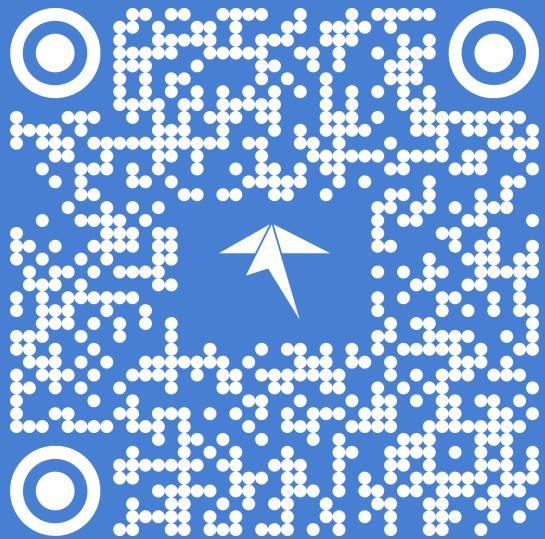


Exploring boto3 Events With Mitmproxy

Michael Twomey



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!

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!

AWS S3 

boto3 

HTTP 

Python 

TLS 

mitmproxy



I'll be talking about a few different things:

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

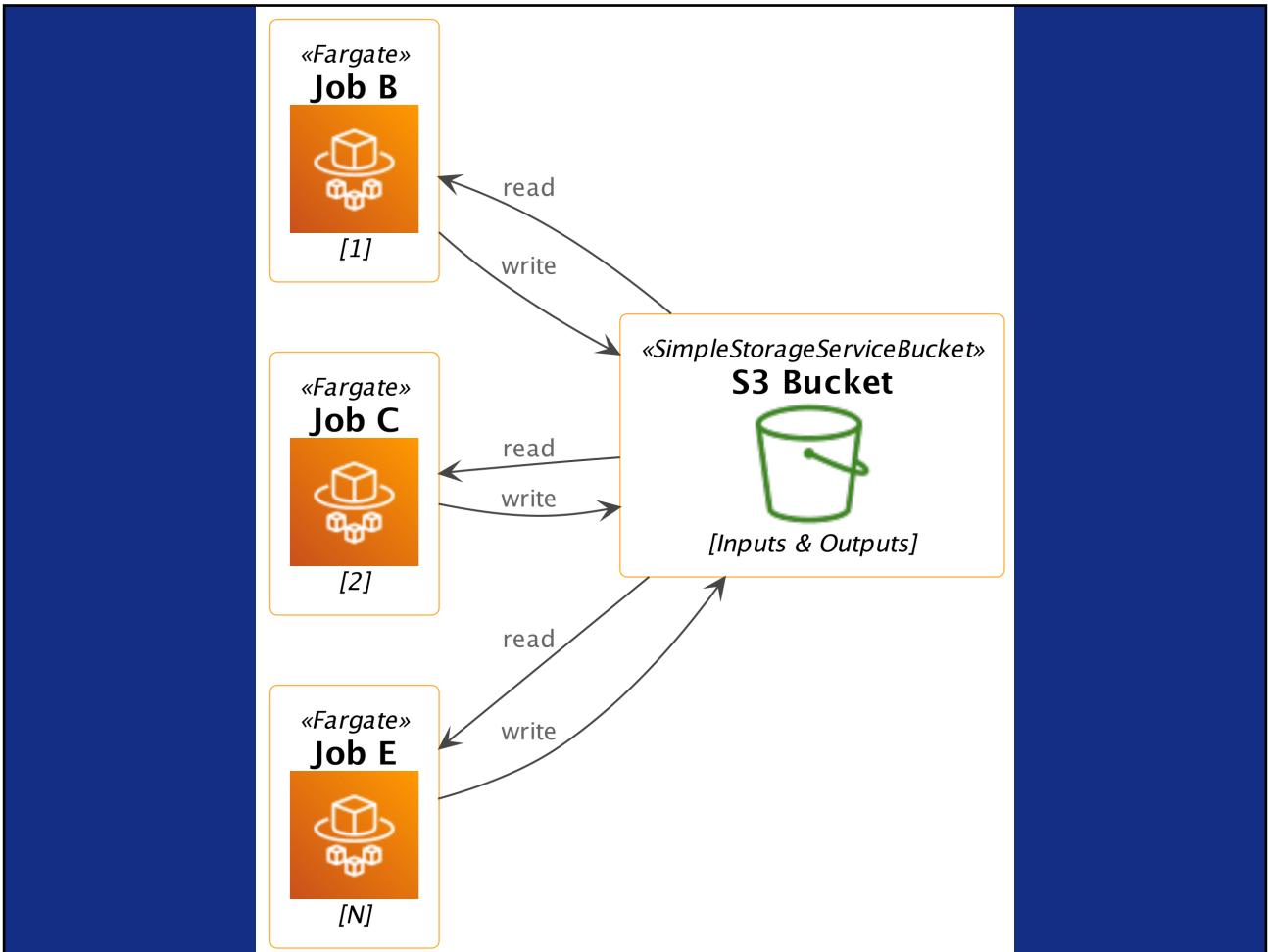
Some boto3, the python AWS client SDK
A little bit about HTTP and TLS

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

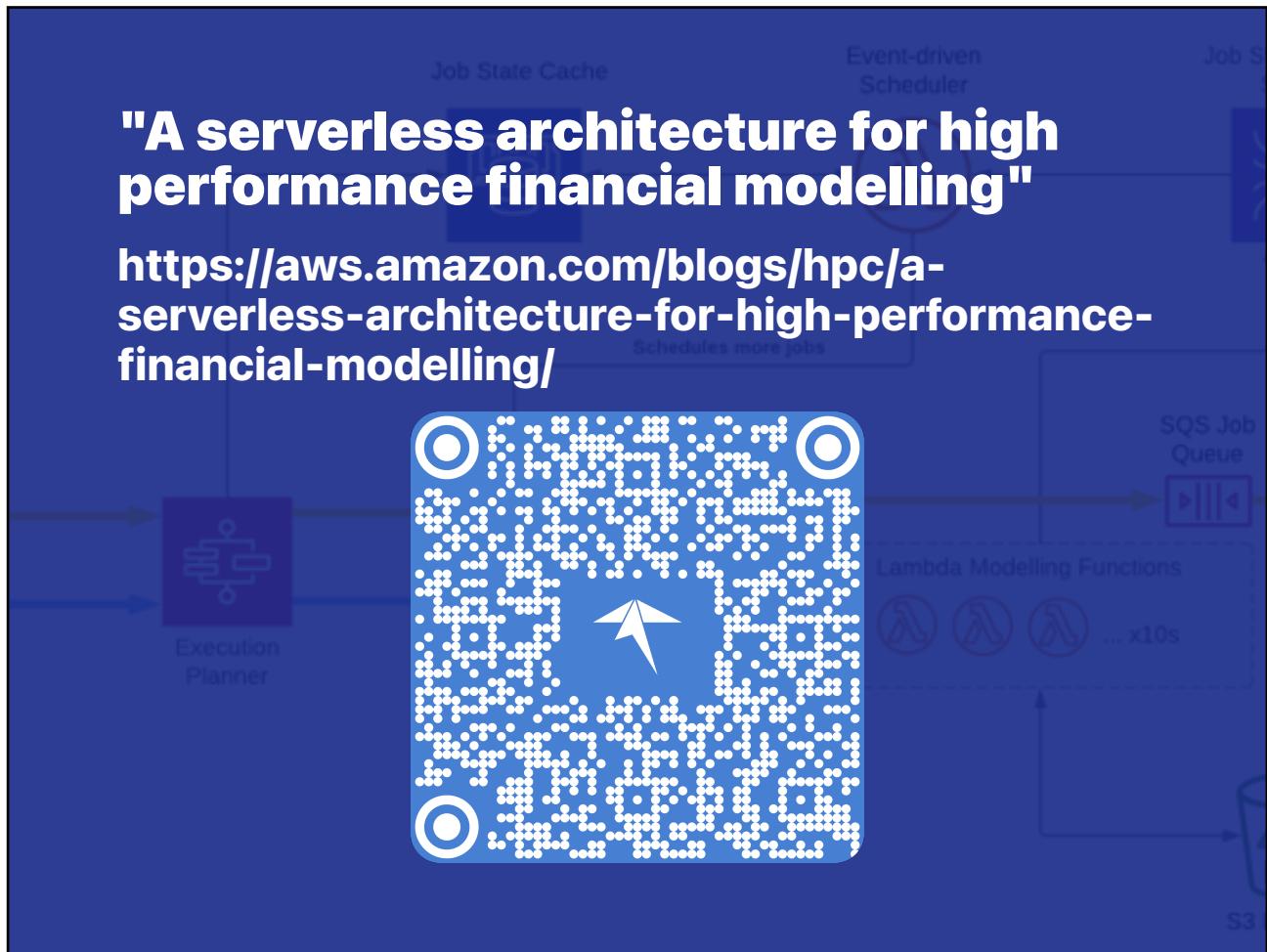
The Setup

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

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



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/>

The Problem:

We got one
of the
dreaded
questions

*"Why is my compute
job so slow?"*

We got a question.

One of the questions you
dread when operating a system

During very large job runs we
would occasionally see
inexplicable slow downs.

