

Automating and
attacking
complex HTTP
processes with
OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

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

Automating and
attacking
complex HTTP
processes with
OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

1 Introduction

2 Methodology

3 Demo

4 Conclusions

5 Q/A

\$ whoami

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru



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

Automating and
attacking
complex HTTP
processes with
OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

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

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

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

The screenshot displays the BurpSuite interface with a Repeater tab selected. The target URL is `https://leadids.msauth.net`. The request is a GET request to `/jslibs.svg` with various headers including `Host: www.example.com`, `User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:107.0) Gecko/20100101 Firefox/107.0`, and `Accept: image/avif,image/webp,*/*`. The response is empty. The interface includes a 'Send' button, a 'Raw' view toggle, and a search bar at the bottom.

```
1 GET /jslibs.svg HTTP/2
2 Host: www.example.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:107.0) Gecko/20100101 Firefox/107.0
4 Accept: image/avif,image/webp,*/*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Sec-Fetch-Dest: image
8 Sec-Fetch-Mode: no-cors
9 Sec-Fetch-Site: cross-site
10
11
12
```

BurpSuite

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

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

Macro editor

Use the configuration below to define the items that are included in the macro, and the order they will be issued. You can configure how parameters and cookies are handled for each item. You can also test the macro to confirm it is working correctly.

Macro descriptor: Macro 1

Macro items:

#	Host	Method	URL	Status	Cookies received	Derived parameters	Preset parameters	Accept cookies	Use cookies
1	https://www.eur-de.us-hrings...	GET	/	200	ABMAFtrivy, ABMAFtrivySessId				✓
2	https://login.microsoftonline.com	GET	/	200	lsaid, fpc, exch, x-ms-gateway...	response_type, client_id, state...		✓	✓
3	https://login.live.com	GET	/	200	lsaid, MSRPKey			✓	✓
4	https://login.microsoftonline.com	POST	/	200	lsaid, fpc, x-ms-gateway...			✓	✓
5	https://login.microsoftonline.com	POST	/	200	lsaid, fpc, x-ms-gateway...			✓	✓
6	https://login.microsoftonline.com	GET	/	200	lsaid, fpc, x-ms-gateway...			✓	✓

Request | **Response**

```
POST /connect/SRP/RESP HTTP/1.1
Host: login.microsoftonline.com
Cookie: lsaid=...
```

Configure Custom Item

Configure how cookies and request parameters are handled for this macro item.

Add cookies received in responses to the session handling cookie jar

Use cookies from the session handling cookie jar in requests

Parameter handling

Parameter	Derive from prior response	Response
sts	Derive from prior response	Response 1
apprequestid	Derive from prior response	Response 3
flowtoken	Derive from prior response	Response 4
currentpassword	Use preset value	password123
refreshpassword	Use preset value	password106

Custom parameter locations in response

Name	From	Value
domain	From {domain} to {express}	

Define Custom Parameter

Configure the details of the custom parameter location. You need to specify the name that is used for this parameter in subsequent macro requests, and the location within this response from which the parameter's value should be derived.

Parameter name:

Extracted value is URL-encoded

Define the location of the parameter value. Selecting the item in the response panel will create a suitable configuration automatically. You can also modify the configuration manually to ensure it works effectively.

Define start and end

Start after expression:

Start at offset: Case sensitive

End at delimiter:

End at fixed length:

Exclude HTTP headers Update config based on selection below

```
51 <meta name="LACL" content="en-US" />
52
53 <meta name="format-detection" content="telephone=no" />
54
55 </NOCSRF>
56 <meta http-equiv="refresh" content="5;
57 URL=https://login.microsoftonline.com/jwtbearer" />
58 </NOCSRF>
59
60
61
62 <meta name="robots" content="noindex" />
63
64 <script type="text/javascript">+function() {
65   $ajax$.ajax({url:"https://login.microsoftonline.com/connect/ProcessAuth",
66     "url":$urlPostRedirect});
67 }
```

