /desktopapi/v1/sidebar_insertion?include=           200 …plication/json 29.7k 346ms
  21:49:11 HTTPS GET   www.reddit.com /account/sso/one_tap/?experiment_d2x_2020ify_buttons… 200       text/html 10.2k 140ms
  21:49:12 HTTPS GET   www.reddit.com /login/?experiment_d2x_2020ify_buttons=enabled&exper… 200       text/html 14.5k 574ms
  21:49:26 HTTPS POST  www.reddit.com /login                                              200 …plication/json   30b 487ms
  21:49:32 HTTPS POST  www.reddit.com /login                                              200 …plication/json   34b 413ms
  21:49:33 HTTPS POST  www.reddit.com /                                                   200 …plication/json    2b 122ms
  21:49:33 HTTPS GET   www.reddit.com /                                                   200       text/html  1.5m 2.73s
  21:49:36 HTTPS GET   www.reddit.com /account/sso/one_tap/?experiment_d2x_2020ify_buttons… 200       text/html  3.5k 220ms
  21:49:38 HTTPS POST  www.reddit.com /errors                                             204                           629ms
  21:49:39 HTTPS POST …way.reddit.com /desktopapi/v1/sidebar_insertion?allow_over18=1&incl… 200 …plication/json 30.0k 689ms
  21:49:41 HTTPS GET   …tar.reddit.com /api/account                                       200 …plication/json  333b 487ms
[1/14]  [anticache:anticomp]                                                                                        [*:8888]
Flow:    ⏎ Select     D Duplicate   r Replay     e Export     d Delete     m Mark       b Save body
Proxy:   ? Help       q Quit        E Events     O Options    i Intercept  f Filter     w Save flows  - Layout
```

(Understanding authentication)

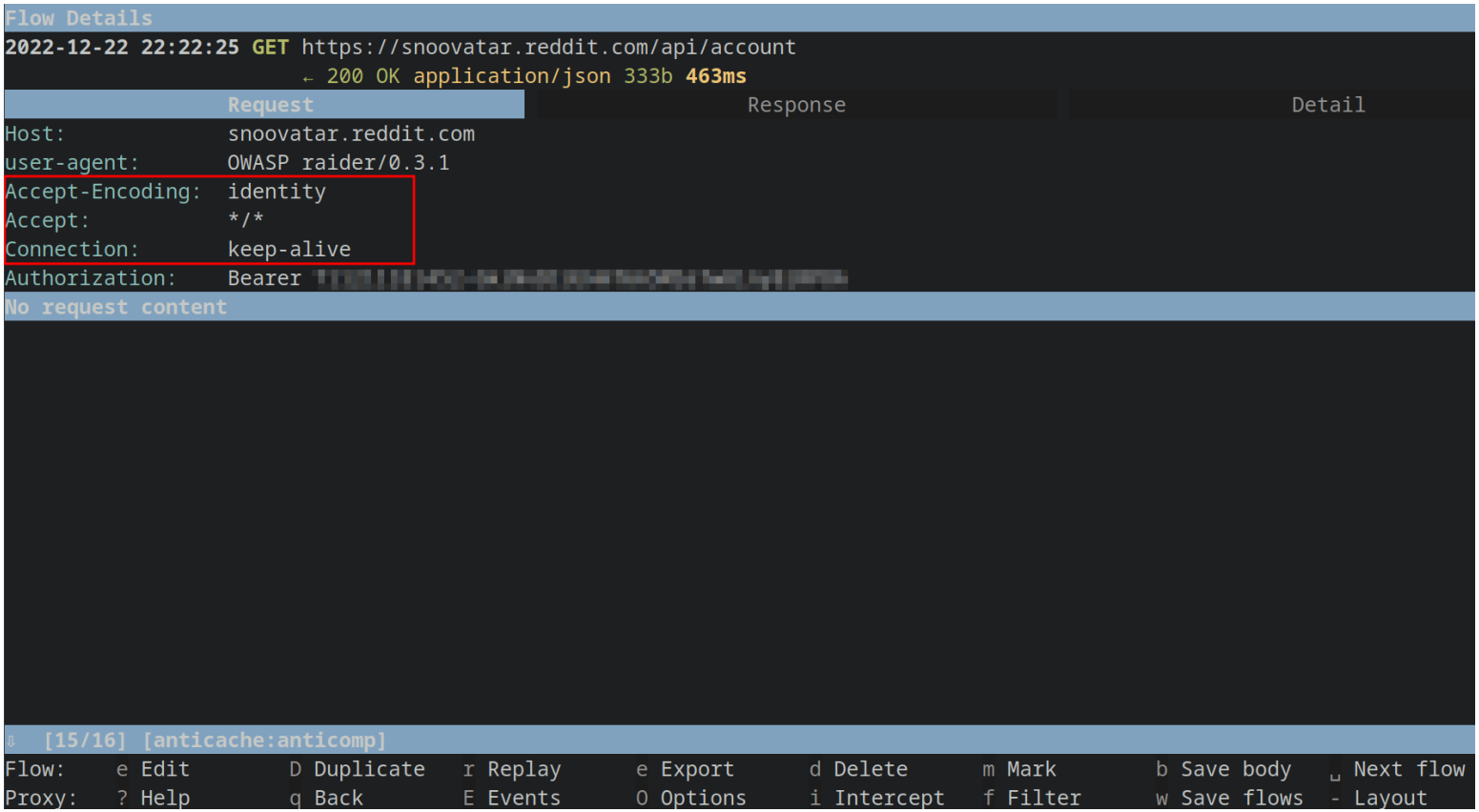
Find one request which only works when authenticated.