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
```

Finally we saw the occasional too many requests error which pointed us in the right direction.

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

The Rate Limit Errors
prompted the question:

**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`

If we were hitting this rate limit so much why didn't we see more errors?

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

boto3 retries requests automatically

```
import boto3

s3 = boto3.client("s3")
# Implicitly retries, sleeping
each time:
response = s3.get_object(...)
```

**Default: 5 attempts
(4 retries)**

Up to 30 seconds sleeping

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

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

Theory

Even though we mostly eventually succeed, we take a really long time doing stuff because of all the retries and sleeps

How Can We Be Sure it's Retrying?

Logging?

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

Could use logging at DEBUG level

```
2023-11-07 12:05:04 DEBUG botocore.hooks Event before-parameter-build.s3.GetObject: calling handler <function sse_md5 at 0x110d3f920>
2023-11-07 12:05:04 DEBUG botocore.hooks Event before-parameter-build.s3.GetObject: calling handler <function validate_bucket_name at 0x110d3f880>
2023-11-07 12:05:04 DEBUG botocore.hooks Event before-parameter-build.s3.GetObject: calling handler <function remove_bucket_from_uri_paths from model at 0x110d5d9e0>
2023-11-07 12:05:04 DEBUG botocore.hooks Event before-parameter-build.s3.GetObject: calling handler <bound method S3RegionRedirectorv2.annotate_request_context of <botocore.utils.S3RegionRedirectorv2 object at 0x114319fd0>>
2023-11-07 12:05:04 DEBUG botocore.hooks Event before-parameter-build.s3.GetObject: calling handler <function generate_idempotent_uuid at 0x110d3f6a0>
2023-11-07 12:05:04 DEBUG botocore.hooks Event before-endpoint-resolution.s3: calling handler <function customize_endpoint_resolver_builtins at 0x110d5dbc0>
2023-11-07 12:05:04 DEBUG botocore.hooks Event before-endpoint-resolution.s3: calling handler <bound method S3RegionRedirectorv2.redirect from cache of <botocore.utils.S3RegionRedirectorv2 object at 0x114319fd0>>
2023-11-07 12:05:04 DEBUG botocore.regions Calling endpoint provider with parameters: {'Bucket': 'micktwomey-scratch-example', 'Region': 'eu-west-1', 'UsePIPS': False, 'UseDualStack': False, 'ForcePathStyle': False, 'Accelerate': False, 'UseGlobalEndpoint': False, 'DisableMultiRegionAccessPoints': False, 'UseArnRegion': True}
2023-11-07 12:05:04 DEBUG botocore.regions Endpoint provider result: https://micktwomey-scratch-example.s3.eu-west-1.amazonaws.com
2023-11-07 12:05:04 DEBUG botocore.regions Selecting from endpoint provider's list of auth schemes: "sigv4". User selected auth scheme is: "None"
2023-11-07 12:05:04 DEBUG botocore.regions Selected auth type "v4" as "v4" with signing context params: {'region': 'eu-west-1', 'signing_name': 's3', 'disableDoubleEncoding': True}
2023-11-07 12:05:04 DEBUG botocore.hooks Event before-call.s3.GetObject: calling handler <function add_expect_header at 0x110d3fc40>
2023-11-07 12:05:04 DEBUG botocore.hooks Event before-call.s3.GetObject: calling handler <function add_recursion_detection_header at 0x110d3df80>
2023-11-07 12:05:04 DEBUG botocore.hooks Event before-call.s3.GetObject: calling handler <function inject_api_version_header_if_needed at 0x110d5d1c0>
2023-11-07 12:05:04 DEBUG botocore.endpoint Making request for OperationModel(name=GetObject) with params: {'url_path': '/test/prefix1/ham/spam/foo.txt', 'query_string': (), 'method': 'GET', 'headers': {'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#Cython cfg/retry-mode#legacy Botcore/1.31.79'}, 'body': b'', 'auth_path': '/micktwomey-scratch-example/test/prefix1/ham/spam/foo.txt', 'context': {'https://micktwomey-scratch-example.s3.eu-west-1.amazonaws.com/test/prefix1/ham/spam/foo.txt': 'https://micktwomey-scratch-example.s3.eu-west-1.amazonaws.com/test/prefix1/ham/spam/foo.txt', 'client_region': 'eu-west-1', 'client_config': <botocore.config.Config object at 0x114319550>, 'has_streaming_input': False, 'auth_type': 'v4', 's3_redirect': {'redirected': False, 'bucket': 'micktwomey-scratch-example', 'params': {'Bucket': 'micktwomey-scratch-example', 'Key': 'test/prefix1/ham/spam/foo.txt'}, 'signing': {'region': 'eu-west-1', 'signing_name': 's3', 'disableDoubleEncoding': True}}}}
2023-11-07 12:05:04 DEBUG botocore.hooks Event request-created.s3.GetObject: calling handler <bound method RequestSigner.handler of <botocore.signers.RequestSigner object at 0x114319510>>
2023-11-07 12:05:04 DEBUG botocore.hooks Event choose-signer.s3.GetObject: calling handler <bound method ClientCreator.default_s3_presign_to_sigv2 of <botocore.client.ClientCreator object at 0x110e138d0>>
2023-11-07 12:05:04 DEBUG botocore.hooks Event choose-signer.s3.GetObject: calling handler <function set_operation_specific_signer at 0x110d3f560>
2023-11-07 12:05:04 DEBUG botocore.hooks Event before-sign.s3.GetObject: calling handler <function remove_arn_from_signing_path at 0x110d5d620>
2023-11-07 12:05:04 DEBUG botocore.auth Calculating signature using v4 auth.
2023-11-07 12:05:04 DEBUG botocore.auth CanonicalRequest:
GET
/test/prefix1/ham/spam/foo.txt

host:micktwomey-scratch-example.s3.eu-west-1.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:2023107120504Z
x-amz-security-token:...

host;x-amz-content-sha256;x-amz-date;x-amz-security-token
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
2023-11-07 12:05:04 DEBUG botocore.auth StringToSign:
```

DEBUG logging in boto3 hugely verbose, unusable 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 are an extension mechanism for boto3

boto3 event docs over at

<https://boto3.amazonaws.com/v1/documentation/api/latest/guide/events.html>

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.

```
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

```
S3:  
provide-client-params.s3.ListBuckets  
before-parameter-build.s3.ListBuckets  
before-call.s3.ListBuckets  
request-created.s3.ListBuckets  
choose-signer.s3.ListBuckets  
before-sign.s3.ListBuckets  
before-send.s3.ListBuckets  
response-received.s3.ListBuckets  
needs-retry.s3.ListBuckets  
after-call.s3.ListBuckets  
  
EC2:  
provide-client-params.ec2.DescribeInstances  
before-parameter-build.ec2.DescribeInstances  
before-call.ec2.DescribeInstances  
request-created.ec2.DescribeInstances  
choose-signer.ec2.DescribeInstances  
before-sign.ec2.DescribeInstances  
before-send.ec2.DescribeInstances  
response-received.ec2.DescribeInstances  
needs-retry.ec2.DescribeInstances  
after-call.ec2.DescribeInstances
```

Here you can see all the events which get triggered

```
import boto3

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!

```
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.
Does that attempts count go up?

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

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

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

1. Triggering the rate limit for real



2. Hacking the library



3. Hacking Python



4. Hacking the OS



5. Hacking the network



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

**Bet: boto3
looks at the
HTTP response
code**

**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

From Success



**ERROR
FREE
REQUESTS**

```
HTTP/1.1 200 OK ➡  
Content-Length: 34202  
Content-Type: application/json  
...  
{  
    ...  
}
```



**BROKEN
RATE LIMITS**

imgflip.com

To Failure! 😞

```
HTTP/1.1 429 Rate limit exceeded ➡  
Content-Length: 34202  
Content-Type: application/json;  
...  
{  
    ...  
}
```

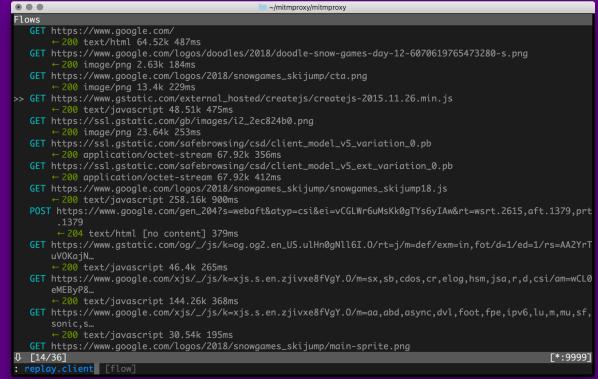
We're only changing one part of the response, so should be easy to implement.

Right?

How: mitmproxy

<https://mitmproxy.org>

mitmproxy is a free
and open source
interactive HTTPS
proxy.



The screenshot shows the mitmproxy interface with a list of network flows. The flows are listed in a table with columns for method, URL, status code, file size, and duration. The table includes rows for various Google services like Google.com, Google Logos, and Google Translate, along with some JavaScript files and images. The interface has a dark theme with light-colored text and icons.

Flows
GET https://www.google.com/
→ 200 text/html 64.52k 48ms
GET https://www.google.com/doodles/2018/doodle-snow-games-day-12-6070619765473280-s.png
→ 200 image/png 2.63k 184ms
GET https://www.google.com/logos/2018/snowgames_skijump_cto.png
>> GET https://www.gstatic.com/external_hosted/createjs/createjs-2015.11.26.min.js
→ 200 text/javascript 49.48k 475ms
GET https://ssl.gstatic.com/images/iz_2_ec824b0.png
→ 200 image/png 23.64k 253ms
GET https://ssl.gstatic.com/safebrowsing/cs/client_model_v5_variation_0.pb
→ 200 application/octet-stream 67.92k 356ms
GET https://www.gstatic.com/translate/v2/js/client_model_v5_ext_variation_0.pb
→ 200 application/octet-stream 67.92k 41ms
GET https://www.google.com/logos/2018/snowgames_skijump/snowgames_skijump18.js
POST https://www.google.com/gen_2047s=webaff&tct=cs1&kei=vCLWrbUmKK0gYs6yIAw&r=wsrt.2615,aft.1379,prt.151
→ 204 text/html [no content] 379ms
GET https://www.gstatic.com/og/_js/k=og_og2.en_US.u1In@Gn16I.O/r=j/m=def/exm=in,fot/d=1/eid=1/rs=A2YrTuVKqgJN
→ 200 text/javascript 46.4k 265ms
GET https://www.google.com/xjs/_/j/s=k-xjs.s.en.zjivxe8FVgY.O/m=sx,sb,cdos,cr,elog,hsm,jsar,r,d,csi/cm=WCL0ek8yP8
→ 200 text/javascript 144.26k 368ms
GET https://www.google.com/xjs/_/js/k=xjs.s.en.zjivxe8FVgY.O/m=aa,dbd,async,dvl,foot,fpe,ipv6,lu,m,mu,sf,sonic,\$
→ 200 text/javascript 30.54k 195ms
GET https://www.google.com/logos/2018/snowgames_skijump/main-sprite.png

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?