ZAPProxy

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

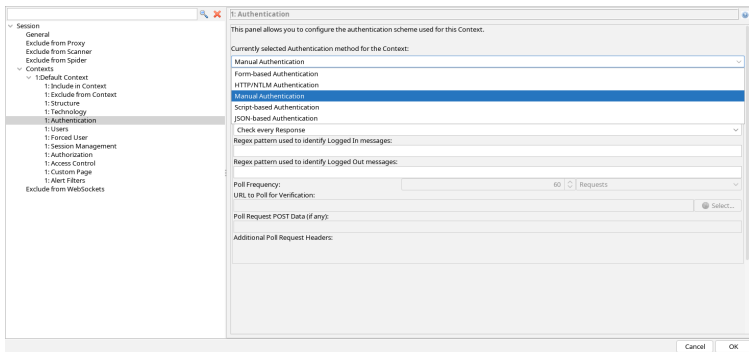
Methodology

Demo

Conclusions

Q/A

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



ZAPProxy

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

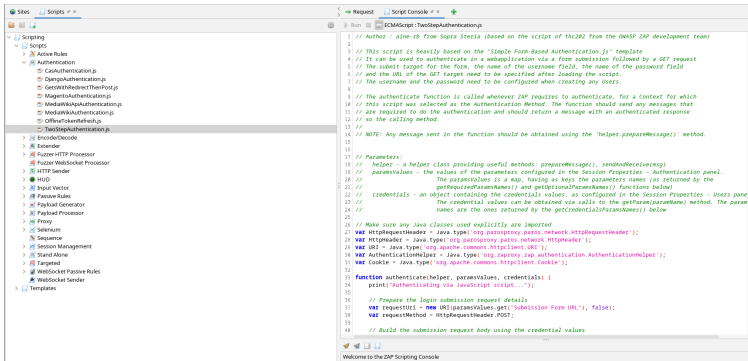
Methodology

Demo

Conclusions

Q/A

ZAPProxy provides some authentication scripts.



```
1 // Author : aine-ib from Sopra Steria (based on the script of thc282 from the OWASP ZAP development team)
2 //
3 // This script is heavily based on the "Simple Form-Based Authentication.js" template
4 // It can be used to authenticate in a webapplication via a form submission followed by a GET request
5 // The submit target for the form, the name of the username field, the name of the password field
6 // and the url of the GET target need to be specified after loading the script.
7 // The username and the password need to be configured when creating any users.
8 //
9 // The authenticate function is called whenever ZAP requires to authenticate, for a context for which
10 // this script was selected as the Authentication Method. The function should send any messages that
11 // are required to do the authentication and should return a message with an authenticated response
12 // to the calling method.
13 //
14 // NOTE: Any message sent in the function should be obtained using the 'helper.prepareMessage()' method.
15 //
16 // Parameters:
17 // helper - a helper class providing useful methods: prepareMessage(), sendMsg/recvMsg()
18 // paramValues - the values of the parameters configured in the Session Properties - Authentication panel.
19 // The paramValues is a map, having as keys the parameters names (as returned by the
20 // getRequiredParameters() and getOptionalParameters() functions below)
21 // credentials - an object containing the credentials values, as configured in the Session Properties - Users pane
22 // The credential values can be obtained via calls to the getParam(paramName) method. The param
23 // names are the ones returned by the getCredentialsParameters() below
24 //
25 // Make sure any Java classes used explicitly are imported
26 //
27 var HttpRequestHeader = Java.type('org.parosproxy.paros.network.HttpRequestHeader');
28 var HttpHeaders = Java.type('org.parosproxy.paros.network.HttpHeader');
29 var URL = Java.type('org.apache.commons.httpclient.URI');
30 var AuthenticationHeader = Java.type('org.zaproxy.zap.authentication.AuthenticationHeader');
31 var Cookie = Java.type('org.apache.commons.httpclient.Cookie');
32
33 function authenticate(helper, paramValues, credentials) {
34     print("Authenticating via Javascript script...");
35
36     // Prepare the login submission request details
37     var requestURI = new URL(paramValues.get("Submission Form URL"), false);
38     var requestMethod = HttpRequestHeader.POST;
39
40     // Build the submission request body using the credential values
```


ZAProxy

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

HTTP processes can also be automated using Zest scripts.

