Exploring boto3 Events With Mitmproxy

This QR code links to these slides and the code examples.

I'll post it up at the end but I also have it here so you can follow along on your own device if you want.

- Hi! 👋 I'm Michael Twomey 🇮🇪
- I started my career in Sun Microsystems working on the Solaris OS in 1999 (when Y2K was a thing)
- I've Been coding in Python for over 20 years 🐍
- Started kicking the tyres of AWS back when it was just S3, EC2 and SQS
- I'm now a Senior Cloud Architect at fourTheorem

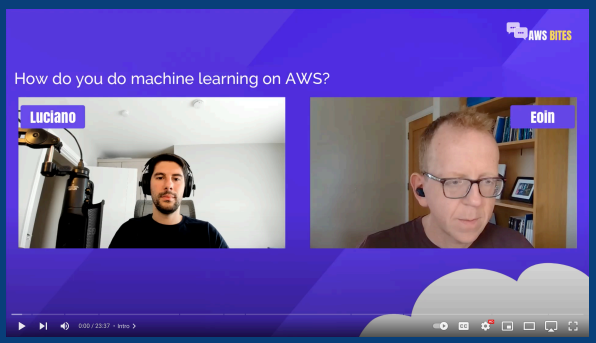
A quick note on fourTheorem:

We are a technology consultancy focused on AWS and serverless

We help with cloud architecture, application modernisation and even HPC.

I'm giving fourTheorem a shout out as it was in the context of a problem a client was having that this whole talk came to be!

AWS Bites



I'd also be remiss if I didn't give a plug to the podcast AWS Bites, a weekly podcast from my colleagues Eoin and Luciano

I'll be talking about solving a problem

I'm going to go through a problem from beginning to end

I'll show what issues I ran into and how I solved them

I will try to give just enough explanation of everything I use

There are many ways to achieve what I wanted, this is just one path!

I'll be talking about a few different things:

A bit of AWS (just enough to understand the problem)

Some boto3, the python AWS client SDK

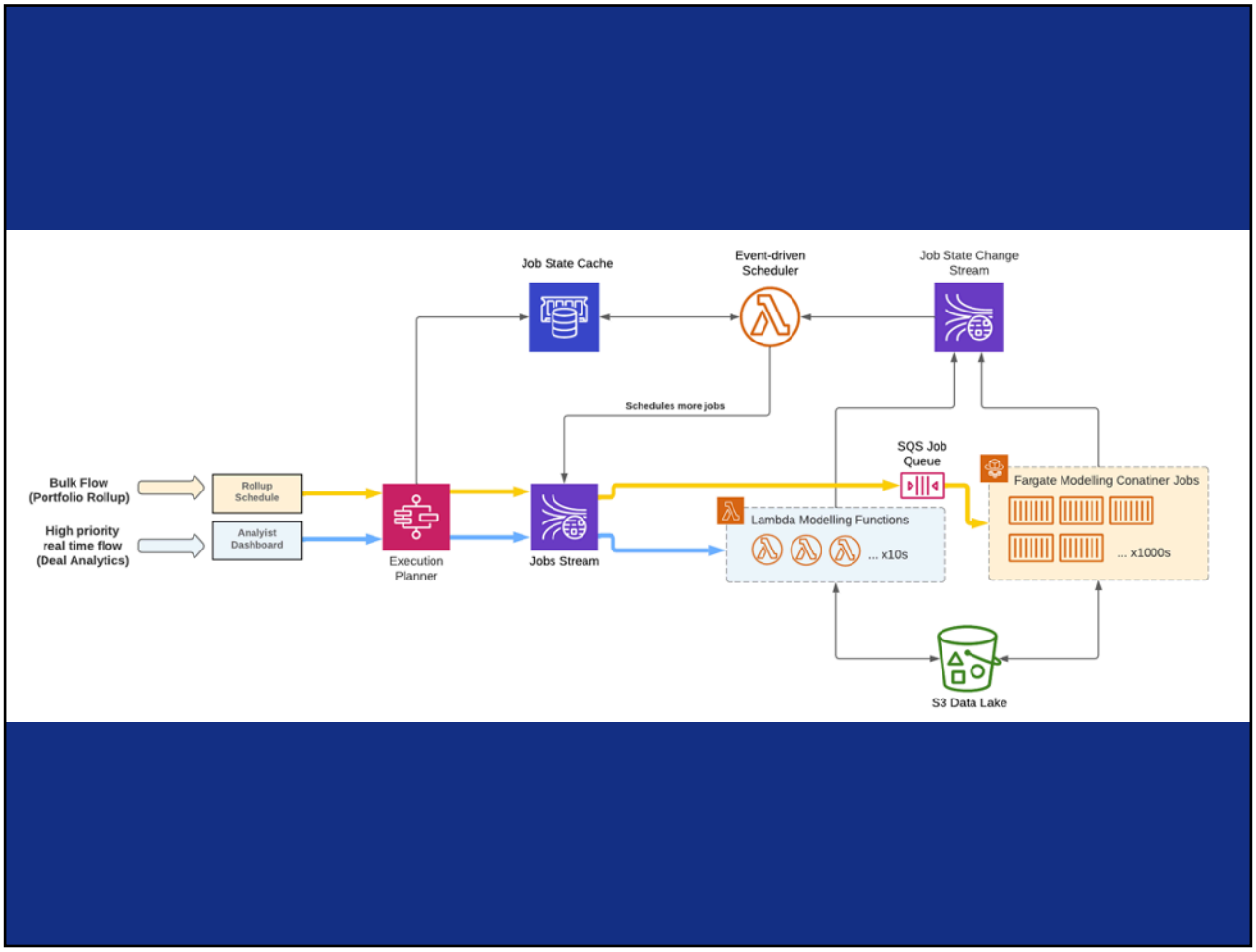
A little bit about HTTP and TLS

Finally I'll be showing a tool I like to use called mitmproxy

The goal is to show how I tackled a problem using these tools and the speed bumps I hit along the way

The Setup

Before I talk about the problem I need to talk a little bit about the system and how the relevant parts of it work.



The code base uses Python and the boto3 library to talk to AWS

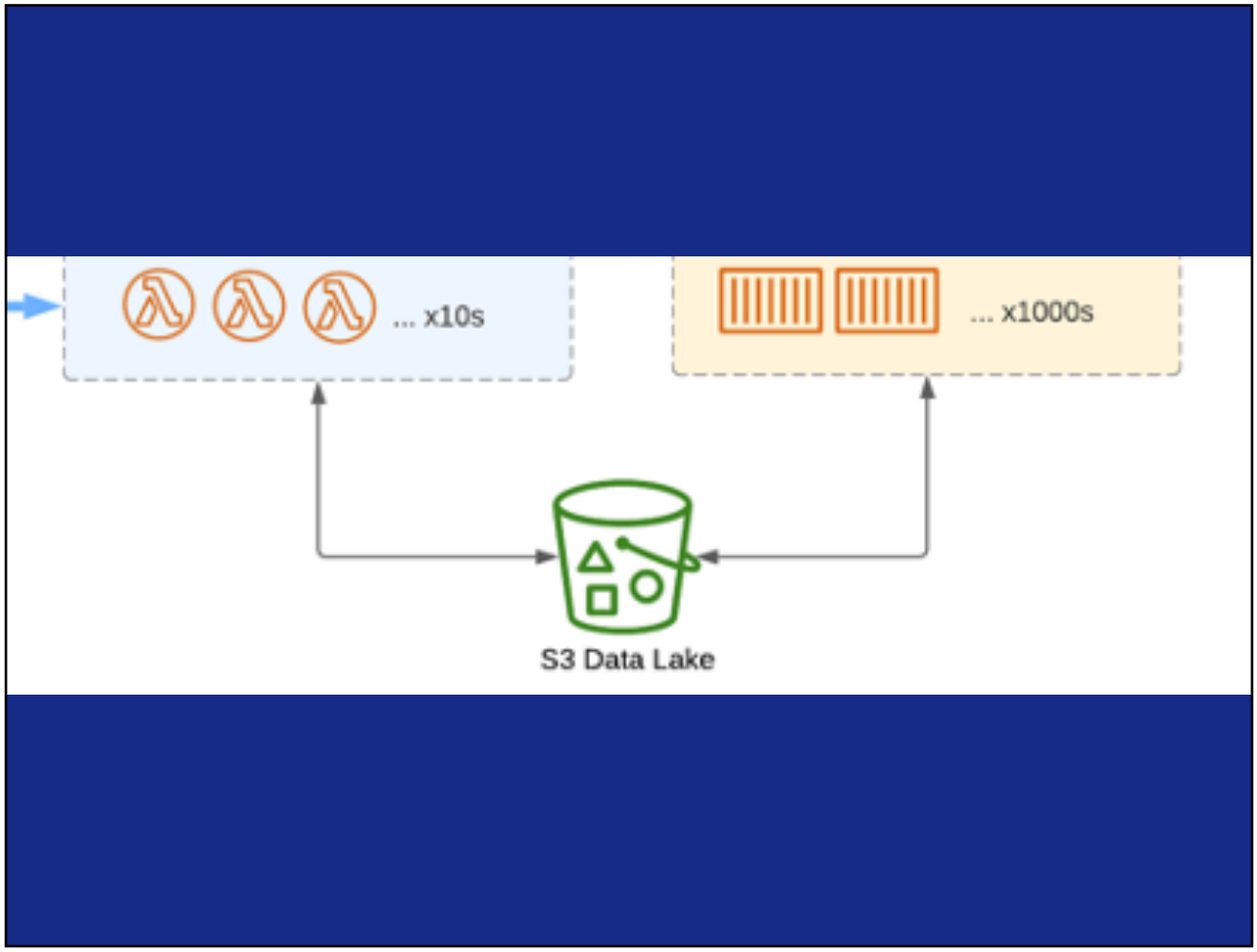
The core of the system runs a huge amount of computations spread over a large amount of jobs, running on either Lambda or Fargate containers.

Each job will read and write data from and to S3.

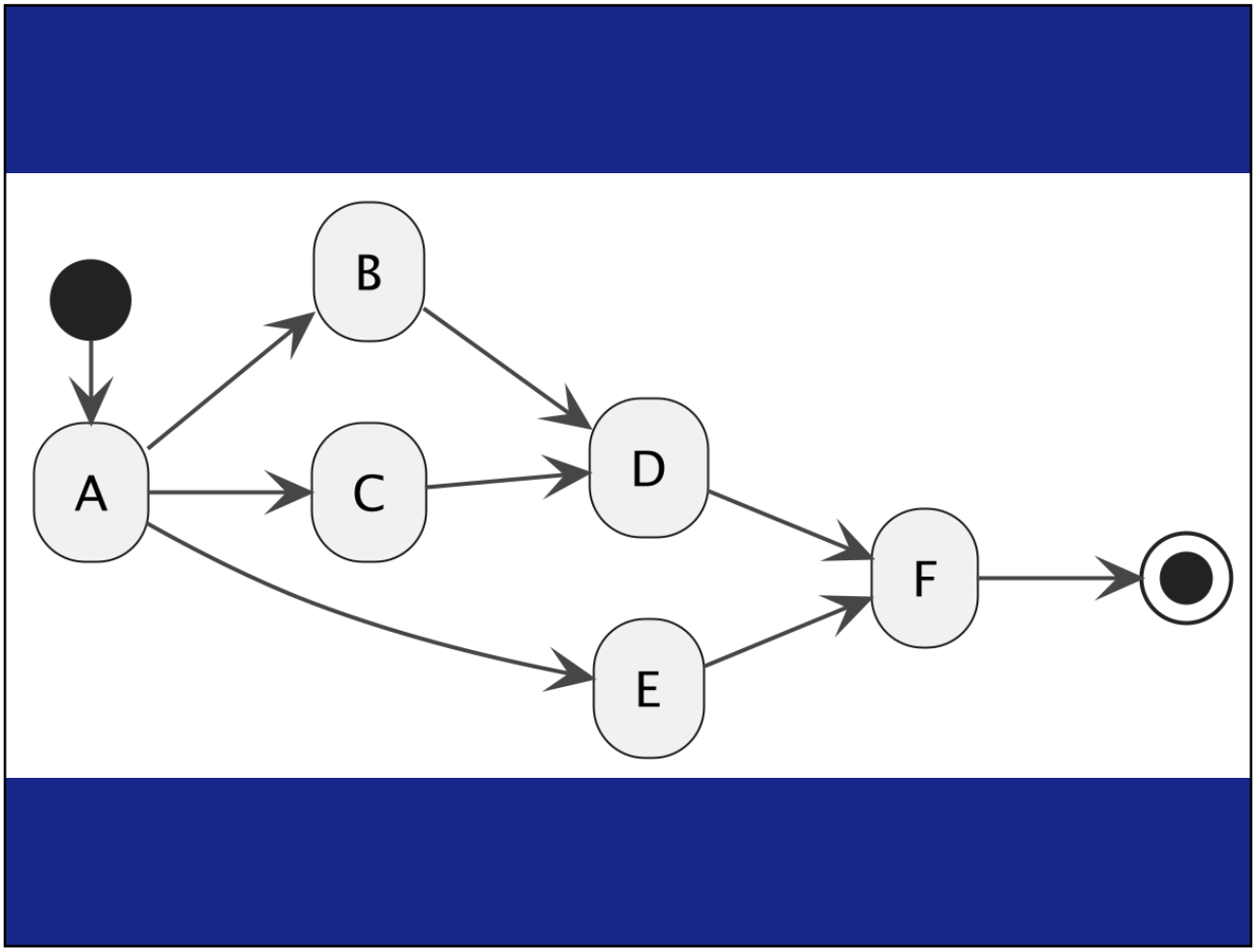
It wouldn't be unusual to have thousands of containers running many compute jobs per second.

Typically talking about tens of thousands to millions of jobs in each run.

The majority of orchestration and compute is run using AWS infrastructure such as step functions and SQS queues. They're not super relevant to this problem!



The bit to focus on for now is lots and lots of Fargate containers talking to S3

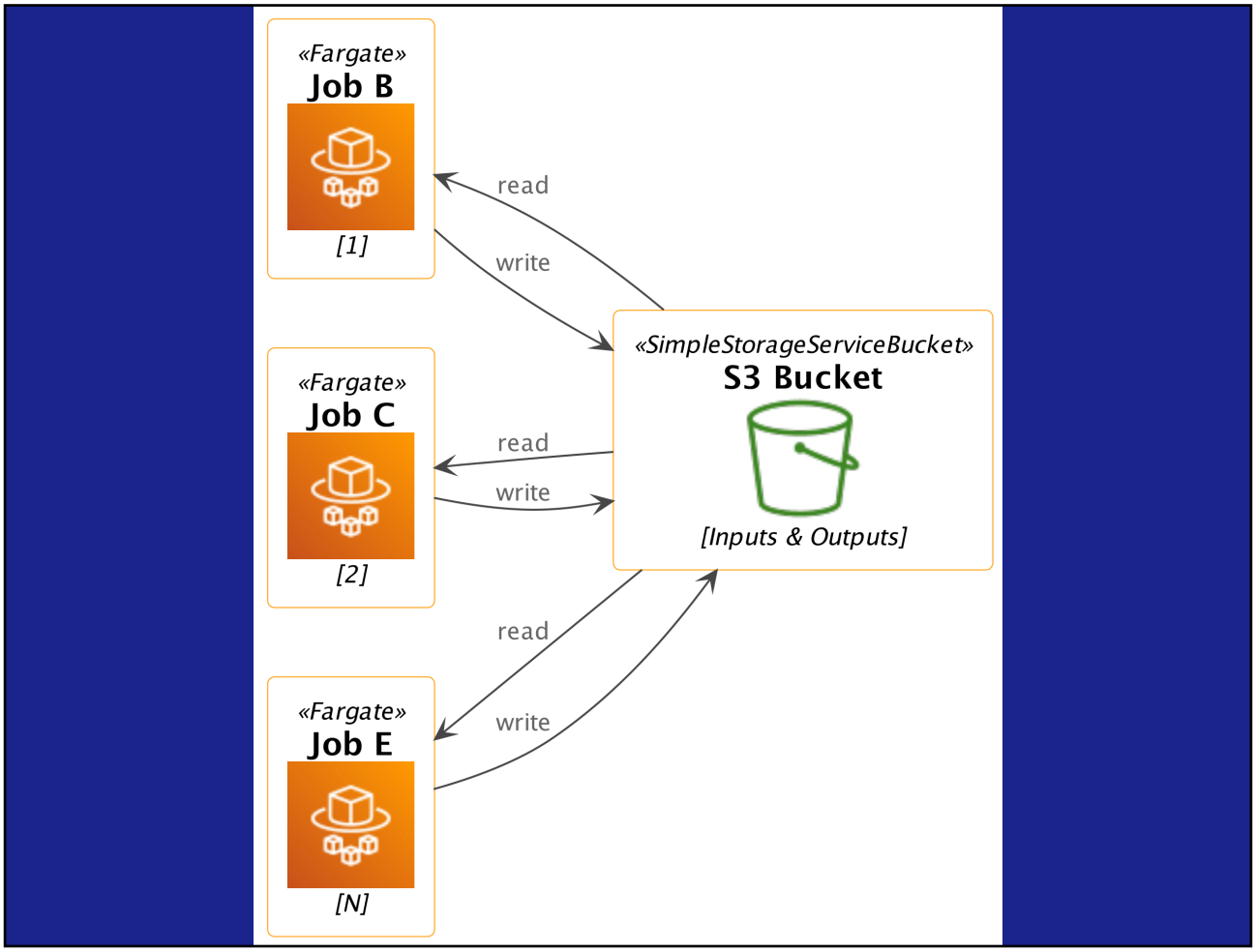


One important aspect of this system is that it runs tasks in order based on a directed acyclic graph, or DAG

This is the same sort of graph you'd see in a makefile or build system.

It's relevant as some jobs will have to wait on others, so any impact on performance can have quite a big knock on effect.

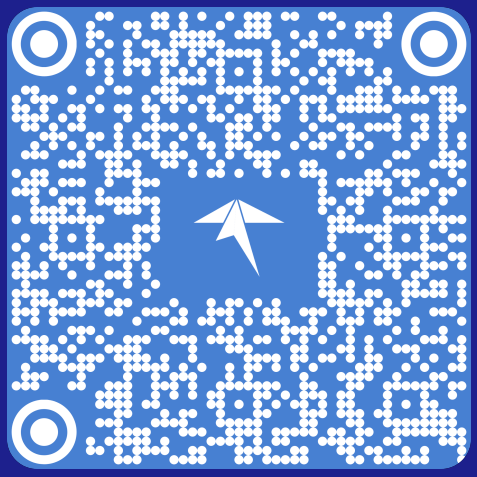
In this example you can't start D until A, B and C are complete.



The interaction between each container and S3 is quite simple, they'll either read data from S3 or write data to S3

"A serverless architecture for high performance financial modelling"

https://aws.amazon.com/blogs/hpc/a-serverless-architecture-for-high-performance-financial-modelling/



For more details check out the AWS HPC blog post "A serverless architecture for high performance financial modelling"

https://aws.amazon.com/blogs/hpc/a-serverless-architecture-for-high-performance-financial-modelling/

```
import boto3

s3 = boto2.client("s3")
for input in stuff_to_process:
  # HTTP GET https://example.s3.eu-west-
1.amazonaws.com/my-input-data
  response = s3.get_object(
    Bucket="example", Key="my-input-data"
  )
  data = do_stuff_with_input(response)
  # HTTP PUT https://example.s3.eu-west-
1.amazonaws.com/my-output-data
  response = s3.put_object(
    Bucket="example", Key="my-output-data", Body=data
  )
  # Fancier: multi-part upload with threaded transfer
manager
  s3.upload_fileobj(data, 'example', 'my-output-data')
```

What does a compute job look like?

Basically every worker does some combination of this

Read data (sometimes lots of S3 keys), process, then write output to S3 (sometimes lots of S3 keys).

For larger objects we use the transfer manager which splits the object into lots of smaller chunks and uploads or downloads in parallel.

The Problem: We got one of the dreaded questions

We got a question.

One of the questions you dread when operating a system

"Why is my compute job so slow?"

During very large job runs we would occasionally see inexplicable slow downs and sometimes outright failures

We did the usual and checked our dashboards and alarms, nothing stood out.

A clue in the logs

```
Traceback (most recent call last):
  File "...examples/s3_get_object.py", line 12, in
<module>
    response = s3.get_object(Bucket=bucket, Key=key)
               ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "...botocore/client.py", line 535, in _api_call
    return self._make_api_call(operation_name, kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "...botocore/client.py", line 980, in
_make_api_call
    raise error_class(parsed_response, operation_name)
botocore.exceptions.ClientError: An error occurred
(429) when calling the GetObject operation (reached max
retries: 4): Too Many Requests
```

Note that frequently we'd have a slow run but no errors.

Finally we saw the occassional error which pointed us in the right direction.

"Are we triggering a lot of S3 request retries?"

The Rate Limit Errors prompted the question:

This could also equally apply to Kinesis or SQS or other APIs

Request retries?

Rate limits?

Isn't the cloud as infinite as your wallet?

S3 PUT object

3,500 requests per second

AWS has rate limits on their APIs (which is very sensible!)

In fact they have many carefully thought out quotas and limits which are a combination of system limitations and economic models.

S3 PUT object might have a rate limit of 3,500 requests per second

When you hit this you might get back a `HTTP 429` or `HTTP 503`

https://docs.aws.amazon.com/AmazonS3/latest/userguide/optimizing-performance.html

boto3 tries to be helpful

```
import boto3

s3 = boto2.client("s3")
# Implicitly retries, blocking:
response = s3.get_object(...)
```

boto3 attempts to handle this invisibly via retries[1] to minimize impact on your application

https://boto3.amazonaws.com/v1/documentation/api/latest/guide/retries.html#standard-retry-mode

Digression: boto3 Retry Mechanism

1. For a known set of errors catch them
2. Keep count of the number of times we've tried
3. If we've hit a maximum retry count fail and allow the error to bubble up
4. Otherwise take the count and multiply by some random number and some scale factor
5. Sleep for that long
6. Retry the call

boto3's default retry handler[1] implements the classic "retry with jitter" approach[2]:

```
# From
https://github.com/boto/botocore/blob/develop/botocore/retryhandler.py

base * (growth_factor ** (attempts - 1))

base = random.random()  # random float between 0.0 and 1.0
growth_factor = 2
attempts = 2

random.random() * (growth_factor ** (attempts - 1))
0.75 * (2 ** (2 - 1)) = 1.5

attempt 1 = 1 second max
attempt 2 = 2 second max
attempt 3 = 8 second max
attempt 4 = 16 second max
attempt 5 = 32 second max

# Default of 5 retries
32 + 16 + 8 + 2 + 1 = 59 seconds max sleep total, with 5x requests
```

Retry sleep formula

Site note: note how it carefully adds up to less than 60 seconds. This is the default request timeout in many clients, servers and proxies.

Theory

> Even though we mostly eventually succeed, we take a really long time doing stuff

Where we are in the problem

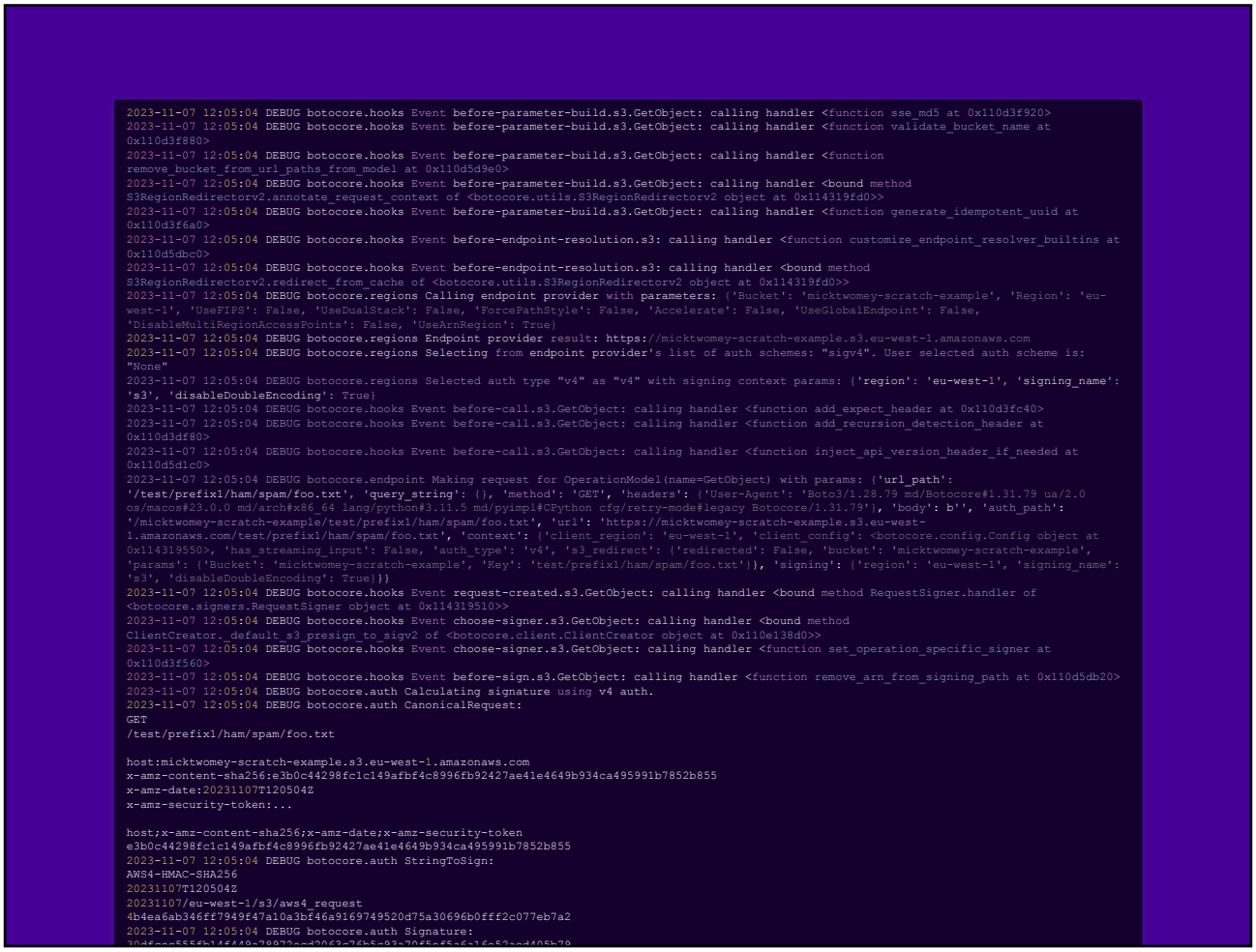
1. Got some jobs taking a long time
2. Guessed it's retries causing this
3. ??