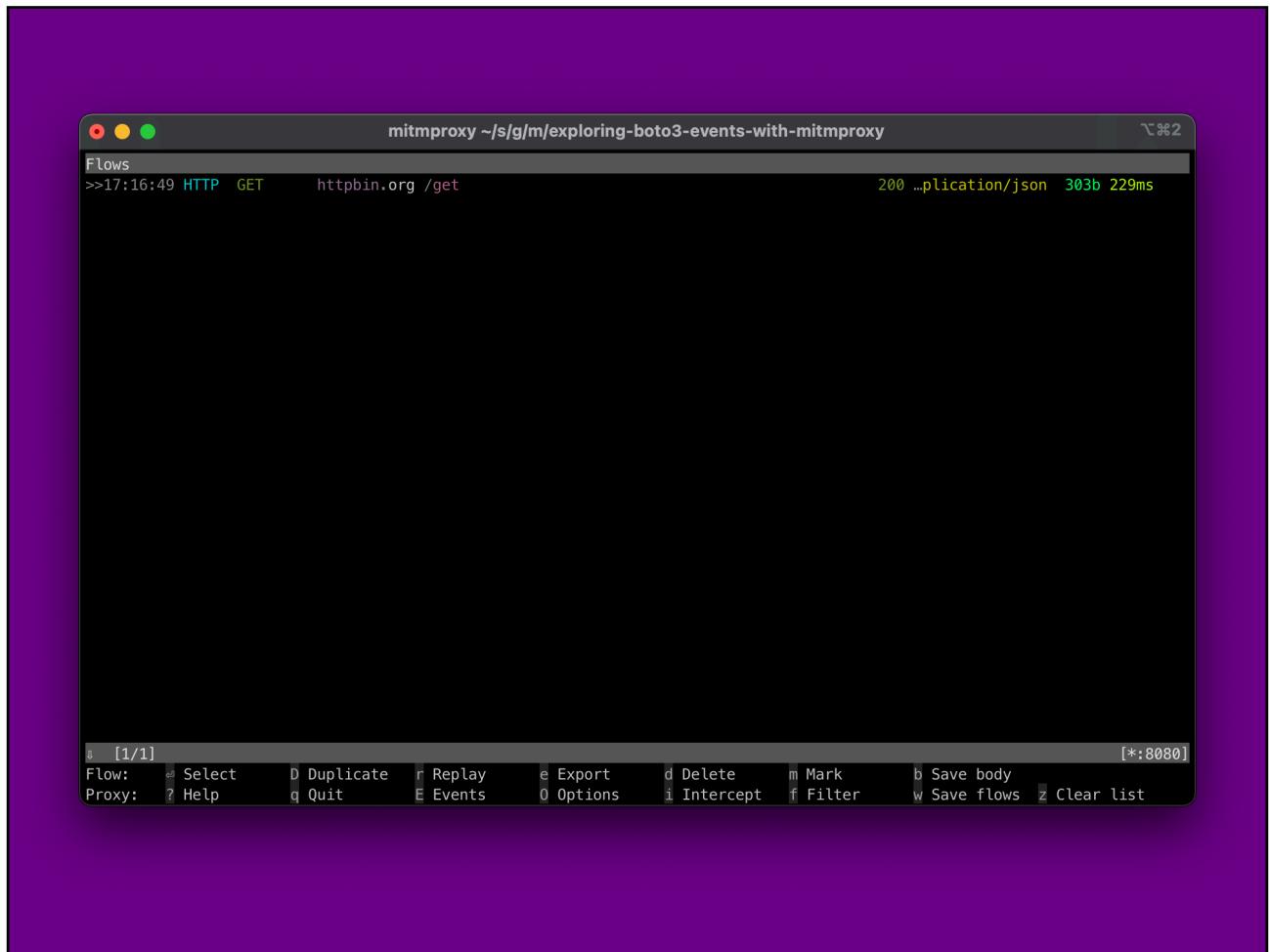
HTTP GET Example

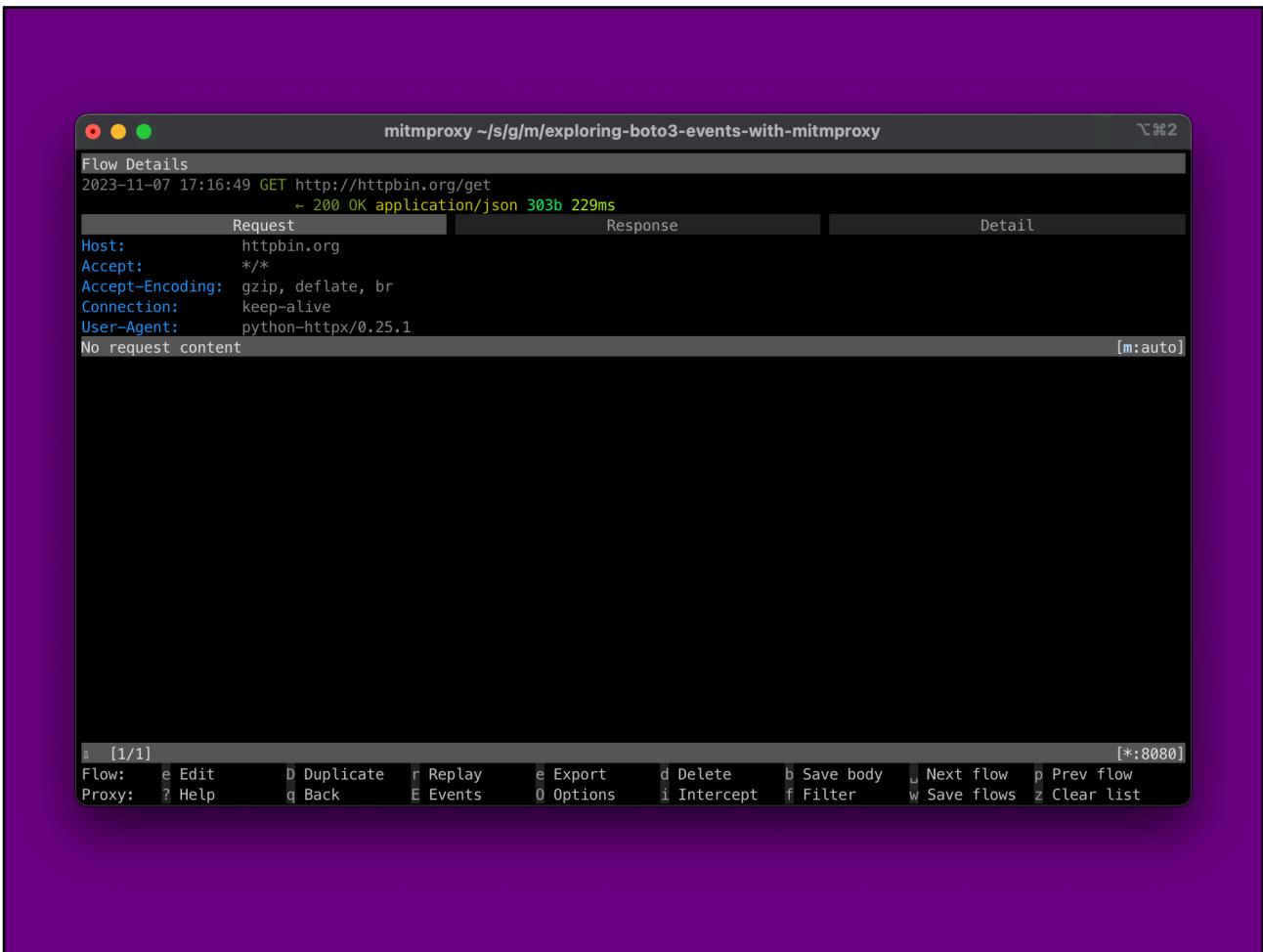
```
import httpx

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

```
export http_proxy=localhost:8080

python http_get.py
```





```
mitmproxy ~s/g/m/exploring-boto3-events-with-mitmproxy
```

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		
ls:		
JSON		[m:auto]
{		
"args": {},		
"headers": {		
"Accept": "*/*",		
"Accept-Encoding": "gzip, deflate, br",		
"Host": "httpbin.org",		
"User-Agent": "python-httpsx/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 n 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 n 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",
)
```