The screenshot displays the OWASP ZAP interface. On the left, a tree view shows various scripts, with 'Test Zest scripts' selected. The main area shows a Zest script named 'Zest: Test Zest script' with the following code:

```
1 | {
2 |   "about": "This is a Zest script. For more details about Zest visit https://github.com/zaproxy/zest/",
3 |   "testVersion": "0.0.0",
4 |   "title": "Test Zest script",
5 |   "description": "",
6 |   "url": "http://www.example.com",
7 |   "type": "StandAlone",
8 |   "parameters": {
9 |     "tokenStart": "{{",
10 |    "tokenEnd": "}}",
11 |    "tokens": {
12 |      "part1": "ng",
13 |      "part2": "3"
14 |    }
15 |   },
16 |   "elementType": "ZestVariables",
17 |   "statements": [
18 |     {
19 |       "url": "http://www.example.com",
20 |       "data": "",
21 |       "method": "GET",
22 |       "headers": "DNT: 1;Icmpgrade-Insecure-Requests: 1;tr",
23 |       "response": {
24 |         "url": "http://www.example.com/",
25 |         "headers": "HTTP/1.1 200 OK(nginx): 527094c1u4Cache-Control: max-age=0;36044800;trContent-Type: text/html;
26 |         body": "u0802c;duType: html;u081e;u083c;html;u083e;u083c;head;u083e;u083c;title;u083e;u083c;example: Domain;
27 |         statusCode": 200,
28 |         responseTimeInfo": 207,
29 |         "elementType": "ZestResponse"
30 |       },
31 |       "assertions": [
32 |         {
33 |           "rootExpression": {
34 |             "code": 200,
35 |             "root": false,
36 |             "elementType": "ZestExpressionStatusCode"
37 |           },
38 |           "elementType": "ZestAssertion"
39 |         }
40 |       ]
41 |     }
42 |   ]
43 | }
```

Below the script editor, the 'Script Console' shows the output: 'Welcome to the ZAP Scripting Console'.

ZAProxy

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

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.

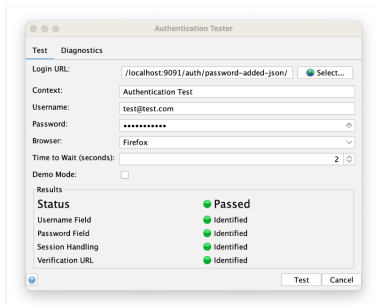


Table of Contents

[Authentication Tester Dialog](#)
[What if it Doesn't Work?](#)
[Give Us Feedback](#)
[Sponsored by Jit](#)

Tags

- [blog](#)
- [authentication](#)

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 (2FA) 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?

Automating and
attacking
complex HTTP
processes with
OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

- 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?)

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

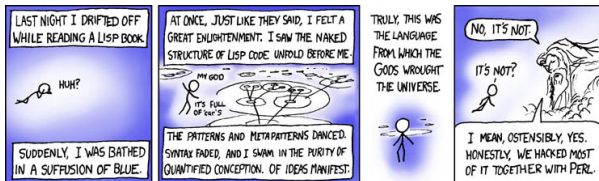
Methodology

Demo

Conclusions

Q/A

- 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?)

Automating and
attacking
complex HTTP
processes with
OWASP Raider

Daniel Neagaru

Introduction

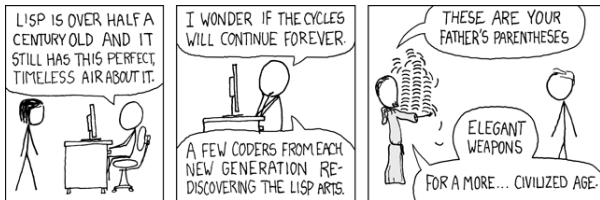
Methodology

Demo

Conclusions

Q/A

- 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)

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

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



(Understanding authentication)