How Can We Be Sure it's Retrying?

Logging?

```
logging.basicConfig(level=logging.DEBUG)
```

Could use logging at DEBUG level



DEBUG logging in boto3 hugely verbose, unusuble in most situations.

Particularly unsusable for us, this would likely generate a TiB of logs!

If only there was some kind of hook or event we could use...

Events?

Events[1] are an extension mechanism for boto3

```python
event_hooks = []

def event_emitting_call(arg):
    # ... do some work
    for hook in event_hooks:
      hook(("event", arg))

def register(callable):
  event_hooks.append(callable)

register(print)
event_emitting_call("hello")
# prints ("event", "hello")
```

This is an example of how you could implement events in python.

The core idea is you have hooks in your library code which can call handlers with some data. In some cases the hooks could modify data too.

As a user you can register your own functions to react to these events.

More advanced use allows you to modify data from the hook. Handy for things like authentication.

```
s3 = boto3.client("s3")
s3.meta.events.register(
  "needs-retry.*", my_function
)
```

```
"provide-client-
params.s3.ListObjects"
"provide-client-params.s3.*"
"provide-client-params.*"
"*"
```

boto3 Events

You register a function to be called when an event matching a pattern happens.

Wildcards (*) are also allowed for patterns.

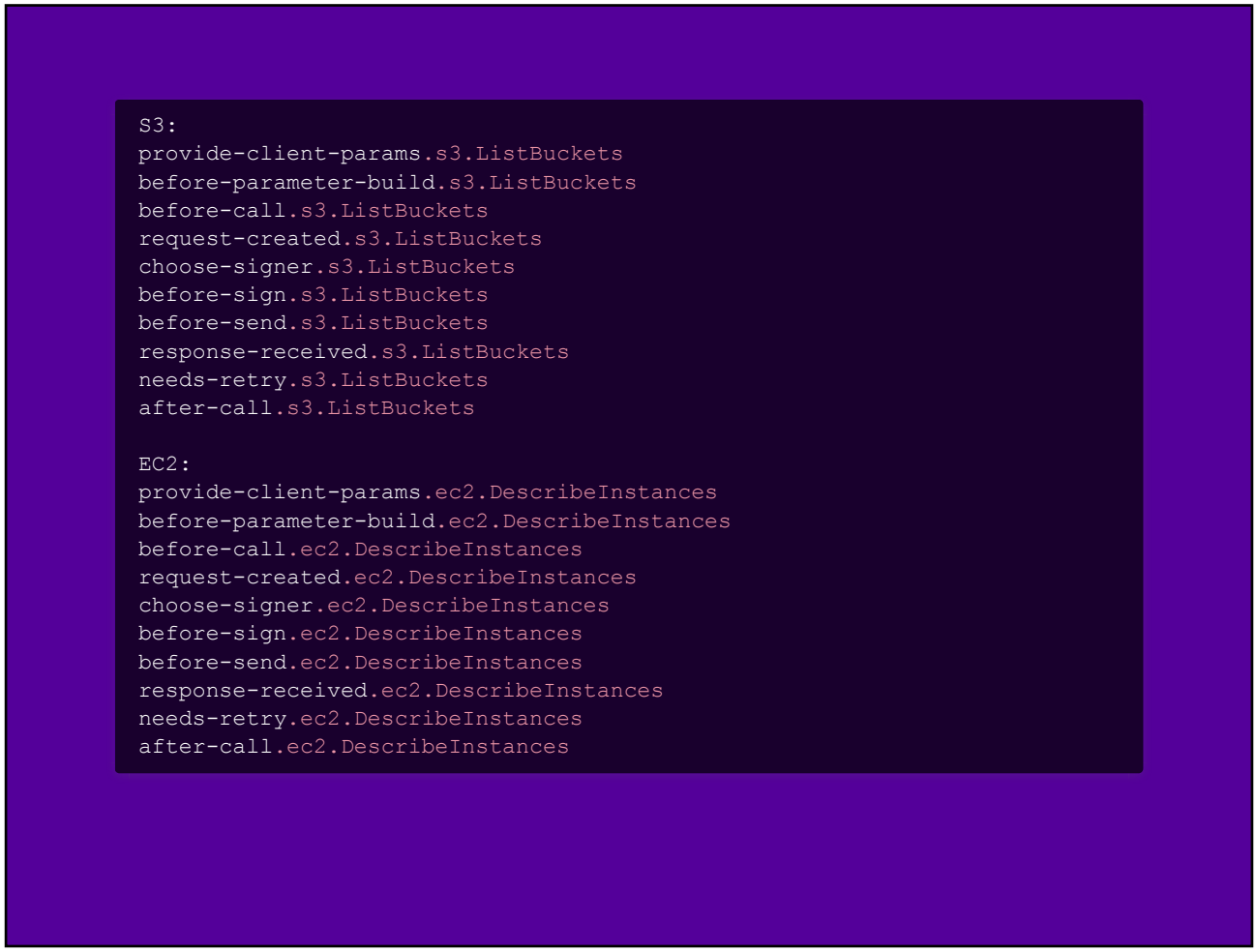
However I don't believe you can glob anywhere except at the end.

Annoyance: even though you specify s3 in the event name you still have to use a s3 specific event register call.

Vice versa, why have s3 in the name if using a specific call.

```python
import boto3

def print_event(event_name, **kwargs):
    print(event_name)

print("\nS3:")
s3 = boto3.client("s3")
s3.meta.events.register("*", print_event)
s3.list_buckets()

print("\nEC2:")
ec2 = boto3.client("ec2")
ec2.meta.events.register("*", print_event)
ec2.describe_instances()
```

Here you can see the code to print out all events received by s3 list_buckets and ec2 describe_instances



Here you can see all the events which get triggered

```python
import boto3
from rich import print

def print_event(event_name,
**kwargs):
    print(event_name, kwargs)


s3 = boto3.client("s3")
s3.meta.events.register("*",
print_event)
```

There's more than just the event name, there are a bunch of params being passed in. However we've no idea what they are.

Python can capture keyword arguments (`foo="bar"`) into a variable using `**myvar`.

This is way more useful, now you are getting arguments too!

```
provide-client-params.s3.ListBuckets
{
    'params': {},
    'model':
OperationModel(name=ListBuckets),
    'context': {
        'client_region': 'eu-west-1',
        'client_config':
<botocore.config.Config>,
        'has_streaming_input': False,
        'auth_type': None
    }
}
```

```
request-created.s3.ListBuckets
{
    'request':
<botocore.awsrequest.AWSRequest>,
    'operation_name': 'ListBuckets'
}

# example of how your hook is called
my_hook(
  "request-created.s3.ListBuckets",
  request=...,
  operation_name=...
)
```

```
needs-retry.s3.ListBuckets
{
    'response': (
        <botocore.awsrequest.AWSResponse>,
        {
            'ResponseMetadata': {
                'RequestId': 'QZV9EWHJMR4T8VQ9',
    ...
    'endpoint': s3(https://s3.eu-west-1.amazonaws.com),
    'operation': OperationModel(name=ListBuckets),
    'attempts': 1,
    'caught_exception': None,
    'request_dict': {
        'url_path': '/',
        'query_string': '',
        'method': 'GET',
    ...
}
```

This one looks relevant

```
needs-retry.s3.ListBuckets
...
'attempts': 1,
'caught_exception': None,
...

# example of how your hook is called
my_hook(
  "needs-retry.s3.ListBuckets",
  attempts=1,
  caught_exception=None,
  ...
)
```

This looks really promising. Doest that attempts count go up?

Is this documented anywhere? Nope!

We have an event to watch but we don't know what failure looks like

No Documentation of needs-retry.s3.ListBuckets Event 😭

Where we are

1. Got some jobs taking a long time
2. Guessed it's retries causing this
3. Want to use events to monitor these but don't have docs
4. ??

How do we figure out what the event looks like for real?

This is time consuming and really expensive. We could run up a bill of thousands doing this.

We could add debugging or modify the library (botocore or boto3)

This requires building a mental model of the library

You've no guarantee your changes reflect reality

Sometimes printing something accidentally consumes a resource and changes the code's behaviour!

Also, we can't debug in the production system.

We could fiddle with the python implementation or libraries.

This could give us some fairly low level info

Could be a bunch of work though

We could use something like gdb or ebpf to debug this

Starting to get into very low level debugging though!

Semi realistic, the code will behave as it would if this really happened.

Can treat all the code as a black box and change the only point of interaction with AWS you have: the network.

This is the one I chose.

HTTP: The Bits We Care About

Client Request 💻 ➡️ ☁️

```
GET /cat.jpeg HTTP/1.1
Host: example.s3.eu-west-
2.amazonaws.com
...
```

Server Response 💻 ⬅️ 🐱

```
HTTP/1.1 200 OK  👈🧐
Content-Type: image/jpeg
...
```

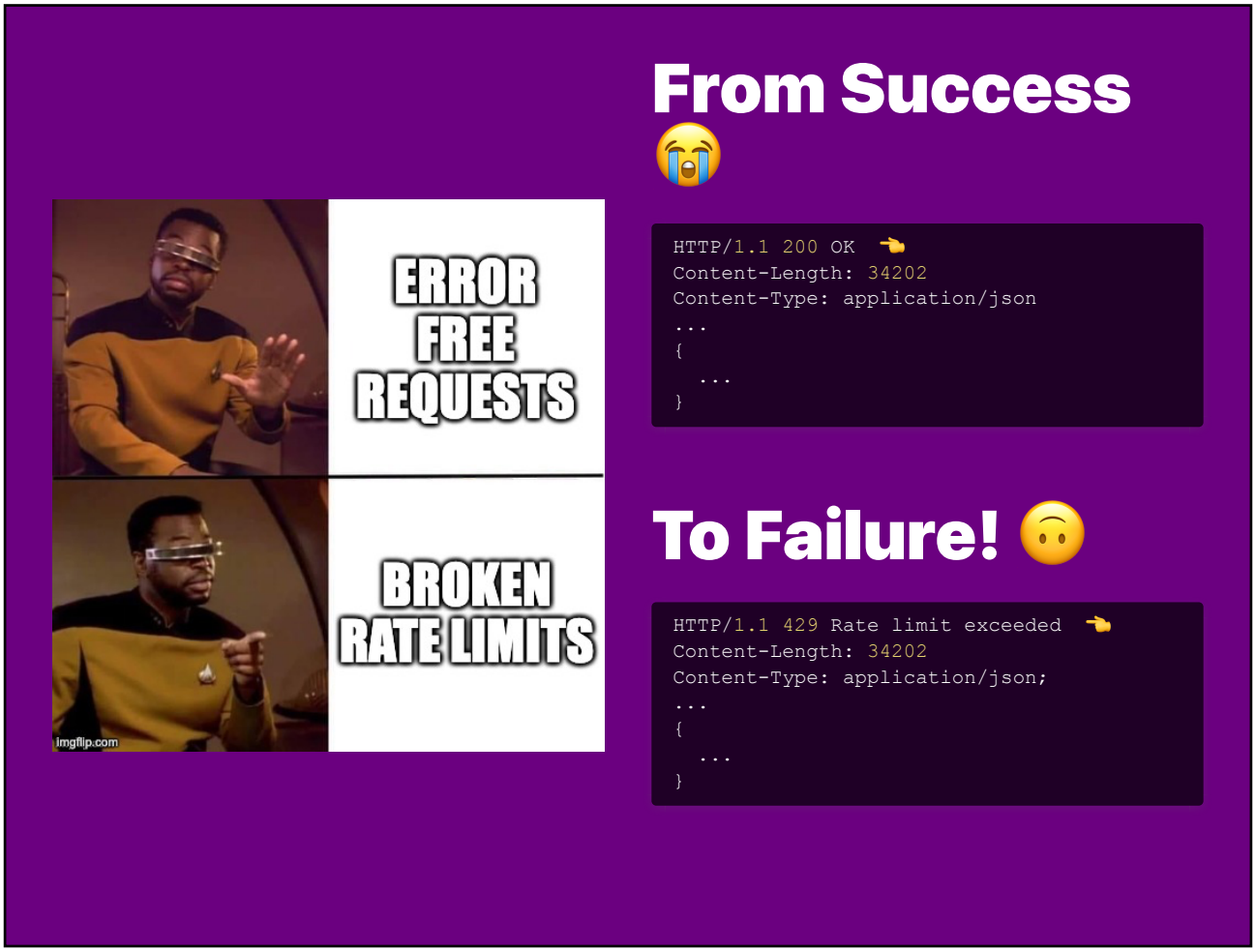
Hacking HTTP

Bet: boto3 looks at the HTTP response code

Bet: AWS uses "429 Too Many Requests" for rate limits

Bet: boto3 ignores everything except that 429

- For a rate limit I'm betting boto3 looks at the HTTP response code
- I'm also betting it'll be HTTP 429
- I'm also betting the code doesn't care about the payload too much once it's a 429
- Finally I'm betting boto3 doesn't verify a response checksum



We're only changing one part of the response, so should be easy to implement.

Right?

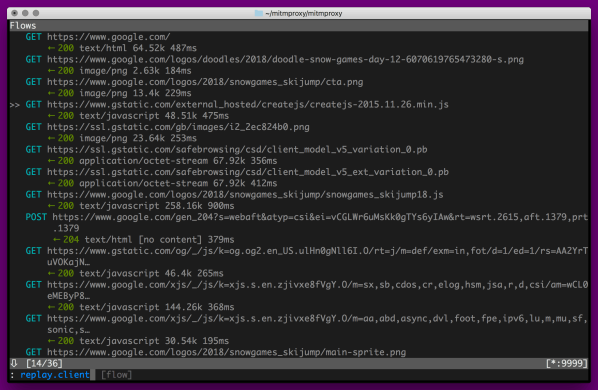
Where We Are

1. Got some jobs taking a long time
2. Guessed it's retries causing this
3. Want to use events to monitor these but don't have docs
4. Want to somehow modify responses to simulate retries and see what happens
5. ??

How: mitmproxy

https://mitmproxy.org

> mitmproxy is a free and open source interactive HTTPS proxy.



What?

Let's you mess with the HTTP requests and responses from programs

Bit like Chrome Dev Tools for all your HTTP speaking commands

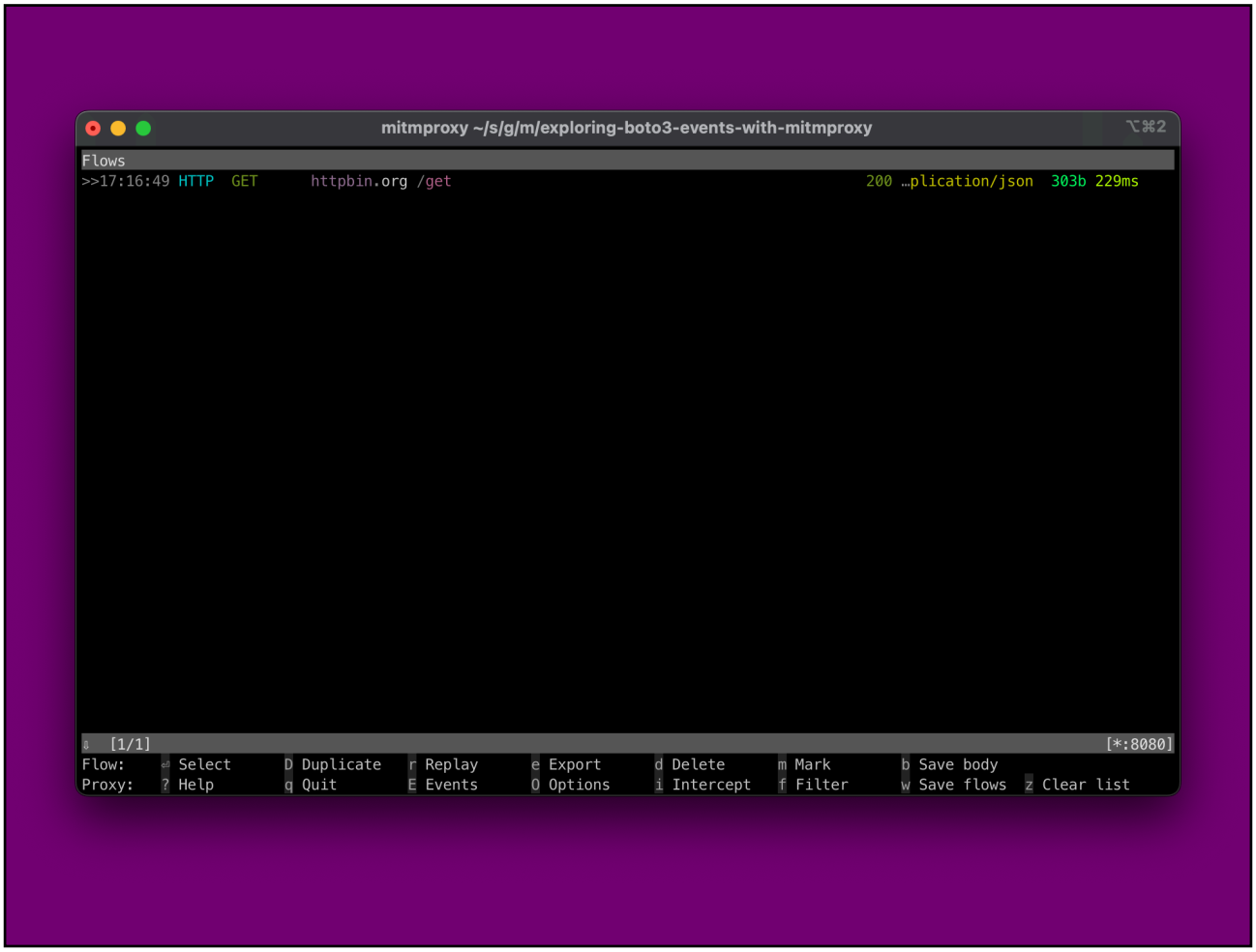
1. Run `mitmproxy` (or `mitmweb` for a fancier web interface)
2. Set the HTTP proxy settings to mitmproxy's (defaults to http://localhost:8080)
3. Run your program
4. Watch in mitmproxy

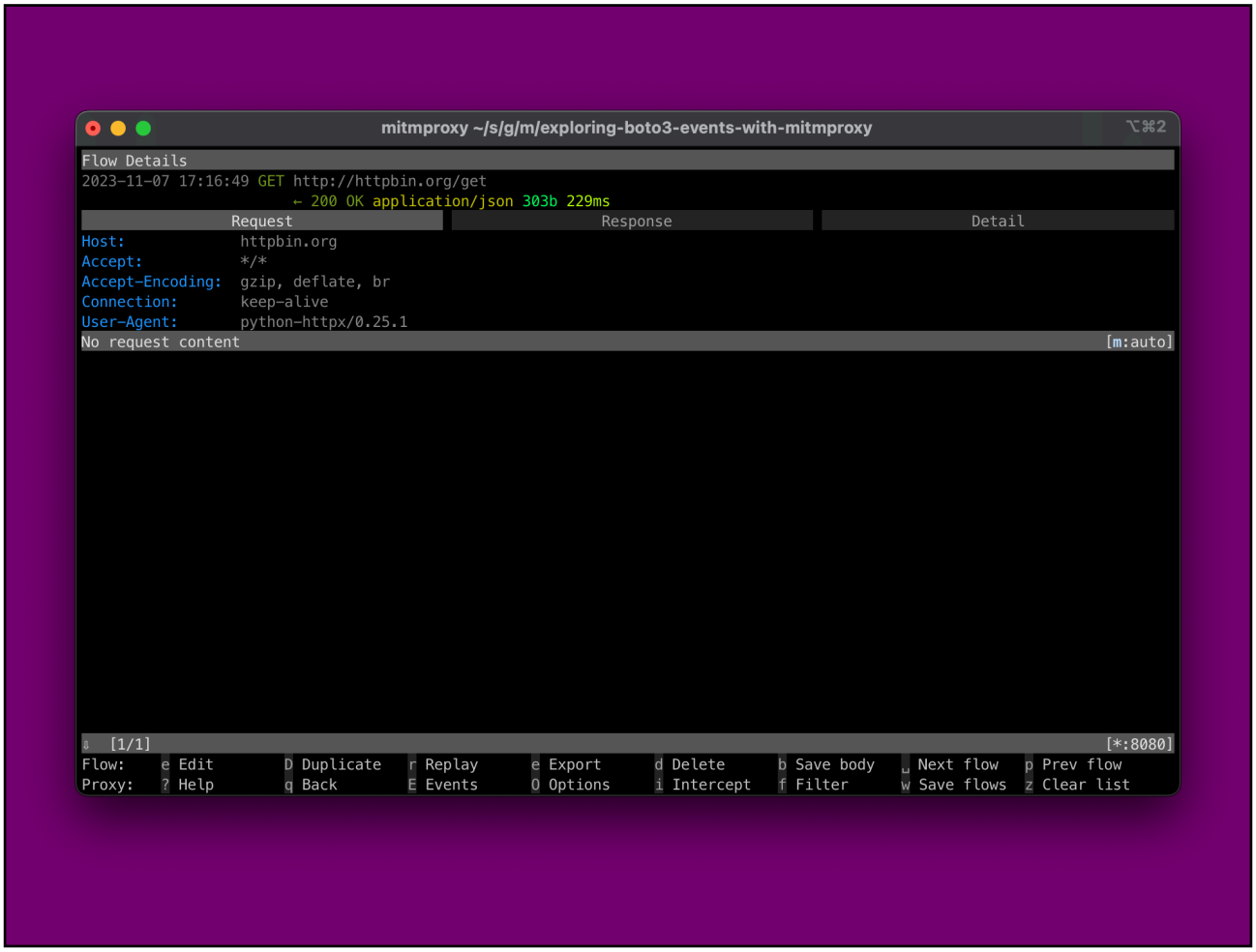
Easy right?