```
Flow Details
2022-12-22 21:49:41 GET https://snoovatar.reddit.com/api/account HTTP/2.0
                    ← 200 application/json 333b 487ms
                  Request                    Response                  Detail
user-agent:       Mozilla/5.0 (X11; Linux x86_64; rv:108.0) Gecko/20100101 Firefox/108.0
accept:           */*
accept-language:  en-US,en;q=0.5
accept-encoding:  identity
referer:          https://www.reddit.com/
content-type:     application/x-www-form-urlencoded
x-reddit-loid:    [redacted]
x-reddit-session: [redacted]
origin:           https://www.reddit.com
dnt:              1
sec-fetch-dest:   empty
sec-fetch-mode:   cors
sec-fetch-site:   same-site
authorization:    Bearer [redacted]
te:               trailers
No request content                                                    [m:auto]

[14/14] [anticache:anticomp]                                          [*:8888]
Flow:    e Edit       D Duplicate   r Replay     e Export     d Delete     m Mark       b Save body   ␣ Next flow
Proxy:   ? Help       q Back        E Events     O Options    i Intercept  f Filter     w Save flows  - Layout
```

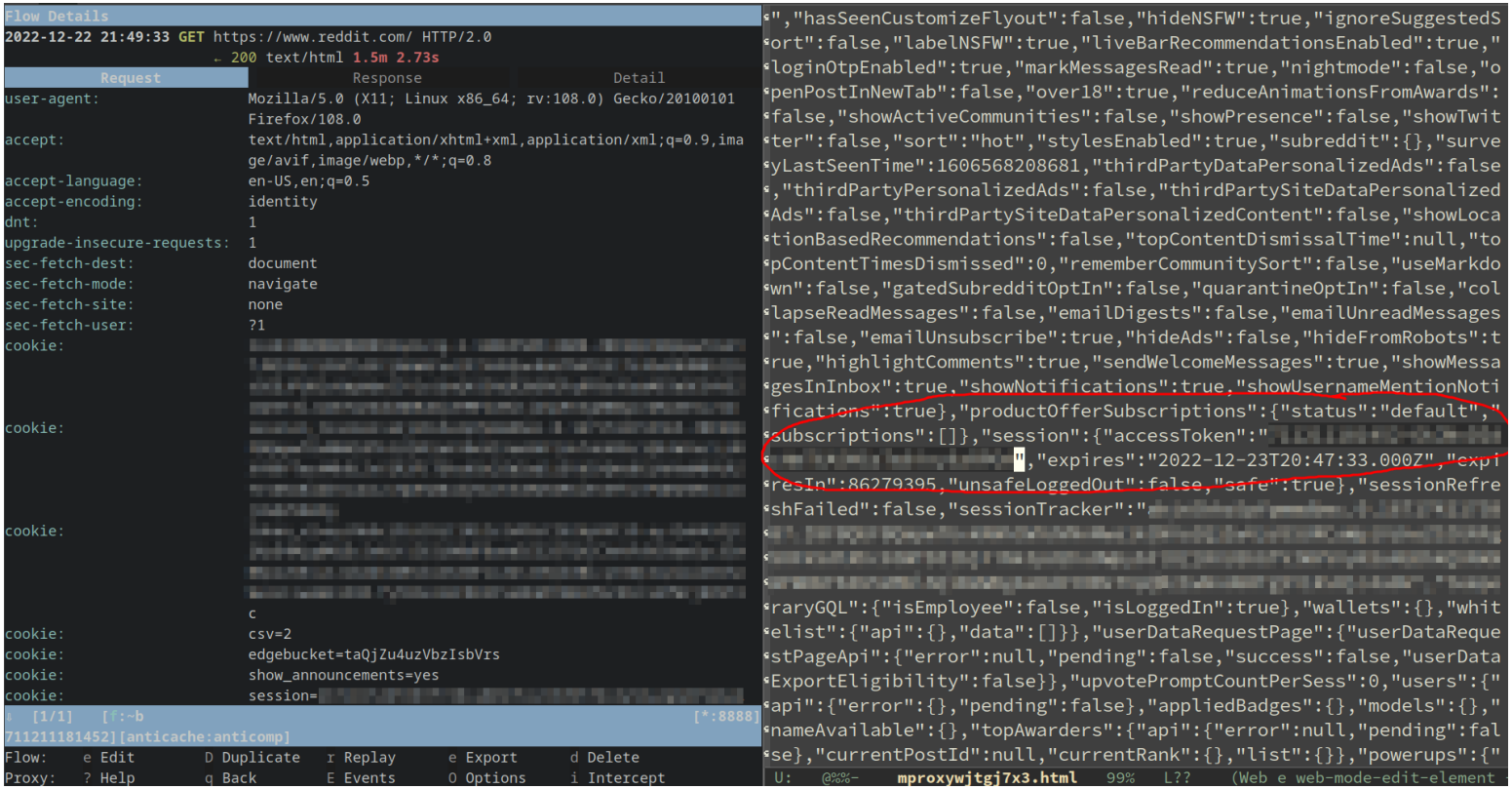
(Understanding authentication)

Only Authorization header is needed here. Host/User-agent are required by HTTP. Highlighted headers are generated automatically.



(Understanding authentication)

Finding the Authorization header token in the response body of a previous request.



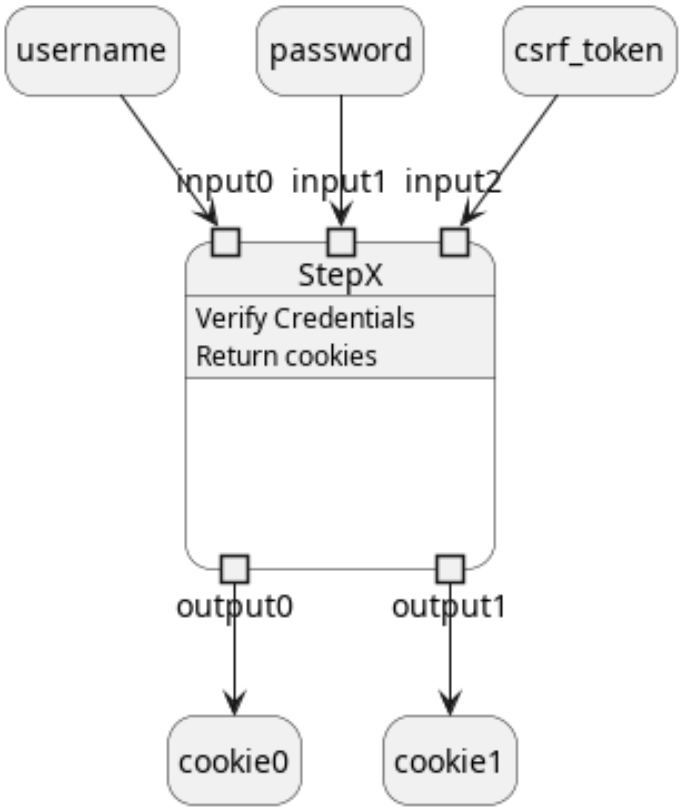
(Abstracting Authentication using Finite State Machines)

Finite State Machine (FSM) is a mathematical model of computation. It allows for a detailed analysis of how a computer system functions.