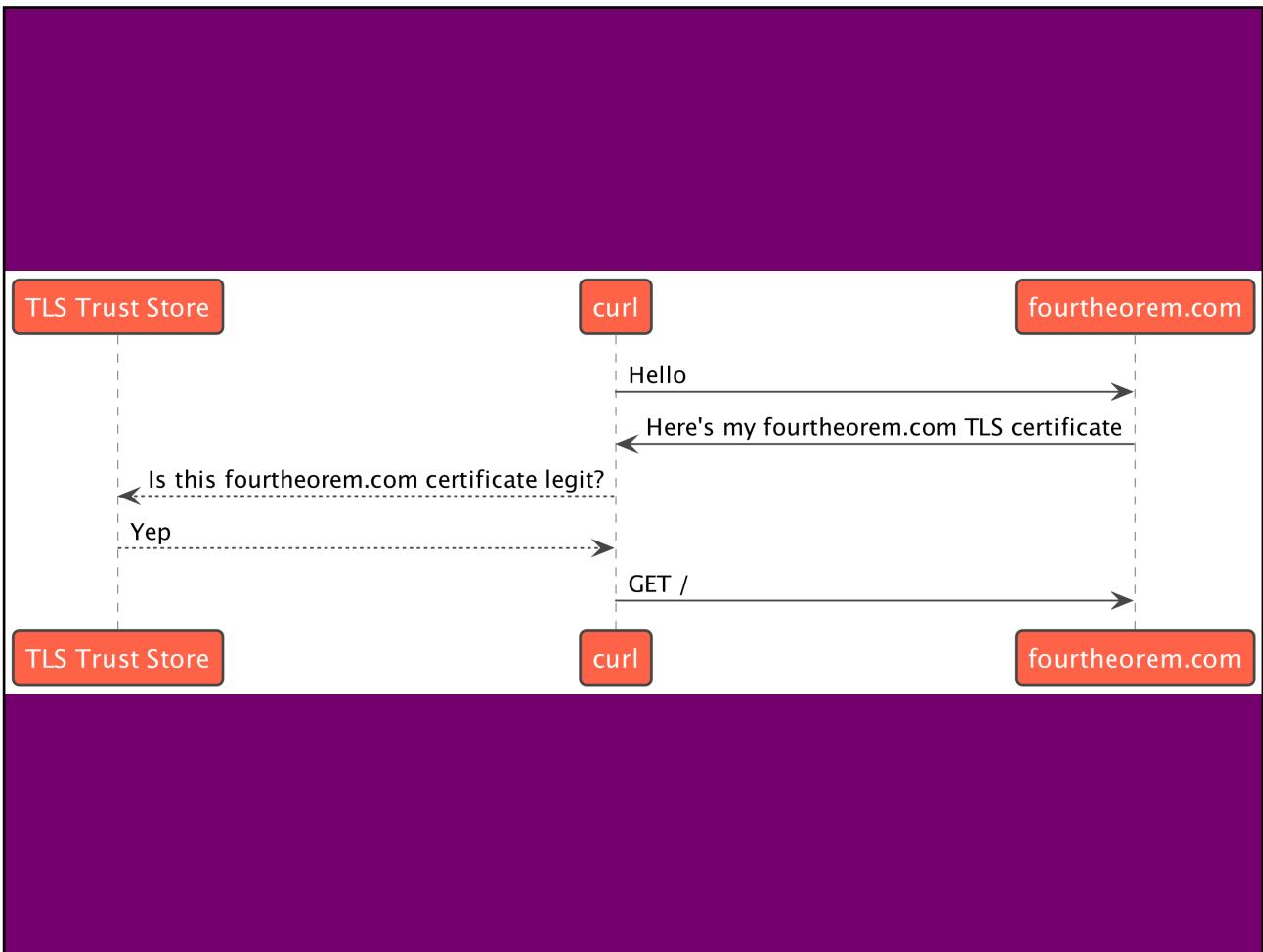
```
export http_proxy=localhost:8080
export https_proxy=localhost:8080
```

```
python s3_get_object.py
```

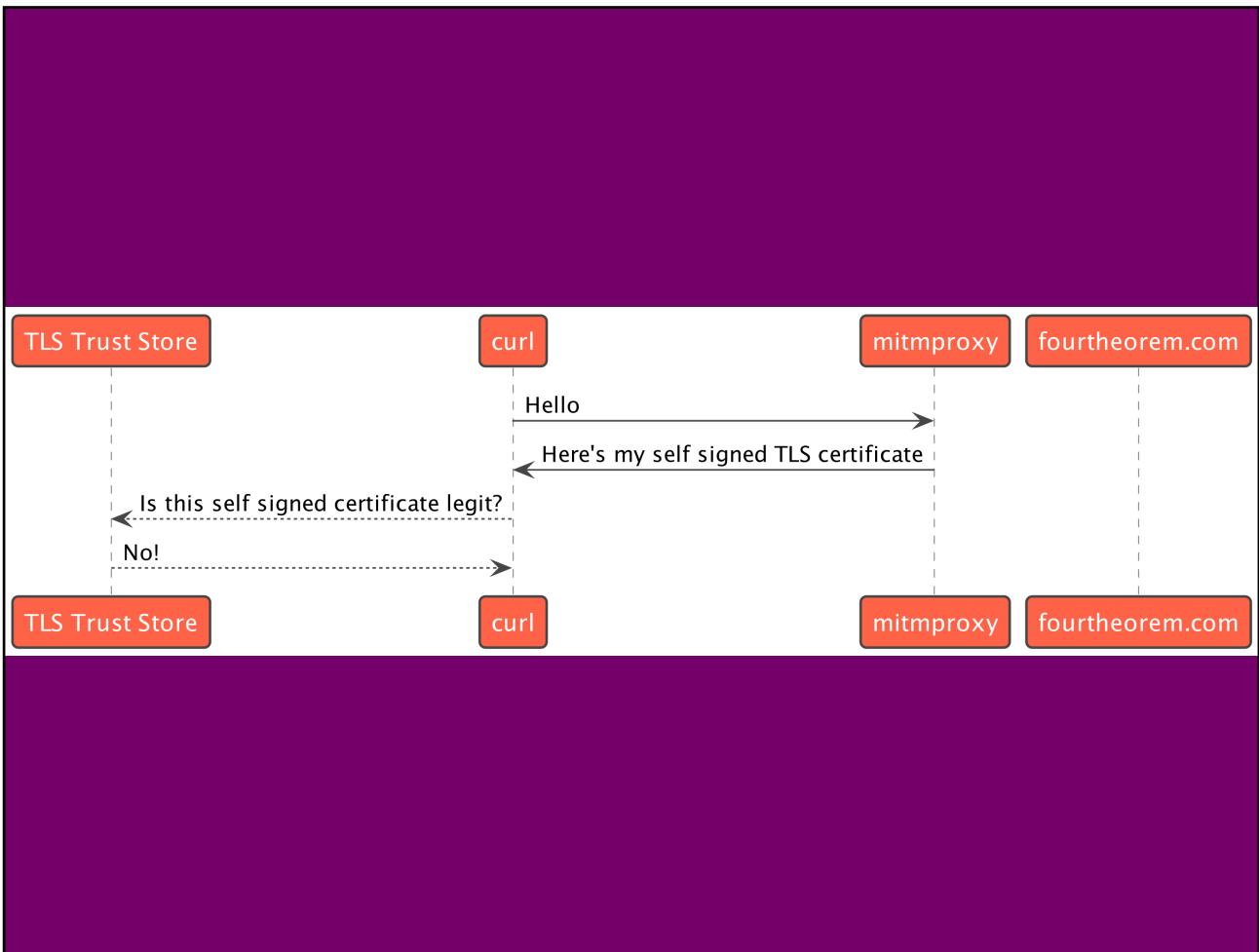
Wut?

```
...
  File "/Users/mick/src/github.com/micktowmey/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/micktowmey/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://micktowmey-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**



1. We connect to a server and get back its certificate
2. We check with the cert store on the OS to see if that cert is ok
3. It's good so we proceed with the connection



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:

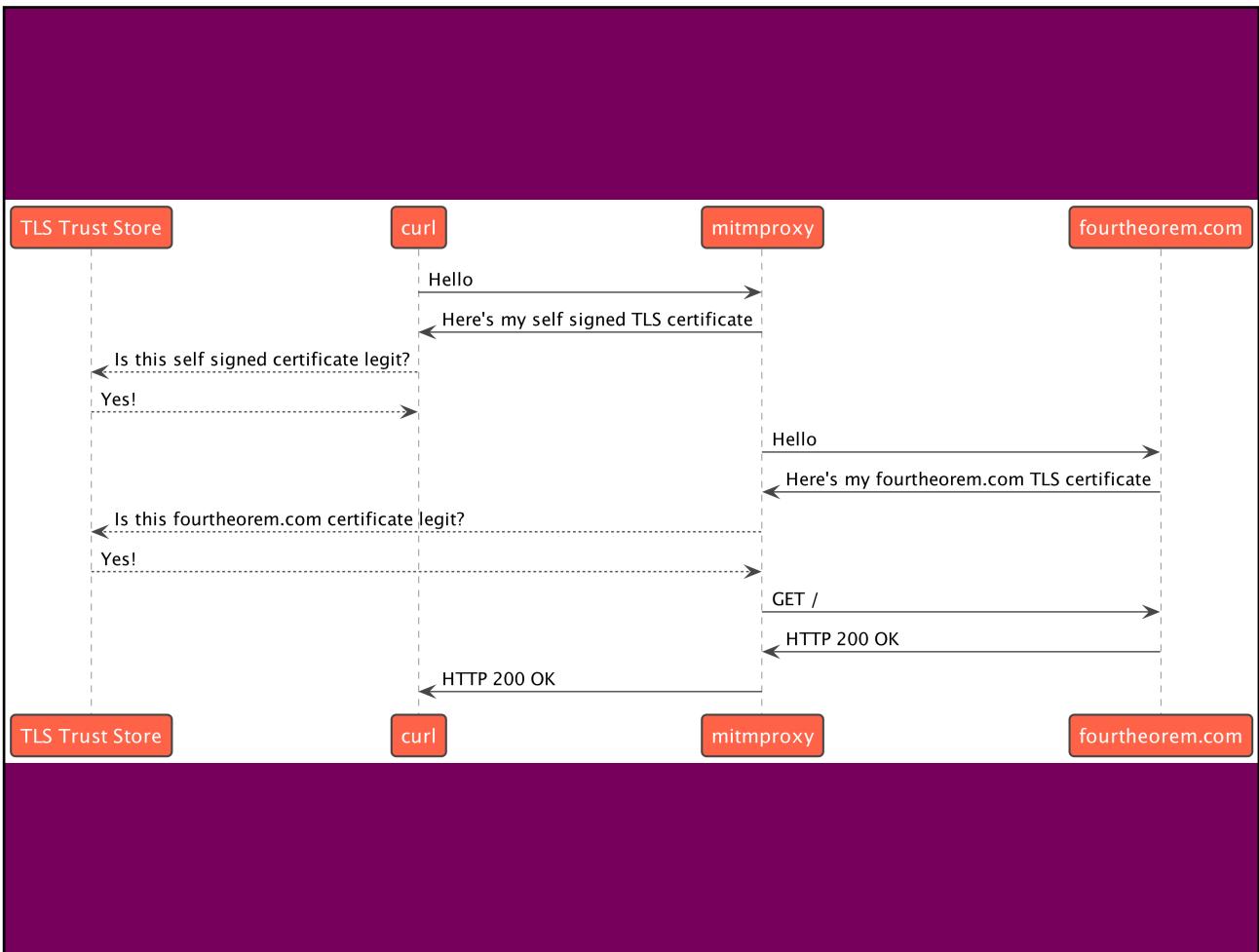
<https://docs.mitmproxy.org/stable/concepts-certificates/>

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



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:

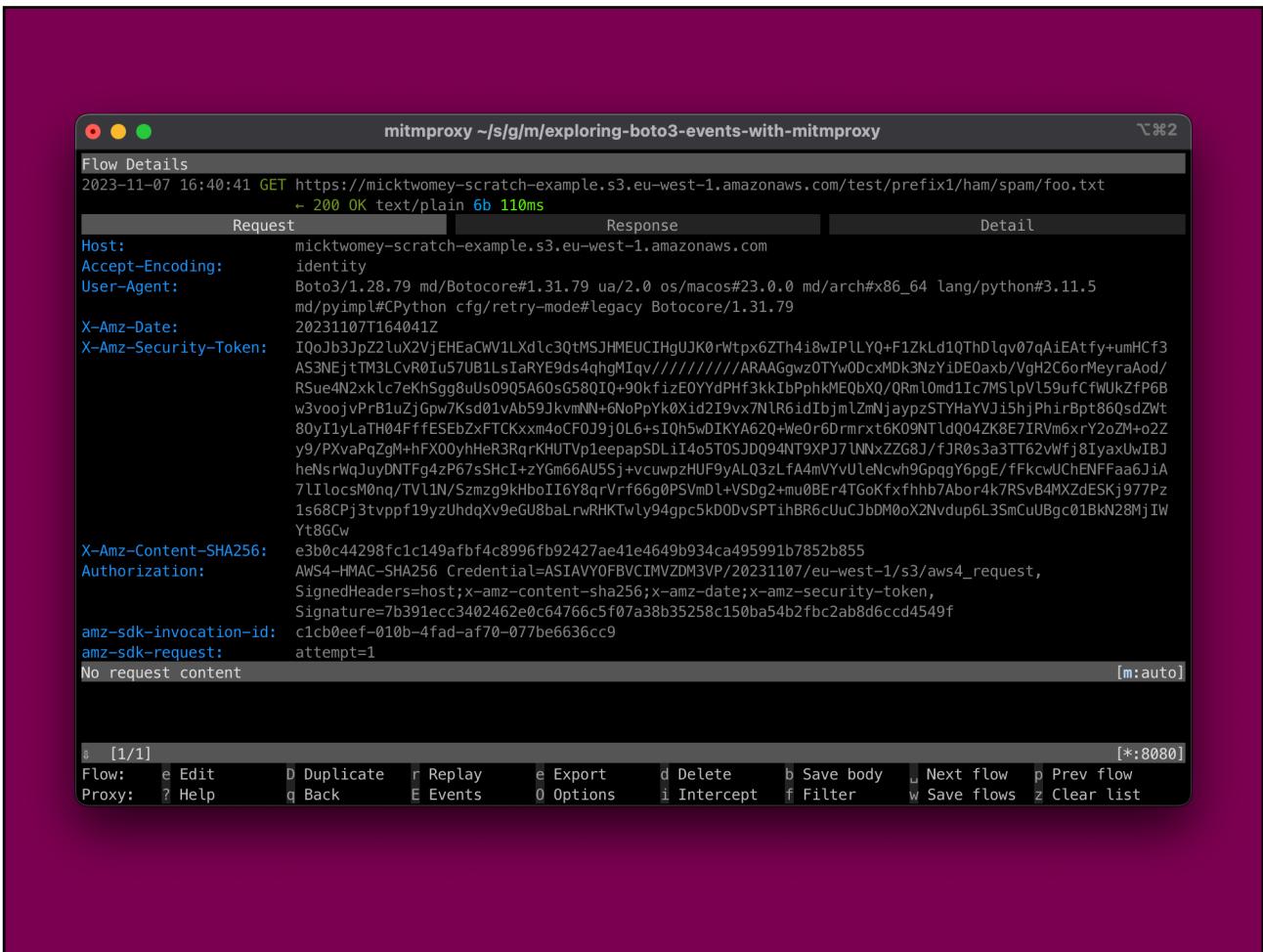
We tell boto3 to use a different cert bundle
(AWS_CA_BUNDLE)