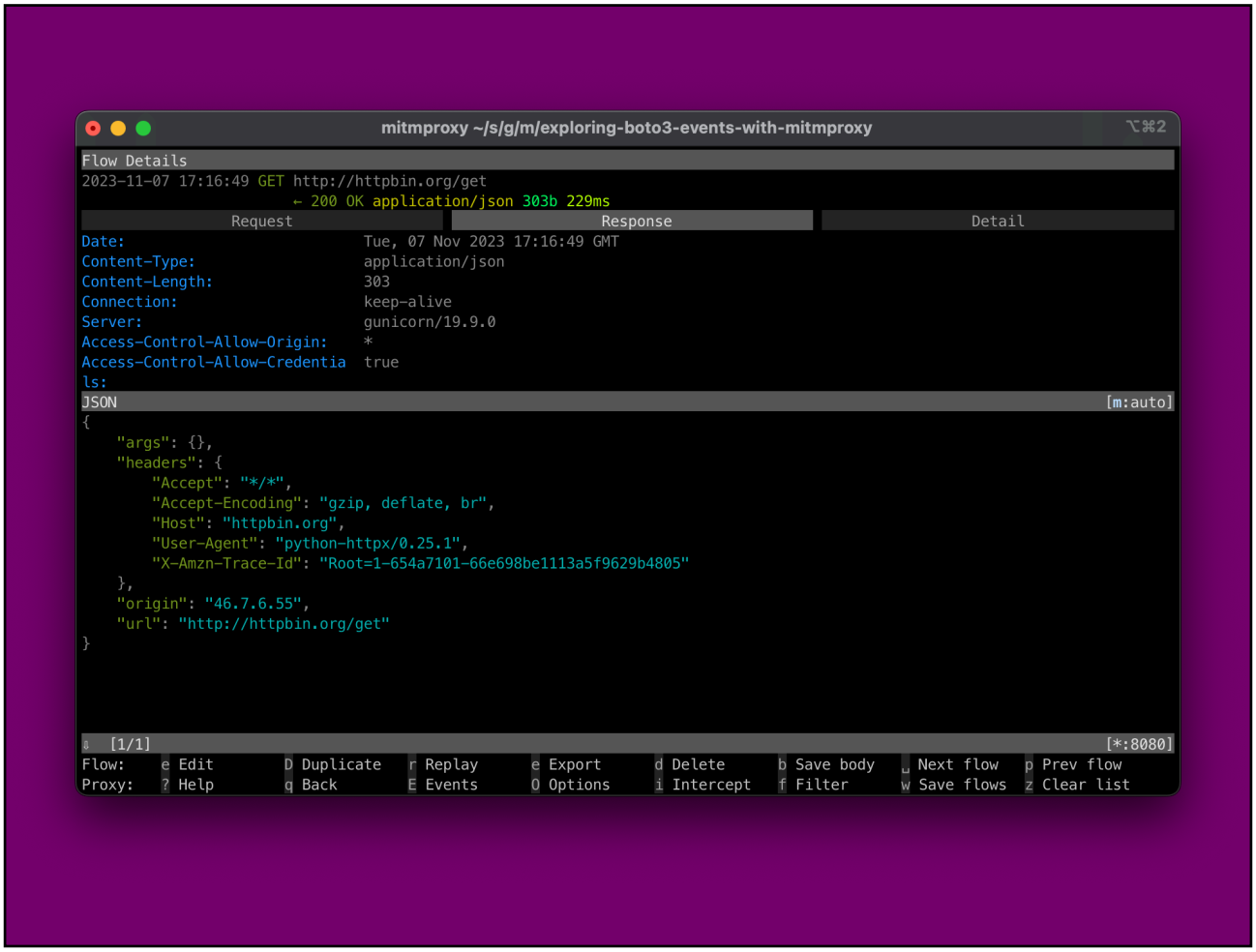
HTTP GET Example

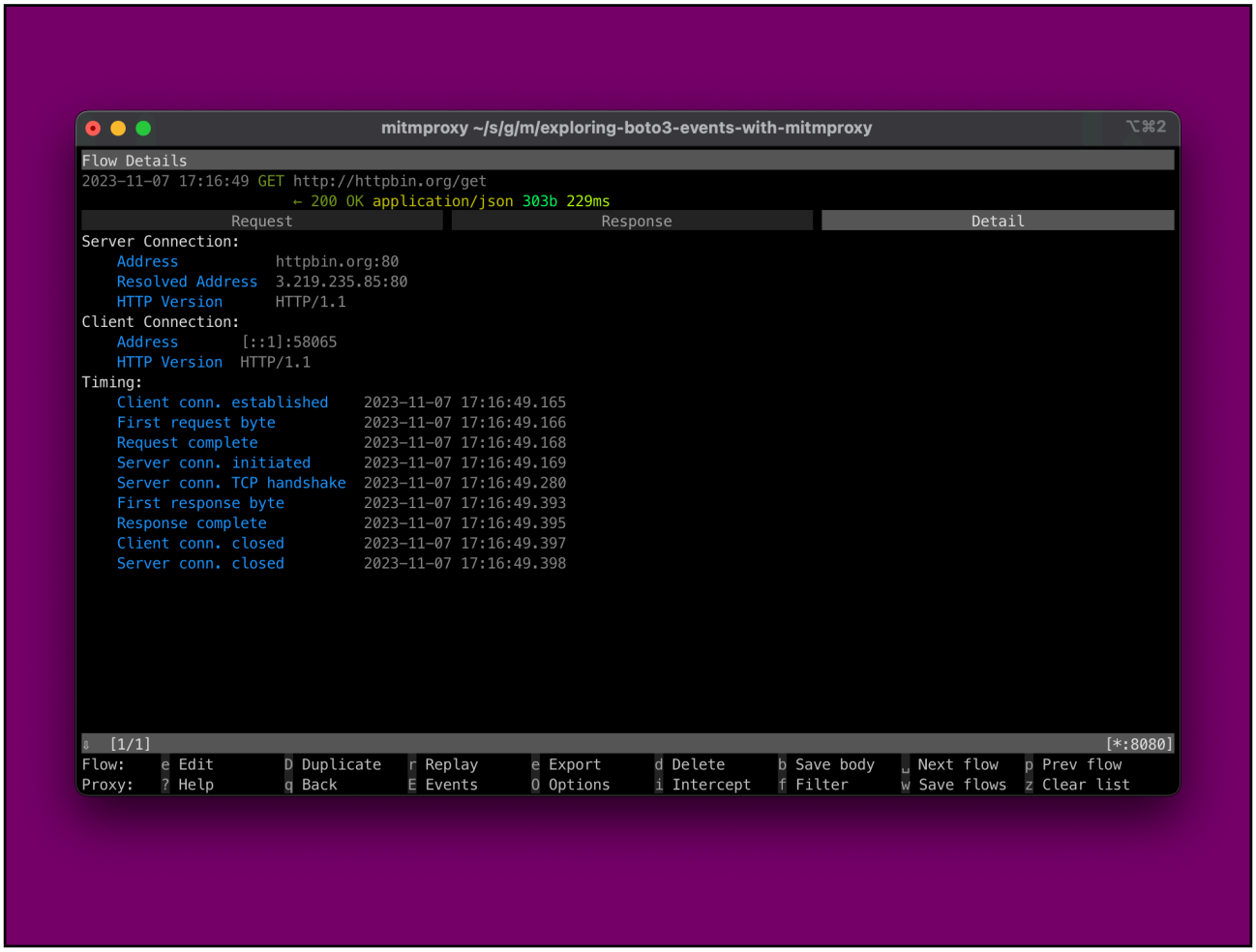
```
import httpx

httpx.get("http://httpbin.org/get")
```

```
export http_proxy=localhost:8080
export https_proxy=localhost:8080

python http_get.py
```


mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
⌥⌘2
Flows
>>17:16:49 HTTP GET httpbin.org /get 200 …plication/json 303b 229ms
[1/1]
[*:8080]
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 z Clear list


mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
⌥⌘2
Flow Details
2023-11-07 17:16:49 GET http://httpbin.org/get
← 200 OK application/json 303b 229ms
Request
Response
Detail
Host: httpbin.org
Accept: */*
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
User-Agent: python-httpx/0.25.1
No request content
[m:auto]
[1/1]
[*:8080]
Flow: e Edit D Duplicate r Replay e Export d Delete b Save body ␣ Next flow p Prev flow
Proxy: ? Help q Back E Events O Options i Intercept f Filter w Save flows z Clear list


mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
⌥⌘2
Flow Details
2023-11-07 17:16:49 GET http://httpbin.org/get
← 200 OK application/json 303b 229ms
Request
Response
Detail
Date: Tue, 07 Nov 2023 17:16:49 GMT
Content-Type: application/json
Content-Length: 303
Connection: keep-alive
Server: gunicorn/19.9.0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
JSON
[m:auto]
{
"args": {},
"headers": {
"Accept": "*/*",
"Accept-Encoding": "gzip, deflate, br",
"Host": "httpbin.org",
"User-Agent": "python-httpx/0.25.1",
"X-Amzn-Trace-Id": "Root=1-654a7101-66e698be1113a5f9629b4805"
},
"origin": "46.7.6.55",
"url": "http://httpbin.org/get"
}
[1/1]
[*:8080]
Flow: e Edit D Duplicate r Replay e Export d Delete b Save body ␣ Next flow p Prev flow
Proxy: ? Help q Back E Events O Options i Intercept f Filter w Save flows z Clear list


mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
⌥⌘2
Flow Details
2023-11-07 17:16:49 GET http://httpbin.org/get
← 200 OK application/json 303b 229ms
Request
Response
Detail
Server Connection:
Address httpbin.org:80
Resolved Address 3.219.235.85:80
HTTP Version HTTP/1.1
Client Connection:
Address [::1]:58065
HTTP Version HTTP/1.1
Timing:
Client conn. established 2023-11-07 17:16:49.165
First request byte 2023-11-07 17:16:49.166
Request complete 2023-11-07 17:16:49.168
Server conn. initiated 2023-11-07 17:16:49.169
Server conn. TCP handshake 2023-11-07 17:16:49.280
First response byte 2023-11-07 17:16:49.393
Response complete 2023-11-07 17:16:49.395
Client conn. closed 2023-11-07 17:16:49.397
Server conn. closed 2023-11-07 17:16:49.398
[1/1]
[*:8080]
Flow: e Edit D Duplicate r Replay e Export d Delete b Save body ␣ Next flow p Prev flow
Proxy: ? Help q Back E Events O Options i Intercept f Filter w Save flows z Clear list

S3 Example

```
import boto3

s3 = boto3.client("s3")
s3.get_object(
  Bucket="micktwomey-scratch-example",
  Key="test/prefix1/ham/spam/foo.txt",
)
```

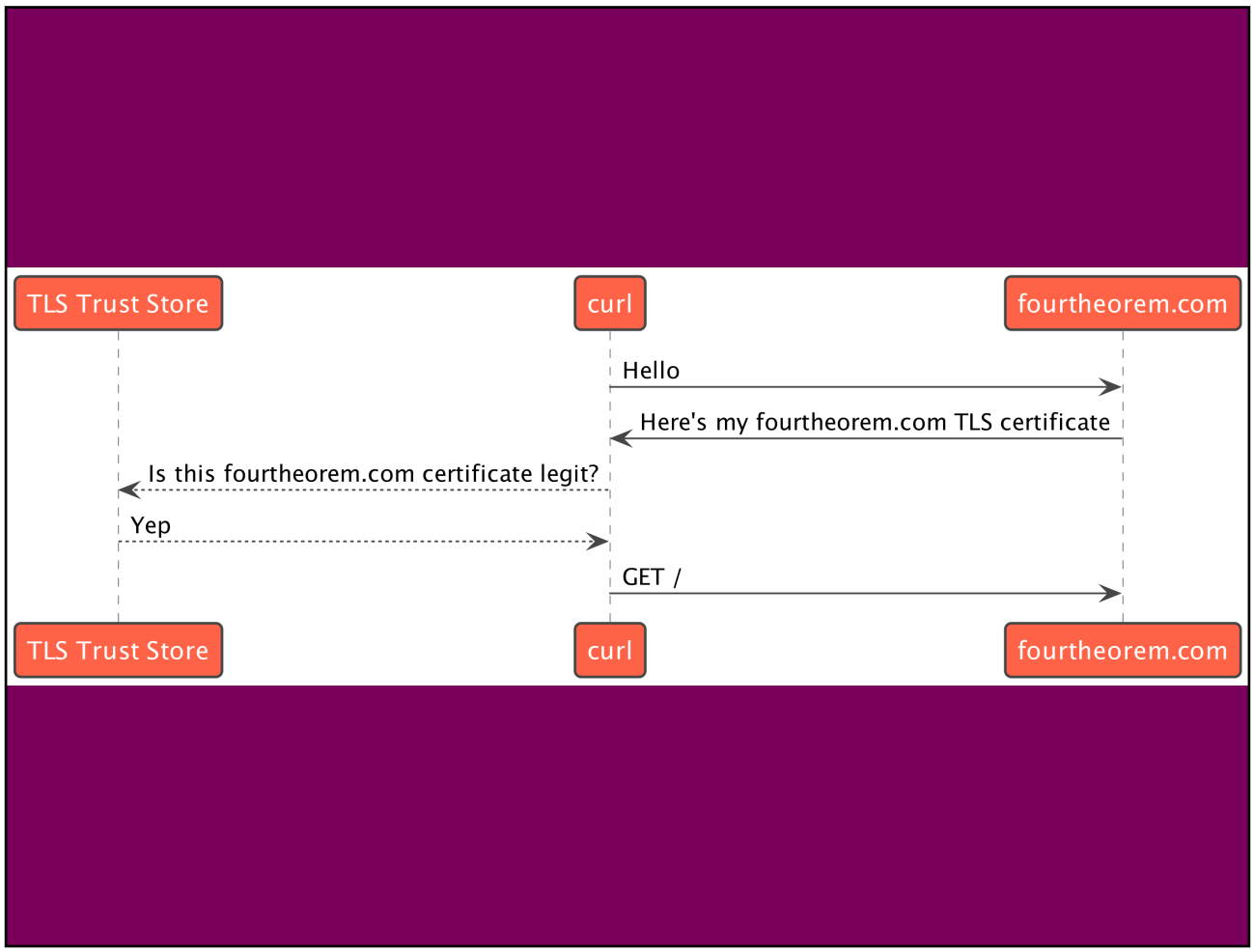
```
export http_proxy=localhost:8080
export https_proxy=localhost:8080

python s3_get_object.py
```

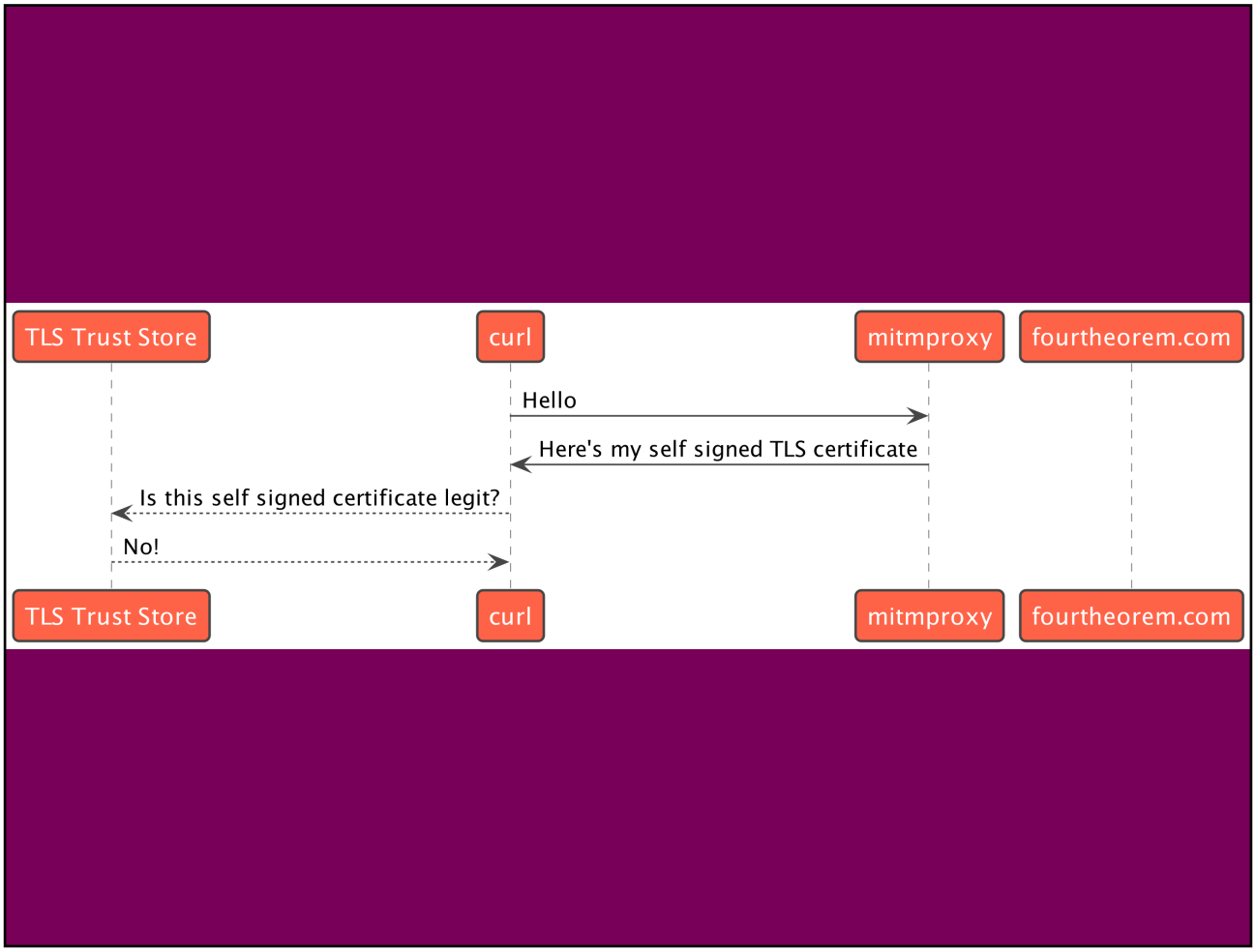
Wut?

```
...
  File "/Users/mick/src/github.com/micktwomey/exploring-
boto3-events-with-mitmproxy/.venv/lib/python3.11/site-
packages/botocore/endpoint.py", line 377, in _send
    return self.http_session.send(request)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/mick/src/github.com/micktwomey/exploring-
boto3-events-with-mitmproxy/.venv/lib/python3.11/site-
packages/botocore/httpsession.py", line 491, in send
    raise SSLError(endpoint_url=request.url, error=e)
botocore.exceptions.SSLError: SSL validation failed for
https://micktwomey-scratch-example.s3.eu-west-
1.amazonaws.com/test/prefix1/ham/spam/foo.txt [SSL:
CERTIFICATE_VERIFY_FAILED] certificate verify failed: unable
to get local issuer certificate (_ssl.c:1006)
```

TLS is doing its job

```
http_proxy=localhost:8080 \
https_proxy=localhost:8080 \
curl https://fourtheorem.com/

curl: (60) SSL certificate problem: unable to get
local issuer certificate
More details here:
https://curl.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server
and therefore could not establish a secure
connection to it. To learn more about this
situation and how to fix it, please visit the web
page mentioned above.
```


TLS Trust Store
curl
fourtheorem.com
Hello
Here's my fourtheorem.com TLS certificate
Is this fourtheorem.com certificate legit?
Yep
GET /
TLS Trust Store
curl
fourtheorem.com



What's happening?

1. We tell curl to connect via mitmproxy to www.fourtheorem.com using HTTPS
2. curl connects to mitmproxy and tries to verify the TLS certificate
3. curl decides the certificate in the proxy isn't to be trusted and rejects the connection

curl (and TLS) is doing its job: preventing someone from injecting themselves into the HTTP connection and intercepting traffic.

A man in the middle attack (or MITM) was prevented!

Unfortunately that's what we want to do!

(ab)using self signed certs for the win!

```
$ ls -l ~/.mitmproxy/
total 48
-rw-r--r--  1 mick  staff  1172 Sep  4 19:26 mitmproxy-ca-
cert.cer
-rw-r--r--  1 mick  staff  1035 Sep  4 19:26 mitmproxy-ca-
cert.p12
-rw-r--r--  1 mick  staff  1172 Sep  4 19:26 mitmproxy-ca-
cert.pem
-rw-------  1 mick  staff  2411 Sep  4 19:26 mitmproxy-
ca.p12
-rw-------  1 mick  staff  2847 Sep  4 19:26 mitmproxy-
ca.pem
-rw-r--r--  1 mick  staff   770 Sep  4 19:26 mitmproxy-
dhparam.pem
```

Luckily mitmproxy generates TLS certificates for you to use:

If you can somehow tell your command to trust these it will talk via mitmproxy!

⚠️ Danger! ⚠️
Here be Dragons! 

Full guide:
https://docs.mitmproxy.org/stable/concepts-certificates/

To work mitmproxy requires clients to trust these certificates

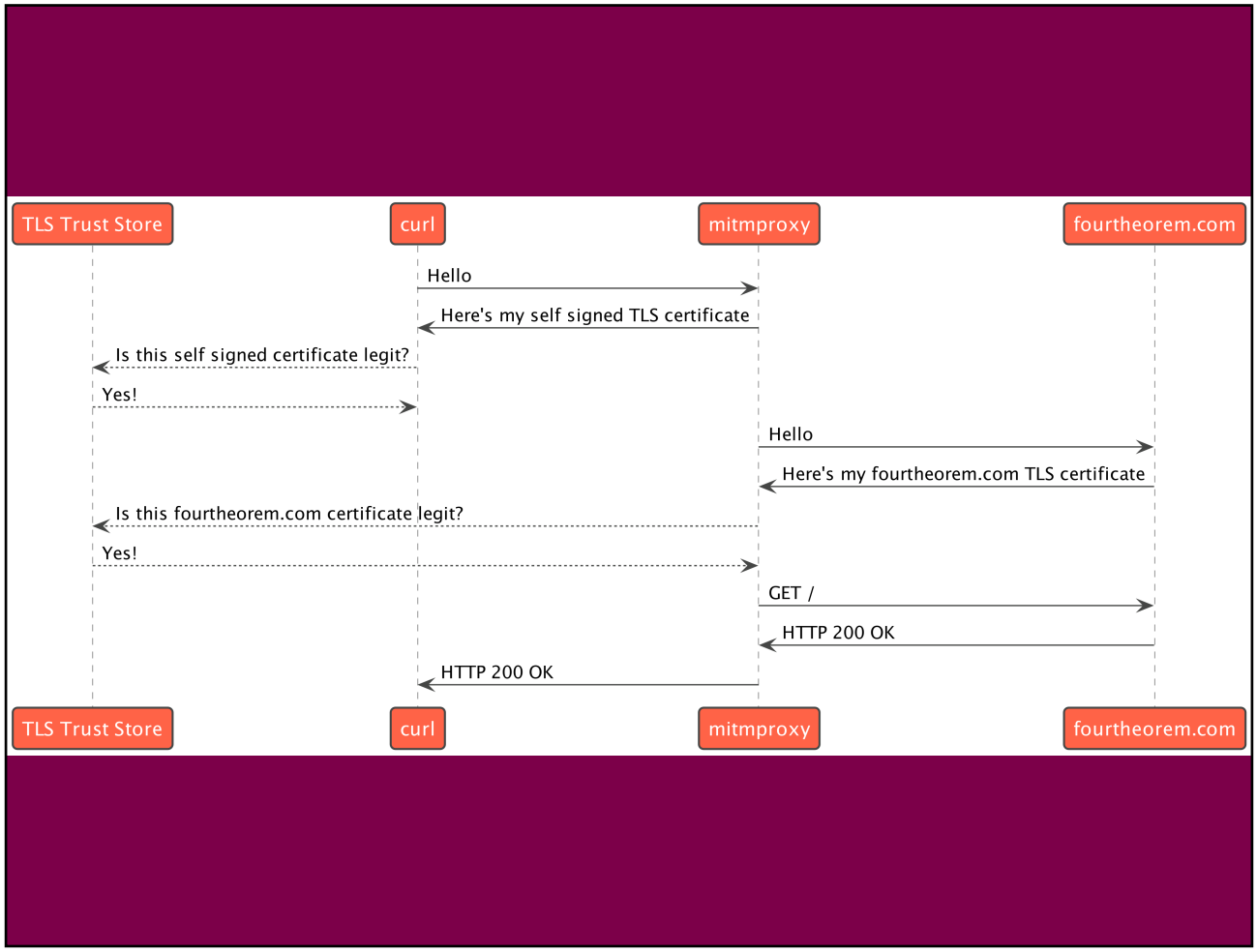
This potentially opens up a massive security hole on your machine depending how this is set up

Recommendation: if possible restrict to one off command line invocations rather than install system wide

Luckily we can override on a per invocation basis in curl and boto3

```
$ https_proxy=localhost:8080 \
curl --cacert ~/.mitmproxy/mitmproxy-
ca-cert.pem \
-I https://www.fourtheorem.com
HTTP/1.1 200 Connection established

HTTP/1.1 200 OK
Server: openresty
Date: Sun, 04 Sep 2022 20:09:03 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 520810
Connection: keep-alive
...
```

curl offers a simple way to trust a cert: `--cacert`


TLS Trust Store
curl
mitmproxy
fourtheorem.com
Hello
Here's my self signed TLS certificate
Is this self signed certificate legit?
Yes!
Hello
Here's my fourtheorem.com TLS certificate
Is this fourtheorem.com certificate legit?
Yes!
GET /
HTTP 200 OK
HTTP 200 OK
TLS Trust Store
curl
mitmproxy
fourtheorem.com

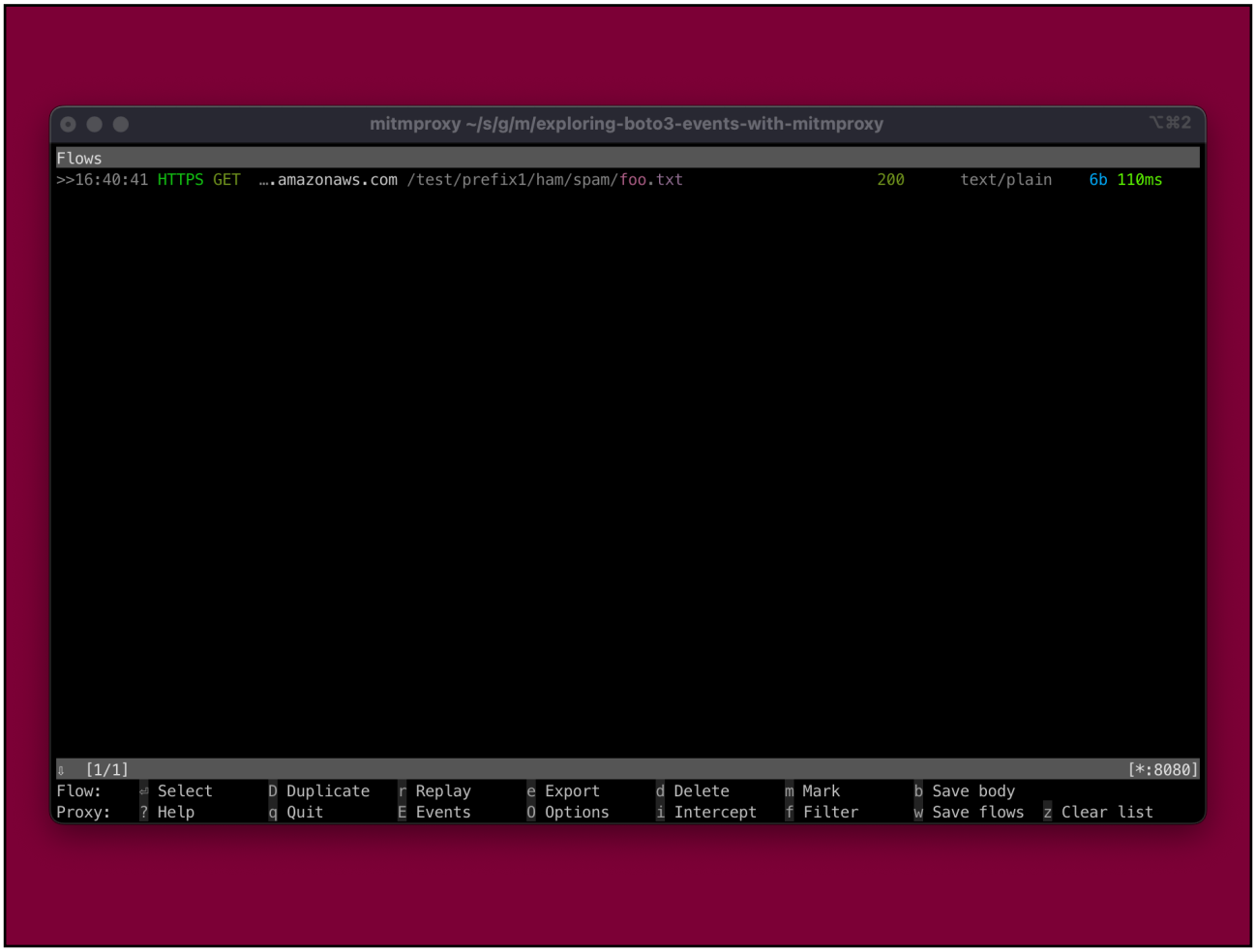
```
* Uses proxy env variable https_proxy == 'localhost:8080'
*   Trying 127.0.0.1:8080...
* Connected to localhost (127.0.0.1) port 8080 (#0)
* allocate connect buffer!
* Establish HTTP proxy tunnel to www.google.ie:443
...
* Proxy replied 200 to CONNECT request
...
* successfully set certificate verify locations:
*  CAfile: /Users/mick/.mitmproxy/mitmproxy-ca-cert.pem
*  CApath: none
* (304) (OUT), TLS handshake, Client hello (1):
* (304) (IN), TLS handshake, Server hello (2):
* (304) (IN), TLS handshake, Unknown (8):
* (304) (IN), TLS handshake, Certificate (11):
* (304) (IN), TLS handshake, CERT verify (15):
* (304) (IN), TLS handshake, Finished (20):
* (304) (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / AEAD-AES256-GCM-SHA384
* ALPN, server accepted to use h2
* Server certificate:
*  subject: CN=*.google.ie
*  start date: Sep 17 11:58:59 2022 GMT
*  expire date: Sep 19 11:58:59 2023 GMT
*  subjectAltName: host "www.google.ie" matched cert's "*.google.ie"
*  issuer: CN=mitmproxy; O=mitmproxy
*  SSL certificate verify ok.
...
```