- A system is stateful if it remembers preceding events
- A state is the remembered information about the system
- Can be in exactly one of the finite number of states at any given time
- *Mealy* FSMs can be used to model authentication systems
- Output values are determined both by current state and inputs

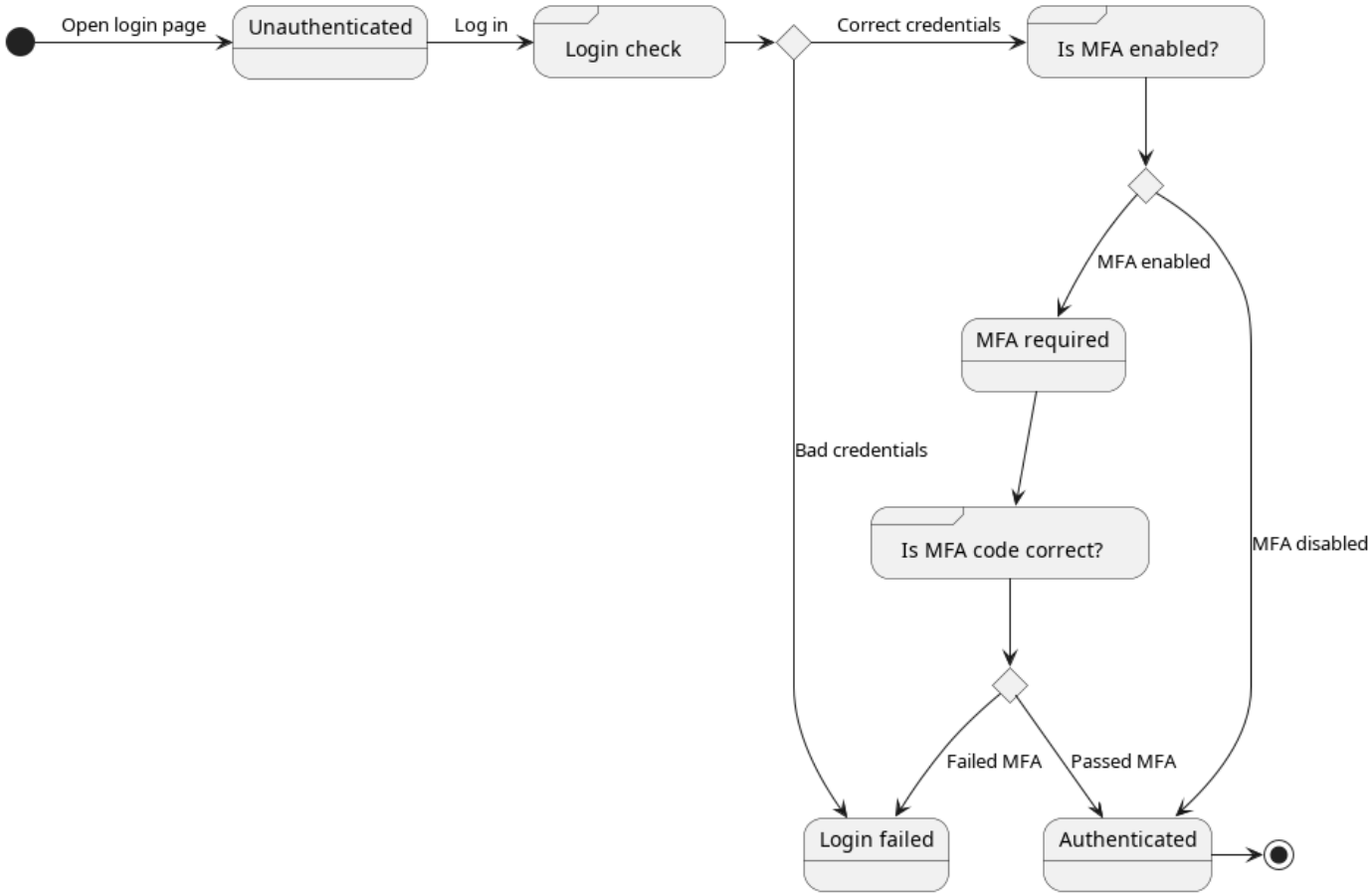
(Abstracting Authentication using Finite State Machines)

One step (state) of the authentication with its inputs and outputs:



(Abstracting Authentication using Finite State Machines)

Authentication represented as FSM with its inputs/outputs hidden:



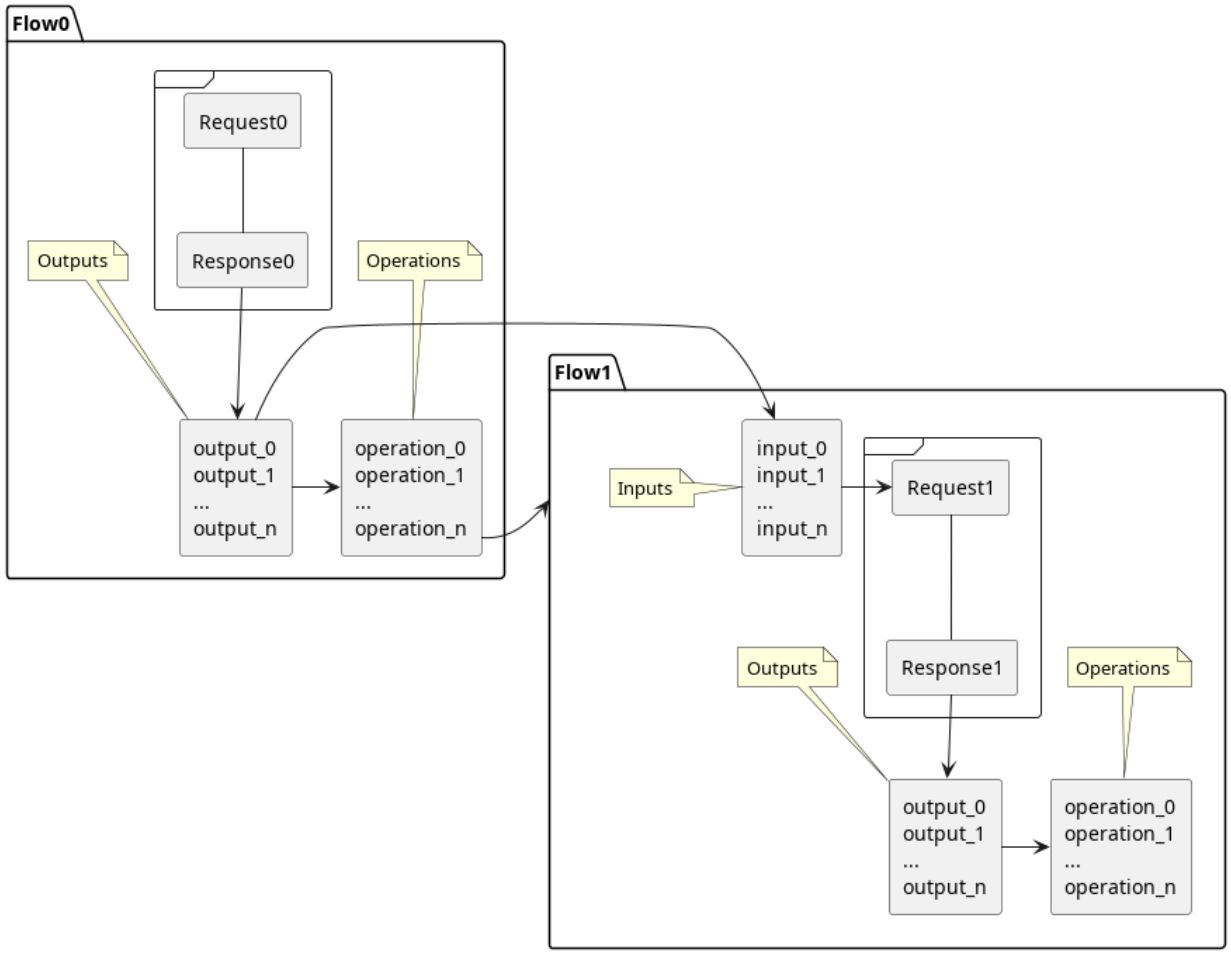
(Flows)

Flows are used to describe the information exchange between the client and the server with one pair of HTTP request and the response.

- Requires a Request with URL
- Optionally `outputs`
  - defines what to extract from the response
- Optionally `operations`
  - arbitrary actions to run after receiving response
  - links to other Flows (could be conditional, and nested)

(Flows)

Raider Flows represent one state in the FSM.



(Request)

- The only required parameter is the URL, rest are optional
  - Requests HTTP method are specified via the class methods. `Request.get`, `Request.post`, `Request.put`, `Request.custom`
  - `:cookies`, `:headers`
  - `:params`, `:data`, `:json`, `:multipart`
- All Request parameters can contain Plugins. Use those to share data between Flows.