```
mitmproxy ~s/g/m/exploring-boto3-events-with-mitmproxy
```

Flows
>>16:40:41 HTTPS GETamazonaws.com /test/prefix1/ham/spam/foo.txt 200 text/plain 6b 110ms

[*:8080] [1/1]

Flow: **S**elect **D**uplicate **R**Reply **E**xport **D**elete **M**ark **S**ave body
Proxy: **H**elp **Q**uit **E**vents **O**ptions **I**ntercept **F**ilter **S**ave flows **C**lear 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: ZtbuzyAlwmED0MWZcp9I0un7uGin2a0Ukmeo0NeDy13UfxwC/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 n 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
```

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	3497643272326306858947754322096787020	
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	

[*:8080]
[1/1]

Flow: e Edit D Duplicate r Replay e Export d Delete b Save body u 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 u Next flow p Prev flow
Proxy: ? Help q Back E Events O Options i Intercept f Filter w Save flows z Clear list
```

More than a HTTP debugger

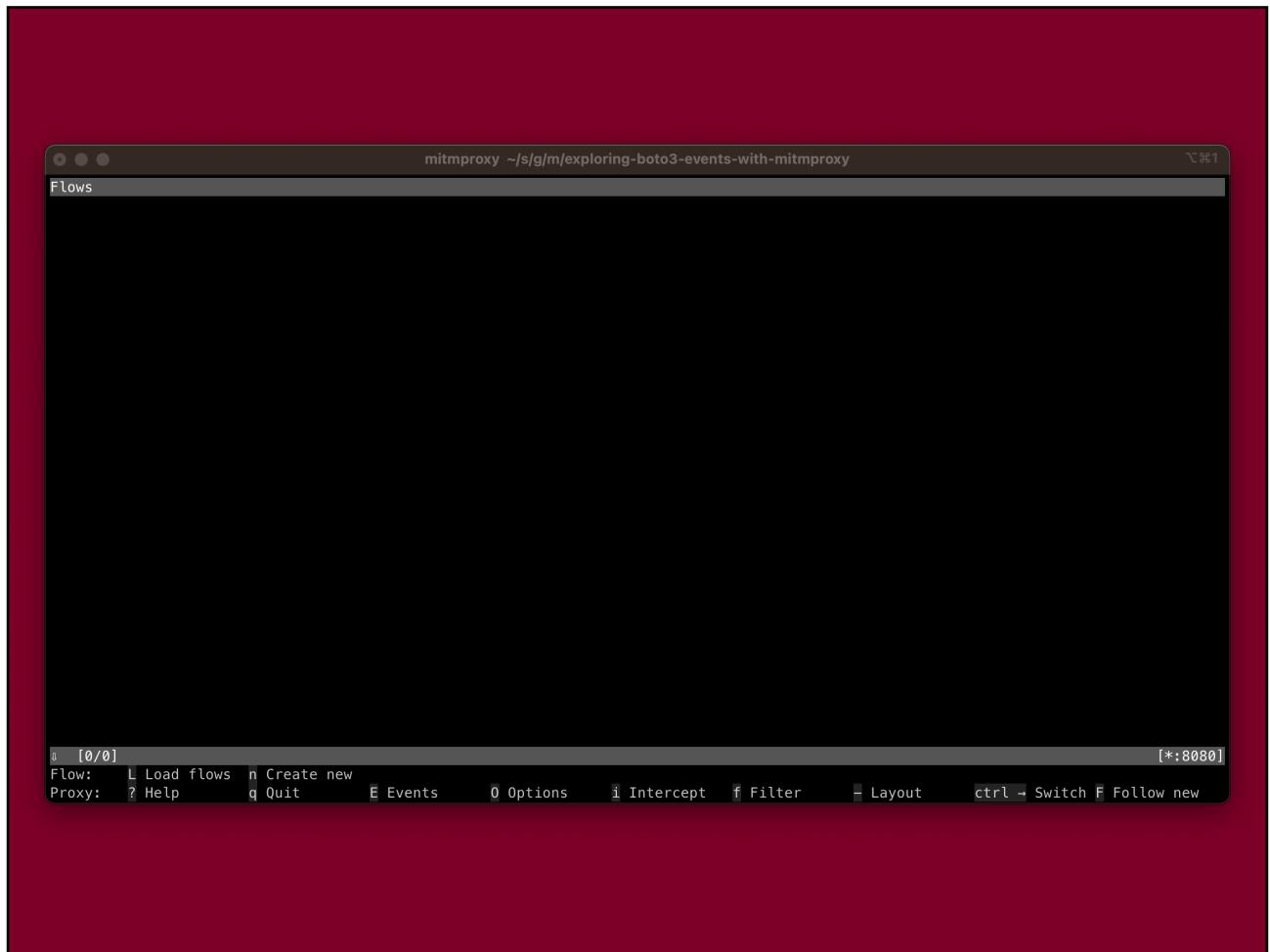