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

392 requests captured during Reddit authentication.
Most are irrelevant.

```
time
>>21:49:00 HTTPS GET www.reddit.com / 301 21ms
21:49:00 HTTPS GET www.reddit.com / 200 text/html 1.5m 2.78s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/runtime-Reddit_fe6d977fb5872271b4fc.js 200 _lon/javascript 89.8k 1.84s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/Governance-Reddit-Subreddit-reddit-components-ClassicPost-reddit-com... 200 _lon/javascript 94.2k 2.41s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/Governance-ModListing-Reddit-ReportFlow-Subreddit_e02b42a7e011e7b7f9... 200 _lon/javascript 61.3k 2.27s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/Reddit-RichTextEditor-reddit-components-MediumPost-reddit-components... 200 _lon/javascript 46.6k 1.95s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/Chat-Governance-Reddit_1acfd865b338377eb23.js 200 _lon/javascript 1.5m 3.70s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/Reddit-StandalonePostPage-reddit-components-MediumPost_12fb77fc7367... 200 _lon/javascript 81.1k 2.57s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/Governance-Reddit-SubredditForkingCTA_2bab5d5ec038e3e53160.js 200 _lon/javascript 63.0k 2.43s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/Governance-ModListing-Reddit_12b609ae63d6845b244b.js 200 _lon/javascript 77.8k 2.32s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/Governance-Reddit_20ade11d84422d5b35f5.js 200 _lon/javascript 424k 3.10s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/ModListing-Reddit_a206daa5e1f6b128a9f.js 200 _lon/javascript 63.7k 2.33s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/Reddit_22f3814dc9f897c22743.js 200 _lon/javascript 546k 3.24s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/CollectionCommentsPage-CommentsPage-CountryPage-Frontpage-Governance... 200 _lon/javascript 474.7k 3.33s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/CountryPage-Frontpage-ModListing-Multireddit-ProfileComments-Profile... 200 _lon/javascript 55.3k 2.22s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/Frontpage_358fb9e047430550e83.js 200 _lon/javascript 291k 2.97s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/FeaturedLiveEntryPointAnnouncementsCarousel_9cc033a0e815f38c8b61.js 200 _lon/javascript 7.6k 2.05s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/CollectionCommentsPage-CommentsPage-CountryPage-FramedGild-GildModal... 200 _lon/javascript 80.1k 2.57s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/CollectionCommentsPage-CommentsPage-CountryPage-GovernanceReleaseNot... 200 _lon/javascript 47.2k 2.03s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/CollectionCommentsPage-CommentsPage-EconTopAwardsModal-ModQueuePag... 200 _lon/javascript 95.5k 2.53s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/CollectionCommentsPage-CommentsPage-GovernanceReleaseNotesModal-Mode... 200 _lon/javascript 53.3k 2.39s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/CollectionCommentsPage-CommentsPage-PostCreation-ProfileComments-Pro... 200 _lon/javascript 63.2k 2.46s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/CollectionCommentsPage-ProfileComments-ProfileOverview-ProfilePrivat... 200 _lon/javascript 97.5k 2.46s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/reddit-components-LargePost_5f34facballa1fa2232e.js 200 _lon/javascript 298k 2.95s
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/js/ads.js 200 _lon/javascript 142b 0.11ms
21:49:03 HTTPS GET www.redditstatic.com /gold/awards/icon/allver_32.png 200 image/png 1.8k 0.12ms
21:49:03 HTTPS GET www.redditstatic.com /desktop2x/img/loading.gif 200 image/gif 33.9k 2.06s
21:49:04 HTTPS GET www.redditstatic.com /desktop2x/vendors-CommentsPage-ModerationPages-Reddit-reddit-components-Classi... 200 _lon/javascript 37.8k 346ms
21:49:04 HTTPS GET www.redditstatic.com /desktop2x/vendors-Reddit_b93221cb0f1f80012911.js 200 _lon/javascript 33.7k 432ms
21:49:04 HTTPS GET www.redditstatic.com /desktop2x/vendors-Chat-Governance-Reddit_158dc745d488d965ed84.js 200 _lon/javascript 971k 1.89s
21:49:04 HTTPS GET www.redditstatic.com /desktop2x/CommentsPage-Governance-Reddit-ReportFlow-Subreddit-reddit-component... 200 _lon/javascript 247k 600ms
[1/392] [anti/trace: anti/cookie]
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 ctrl - Switch F Follow new
```


(Understanding authentication)

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

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)

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

Find one request which only works when authenticated.