(Plugins)

- Small pieces of code
- Used as inputs and/or outputs
- Extract data
- Parse data/Plugins
- Encode/Decode data/Plugins
- Some can be nested
- User can write their own without touching the core

(Operations)

- Run *after* Response is received
- Execute arbitrary code
- Control the information flow
  - `Next, Success, Failure`
- Run other Operations conditionally (can be nested)
- Can run even real hylang code by using the LISP *quote*
- User can write their own without touching the core

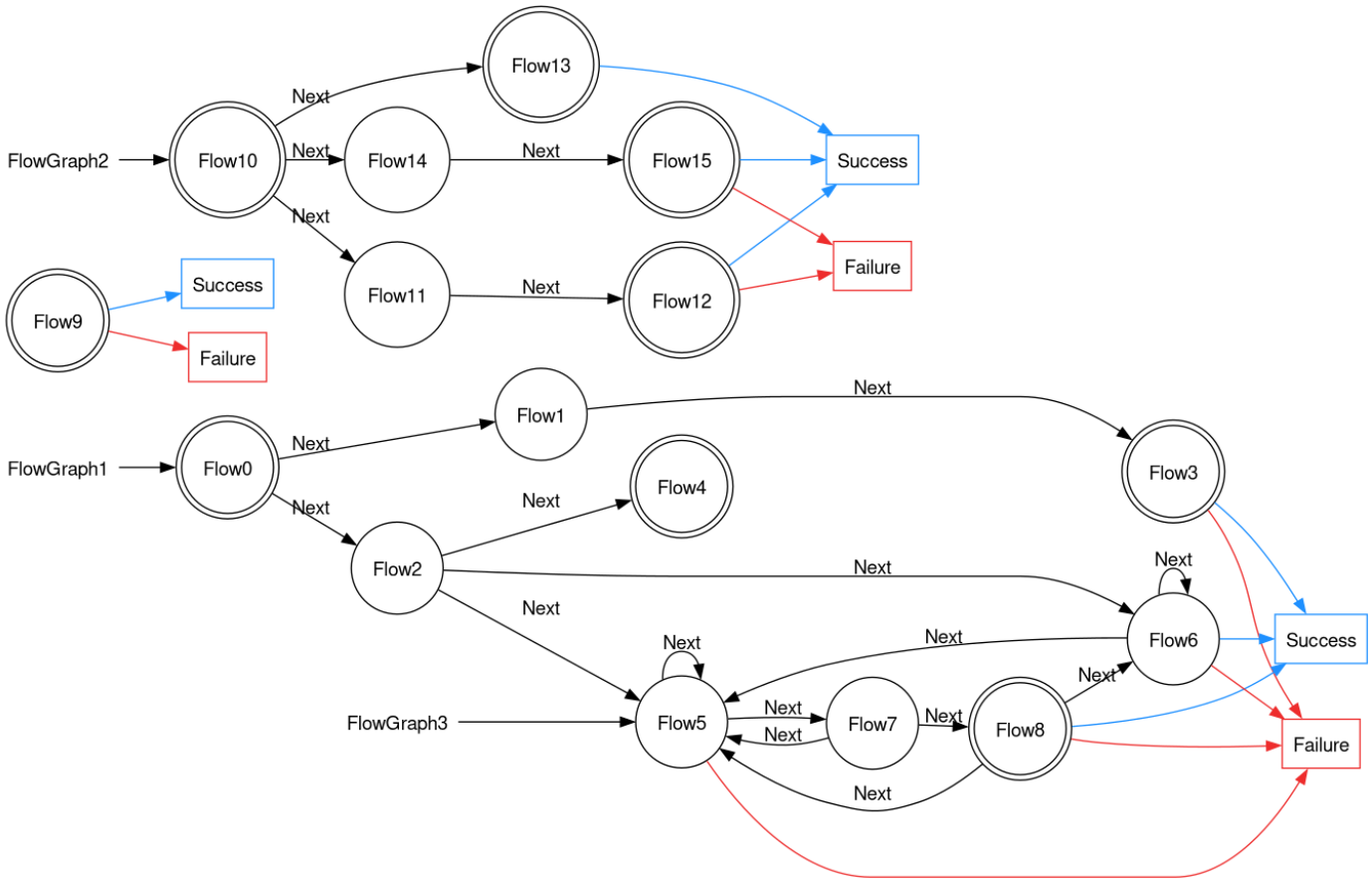
(FlowGraphs)

FlowGraphs are used to chain multiple Flows together and follow the links until the end, or until `(Success)/(Failure)` operations.