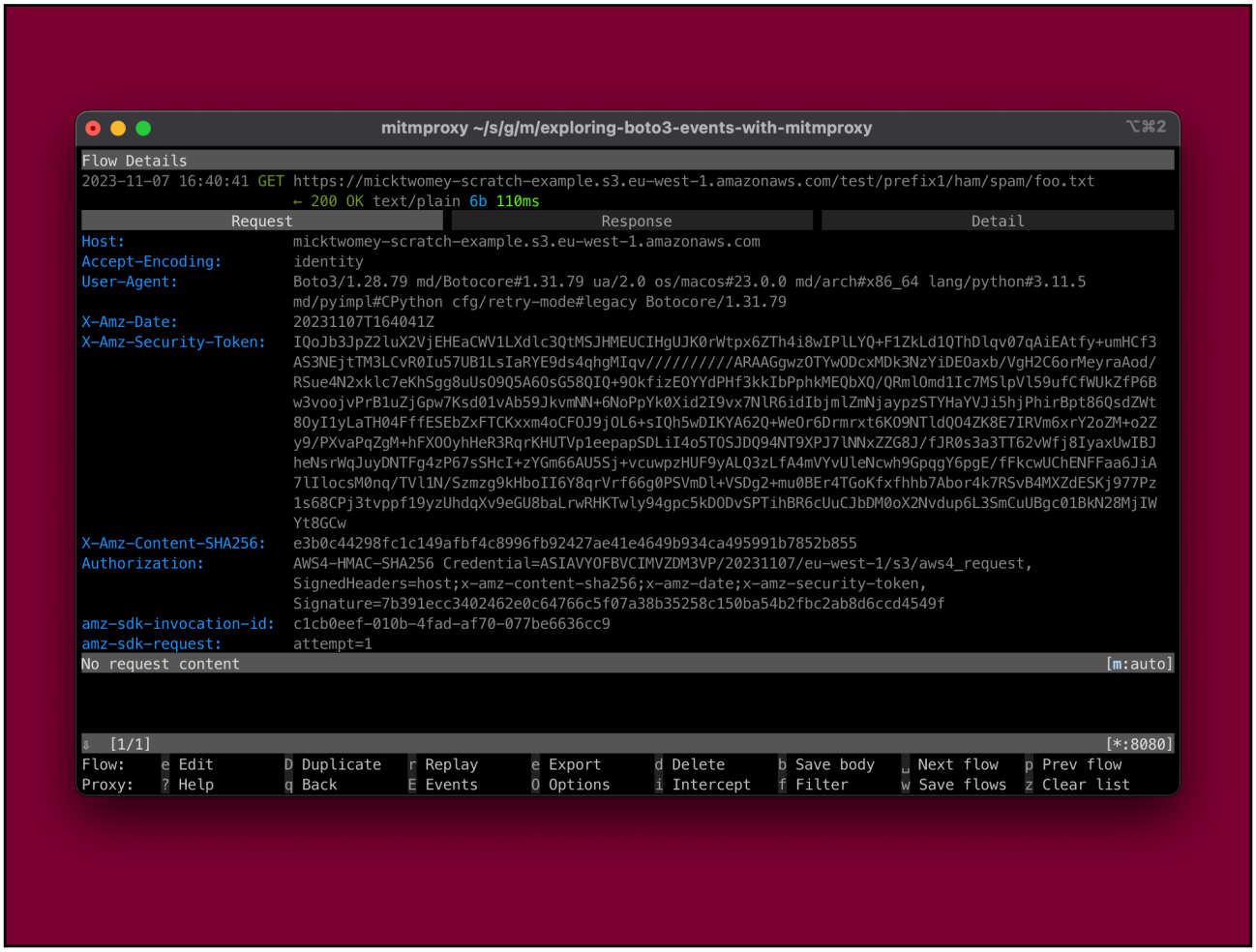
AWS_CA_BUNDLE

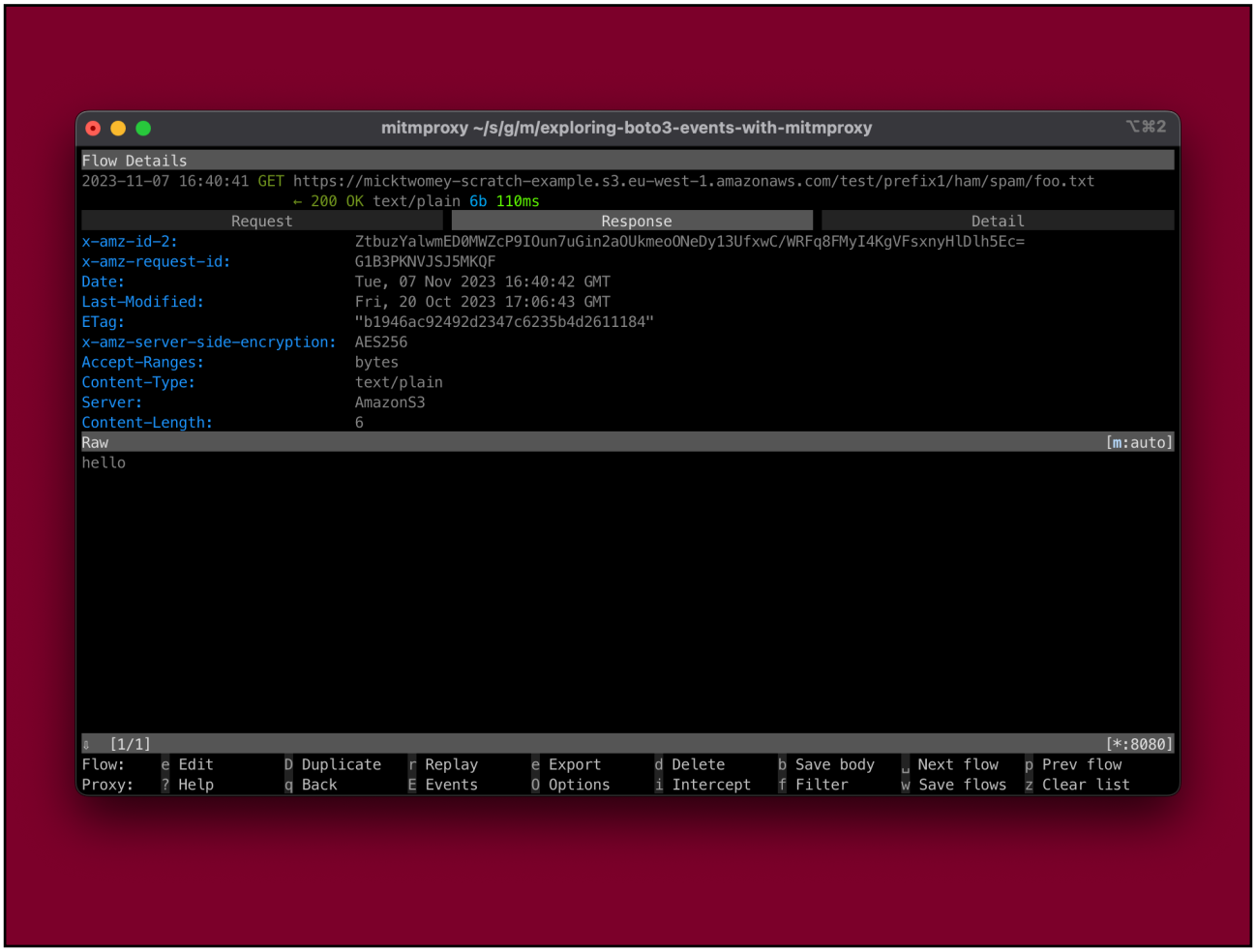
```
https_proxy=localhost:8080 \
AWS_CA_BUNDLE=$HOME/.mitmproxy/mitmproxy-ca-
cert.pem \
python examples/s3_get_object.py
```

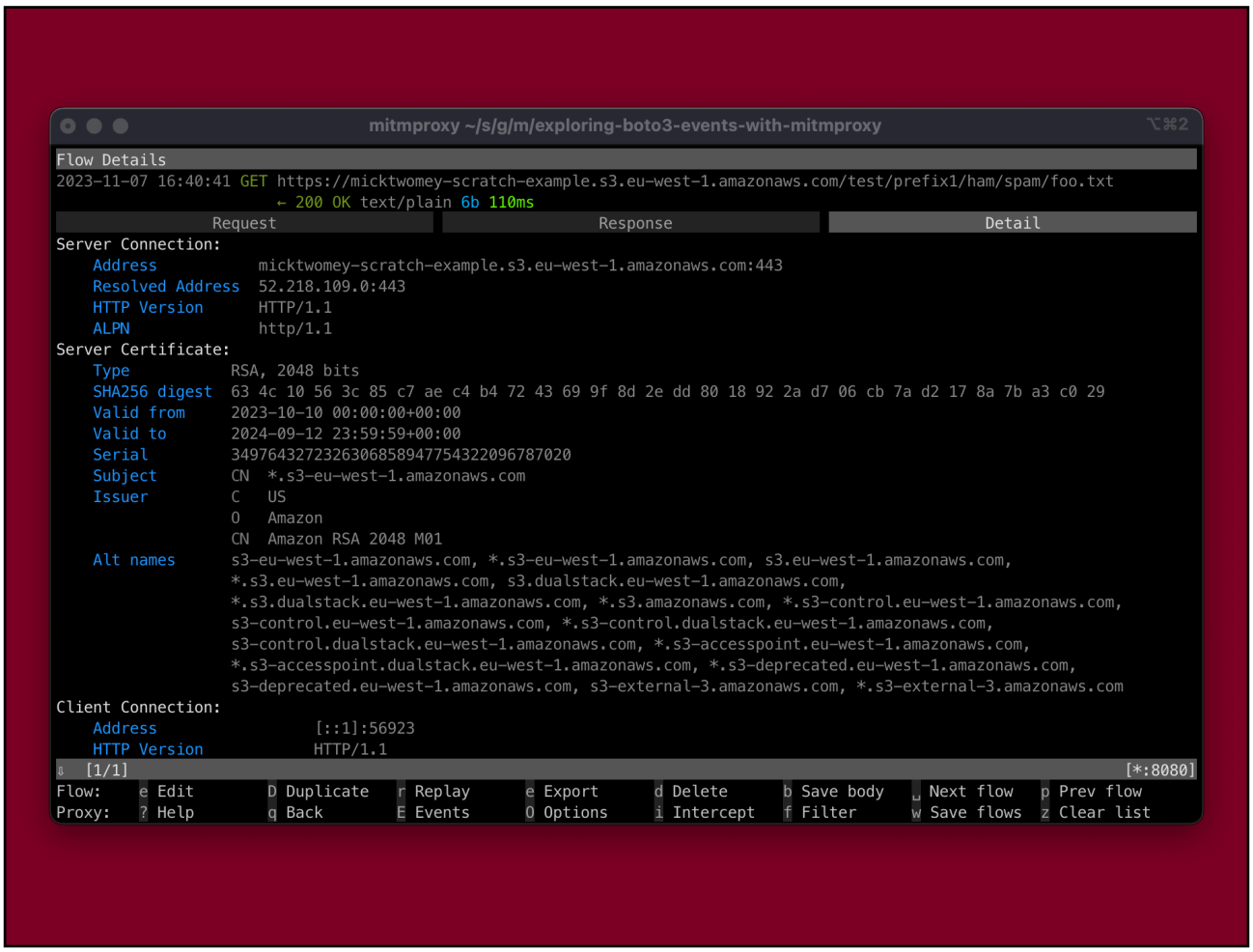
We can do something similar with boto3:

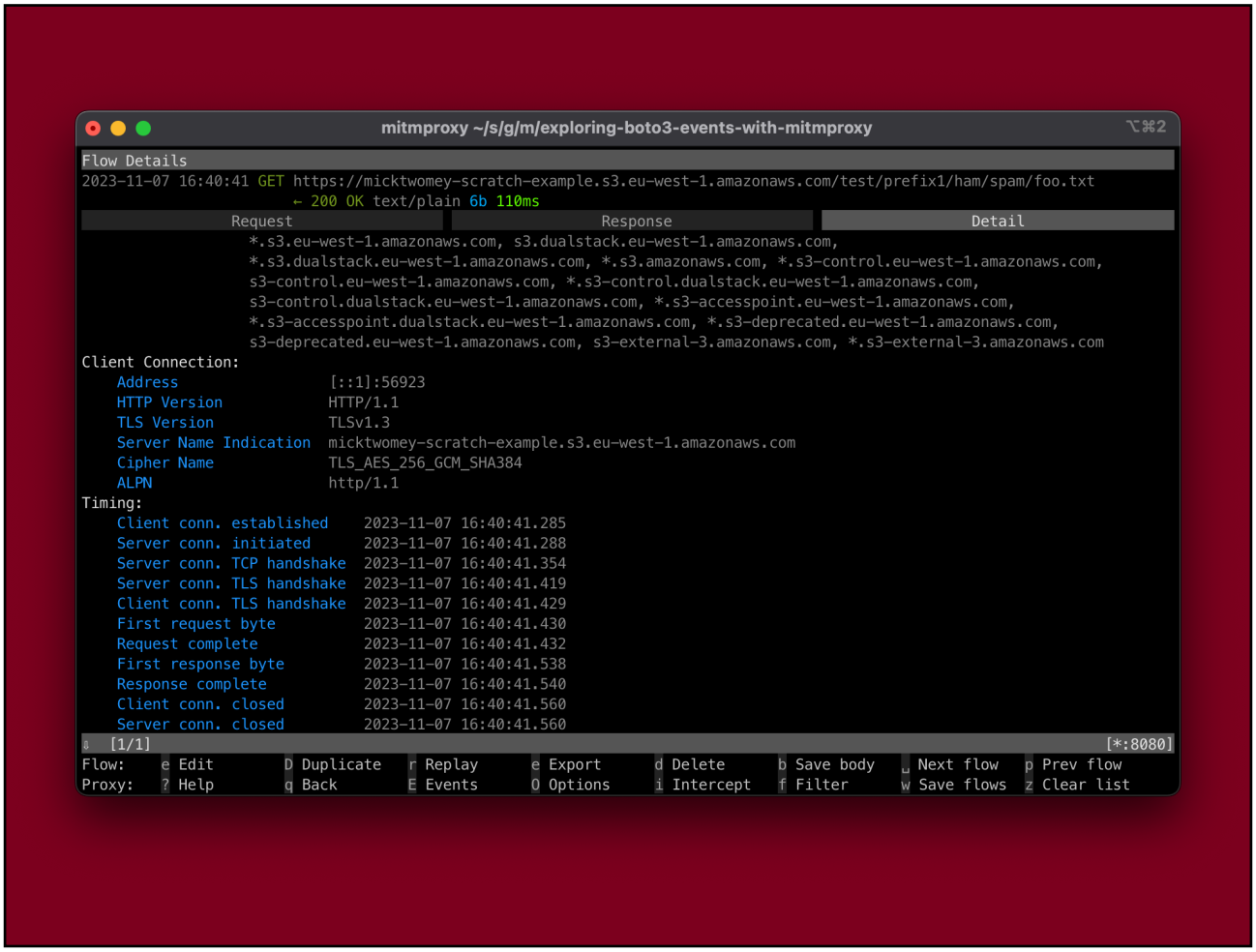
We tell boto3 to use a different cert bundle (`AWS_CA_BUNDLE`)


mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
⌥⌘2
Flows
>>16:40:41 HTTPS GET ….amazonaws.com /test/prefix1/ham/spam/foo.txt 200 text/plain 6b 110ms
[1/1]
[*:8080]
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 z Clear list


mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
⌥⌘2
Flow Details
2023-11-07 16:40:41 GET https://micktwomey-scratch-example.s3.eu-west-1.amazonaws.com/test/prefix1/ham/spam/foo.txt
← 200 OK text/plain 6b 110ms
Request
Response
Detail
Host: micktwomey-scratch-example.s3.eu-west-1.amazonaws.com
Accept-Encoding: identity
User-Agent: Boto3/1.28.79 md/Botocore#1.31.79 ua/2.0 os/macos#23.0.0 md/arch#x86_64 lang/python#3.11.5
md/pyimpl#CPython cfg/retry-mode#legacy Botocore/1.31.79
X-Amz-Date: 20231107T164041Z
X-Amz-Security-Token: IQoJb3JpZ2luX2VjEHEaCWV1LXdlc3QtMSJHMEUCIHgUJK0rWtpx6ZTh4i8wIPlLYQ+F1ZkLd1QThDlqv07qAiEAtfy+umHCf3
AS3NEjtTM3LCvR0Iu57UB1LsIaRYE9ds4qhgMIqv//////////ARAAGgwzOTYwODcxMDk3NzYiDEOaxb/VgH2C6orMeyraAod/
RSue4N2xklc7eKhSgg8uUsO9Q5A6OsG58QIQ+9OkfizEOYYdPHf3kkIbPphkMEQbXQ/QRmlOmd1Ic7MSlpVl59ufCfWUkZfP6B
w3voojvPrB1uZjGpw7Ksd01vAb59JkvmNN+6NoPpYk0Xid2I9vx7NlR6idIbjmlZmNjaypzSTYHaYVJi5hjPhirBpt86QsdZWt
8OyI1yLaTH04FffESEbZxFTCKxxm4oCFOJ9jOL6+sIQh5wDIKYA62Q+WeOr6DrmrxtGKO9NTldQO4ZK8E7IRVm6xrY2oZM+o2Z
y9/PXvaPqZgM+hFXOOyhHeR3RqrKHUTVp1eepapSDLiI4o5TOSJDQ94NT9XPJ7lNNxZZG8J/fJR0s3a3TT62vWfj8IyaxUwIBJ
heNsrWqJuyDNTFg4zP67sSHcI+zYGm66AU5Sj+vcuwpzHUF9yALQ3zLfA4mVYvUleNcwh9GpqgY6pgE/fFkcwUChENFFaa6JiA
7lIlocsM0nq/TVl1N/Szmzg9kHboII6Y8qrVrf66g0PSVmDl+VSDg2+mu0BEr4TGoKfxfhhb7Abor4k7RSvB4MXZdESKj977Pz
1s68CPj3tvppf19yzUhdqXv9eGU8baLrwRHKTwly94gpc5kDODvSPTihBR6cUuCJbDM0oX2Nvdup6L3SmCuUBgc01BkN28MjIW
Yt8GCw
X-Amz-Content-SHA256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
Authorization: AWS4-HMAC-SHA256 Credential=ASIAVYOFBVCIMVZDM3VP/20231107/eu-west-1/s3/aws4_request,
SignedHeaders=host;x-amz-content-sha256;x-amz-date;x-amz-security-token,
Signature=7b391ecc3402462e0c64766c5f07a38b35258c150ba54b2fbc2ab8d6ccd4549f
amz-sdk-invocation-id: c1cb0eef-010b-4fad-af70-077be6636cc9
amz-sdk-request: attempt=1
No request content
[m:auto]
[1/1]
[*:8080]
Flow: e Edit D Duplicate r Replay e Export d Delete b Save body ␣ Next flow p Prev flow
Proxy: ? Help q Back E Events O Options i Intercept f Filter w Save flows z Clear list


mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy ⌥⌘2
Flow Details
2023-11-07 16:40:41 GET https://micktwomey-scratch-example.s3.eu-west-1.amazonaws.com/test/prefix1/ham/spam/foo.txt
← 200 OK text/plain 6b 110ms
Request
Response
Detail
x-amz-id-2: ZtbuzYalwmED0MWZcP9IOun7uGin2aOUkmeoONeDy13UfxwC/WRFq8FMyI4KgVFsxnyHlDlh5Ec=
x-amz-request-id: G1B3PKNVJSJ5MKQF
Date: Tue, 07 Nov 2023 16:40:42 GMT
Last-Modified: Fri, 20 Oct 2023 17:06:43 GMT
ETag: "b1946ac92492d2347c6235b4d2611184"
x-amz-server-side-encryption: AES256
Accept-Ranges: bytes
Content-Type: text/plain
Server: AmazonS3
Content-Length: 6
Raw
[m:auto]
hello
[1/1]
[*:8080]
Flow: e Edit D Duplicate r Replay e Export d Delete b Save body ␣ Next flow p Prev flow
Proxy: ? Help q Back E Events O Options i Intercept f Filter w Save flows z Clear list


mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
⌥⌘2
Flow Details
2023-11-07 16:40:41 GET https://micktwomey-scratch-example.s3.eu-west-1.amazonaws.com/test/prefix1/ham/spam/foo.txt
← 200 OK text/plain 6b 110ms
Request
Response
Detail
Server Connection:
Address micktwomey-scratch-example.s3.eu-west-1.amazonaws.com:443
Resolved Address 52.218.109.0:443
HTTP Version HTTP/1.1
ALPN http/1.1
Server Certificate:
Type RSA, 2048 bits
SHA256 digest 63 4c 10 56 3c 85 c7 ae c4 b4 72 43 69 9f 8d 2e dd 80 18 92 2a d7 06 cb 7a d2 17 8a 7b a3 c0 29
Valid from 2023-10-10 00:00:00+00:00
Valid to 2024-09-12 23:59:59+00:00
Serial 349764327232630685894775432209678702O
Subject CN *.s3-eu-west-1.amazonaws.com
Issuer C US
O Amazon
CN Amazon RSA 2048 M01
Alt names s3-eu-west-1.amazonaws.com, *.s3-eu-west-1.amazonaws.com, s3.eu-west-1.amazonaws.com,
*.s3.eu-west-1.amazonaws.com, s3.dualstack.eu-west-1.amazonaws.com,
*.s3.dualstack.eu-west-1.amazonaws.com, *.s3.amazonaws.com, *.s3-control.eu-west-1.amazonaws.com,
s3-control.eu-west-1.amazonaws.com, *.s3-control.dualstack.eu-west-1.amazonaws.com,
s3-control.dualstack.eu-west-1.amazonaws.com, *.s3-accesspoint.eu-west-1.amazonaws.com,
*.s3-accesspoint.dualstack.eu-west-1.amazonaws.com, *.s3-deprecated.eu-west-1.amazonaws.com,
s3-deprecated.eu-west-1.amazonaws.com, s3-external-3.amazonaws.com, *.s3-external-3.amazonaws.com
Client Connection:
Address [::1]:56923
HTTP Version HTTP/1.1
[1/1]
[*:8080]
Flow: e Edit D Duplicate r Replay e Export d Delete b Save body ␣ Next flow p Prev flow
Proxy: ? Help q Back E Events O Options i Intercept f Filter w Save flows z Clear list


mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
⌥⌘2
Flow Details
2023-11-07 16:40:41 GET https://micktwomey-scratch-example.s3.eu-west-1.amazonaws.com/test/prefix1/ham/spam/foo.txt
← 200 OK text/plain 6b 110ms
Request
Response
Detail
*.s3.eu-west-1.amazonaws.com, s3.dualstack.eu-west-1.amazonaws.com,
*.s3.dualstack.eu-west-1.amazonaws.com, *.s3.amazonaws.com, *.s3-control.eu-west-1.amazonaws.com,
s3-control.eu-west-1.amazonaws.com, *.s3-control.dualstack.eu-west-1.amazonaws.com,
s3-control.dualstack.eu-west-1.amazonaws.com, *.s3-accesspoint.eu-west-1.amazonaws.com,
*.s3-accesspoint.dualstack.eu-west-1.amazonaws.com, *.s3-deprecated.eu-west-1.amazonaws.com,
s3-deprecated.eu-west-1.amazonaws.com, s3-external-3.amazonaws.com, *.s3-external-3.amazonaws.com
Client Connection:
Address [::1]:56923
HTTP Version HTTP/1.1
TLS Version TLSv1.3
Server Name Indication micktwomey-scratch-example.s3.eu-west-1.amazonaws.com
Cipher Name TLS_AES_256_GCM_SHA384
ALPN http/1.1
Timing:
Client conn. established 2023-11-07 16:40:41.285
Server conn. initiated 2023-11-07 16:40:41.288
Server conn. TCP handshake 2023-11-07 16:40:41.354
Server conn. TLS handshake 2023-11-07 16:40:41.419
Client conn. TLS handshake 2023-11-07 16:40:41.429
First request byte 2023-11-07 16:40:41.430
Request complete 2023-11-07 16:40:41.432
First response byte 2023-11-07 16:40:41.538
Response complete 2023-11-07 16:40:41.540
Client conn. closed 2023-11-07 16:40:41.560
Server conn. closed 2023-11-07 16:40:41.560
[1/1]
[*:8080]
Flow: e Edit D Duplicate r Replay e Export d Delete b Save body ␣ Next flow p Prev flow
Proxy: ? Help q Back E Events O Options i Intercept f Filter w Save flows z Clear list

What were we doing again?

1. Got some jobs taking a long time
2. Guessed it's retries causing this
3. Want to use events to monitor these but don't have docs
4. Want to somehow modify responses to simulate retries and see what happens
5. Can intercept requests using mitmproxy
6. ??