The screenshot shows the OWASP Raider interface with a REST client flow details view. The flow is titled "Flow Details" and shows a GET request to "https://snoovatar.reddit.com/api/account HTTP/2.0" with a status of 200 and a response size of 333b, taking 487ms. The request headers are:

```
Request
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
```

The response content is partially visible as "to request content". The interface also shows a menu at the bottom with options like "Flow: e Edit", "Proxy: ? Help", "D Duplicate", "q Back", "r Replay", "E Events", "e Export", "O Options", "d Delete", "i Intercept", "m Mark", "f Filter", "b Save body", "w Save flows", "Next flow", and "Layout".

(Understanding authentication)

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

Flow Details

2022-12-22 22:22:25 GET https://snoovatar.reddit.com/api/account
- 200 OK application/json 333b 463ms

Request	Response	Detail
Host: snoovatar.reddit.com		
user-agent: OWASP raider/0.3.1		
Accept-Encoding: identity		
Accept: */*		
Connection: keep-alive		
Authorization: Bearer [REDACTED]		

No request content

[15/16] [anticache:anticomp]

Flow: e Edit D Duplicate r Replay e Export d Delete m Mark b Save body u Next flow
Proxy: ? Help q Back E Events O Options i Intercept f Filter w Save flows - Layout

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

(Abstracting Authentication using Finite State Machines)

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

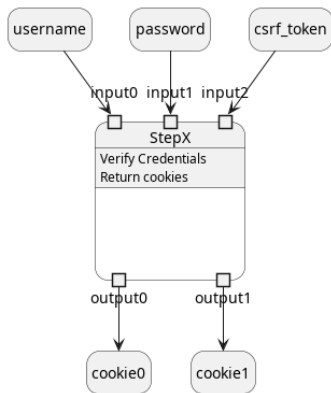
Q/A

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**

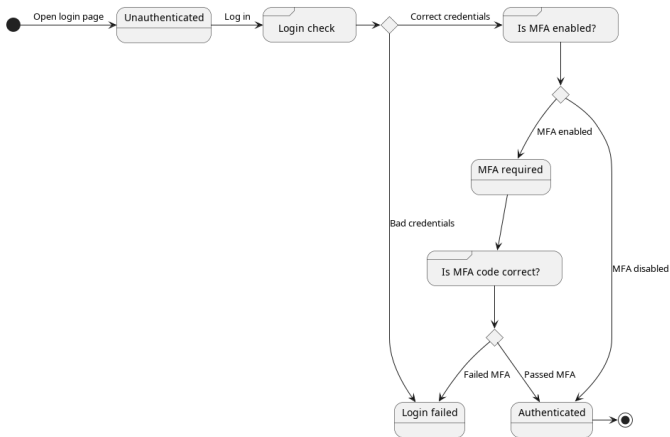
(Abstracting Authentication using Finite State Machines)

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)

Automating and
attacking
complex HTTP
processes with
OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

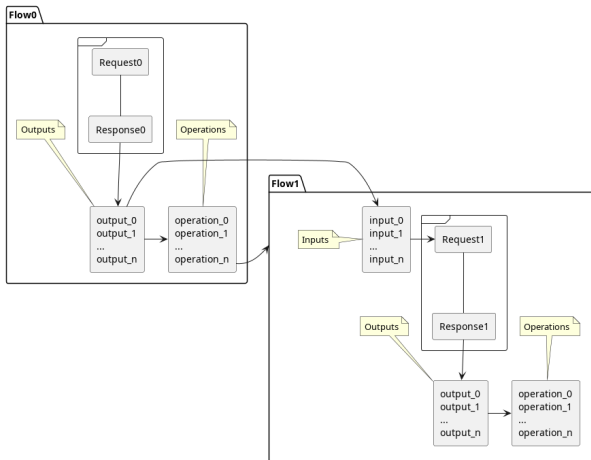
Q/A

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)