- Pointer to a starting Flow
- Optionally a test Flow. Checks if the FlowGraph ran successfully, i.e. if user is authenticated

(FlowGraphs)

Complex systems can be simulated and tested with this architecture:

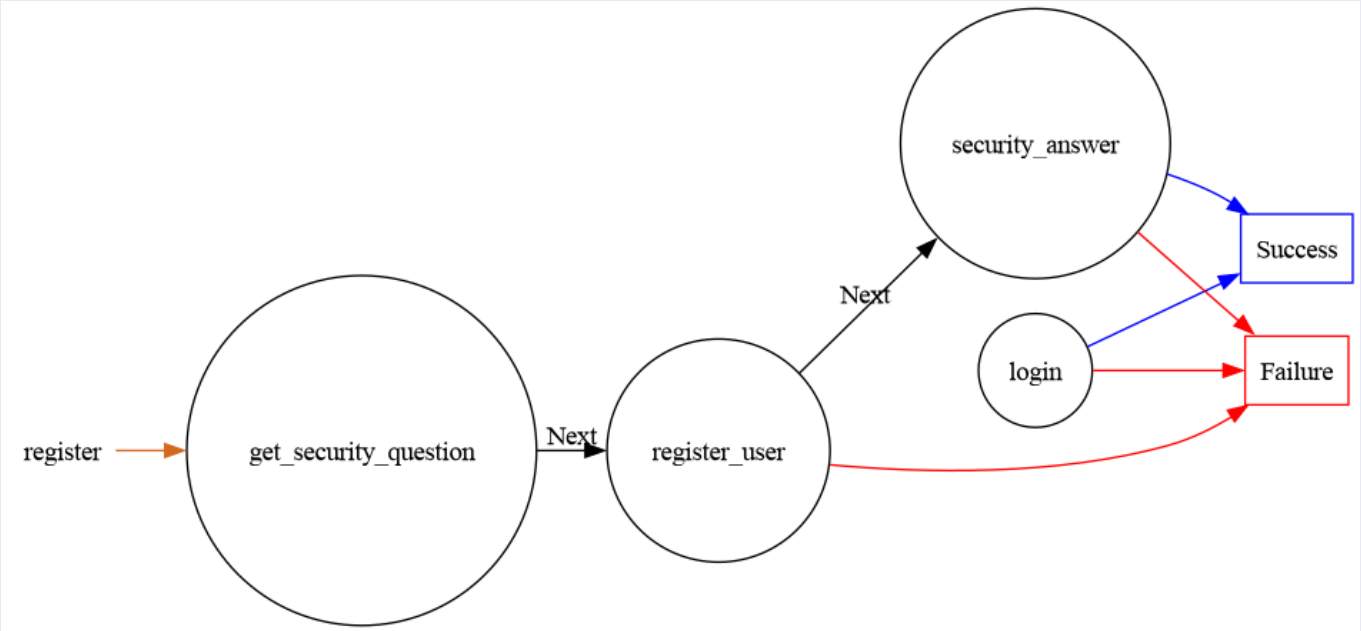


(Demo: Automating juiceshop attacks)

Automate registration and login

Register a new user and log in. Run `register` FlowGraph then the `login` Flow.

```
$ raider run juiceshop register,login   # First run
$ raider run juiceshop login            # Subsequent runs
```

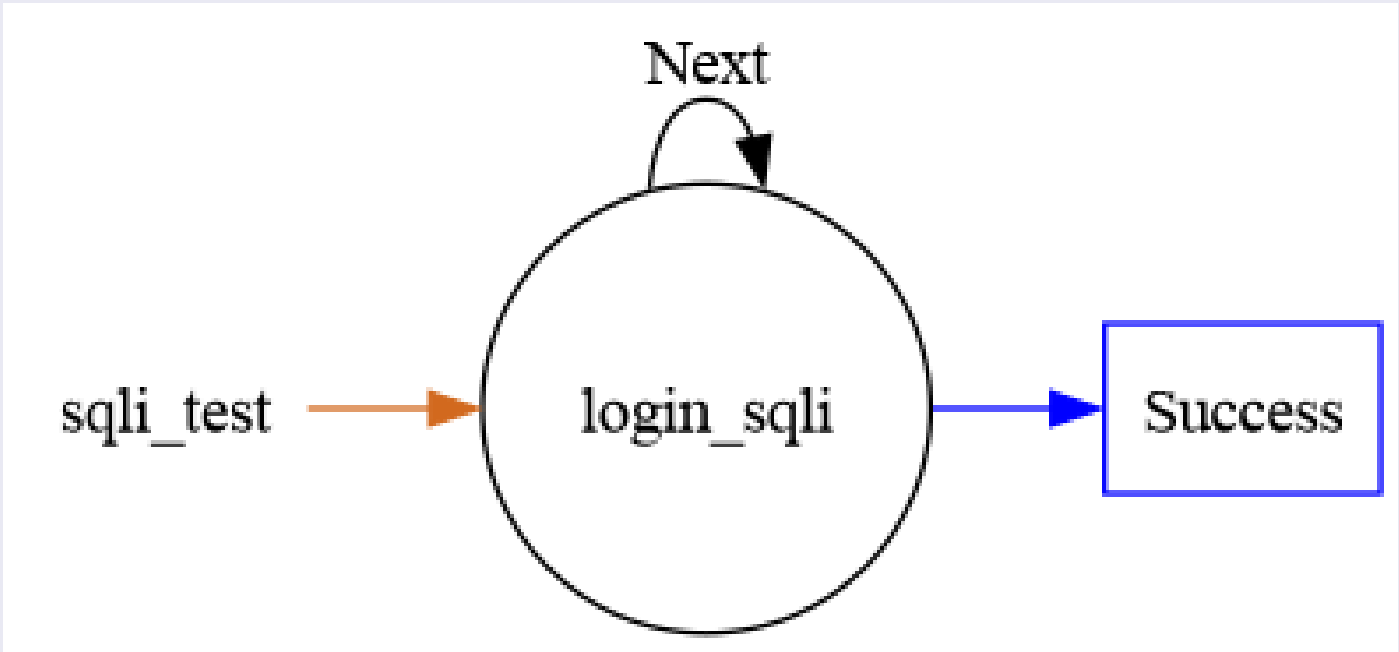


(Demo: Automating juiceshop attacks)

Test SQL injection in email field

Email input is vulnerable to SQL injection. We run the Flow in a loop and exit on `(Success)`.