Cool, we can watch the network traffic, but how does that help us figure out the events?

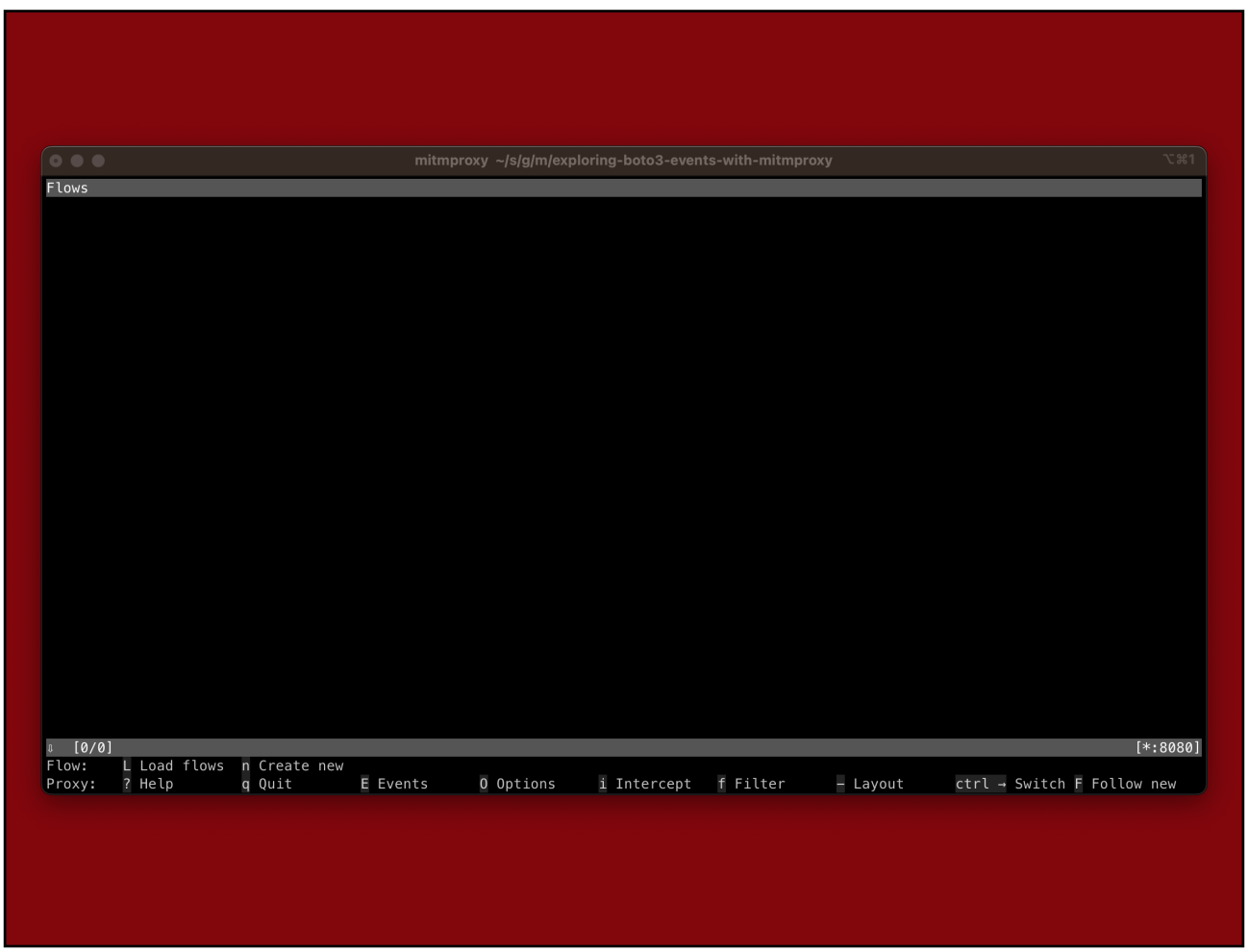
More than a HTTP debugger

https://docs.mitmproxy.org/stable/mitmproxytutorial-interceptrequests/
https://docs.mitmproxy.org/stable/mitmproxytutorial-modifyrequests/

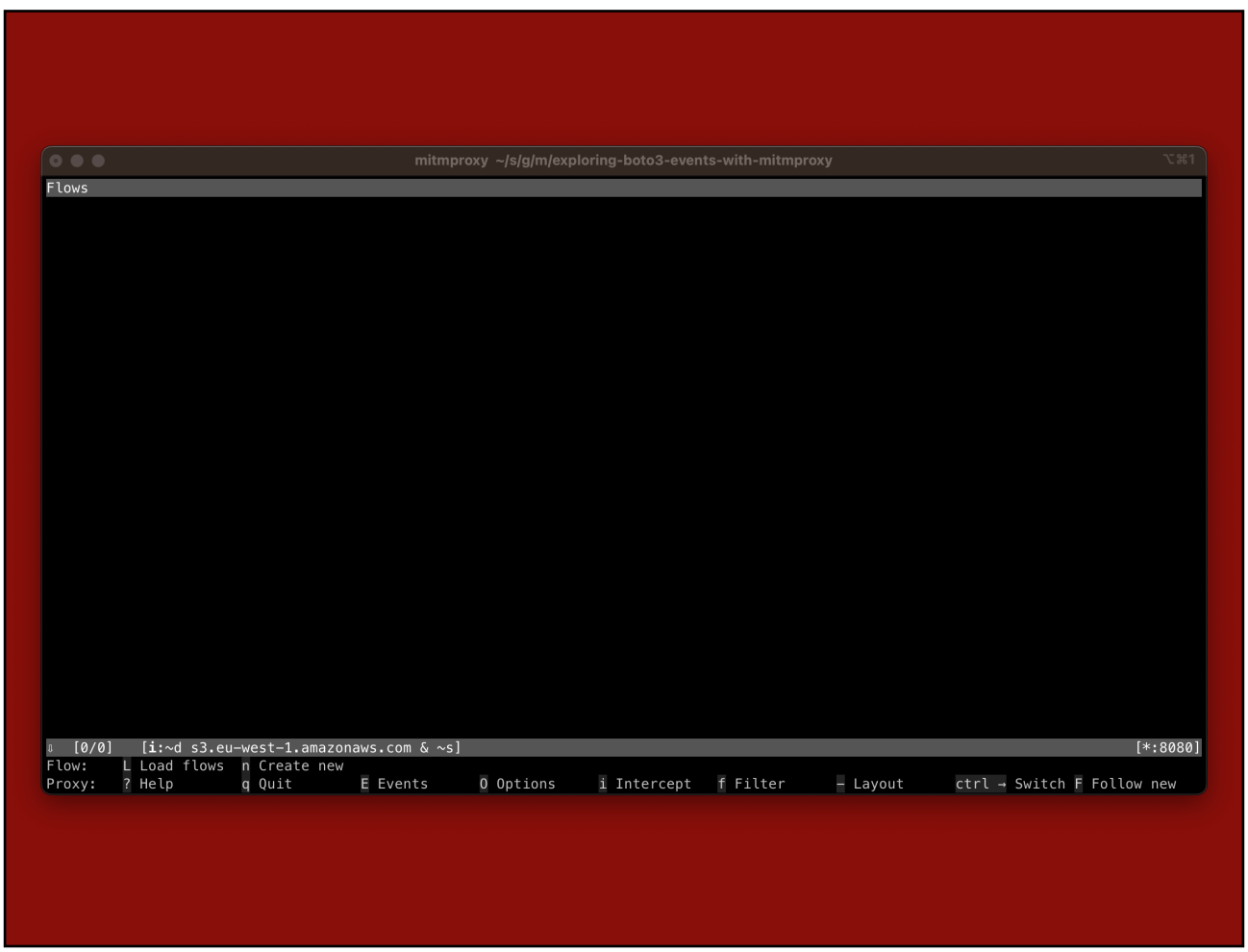
mitmproxy offers the ability to intercept and change HTTP requests

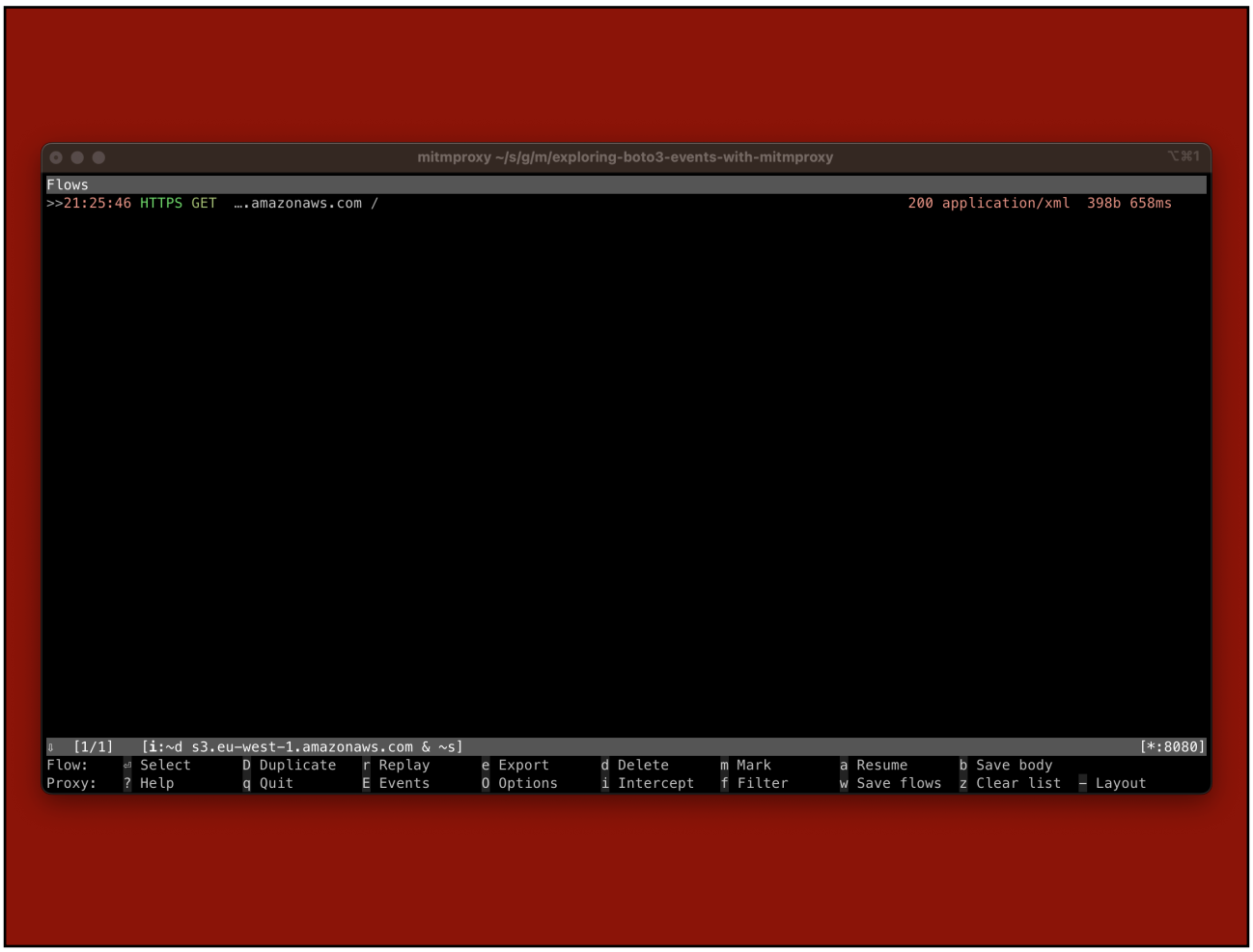
It also offers a full API and the ability to replay requests

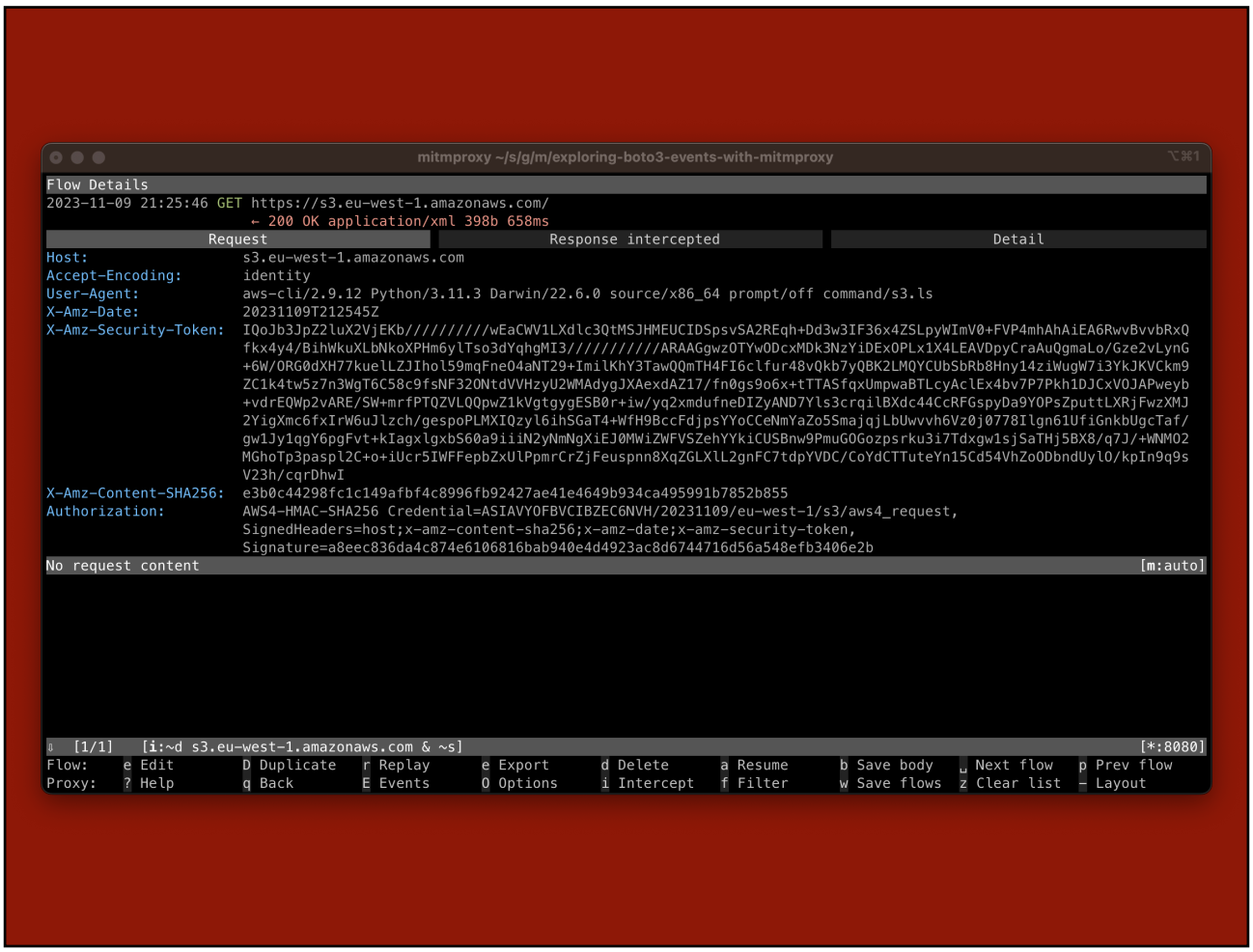
1. Hit `i` to create an intercept
2. `~d s3.eu-west-1.amazonaws.com & ~s ~d` match on domain, `~s` match on server response
3. Run the command
4. In the UI go into the response and hit `e`
5. Change the response code to `429`
6. Hit `a` to allow the request to continue
7. Watch what happens in the command

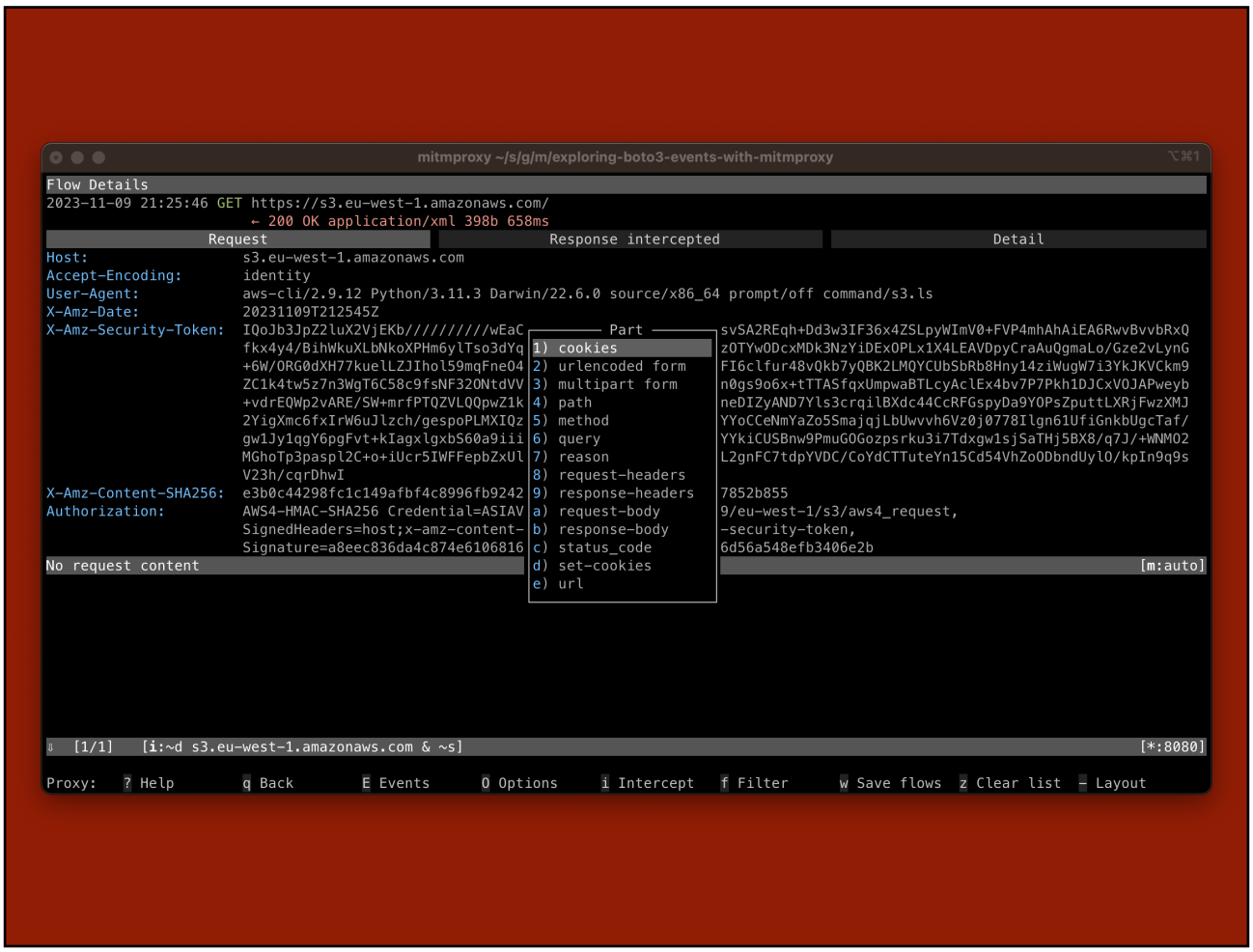

mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
Flows
[0/0]
[*:8080]
Flow: L Load flows n Create new
Proxy: ? Help q Quit E Events O Options i Intercept f Filter - Layout ctrl → Switch F Follow new

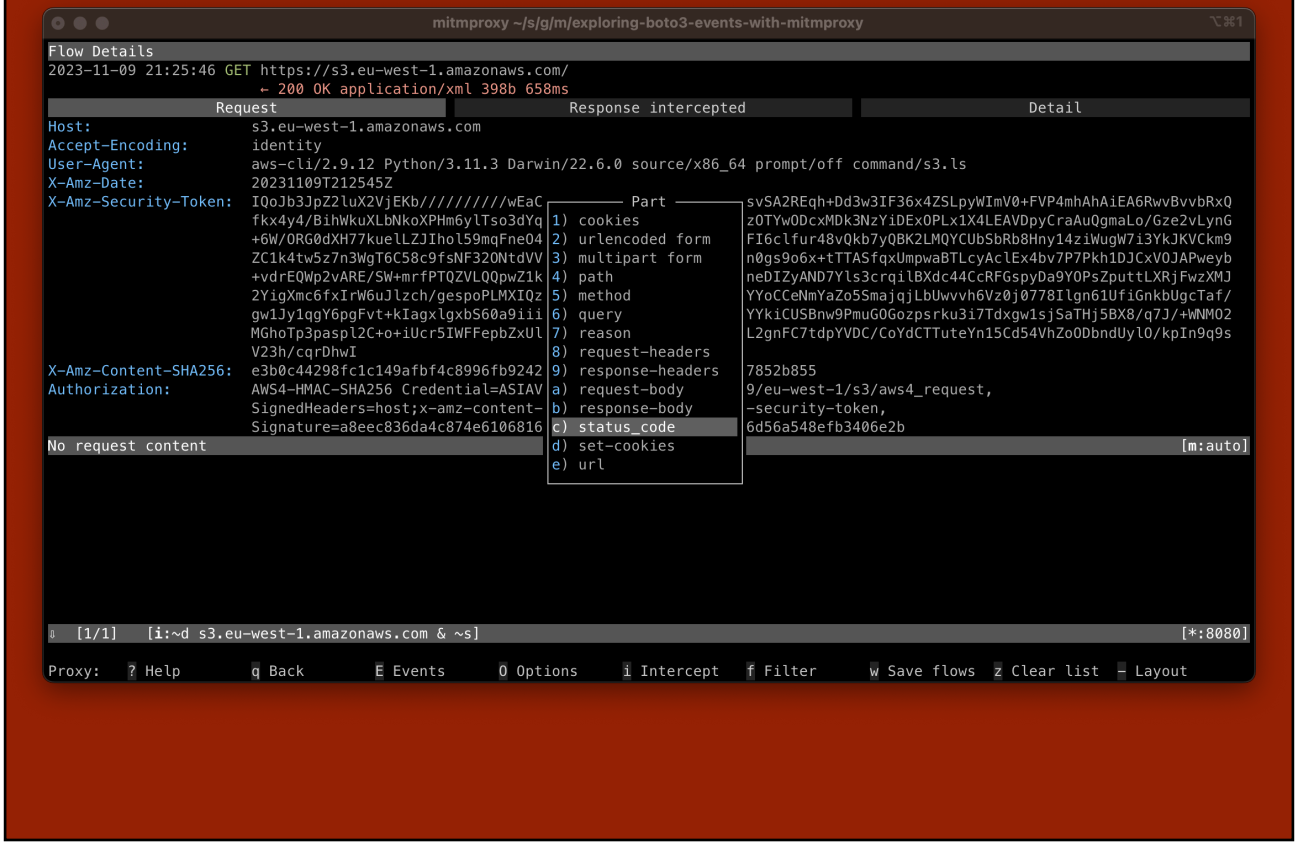

mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
Flows
[0/0]
[*:8080]
: set intercept '~d s3.eu-west-1.amazonaws.com & ~s'

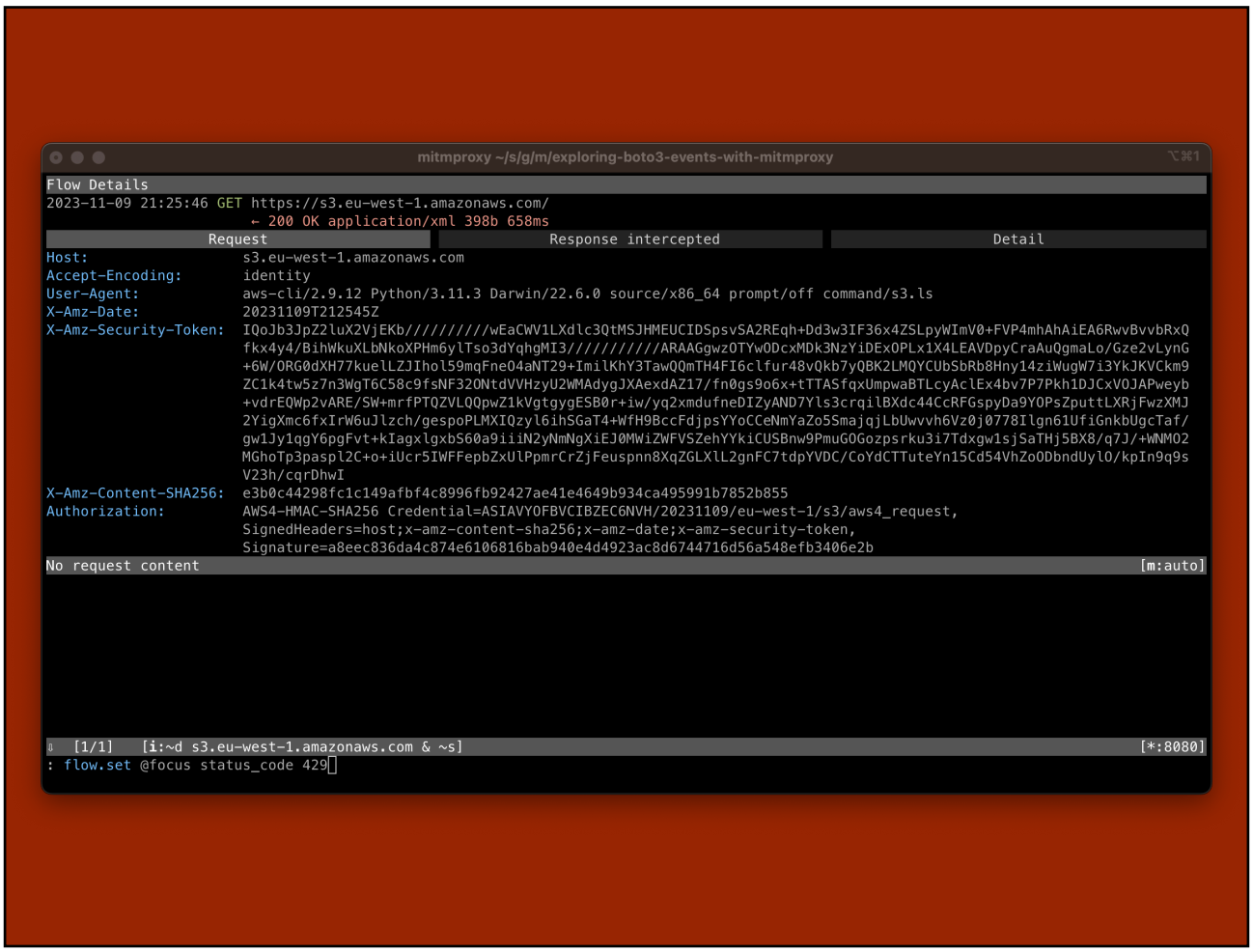

mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
Flows
[0/0] [i:~d s3.eu-west-1.amazonaws.com & ~s] [*:8080]
Flow: L Load flows n Create new
Proxy: ? Help q Quit E Events O Options i Intercept f Filter - Layout ctrl → Switch F Follow new