Automating and
attacking
complex HTTP
processes with
OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

- 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)

Automating and
attacking
complex HTTP
processes with
OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

- 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)

Automating and
attacking
complex HTTP
processes with
OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

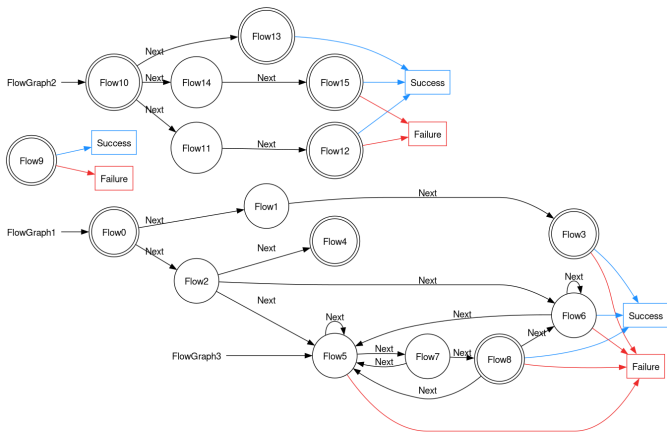
Q/A

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)

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

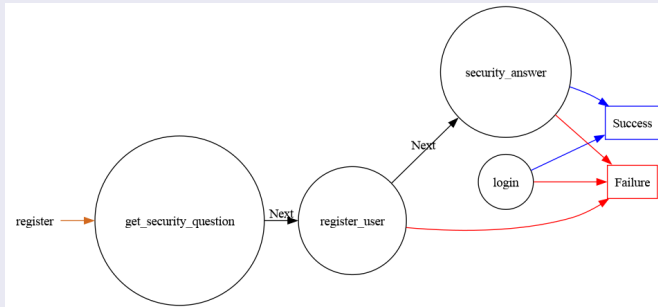
Conclusions

Q/A

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)

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

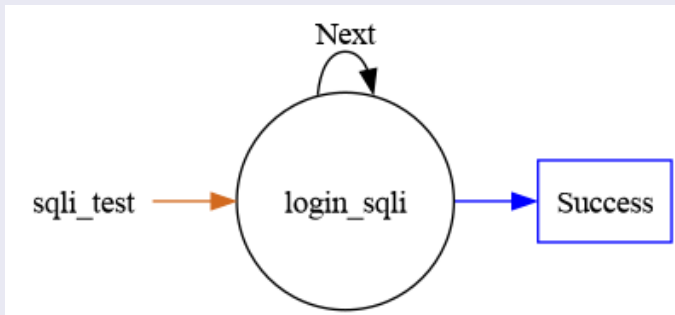
Conclusions

Q/A

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)

Automating and attacking complex HTTP processes with OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

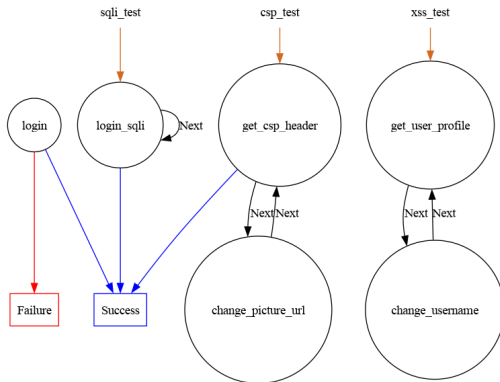
Conclusions

Q/A

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?)

Automating and
attacking
complex HTTP
processes with
OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

- 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)

Automating and
attacking
complex HTTP
processes with
OWASP Raider

Daniel Neagaru

Introduction

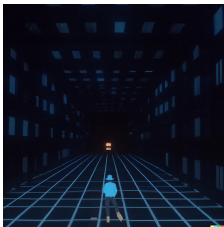
Methodology

Demo

Conclusions

Q/A

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



(Questions/Answers)

Automating and
attacking
complex HTTP
processes with
OWASP Raider

Daniel Neagaru

Introduction

Methodology

Demo

Conclusions

Q/A

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