<https://docs.mitmproxy.org/stable/mitmproxytutorial-interceptrequests/>

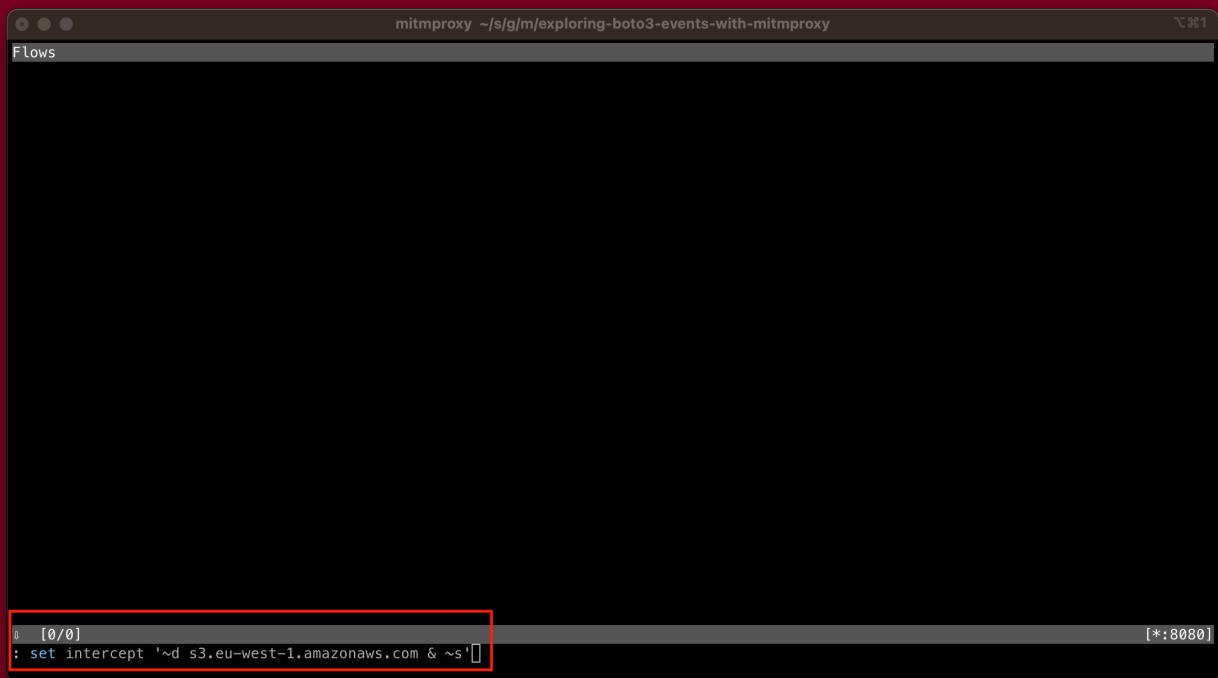
<https://docs.mitmproxy.org/stable/mitmproxytutorial-modifyrequests/>

mitmproxy offers the ability to intercept and change HTTP requests

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

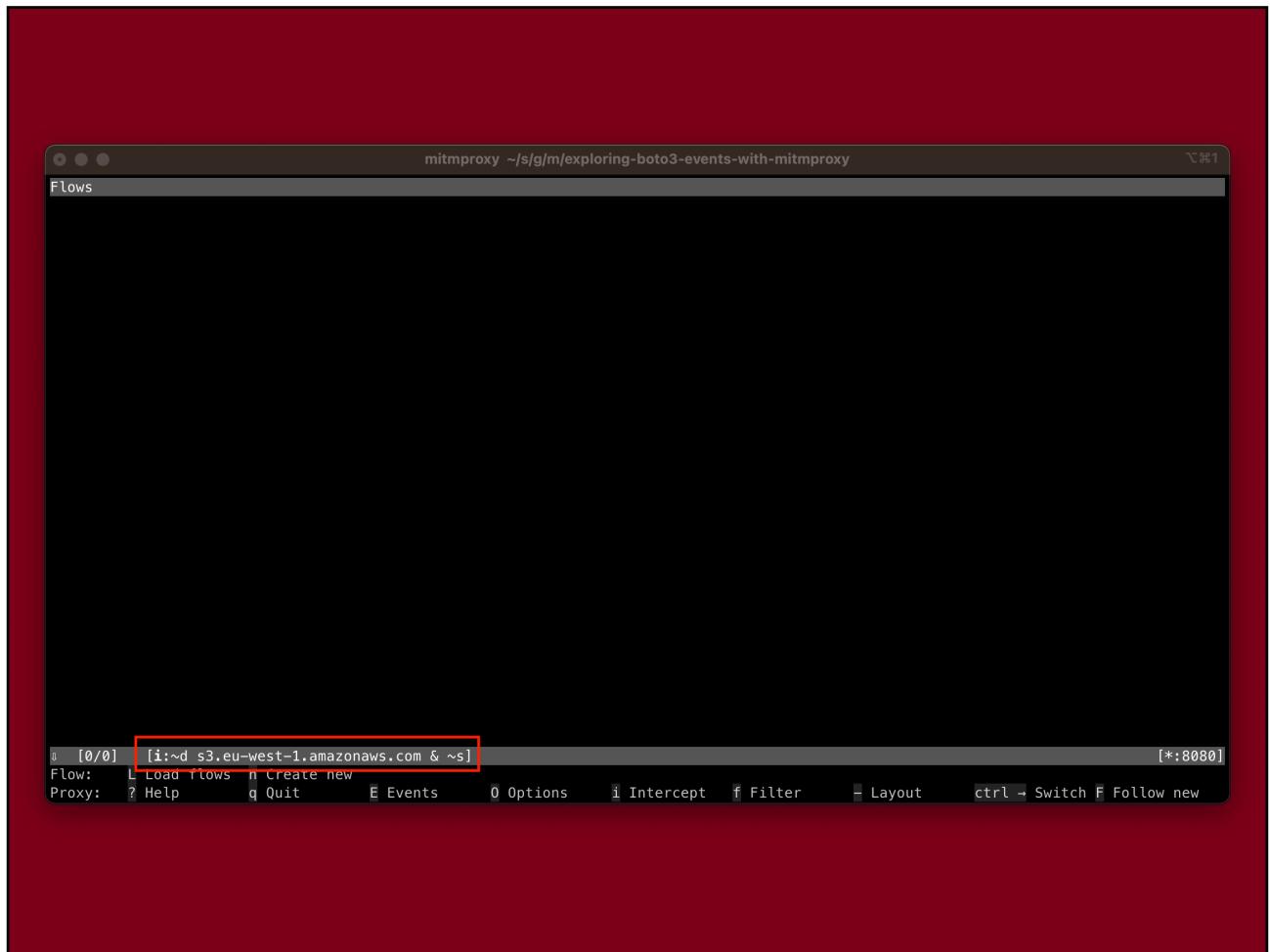
1. Hit `i` to create an intercept
2. `~d s3.eu-west-1.amazonaws.com & ~s`
`~d` match on domain, `~s` match on server response
3. Run the command
4. In the UI go into the response and hit `e`
5. Change the response code to `429`
6. Hit `a` to allow the request to continue
7. Watch what happens in the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

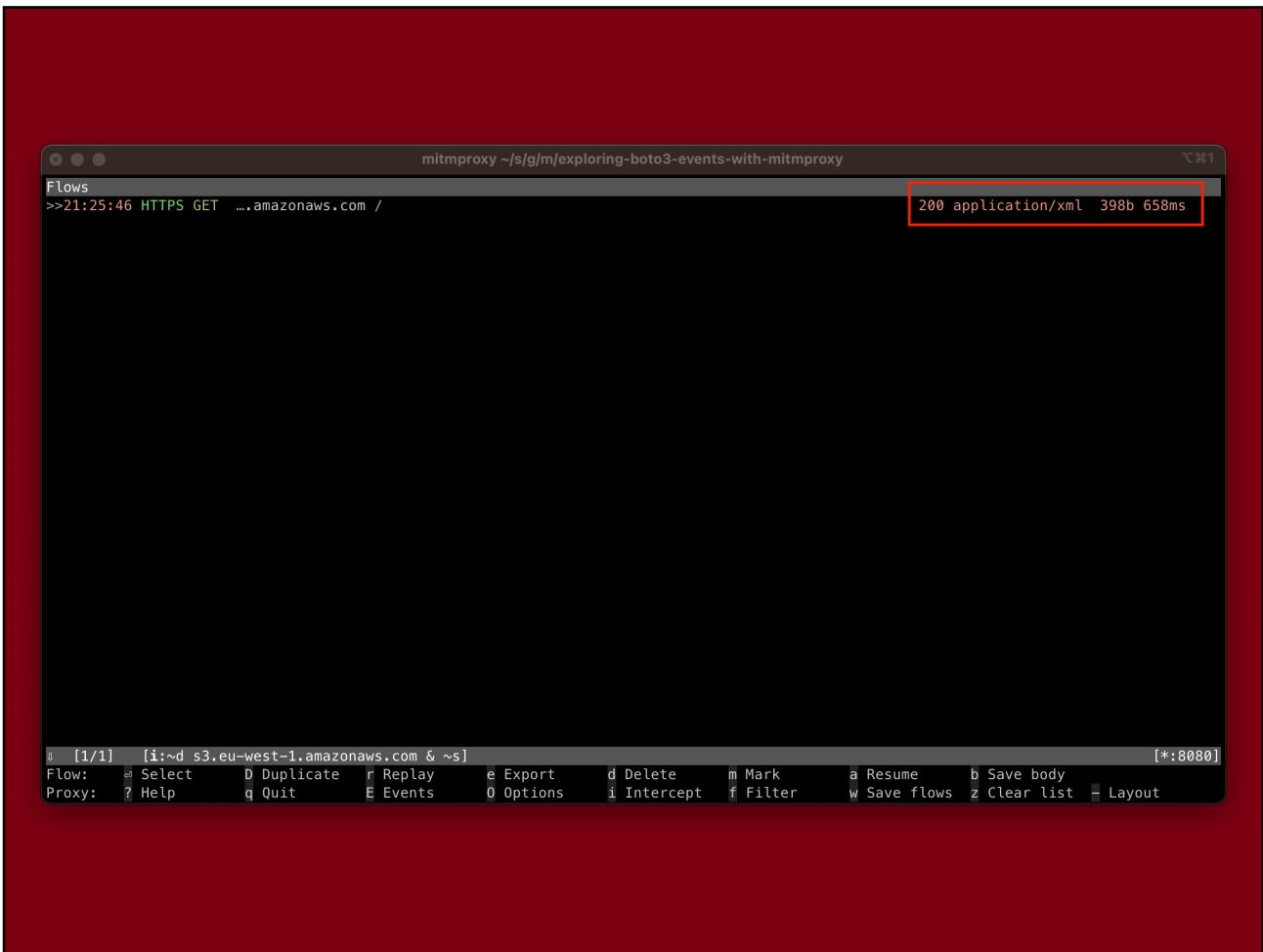




A screenshot of the mitmproxy browser interface. The title bar reads "mitmproxy ~/s/g/m/exploring-boto3-events-with-mitmproxy". The main window is titled "Flows" and is currently empty. At the bottom of the screen, there is a terminal-like interface with a red rectangular highlight around the following command:

```
[0/0] [*:8080]
: set intercept '~d s3.eu-west-1.amazonaws.com & ~s'[]
```





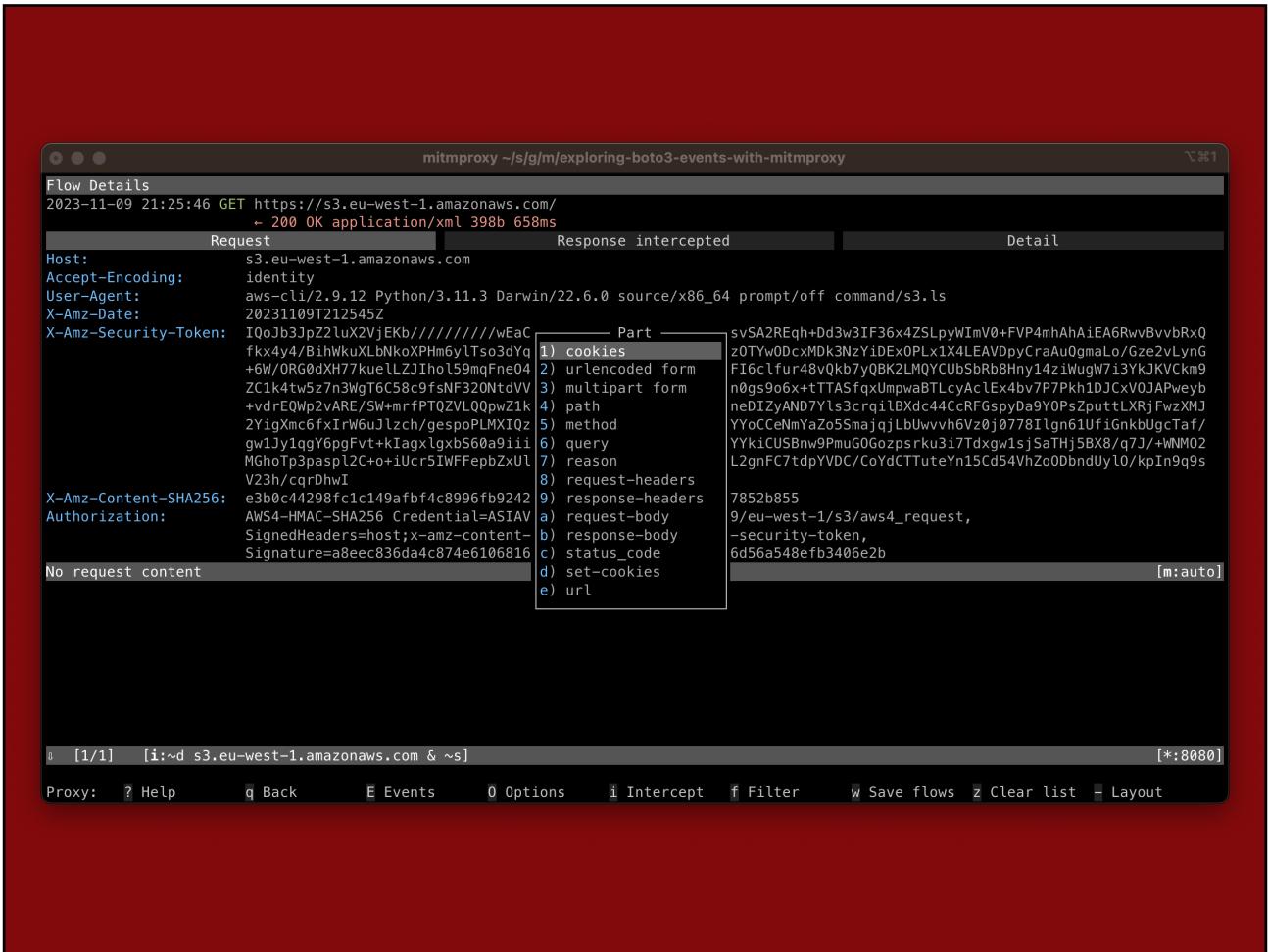
```

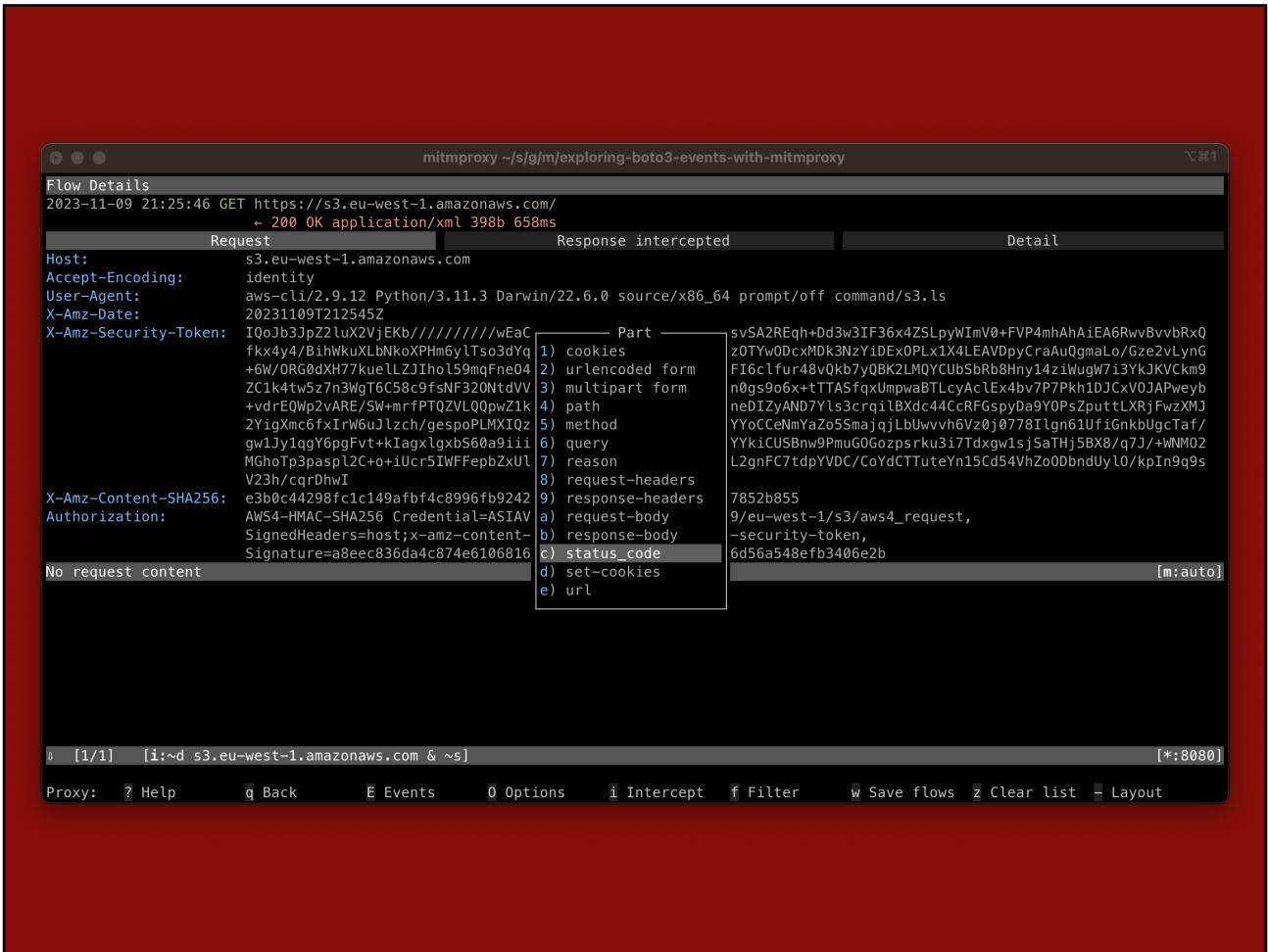
mitmproxy ~ / s / g / m / exploring-boto3-events-with-mitmproxy

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//////////wEAuCWV1LXd1c3QtMSJHMEUCIDSpvSA2REqh+Dd3w3IF36x4ZSLpyWImV0+FVP4mhAhAiEA6RwvBvbRxQfKx4y4/B1hWkuXLbNkoXPhIm6ylTs03dYqhgNI3//////////ARAAGgwzOTy0DcxMDk3Ny1dExoPLx1X4LEAVDpyCrau0gmaLo/Gze2VLynG+6W/0RG0dXH77kueLLZIhol59mqFne04aNT29+InilkhY3TawQ0mTH4F16clfur48vQkb7yQBK2LMQCYCubSrb8hy14ziUwg7i3YkJVKcm9ZC1k4tw5z7n3WqT6C58c9fsNF320NtDVHzyU2WMAdygJXexdAZ17/fn0gs906x+TTASfqxUmpwaBTLCyAcLex4bv7P7Pkh1DJCxV03APweyb+vdre0Wp2vARE/SW+mrfPT0ZVLQ0pwZ1kVgtgygESB0+r+iw/yq2xmdufneDIzYAND7Yls3crqilBXdc44CcRFGspyDa9Y0PsZputtLXRjFwzXMJ2YigXmc6fxIrW6GuJLzch/gespoPLMXIQzyl6ih5GaT4+Wfh9BccFdjpsYYoCceNmYaZo5SmajqjLbuwvvh6Vz0j0778IlgLn61UfiGnkBugcTa/f/gw1iy1qgY6pgFvt+kIagxlgb560a9iin2YNmNgXiEj0MwiZWFV5ZehYYKiCUSBnw9PmuGGozpsrku3iTdxgw1sjSaTHj5BX8/q7J/+WWMO2MGhoTp3aspL2C+o+iUcr5WFepbZxULpmrCrZjFeusppnn8XqZGLXL2gnFC7tdpYVDCoYdCTTuteYn15Cd54VhZo0DbndUylo/KpIn9q9sV23h/cqrDhwI
X-Amz-Content-SHA256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
Authorization: AWS4-HMAC-SHA256 Credential=ASIAYV0FBVCIBZEC6NVH/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 u 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//////////wEaCWV1LXdIc3QtMSJHMEUCIDSpvSA2REqh+Dd3w3IF36x4ZSLpyWImV0+FVP4mhAhAiEA6RwvBvbRxQfKx4y4/B1hWkuXLbNkoPHIm6ylTs03dYqhgNI3//////////ARAAGgwzOTy0DcxMDk3NzyIdExoPLx1X4LEAVDpycrau0gmaLo/Gze2VLynG+6W/0RG0dXH77kueLLZIhol59mqFne04aNT29+InilkhY3TawQ0mTH4F16clfur48vQkb7yQBK2LMQCYCubSrb8hy14ziUwg7i3YKJKVCkm9ZC1k4tw5z7n3WqT6C58c9fsNF320NtDVHzyU2WMAdygJXexdAZ17/fn0gs906x+TTASfqxUmpwaBTLCyAcLex4bv7P7Pkh1DJCxV03APweyb+vdre0Wp2vARE/SW+mrfPT0ZVLQ0pwZ1kVgtgygESB0+r+iw/yq2xmdufneDIzYAND7Yls3crqilBXdc44CcRFGspyDa9Y0PsZputtLXRjFwzXMJ2YigXmc6fxIrW6GuJlzch/gespoPLMXI0zyl6ih5GaT4+Wfh9BccFdjpsYYoCceNmYaZo5SmajqjLbwvvvh6Vz0j07781IlgLn61UfiGnkbuGctaf/gw1iy1qgY6pgFvt+kIagxlgb560a9iiin2YNmNgXiEj0MwiZWFV5ZehYYKicUSBnw9PmuG0Gozpsrku3iTdxgw1sjSaTHj5BX8/q7J/+NNM02MGhoTp3paspl2C+o+iUcr5WFepbZxULpmrCrZjFeusppnn8XqZGLXL2gnFC7tdpYVDCoYdCTTuteYn15Cd54VhZo0DbndUylo/KpIn9q9sV23h/cqrDhwI		
X-Amz-Content-SHA256:	e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855		
Authorization:	AWS4-HMAC-SHA256 Credential=ASIAYV0FBVCIBZEC6NVH/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]