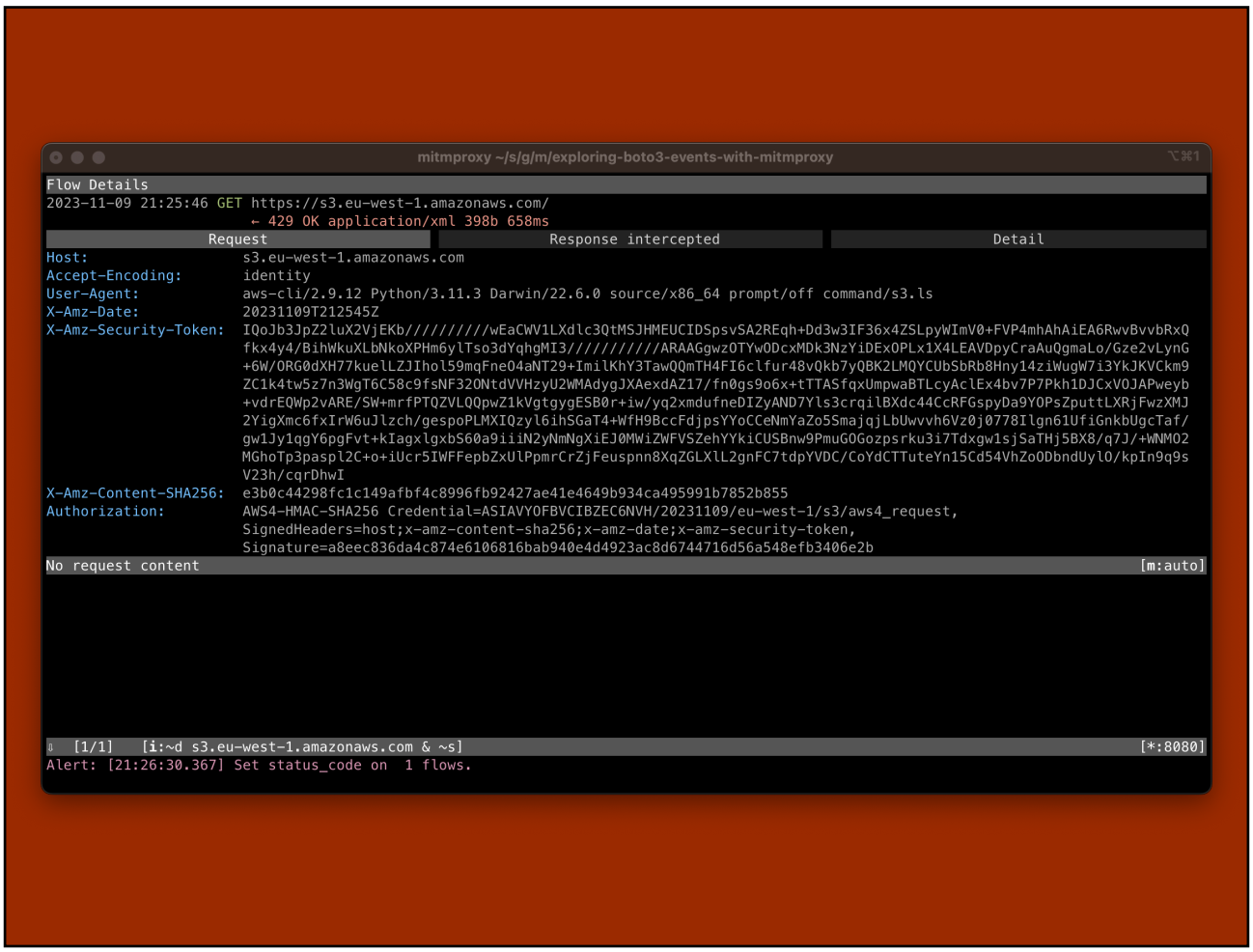

mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
Flows
>>21:25:46 HTTPS GET …amazonaws.com /
200 application/xml 398b 658ms
[1/1] [i:~d s3.eu-west-1.amazonaws.com & ~s]
[*:8080]
Flow: ⏎ Select D Duplicate r Replay e Export d Delete m Mark a Resume b Save body
Proxy: ? Help q Quit E Events O Options i Intercept f Filter w Save flows z Clear list - Layout

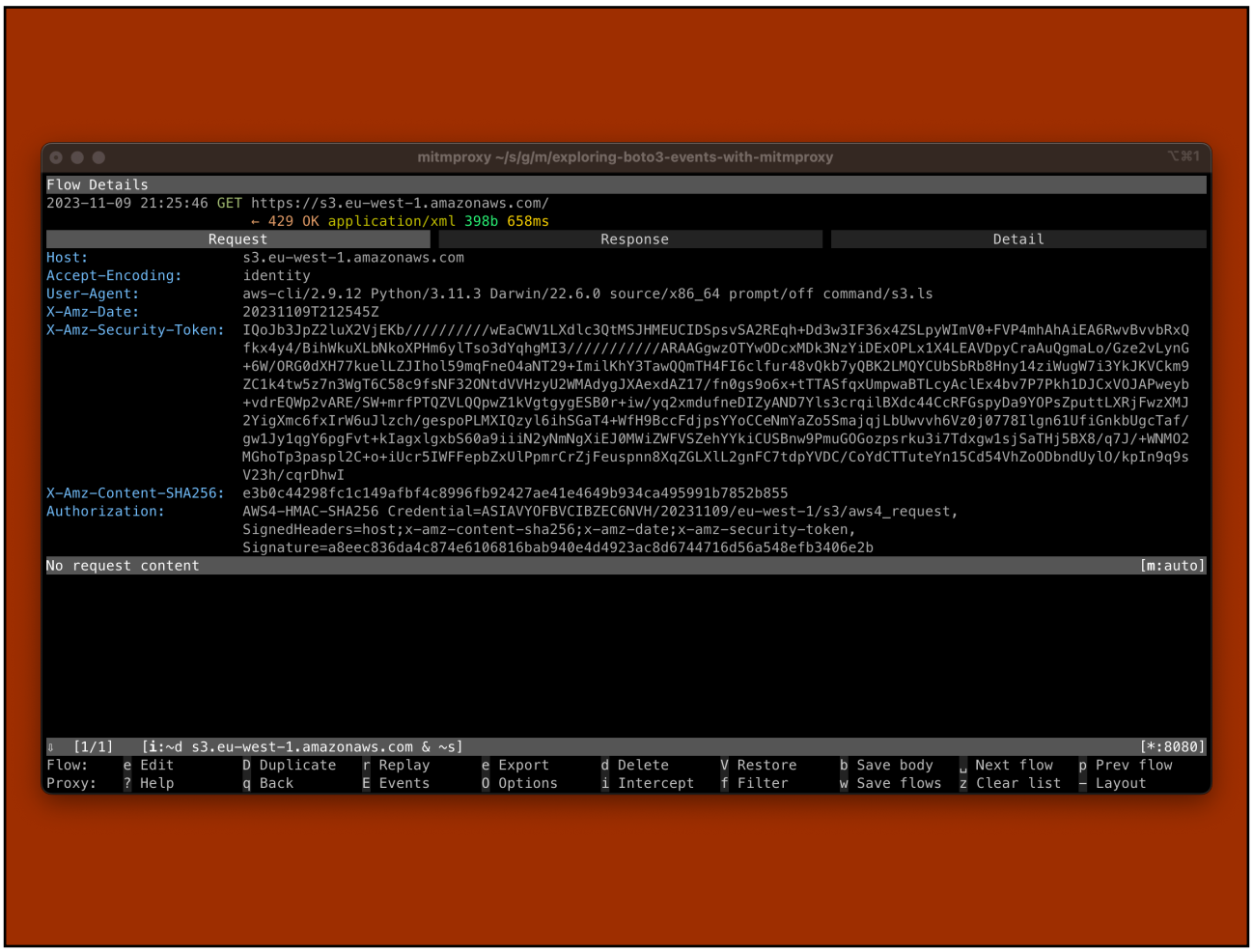

mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
Flow Details
2023-11-09 21:25:46 GET https://s3.eu-west-1.amazonaws.com/
← 200 OK application/xml 398b 658ms
Request
Response intercepted
Detail
Host: s3.eu-west-1.amazonaws.com
Accept-Encoding: identity
User-Agent: aws-cli/2.9.12 Python/3.11.3 Darwin/22.6.0 source/x86_64 prompt/off command/s3.ls
X-Amz-Date: 20231109T212545Z
X-Amz-Security-Token: IQoJb3JpZ2luX2VjEKb/////////wEaCWV1LXdlc3QtMSJHMEUCIDSpsvSA2REqh+Dd3w3IF36x4ZSLpyWImV0+FVP4mhAhAiEA6RwvBvvbRxQ
fkx4y4/BihWkuXLbNkoXPHm6ylTso3dYqhgMI3////////////ARAAGgwzOTYwODcxMDk3NzYiDExOPLx1X4LEAVDpyCraAuQgmaLo/Gze2vLynG
+6W/ORG0dXH77kuelLZJIhol59mqFneO4aNT29+ImilKhY3TawQQmTH4FI6clfur48vQkb7yQBK2LMQYCUbSbRb8Hny14ziWugW7i3YkJKVCkm9
ZC1k4tw5z7n3WgT6C58c9fsNF32ONtdVVHzyU2WMAdygJXAexdAZ17/fn0gs9o6x+tTTASfqxUmpwaBTLcyAclEx4bv7P7Pkh1DJCxVOJAPweyb
+vdrEQWp2vARE/SW+mrfPTQZVLQQpwZ1kVgtgygESB0r+iw/yq2xmdufneDIZyAND7Yls3crqilBXdc44CcRFGspyDa9YOPsZputtLXRjFwzXMJ
2YigXmc6fxIrW6uJlzch/gespoPLMXIQzyl6ihSGaT4+WfH9BccFdjpsYYoCCeNmYaZo5SmajqjLbUwvvh6Vz0j0778Ilgn61UfiGnkbUgcTaf/
gw1Jy1qgY6pgFvt+kIagxlgxbS60a9iiiN2yNmNgXiEJ0MWiZWFVSZehYYkiCUSBnw9PmuGOGozpsrku3i7Tdxgw1sjSaTHj5BX8/q7J/+WNMO2
MGhoTp3paspl2C+o+iUcr5IWFFepbZxUlPpmrCrZjFeuspnn8XqZGLXlL2gnFC7tdpYVDC/CoYdCTTuteYn15Cd54VhZoODbndUylO/kpIn9q9s
V23h/cqrDhwI
X-Amz-Content-SHA256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
Authorization: AWS4-HMAC-SHA256 Credential=ASIAVYOFBVCIBZEC6NVH/20231109/eu-west-1/s3/aws4_request,
SignedHeaders=host;x-amz-content-sha256;x-amz-date;x-amz-security-token,
Signature=a8eec836da4c874e6106816bab940e4d4923ac8d6744716d56a548efb3406e2b
No request content
[m:auto]
[1/1]
[i:~d s3.eu-west-1.amazonaws.com & ~s]
[*:8080]
Flow: e Edit D Duplicate r Replay e Export d Delete a Resume b Save body ␣ Next flow p Prev flow
Proxy: ? Help q Back E Events O Options i Intercept f Filter w Save flows z Clear list - Layout

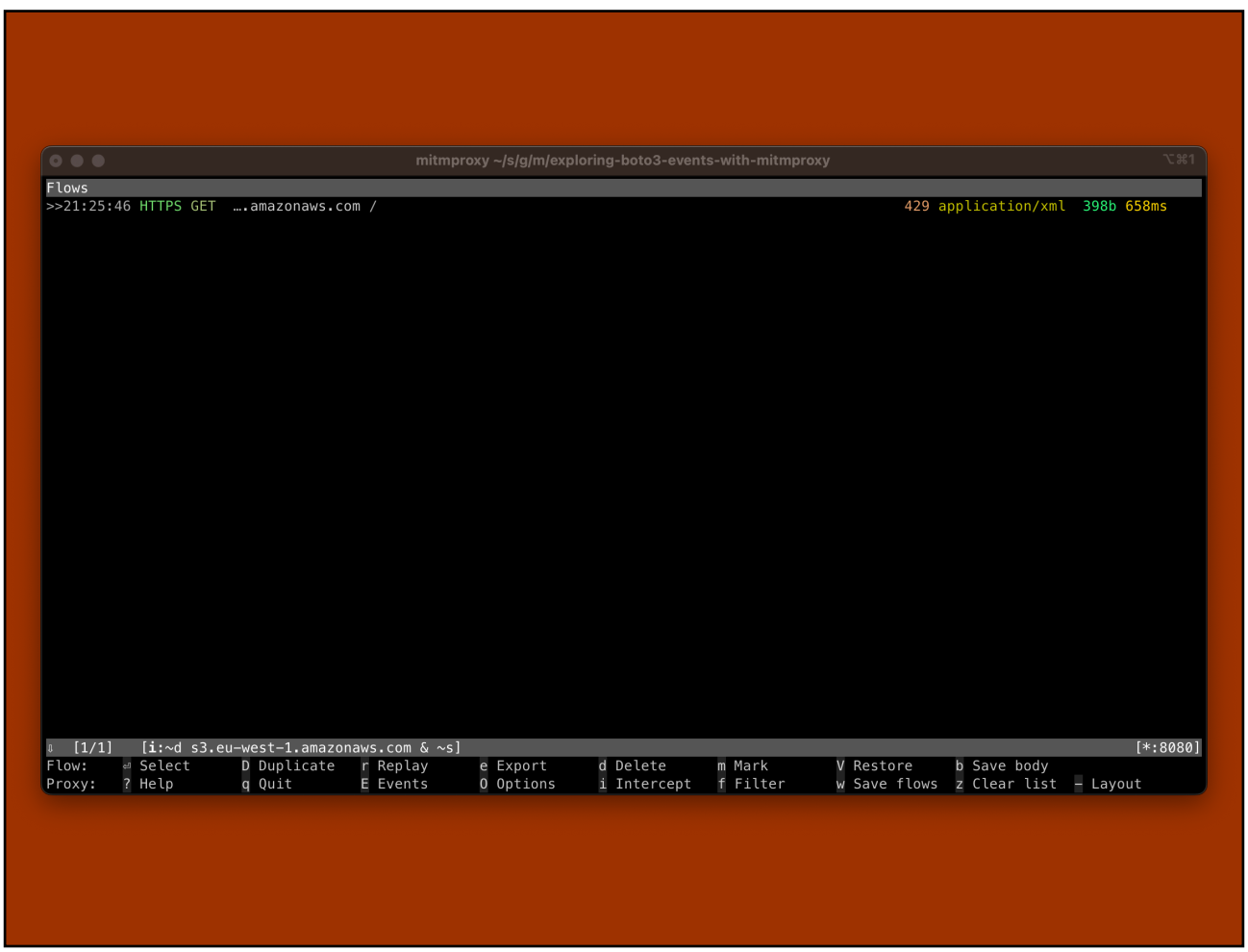

mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
Flow Details
2023-11-09 21:25:46 GET https://s3.eu-west-1.amazonaws.com/
← 200 OK application/xml 398b 658ms
Request
Response intercepted
Detail
Host: s3.eu-west-1.amazonaws.com
Accept-Encoding: identity
User-Agent: aws-cli/2.9.12 Python/3.11.3 Darwin/22.6.0 source/x86_64 prompt/off command/s3.ls
X-Amz-Date: 20231109T212545Z
X-Amz-Security-Token: IQoJb3JpZ2luX2VjEKb/////////wEaC
fkx4y4/BihWkuXLbNkoXPHm6ylTso3dYq
+6W/ORG0dXH77kuelLZJIhol59mqFneO4
ZC1k4tw5z7n3WgT6C58c9fsNF32ONtdVV
+vdrEQWp2vARE/SW+mrfPTQZVLQQpwZ1k
2YigXmc6fxIrW6uJlzch/gespoPLMXIQz
gw1Jy1qgY6pgFvt+kIagxlgxbS60a9iii
MGhoTp3paspl2C+o+iUcr5IWFFepbZxUl
V23h/cqrDhwI
X-Amz-Content-SHA256: e3b0c44298fc1c149afbf4c8996fb9242
Authorization: AWS4-HMAC-SHA256 Credential=ASIAV
SignedHeaders=host;x-amz-content-
Signature=a8eec836da4c874e6106816
Part
1) cookies
2) urlencoded form
3) multipart form
4) path
5) method
6) query
7) reason
8) request-headers
9) response-headers
a) request-body
b) response-body
c) status_code
d) set-cookies
e) url
svSA2REqh+Dd3w3IF36x4ZSLpyWImV0+FVP4mhAhAiEA6RwvBvvbRxQ
zOTYwODcxMDk3NzYiDExOPLx1X4LEAVDpyCraAuQgmaLo/Gze2vLynG
FI6clfur48vQkb7yQBK2LMQYCUbSbRb8Hny14ziWugW7i3YkJKVCkm9
n0gs9o6x+tTTASfqxUmpwaBTLcyAclEx4bv7P7Pkh1DJCxVOJAPweyb
neDIZyAND7Yls3crqilBXdc44CcRFGspyDa9YOPsZputtLXRjFwzXMJ
YYoCCeNmYaZo5SmajqjLbUwvvh6Vz0j0778Ilgn61UfiGnkbUgcTaf/
YYkiCUSBnw9PmuGOGozpsrku3i7Tdxgw1sjSaTHj5BX8/q7J/+WNMO2
L2gnFC7tdpYVDC/CoYdCTTuteYn15Cd54VhZoODbndUylO/kpIn9q9s
7852b855
9/eu-west-1/s3/aws4_request,
-security-token,
6d56a548efb3406e2b
No request content
[m:auto]
[1/1] [i:~d s3.eu-west-1.amazonaws.com & ~s]
[*:8080]
Proxy: ? Help q Back E Events O Options i Intercept f Filter w Save flows z Clear list - Layout

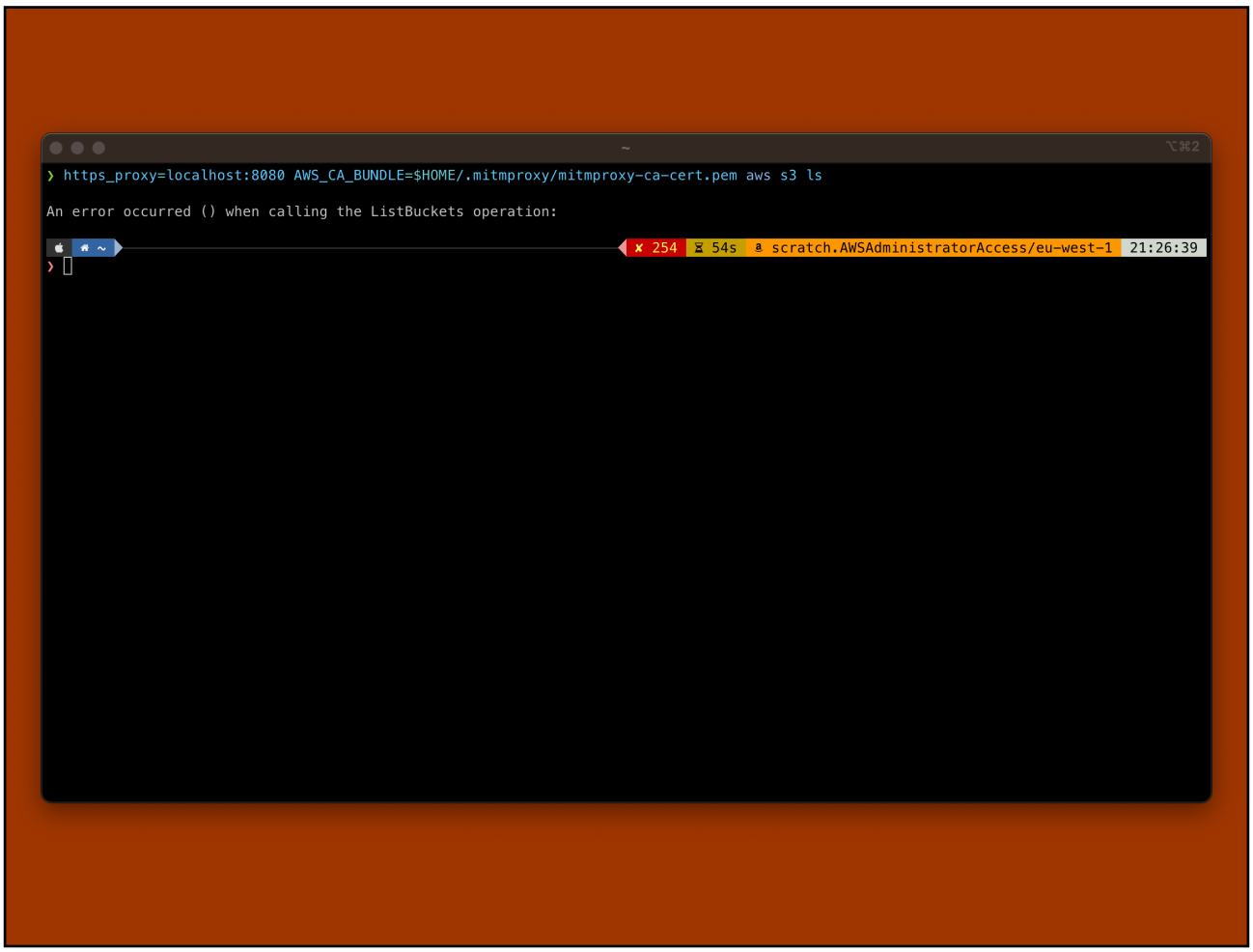

mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
Flow Details
2023-11-09 21:25:46 GET https://s3.eu-west-1.amazonaws.com/
← 200 OK application/xml 398b 658ms
Request
Response intercepted
Detail
Host: s3.eu-west-1.amazonaws.com
Accept-Encoding: identity
User-Agent: aws-cli/2.9.12 Python/3.11.3 Darwin/22.6.0 source/x86_64 prompt/off command/s3.ls
X-Amz-Date: 20231109T212545Z
X-Amz-Security-Token: IQoJb3JpZ2luX2VjEKb/////////wEaC
fkx4y4/BihWkuXLbNkoXPHm6ylTso3dYq
+6W/ORG0dXH77kuelLZJIhol59mqFneO4
ZC1k4tw5z7n3WgT6C58c9fsNF32ONtdVV
+vdrEQWp2vARE/SW+mrfPTQZVLQQpwZ1k
2YigXmc6fxIrW6uJlzch/gespoPLMXIQz
gw1Jy1qgY6pgFvt+kIagxlgxbS60a9iii
MGhoTp3paspl2C+o+iUcr5IWFFepbZxUl
V23h/cqrDhwI
X-Amz-Content-SHA256: e3b0c44298fc1c149afbf4c8996fb9242
Authorization: AWS4-HMAC-SHA256 Credential=ASIAV
SignedHeaders=host;x-amz-content-
Signature=a8eec836da4c874e6106816
Part
1) cookies
2) urlencoded form
3) multipart form
4) path
5) method
6) query
7) reason
8) request-headers
9) response-headers
a) request-body
b) response-body
c) status_code
d) set-cookies
e) url
svSA2REqh+Dd3w3IF36x4ZSLpyWImV0+FVP4mhAhAiEA6RwvBvvbRxQ
zOTYwODcxMDk3NzYiDExOPLx1X4LEAVDpyCraAuQgmaLo/Gze2vLynG
FI6clfur48vQkb7yQBK2LMQYCUbSbRb8Hny14ziWugW7i3YkJKVCkm9
n0gs9o6x+tTTASfqxUmpwaBTLcyAclEx4bv7P7Pkh1DJCxVOJAPweyb
neDIZyAND7Yls3crqilBXdc44CcRFGspyDa9YOPsZputtLXRjFwzXMJ
YYoCCeNmYaZo5SmajqjLbUwvvh6Vz0j0778Ilgn61UfiGnkbUgcTaf/
YYkiCUSBnw9PmuGOGozpsrku3i7Tdxgw1sjSaTHj5BX8/q7J/+WNMO2
L2gnFC7tdpYVDC/CoYdCTTuteYn15Cd54VhZoODbndUylO/kpIn9q9s
7852b855
9/eu-west-1/s3/aws4_request,
-security-token,
6d56a548efb3406e2b
No request content
[m:auto]
[1/1] [i:~d s3.eu-west-1.amazonaws.com & ~s]
[*:8080]
Proxy: ? Help q Back E Events O Options i Intercept f Filter w Save flows z Clear list - Layout


mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
Flow Details
2023-11-09 21:25:46 GET https://s3.eu-west-1.amazonaws.com/
← 200 OK application/xml 398b 658ms
Request
Response intercepted
Detail
Host: s3.eu-west-1.amazonaws.com
Accept-Encoding: identity
User-Agent: aws-cli/2.9.12 Python/3.11.3 Darwin/22.6.0 source/x86_64 prompt/off command/s3.ls
X-Amz-Date: 20231109T212545Z
X-Amz-Security-Token: IQoJb3JpZ2luX2VjEKb/////////wEaCWV1LXdlc3QtMSJHMEUCIDSpsvSA2REqh+Dd3w3IF36x4ZSLpyWImV0+FVP4mhAhAiEA6RwvBvvbRxQ
fkx4y4/BihWkuXLbNkoXPHm6ylTso3dYqhgMI3////////////ARAAGgwzOTYwODcxMDk3NzYiDExOPLx1X4LEAVDpyCraAuQgmaLo/Gze2vLynG
+6W/ORG0dXH77kuelLZJIhol59mqFneO4aNT29+ImilKhY3TawQQmTH4FI6clfur48vQkb7yQBK2LMQYCUbSbRb8Hny14ziWugW7i3YkJKVCkm9
ZC1k4tw5z7n3WgT6C58c9fsNF32ONtdVVHzyU2WMAdygJXAexdAZ17/fn0gs9o6x+tTTASfqxUmpwaBTLcyAclEx4bv7P7Pkh1DJCxVOJAPweyb
+vdrEQWp2vARE/SW+mrfPTQZVLQQpwZ1kVgtgygESB0r+iw/yq2xmdufneDIZyAND7Yls3crqilBXdc44CcRFGspyDa9YOPsZputtLXRjFwzXMJ
2YigXmc6fxIrW6uJlzch/gespoPLMXIQzyl6ihSGaT4+WfH9BccFdjpsYYoCCeNmYaZo5SmajqjLbUwvvh6Vz0j0778Ilgn61UfiGnkbUgcTaf/
gw1Jy1qgY6pgFvt+kIagxlgxbS60a9iiiN2yNmNgXiEJ0MWiZWFVSZehYYkiCUSBnw9PmuGOGozpsrku3i7Tdxgw1sjSaTHj5BX8/q7J/+WNMO2
MGhoTp3paspl2C+o+iUcr5IWFFepbZxUlPpmrCrZjFeuspnn8XqZGLXlL2gnFC7tdpYVDC/CoYdCTTuteYn15Cd54VhZoODbndUylO/kpIn9q9s
V23h/cqrDhwI
X-Amz-Content-SHA256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
Authorization: AWS4-HMAC-SHA256 Credential=ASIAVYOFBVCIBZEC6NVH/20231109/eu-west-1/s3/aws4_request,
SignedHeaders=host;x-amz-content-sha256;x-amz-date;x-amz-security-token,
Signature=a8eec836da4c874e6106816bab940e4d4923ac8d6744716d56a548efb3406e2b
No request content
[m:auto]
[1/1]
[i:~d s3.eu-west-1.amazonaws.com & ~s]
[*:8080]
: flow.set @focus status_code 429


mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
Flow Details
2023-11-09 21:25:46 GET https://s3.eu-west-1.amazonaws.com/
← 429 OK application/xml 398b 658ms
Request
Response intercepted
Detail
Host: s3.eu-west-1.amazonaws.com
Accept-Encoding: identity
User-Agent: aws-cli/2.9.12 Python/3.11.3 Darwin/22.6.0 source/x86_64 prompt/off command/s3.ls
X-Amz-Date: 20231109T212545Z
X-Amz-Security-Token: IQoJb3JpZ2luX2VjEKb/////////wEaCWV1LXdlc3QtMSJHMEUCIDSpsvSA2REqh+Dd3w3IF36x4ZSLpyWImV0+FVP4mhAhAiEA6RwvBvvbRxQ
fkx4y4/BihWkuXLbNkoXPHm6ylTso3dYqhgMI3////////////ARAAGgwzOTYwODcxMDk3NzYiDExOPLx1X4LEAVDpyCraAuQgmaLo/Gze2vLynG
+6W/ORG0dXH77kuelLZJIhol59mqFneO4aNT29+ImilKhY3TawQQmTH4FI6clfur48vQkb7yQBK2LMQYCUbSbRb8Hny14ziWugW7i3YkJKVCkm9
ZC1k4tw5z7n3WgT6C58c9fsNF32ONtdVVHzyU2WMAdygJXAexdAZ17/fn0gs9o6x+tTTASfqxUmpwaBTLcyAclEx4bv7P7Pkh1DJCxVOJAPweyb
+vdrEQWp2vARE/SW+mrfPTQZVLQQpwZ1kVgtgygESB0r+iw/yq2xmdufneDIZyAND7Yls3crqilBXdc44CcRFGspyDa9YOPsZputtLXRjFwzXMJ
2YigXmc6fxIrW6uJlzch/gespoPLMXIQzyl6ihSGaT4+WfH9BccFdjpsYYoCCeNmYaZo5SmajqjLbUwvvh6Vz0j0778Ilgn61UfiGnkbUgcTaf/
gw1Jy1qgY6pgFvt+kIagxlgxbS60a9iiiN2yNmNgXiEJ0MWiZWFVSZehYYkiCUSBnw9PmuGOGozpsrku3i7Tdxgw1sjSaTHj5BX8/q7J/+WNMO2
MGhoTp3paspl2C+o+iUcr5IWFFepbZxUlPpmrCrZjFeuspnn8XqZGLXlL2gnFC7tdpYVDC/CoYdCTTuteYn15Cd54VhZoODbndUylO/kpIn9q9s
V23h/cqrDhwI
X-Amz-Content-SHA256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
Authorization: AWS4-HMAC-SHA256 Credential=ASIAVYOFBVCIBZEC6NVH/20231109/eu-west-1/s3/aws4_request,
SignedHeaders=host;x-amz-content-sha256;x-amz-date;x-amz-security-token,
Signature=a8eec836da4c874e6106816bab940e4d4923ac8d6744716d56a548efb3406e2b
No request content
[m:auto]
[1/1]
[i:~d s3.eu-west-1.amazonaws.com & ~s]
[*:8080]
Alert: [21:26:30.367] Set status_code on 1 flows.


mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
Flow Details
2023-11-09 21:25:46 GET https://s3.eu-west-1.amazonaws.com/
← 429 OK application/xml 398b 658ms
Request
Response
Detail
Host: s3.eu-west-1.amazonaws.com
Accept-Encoding: identity
User-Agent: aws-cli/2.9.12 Python/3.11.3 Darwin/22.6.0 source/x86_64 prompt/off command/s3.ls
X-Amz-Date: 20231109T212545Z
X-Amz-Security-Token: IQoJb3JpZ2luX2VjEKb/////////wEaCWV1LXdlc3QtMSJHMEUCIDSpsvSA2REqh+Dd3w3IF36x4ZSLpyWImV0+FVP4mhAhAiEA6RwvBvvbRxQ
fkx4y4/BihWkuXLbNkoXPHm6ylTso3dYqhgMI3////////////ARAAGgwzOTYwODcxMDk3NzYiDExOPLx1X4LEAVDpyCraAuQgmaLo/Gze2vLynG
+6W/ORG0dXH77kuelLZJIhol59mqFneO4aNT29+ImilKhY3TawQQmTH4FI6clfur48vQkb7yQBK2LMQYCUbSbRb8Hny14ziWugW7i3YkJKVCkm9
ZC1k4tw5z7n3WgT6C58c9fsNF32ONtdVVHzyU2WMAdygJXAexdAZ17/fn0gs9o6x+tTTASfqxUmpwaBTLcyAclEx4bv7P7Pkh1DJCxVOJAPweyb
+vdrEQWp2vARE/SW+mrfPTQZVLQQpwZ1kVgtgygESB0r+iw/yq2xmdufneDIZyAND7Yls3crqilBXdc44CcRFGspyDa9YOPsZputtLXRjFwzXMJ
2YigXmc6fxIrW6uJlzch/gespoPLMXIQzyl6ihSGaT4+WfH9BccFdjpsYYoCCeNmYaZo5SmajqjLbUwvvh6Vz0j0778Ilgn61UfiGnkbUgcTaf/
gw1Jy1qgY6pgFvt+kIagxlgxbS60a9iiiN2yNmNgXiEJ0MWiZWFVSZehYYkiCUSBnw9PmuGOGozpsrku3i7Tdxgw1sjSaTHj5BX8/q7J/+WNMO2
MGhoTp3paspl2C+o+iUcr5IWFFepbZxUlPpmrCrZjFeuspnn8XqZGLXlL2gnFC7tdpYVDC/CoYdCTTuteYn15Cd54VhZoODbndUylO/kpIn9q9s
V23h/cqrDhwI
X-Amz-Content-SHA256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
Authorization: AWS4-HMAC-SHA256 Credential=ASIAVYOFBVCIBZEC6NVH/20231109/eu-west-1/s3/aws4_request,
SignedHeaders=host;x-amz-content-sha256;x-amz-date;x-amz-security-token,
Signature=a8eec836da4c874e6106816bab940e4d4923ac8d6744716d56a548efb3406e2b
No request content
[m:auto]
[1/1]
[i:~d s3.eu-west-1.amazonaws.com & ~s]
[*:8080]
Flow: e Edit D Duplicate r Replay e Export d Delete V Restore b Save body ␣ Next flow p Prev flow
Proxy: ? Help q Back E Events O Options i Intercept f Filter w Save flows z Clear list - Layout

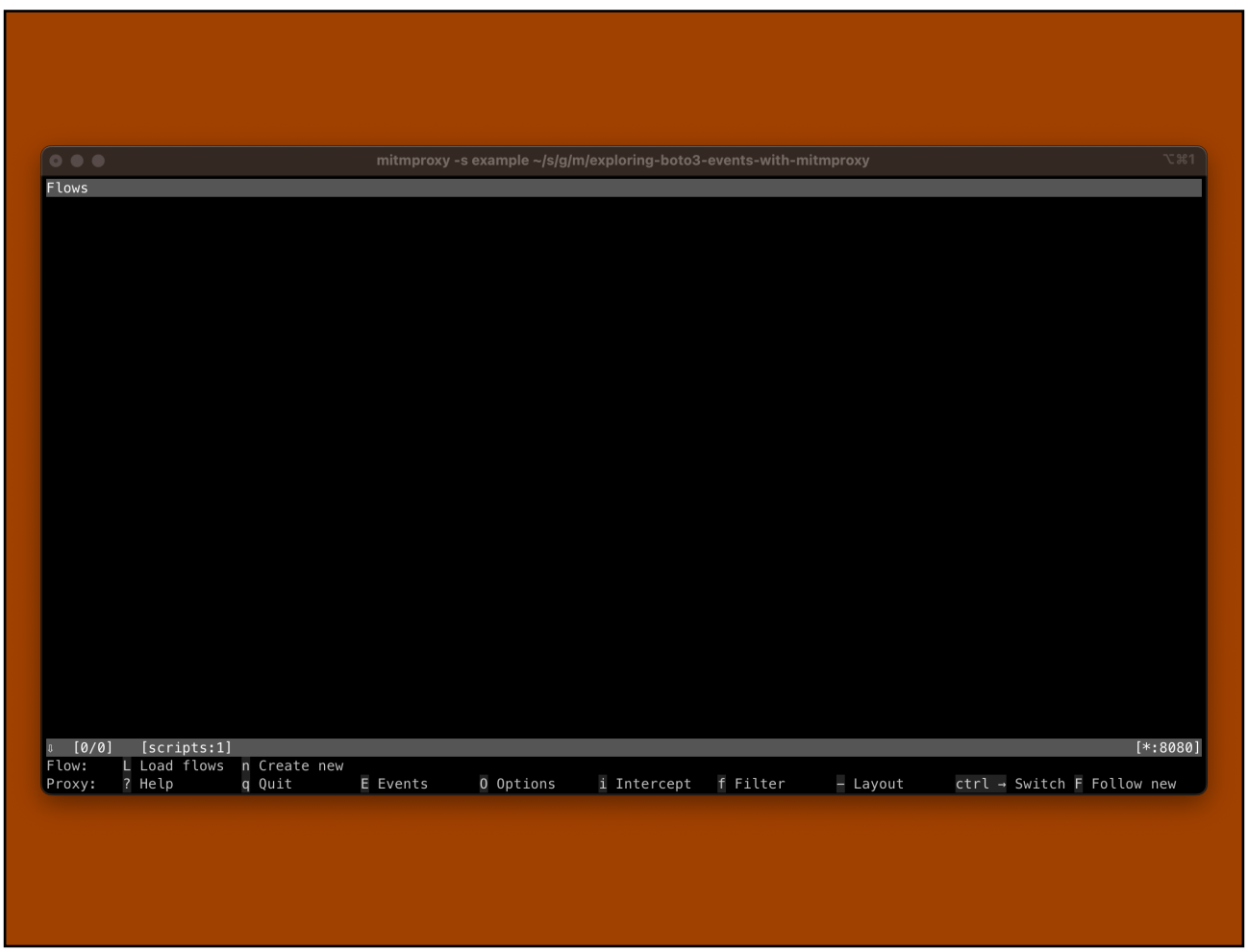

mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy
Flows
>>21:25:46 HTTPS GET …amazonaws.com / 429 application/xml 398b 658ms
[1/1] [i:~d s3.eu-west-1.amazonaws.com & ~s] [*:8080]
Flow: ⏎ Select D Duplicate r Replay e Export d Delete m Mark V Restore b Save body
Proxy: ? Help q Quit E Events O Options i Intercept f Filter w Save flows z Clear list - Layout

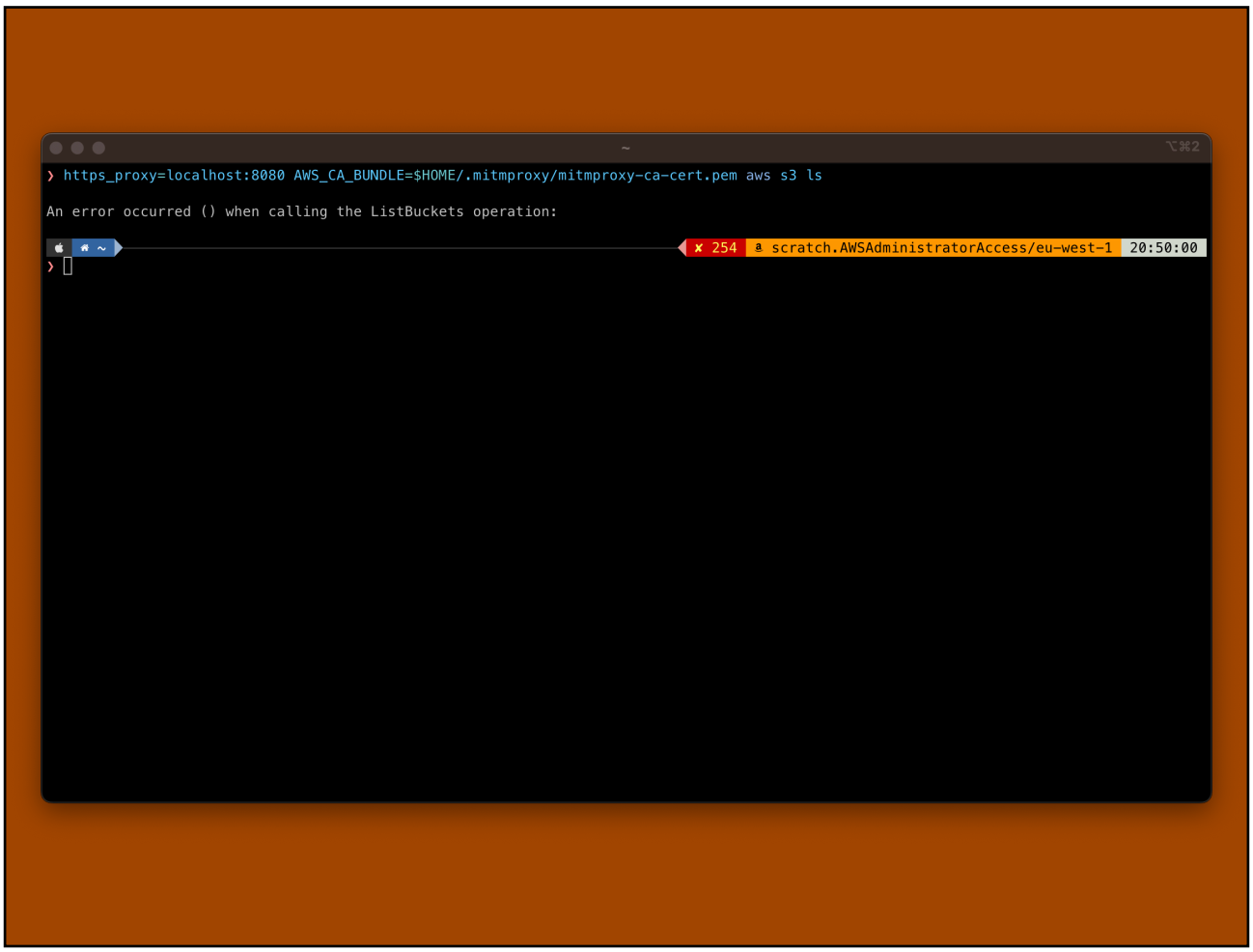

~
⌥⌘2
❯ https_proxy=localhost:8080 AWS_CA_BUNDLE=$HOME/.mitmproxy/mitmproxy-ca-cert.pem aws s3 ls
An error occurred () when calling the ListBuckets operation:
✘ 254
54s
scratch.AWSAdministratorAccess/eu-west-1
21:26:39
❯

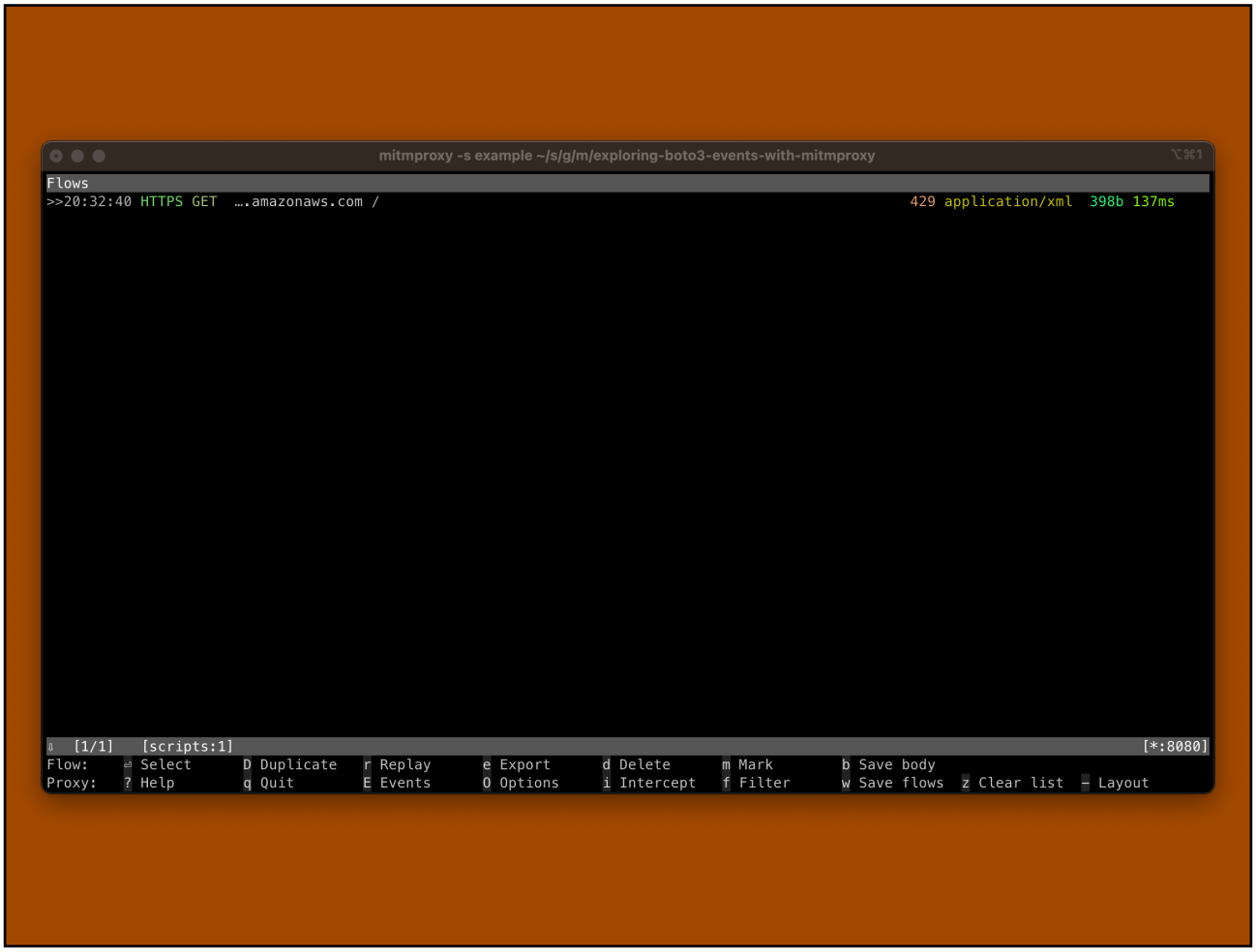
I'm Lazy

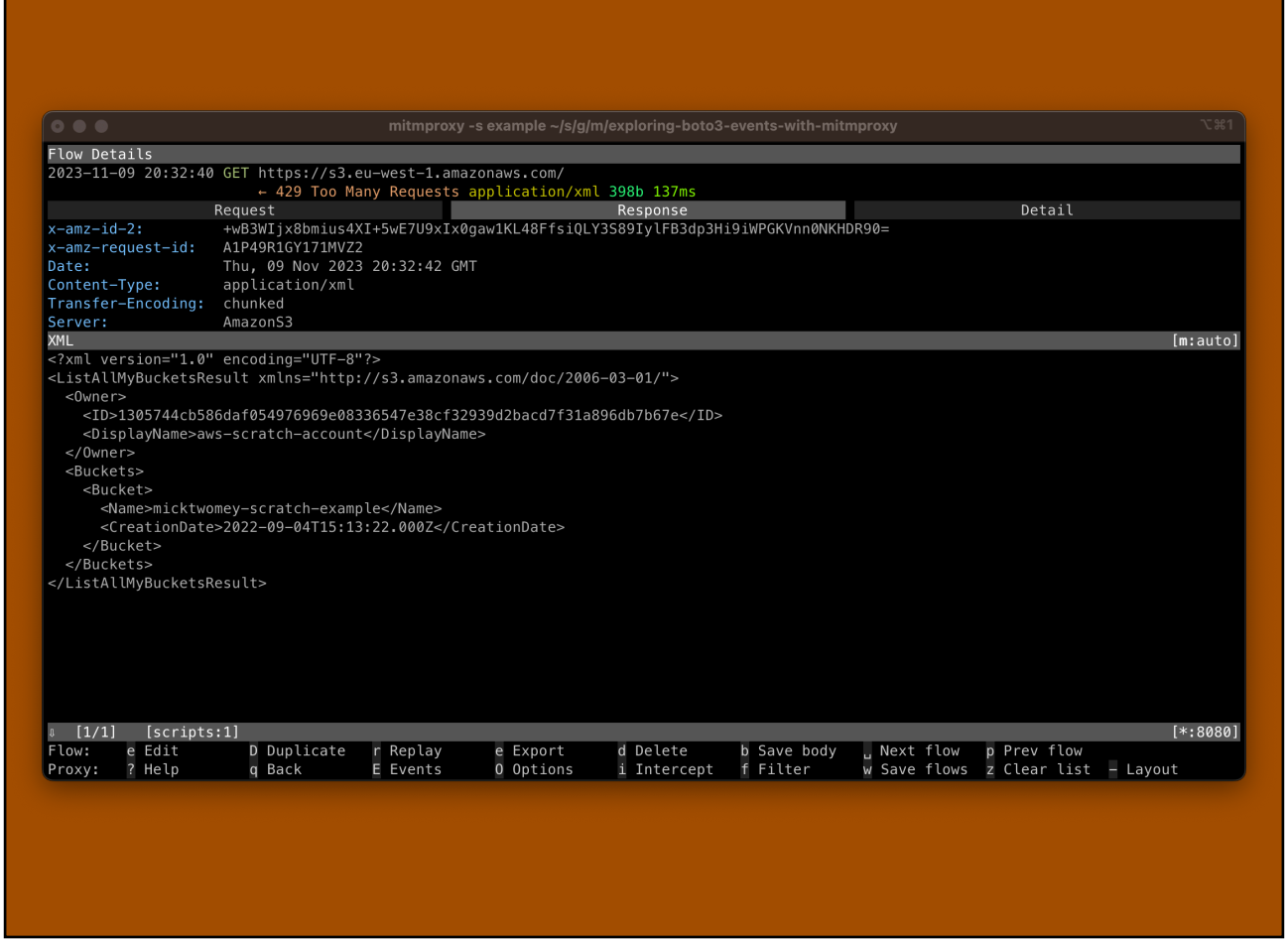
```python
import logging

from mitmproxy import flowfilter
from mitmproxy import http


class RateLimitExceededFilter:
    filter: flowfilter.TFilter

    def __init__(self):
        self.filter = flowfilter.parse("~d s3.eu-west-1.amazonaws.com & ~s")

    def response(self, flow: http.HTTPFlow) -> None:
        if flowfilter.match(self.filter, flow):
            logging.info(
              f"Flow {flow.request} matches filter: setting HTTP 429"
            )
            flow.response.status_code = 429
            flow.response.reason = "Too Many Requests"


addons = [RateLimitExceededFilter()]
```