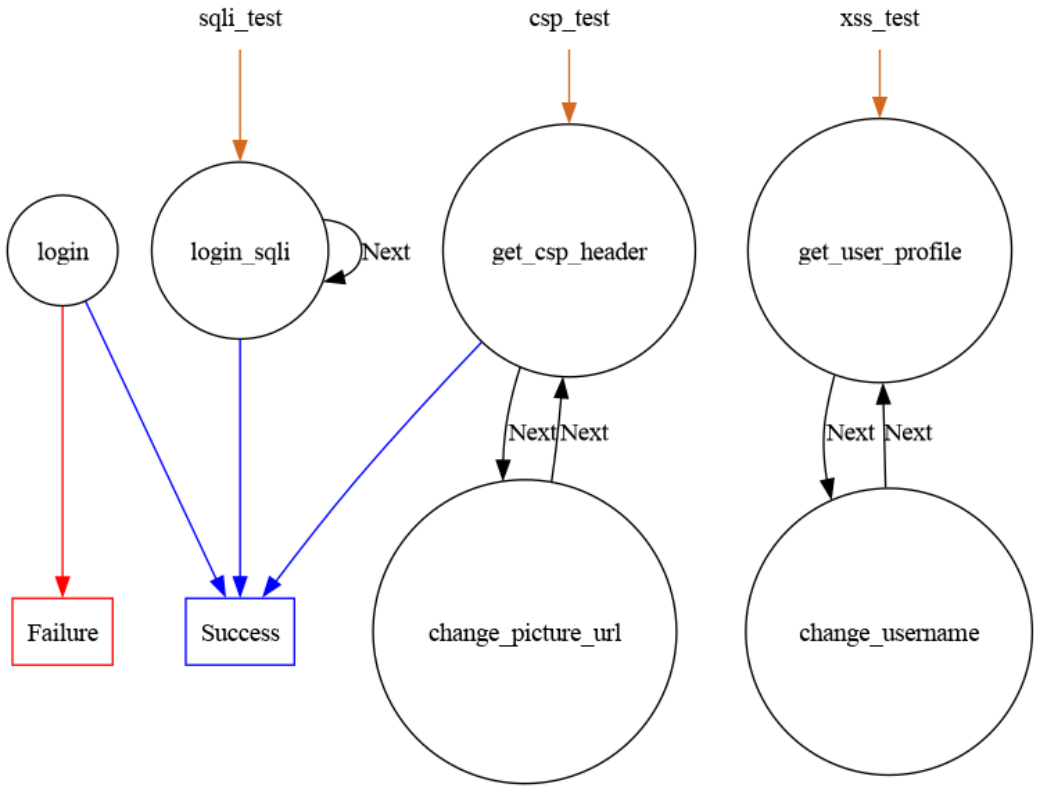
```
$ raider run juiceshop sqli_test
```



(Demo: Automating juiceshop attacks)

Username field is vulnerable to persistent XSS. Input is filtered with a regex, and can be bypassed. Content-Security-Policy header is used as well, and due to another bug can be bypassed too.

```
$ raider run login,csp_test,xss_test
$ raider run sqli_test,csp_test,xss_test
```



(What's next?)

- Documentation
- UI/UX
  - Debugging
  - Improve CLI
  - Raider REPL (read-eval-print loop)
  - Generate hyfiles using LLMs (help needed)
- Features
  - Fuzzing
  - Sessions
  - Macros
- Integrating with other tools

(Limitations)

- Steep learning curve
- LISP parentheses
- Limited community support
- Limited documentation
- Limited OS support

(Questions/Answers)

Raider is not just a toy anymore, it evolved enough to work on complex real life systems. There's still a lot of work to do and room for improvement

Contact me

- Mastodon: @raiderauth@infosec.exchange
- E-mail: hello@raiderauth.com

Links

- Website: raiderauth.com
- Source: github.com/OWASP/raider
- Documentation: docs.raiderauth.com