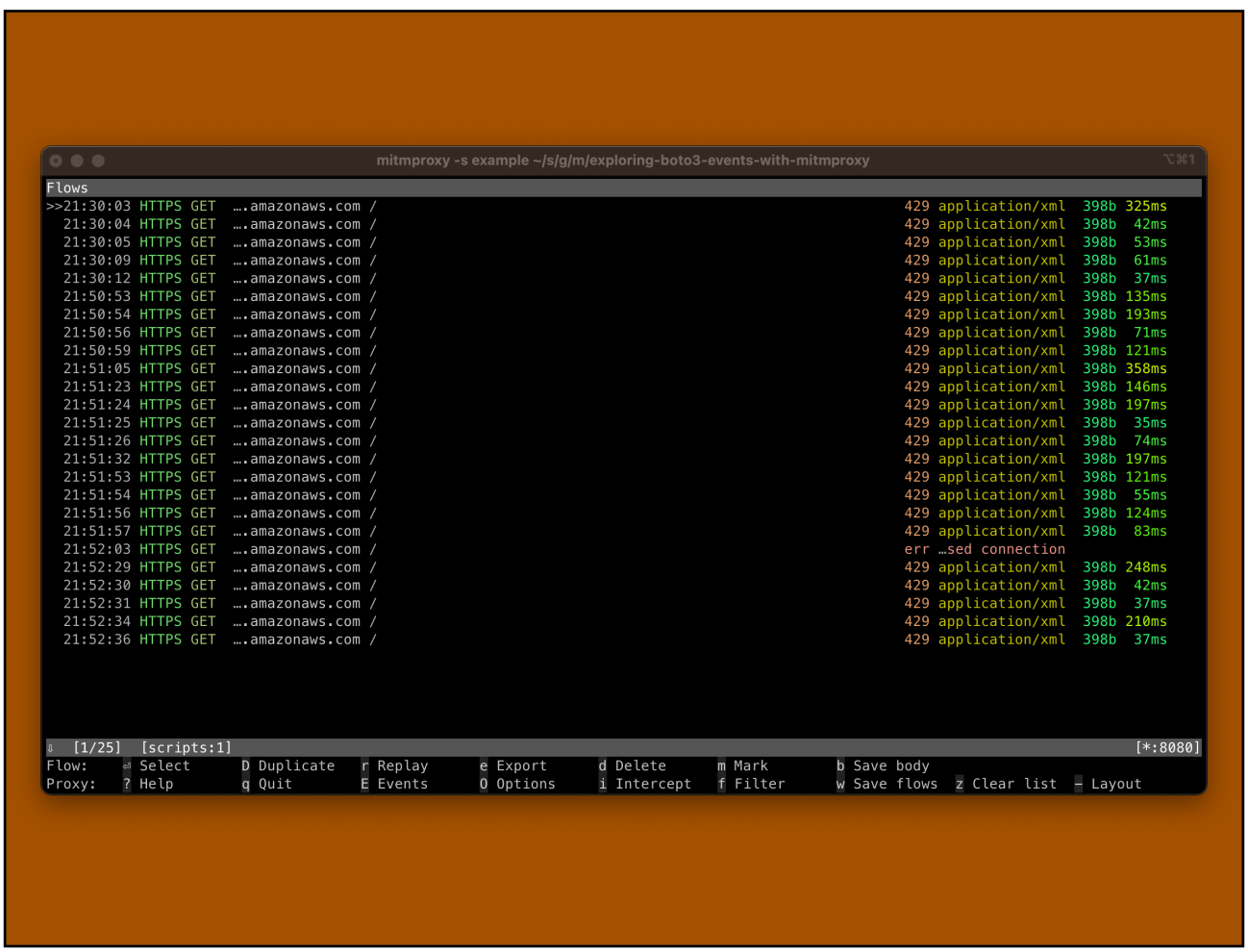
```
mitmproxy -s
examples/mitm_interception.py
```


mitmproxy -s example ~/s/g/m/exploring-boto3-events-with-mitmproxy
⌥⌘1
Flows
[0/0] [scripts:1] [*:8080]
Flow: L Load flows n Create new
Proxy: ? Help q Quit E Events O Options i Intercept f Filter - Layout ctrl → Switch F Follow new


~
⌥⌘2
❯ https_proxy=localhost:8080 AWS_CA_BUNDLE=$HOME/.mitmproxy/mitmproxy-ca-cert.pem aws s3 ls
An error occurred () when calling the ListBuckets operation:
✘ 254
scratch.AWSAdministratorAccess/eu-west-1
20:50:00
❯

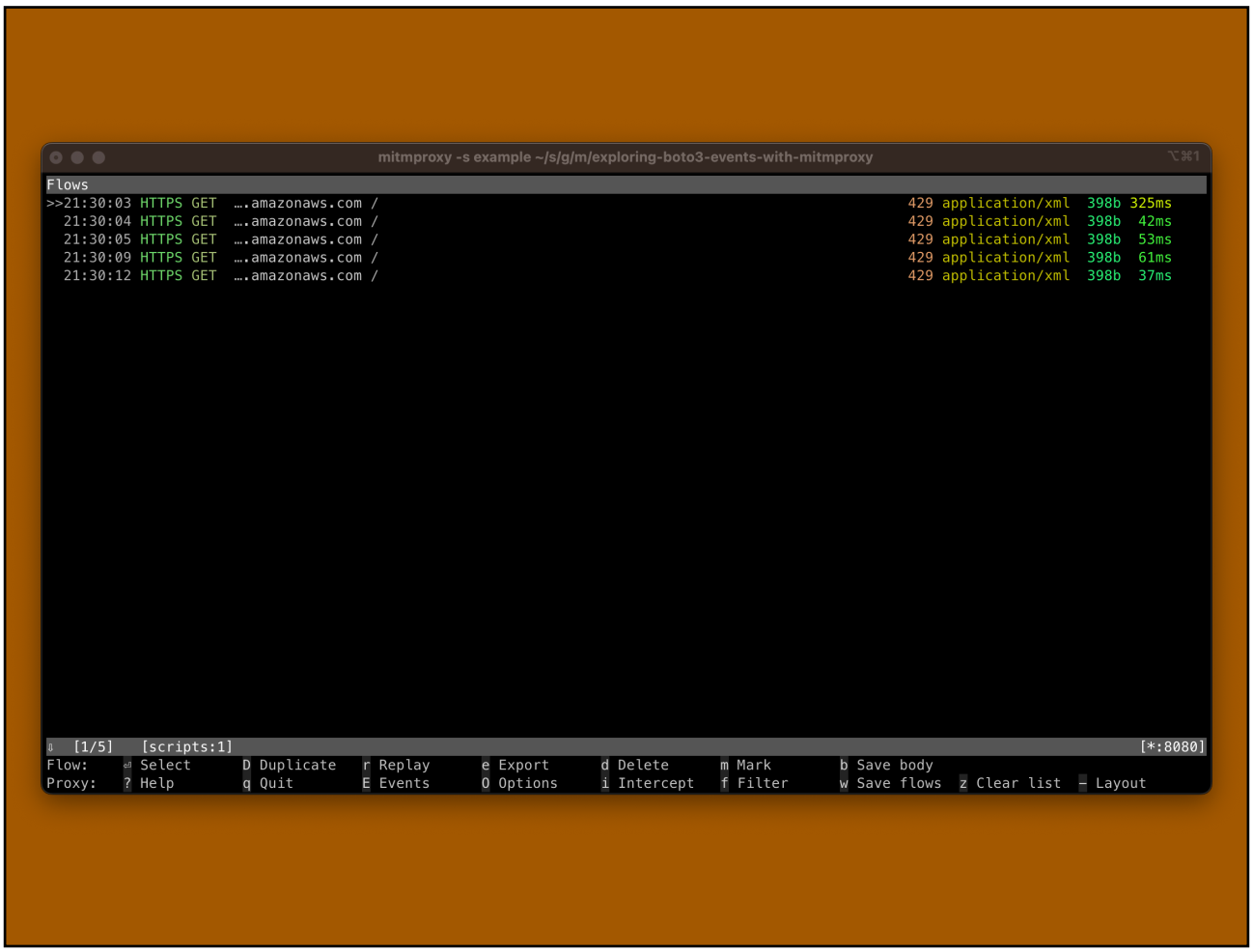

mitmproxy -s example ~/s/g/m/exploring-boto3-events-with-mitmproxy
Flows
>>20:32:40 HTTPS GET ….amazonaws.com /
429 application/xml 398b 137ms
[1/1] [scripts:1]
[*:8080]
Flow: ⏎ Select D Duplicate r Replay e Export d Delete m Mark b Save body
Proxy: ? Help q Quit E Events O Options i Intercept f Filter w Save flows z Clear list - Layout

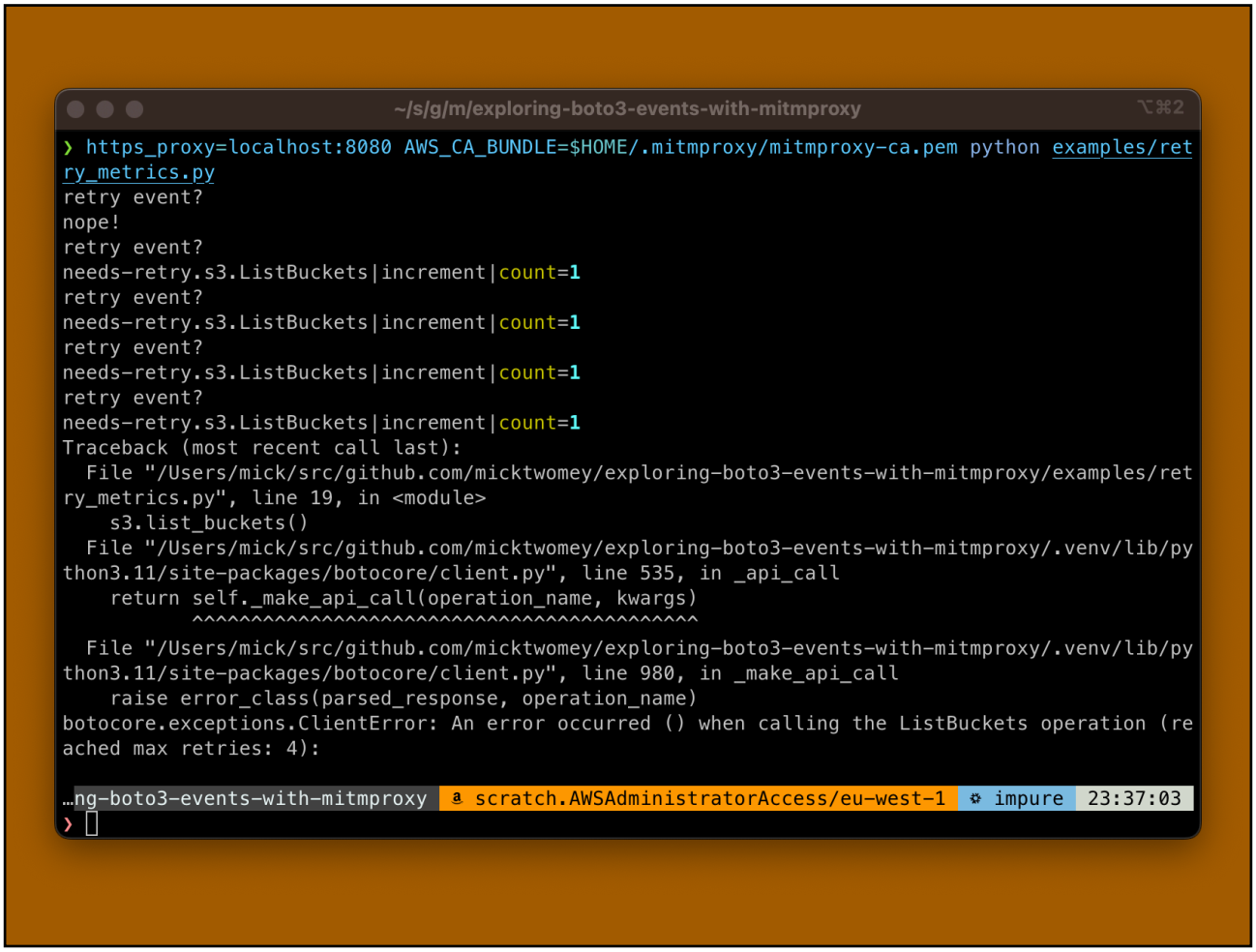

mitmproxy -s example ~/s/g/m/exploring-boto3-events-with-mitmproxy
Flow Details
2023-11-09 20:32:40 GET https://s3.eu-west-1.amazonaws.com/
← 429 Too Many Requests application/xml 398b 137ms
Request
Response
Detail
x-amz-id-2: +wB3WIjx8bmius4XI+5wE7U9xIx0gaw1KL48FfsiQLY3S89IylFB3dp3Hi9iWPGKVnn0NKHDR90=
x-amz-request-id: A1P49R1GY171MVZ2
Date: Thu, 09 Nov 2023 20:32:42 GMT
Content-Type: application/xml
Transfer-Encoding: chunked
Server: AmazonS3
XML
[m:auto]
<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Owner>
<ID>1305744cb586daf054976969e08336547e38cf32939d2bacd7f31a896db7b67e</ID>
<DisplayName>aws-scratch-account</DisplayName>
</Owner>
<Buckets>
<Bucket>
<Name>micktwomey-scratch-example</Name>
<CreationDate>2022-09-04T15:13:22.000Z</CreationDate>
</Bucket>
</Buckets>
</ListAllMyBucketsResult>
[1/1] [scripts:1]
[*:8080]
Flow: e Edit D Duplicate r Replay e Export d Delete b Save body ␣ Next flow p Prev flow
Proxy: ? Help q Back E Events O Options i Intercept f Filter w Save flows z Clear list - Layout


mitmproxy -s example ~/s/g/m/exploring-boto3-events-with-mitmproxy
Flows
>>21:30:03 HTTPS GET ….amazonaws.com / 429 application/xml 398b 325ms
21:30:04 HTTPS GET ….amazonaws.com / 429 application/xml 398b 42ms
21:30:05 HTTPS GET ….amazonaws.com / 429 application/xml 398b 53ms
21:30:09 HTTPS GET ….amazonaws.com / 429 application/xml 398b 61ms
21:30:12 HTTPS GET ….amazonaws.com / 429 application/xml 398b 37ms
21:50:53 HTTPS GET ….amazonaws.com / 429 application/xml 398b 135ms
21:50:54 HTTPS GET ….amazonaws.com / 429 application/xml 398b 193ms
21:50:56 HTTPS GET ….amazonaws.com / 429 application/xml 398b 71ms
21:50:59 HTTPS GET ….amazonaws.com / 429 application/xml 398b 121ms
21:51:05 HTTPS GET ….amazonaws.com / 429 application/xml 398b 358ms
21:51:23 HTTPS GET ….amazonaws.com / 429 application/xml 398b 146ms
21:51:24 HTTPS GET ….amazonaws.com / 429 application/xml 398b 197ms
21:51:25 HTTPS GET ….amazonaws.com / 429 application/xml 398b 35ms
21:51:26 HTTPS GET ….amazonaws.com / 429 application/xml 398b 74ms
21:51:32 HTTPS GET ….amazonaws.com / 429 application/xml 398b 197ms
21:51:53 HTTPS GET ….amazonaws.com / 429 application/xml 398b 121ms
21:51:54 HTTPS GET ….amazonaws.com / 429 application/xml 398b 55ms
21:51:56 HTTPS GET ….amazonaws.com / 429 application/xml 398b 124ms
21:51:57 HTTPS GET ….amazonaws.com / 429 application/xml 398b 83ms
21:52:03 HTTPS GET ….amazonaws.com / err …sed connection
21:52:29 HTTPS GET ….amazonaws.com / 429 application/xml 398b 248ms
21:52:30 HTTPS GET ….amazonaws.com / 429 application/xml 398b 42ms
21:52:31 HTTPS GET ….amazonaws.com / 429 application/xml 398b 37ms
21:52:34 HTTPS GET ….amazonaws.com / 429 application/xml 398b 210ms
21:52:36 HTTPS GET ….amazonaws.com / 429 application/xml 398b 37ms
[1/25] [scripts:1] [*:8080]
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 z Clear list – Layout

```python
import boto3
from rich import print
import time


def print_event(event_name: str, attempts: int, operation,
response, request_dict, **_):
    print(
        event_name,
        operation,
        attempts,
        response[1]["ResponseMetadata"]["HTTPStatusCode"],
        request_dict["context"]["retries"],
    )


s3 = boto3.client("s3")
s3.meta.events.register("needs-retry.s3.ListBuckets",
print_event)
s3.list_buckets()
```

Now we have a way to intercept let's start logging out the bits we might be interested in!


mitmproxy -s example ~/s/g/m/exploring-boto3-events-with-mitmproxy
Flows
>>21:30:03 HTTPS GET ….amazonaws.com / 429 application/xml 398b 325ms
21:30:04 HTTPS GET ….amazonaws.com / 429 application/xml 398b 42ms
21:30:05 HTTPS GET ….amazonaws.com / 429 application/xml 398b 53ms
21:30:09 HTTPS GET ….amazonaws.com / 429 application/xml 398b 61ms
21:30:12 HTTPS GET ….amazonaws.com / 429 application/xml 398b 37ms
[1/5] [scripts:1] [*:8080]
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 z Clear list - Layout



And here you can see the output

Note the attempt count, it always starts on 1 and retries start on 2.

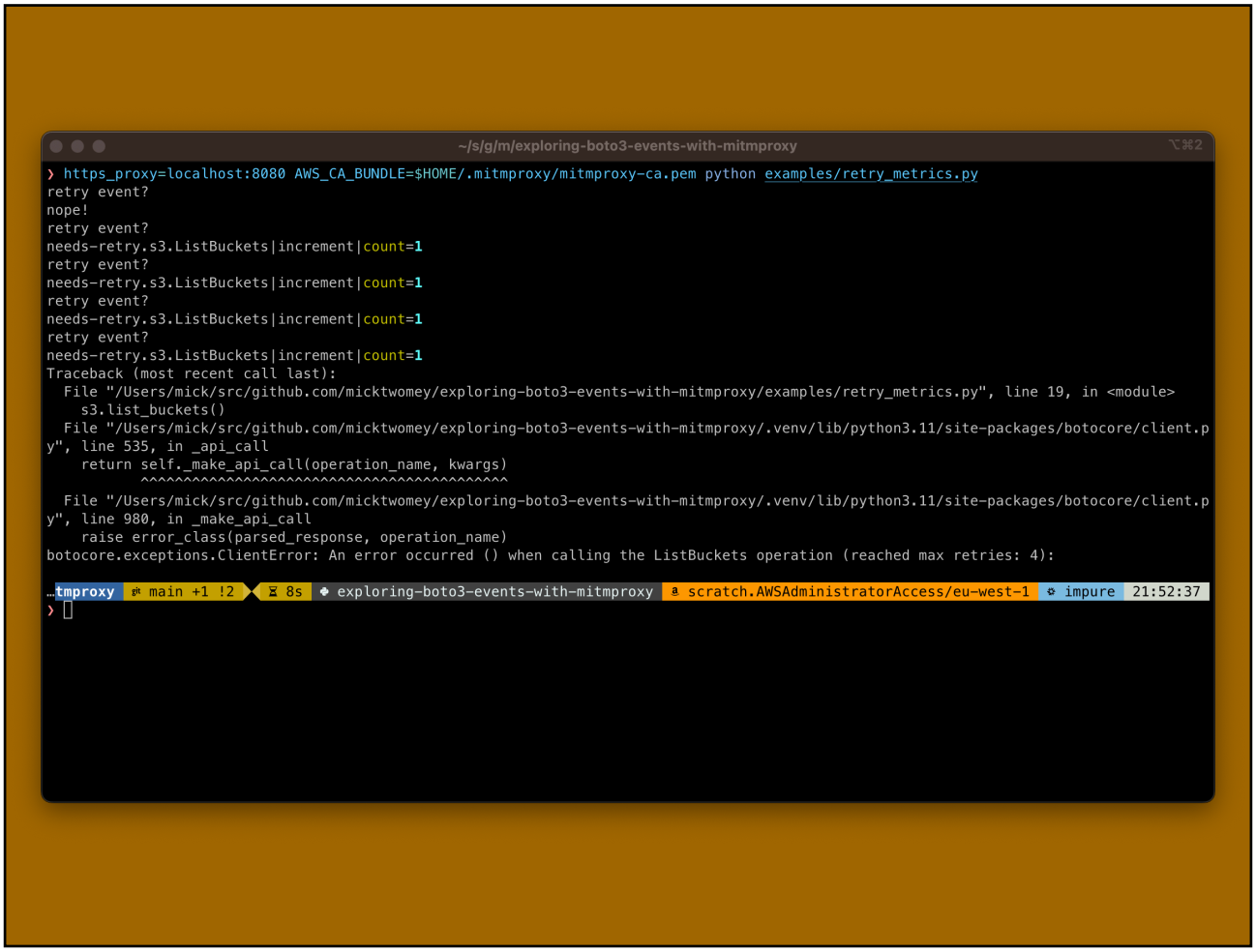
This is counter to my original intuition, I expected 0.

So we now know to only pay attention to the count when > 1

Will this ever end?

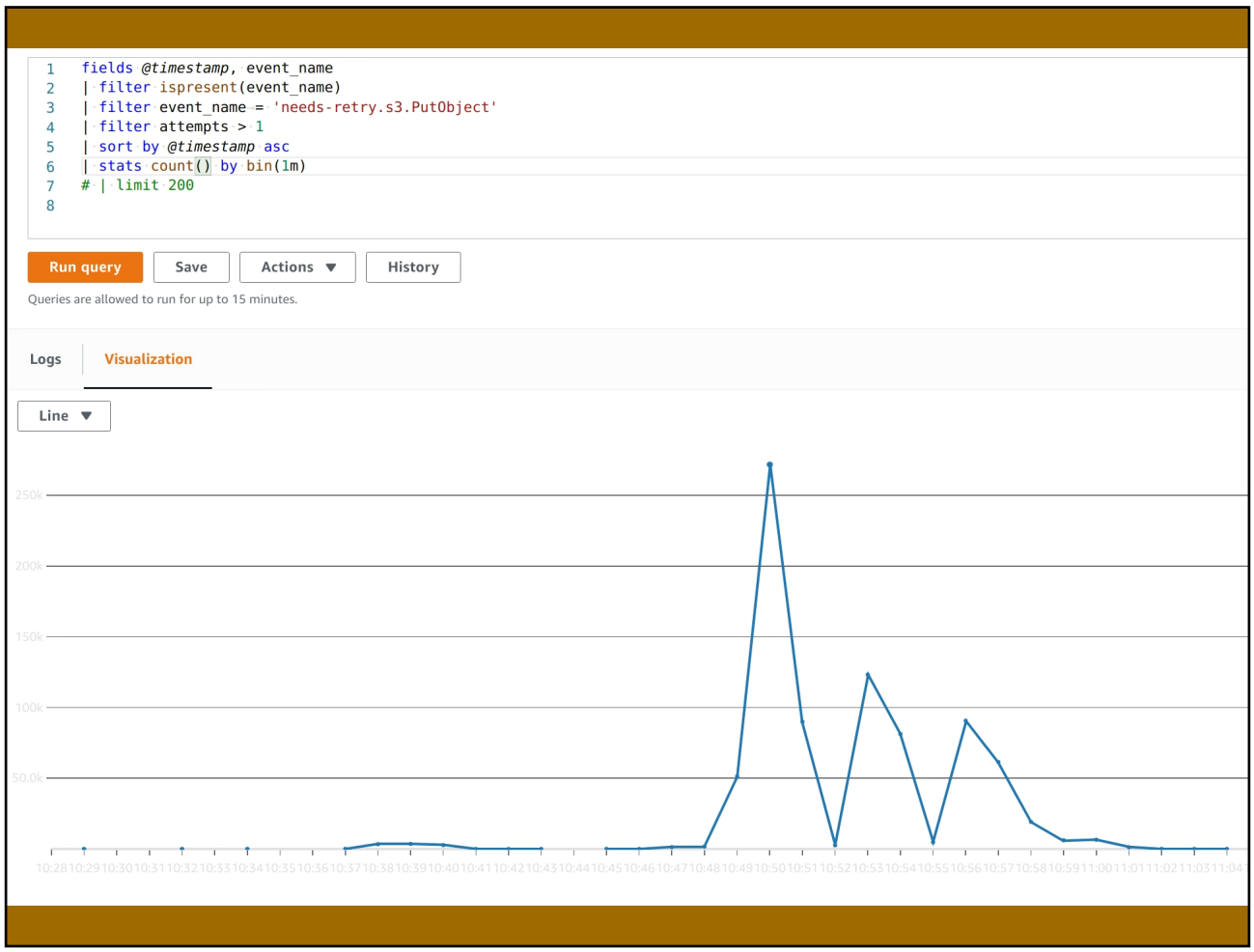
1. Got some jobs taking a long time
2. Guessed it's retries causing this
3. Want to use events to monitor these but don't have docs
4. Want to somehow modify responses to simulate retries and see what happens
5. Can intercept requests using mitmproxy
6. Now know the shape of the data we get in 429 responses
7. ??

```python
import boto3
from rich import print


def increment_metric(name):
    print(f"{name}|increment|count=1")


def handle_retry(event_name: str, attempts: int, **_):
    print("retry event?")
    if attempts > 1:
        increment_metric(event_name)
    else:
        print("nope!")


s3 = boto3.client("s3")
s3.meta.events.register("needs-retry.s3.*", handle_retry)
s3.list_buckets()
print("All done!")
```



Note that we're retrying until the exception is raised, but remember that in production we usually didn't get an error, just slow downs.

Now we can graph these and see how bad things are!



The graph shows over 250K retry attempts at the peak!

It also shows some kind of oscillation, possibly due to so many connections sleeping at the same time.

Each horizontal tick mark is a minute

With some refinements we can now hone in on where this is happening

Finally!

1. Got some jobs taking a long time
2. Guessed it's retries causing this
3. Want to use events to monitor these but don't have docs
4. Want to somehow modify responses to simulate retries and see what happens
5. Can intercept requests using mitmproxy
6. Now know the shape of the data we get in 429 responses
7. Can emit metrics and graph them

What We Covered

boto3's event system

mitmproxy

AWS API limits

How request retries behave

That nothing is ever simple!

Thank You! 🎉

🦣 mastodon.ie/@micktwomey

👨🏽‍💼 fourtheorem.com

github.com/micktwomey/exploring-boto3-events-with-mitmproxy