[*:8080] [i:~d s3.eu-west-1.amazonaws.com & ~s] : 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//////////wEaCWV1LXd1c3QtMSJHMEUCIDSpvSA2REqh+Dd3w3IF36x4ZSLpyWImV0+FVP4mhAhAiEA6RwvBvbRxQfKx4y4/B1hWkuXLbNkoxPhIm6ylTs03dYqhgNI3/////////ARAAggwzOTy0DcxMDk3NyIdExoPLx1X4LEAVDpycrau0gmaLo/Gze2Lyng+6W/0RG0dXH77kueLLZIhol59mqFne04aNT29+InilkhY3TawQ0mTH4F16clfur48vQkb7yQBK2LMQCYCubSrb8hy14ziugw7i3YkJVKcm9ZC1k4tw5z7n3WqT6C58c9fsNF320NtDVHzyU2WMAdygJXexdAZ17/fn0gs906x+TTASfqxUmpwaBTLCyAcLex4bv7P7Pkh1DJCxV03APweyb+vdre0Wp2vARE/SW+mrfPT0ZVLQ0pwZ1kVgtgygESB0+r+iw/yq2xmdufneDIzYAND7Yls3crqilBXdc44CcRFGspyDa9Y0PsZputtLXRjFwzXMJ2YigXmc6fxIrWGuJlzch/gespoPLMXIQzyl6ih5Gat4-Wfh9BccFdjpsYYoCceNmYaZo5SmajqjLbwvvh6Vz0j07781Ilgm61UfiGnkbuGctaf/gw1iy1qgY6pgFvt+kIagxlgb560a9iiin2YNmNgXiEj0MwiZWFVSVzehYYKicUSBnw9PmuG0Gozpsrku3iTdxgw1sjSaTHj5BX8/q7J/+NNM02MGhoTp3paspl2C+o+iUcr5WFepbzULpmrCrZjFeusppnn8XqZGLXL2gnFC7tdpYVDCoYdCTTuteYn15Cd54VhZo0DbndUylo/KpIn9q9sV23h/cqrDhwI
X-Amz-Content-SHA256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
Authorization: AWS4-HMAC-SHA256 Credential=ASIAYV0FBVCIBZEC6NVH/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]

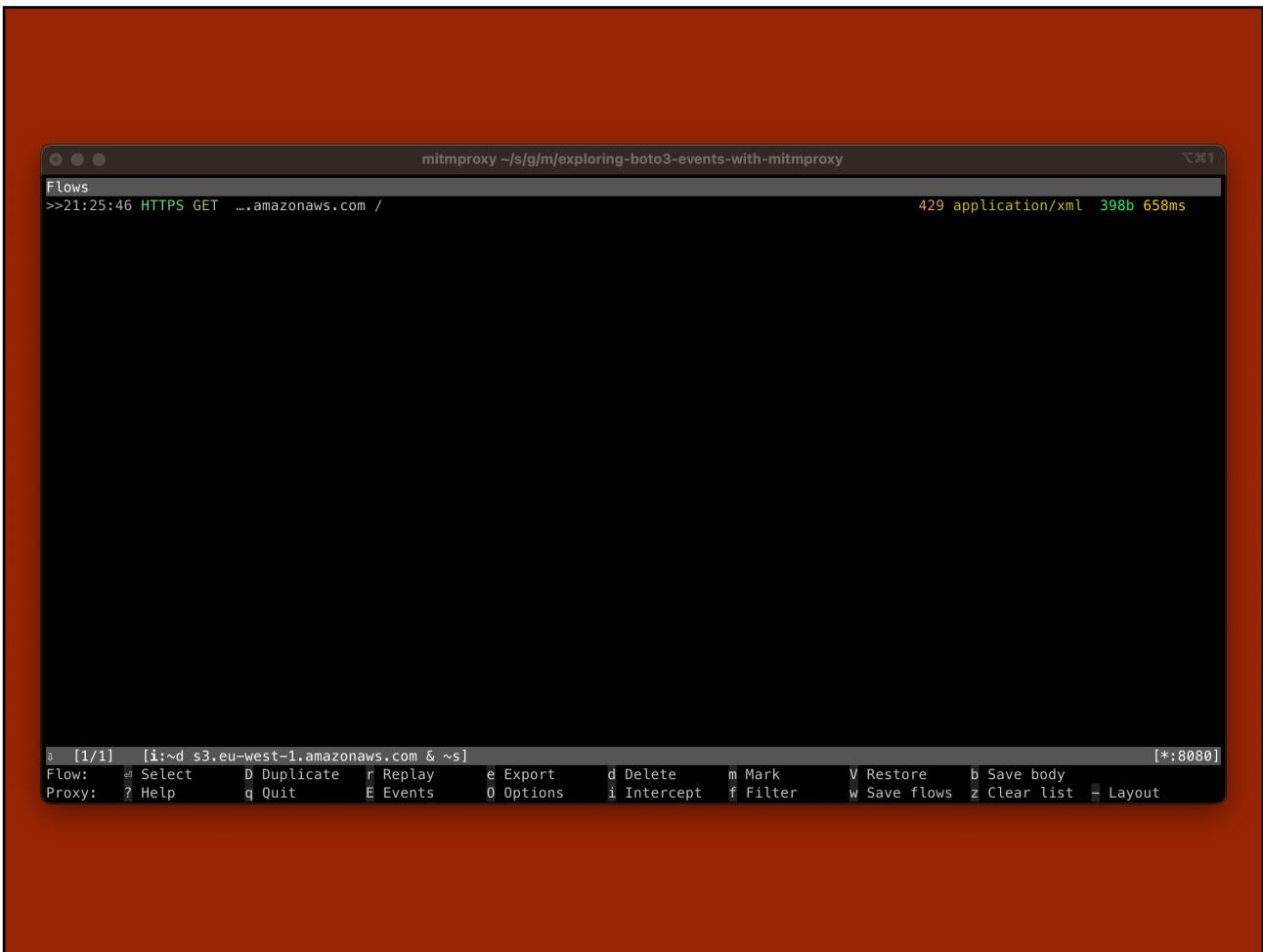
[1/1] [i:~d s3.eu-west-1.amazonaws.com & ~s] [*:8080]
Alert: [21:26:30.367] Set status_code on 1 flows.
```

```
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//////////wEaCWV1LXd1c3QtMSJHMEUCIDSpvSA2REqh+Dd3w3IF36x4ZSLpyWImV0+FVP4mhAhAiEA6RwvBvbRxQfKx4y4/B1hWkuXLbNkoXPhIm6ylTs03dYqhgNI3//////////ARAAggwzOTy0DcxMDk3Ny1dExoPLx1X4LEAVDpycrau0gmaLo/Gze2VlyNG+6W/0RG0dXH77kueLLZIhol59mqFne04aNT29+InilkhY3TawQ0mTH4F16clfur48vQkb7yQBK2LMQCYCubSrb8hy14ziugw7i3YkJVKcm9ZC1k4tw5z7n3WqT6C58c9fsNF320NtDVHzyU2WMAdygJXAxedAZ17/fn0gs906x+TTASfqxUmpwaBTLCyAcLex4bv7P7Pkh1DJCxV03APweyb+vdre0Wp2vARE/SW+mrfPT0ZVLQ0pwZ1kVgtgygESB0+r+iw/yq2xmdufneDIzYAND7Yls3crqilBXdc44CcRFGspyDa9Y0PsZputtLXRjFwzXMJ2YigXmc6fxIrW6GuJLzch/gespoPLMXIQzyl6ih5GaT4+Wfh9BccFdjpsYYoCceNmYaZo5SmajqjLbuwvvh6Vz0j07781lgn61UfiGnkBugcTa/f/gw1iy1qgY6pgFvt+kIagxlgb560a9iiin2YNmNgXiEj0MwiZWFV5ZehYYKiCUSBnw9PmuGGozpsrku3iTdxgw1sjSaTHj5BX8/q7J/+WWMO2MGhoTp3aspL2C+o+iUcr5WFepbZxULpmrCrZjFeusppnn8XqZGLXL2gnFC7tdpYVDCoYdCTTuteYn15Cd54VhZo0DbndUylo/KpIn9q9sV23h/cqrDhwI X-Amz-Content-SHA256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 Authorization: AWS4-HMAC-SHA256 Credential=ASIAVY0FBVCIBZEC6NVH/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]

[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 u Next flow p Prev flow
Proxy: ? Help q Back 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:
x 254 54s a scratch.AWSAdministratorAccess/eu-west-1 21:26:39
```

I'm Lazy

```
import logging
from mitmproxy import flowfilter
from mitmproxy import http

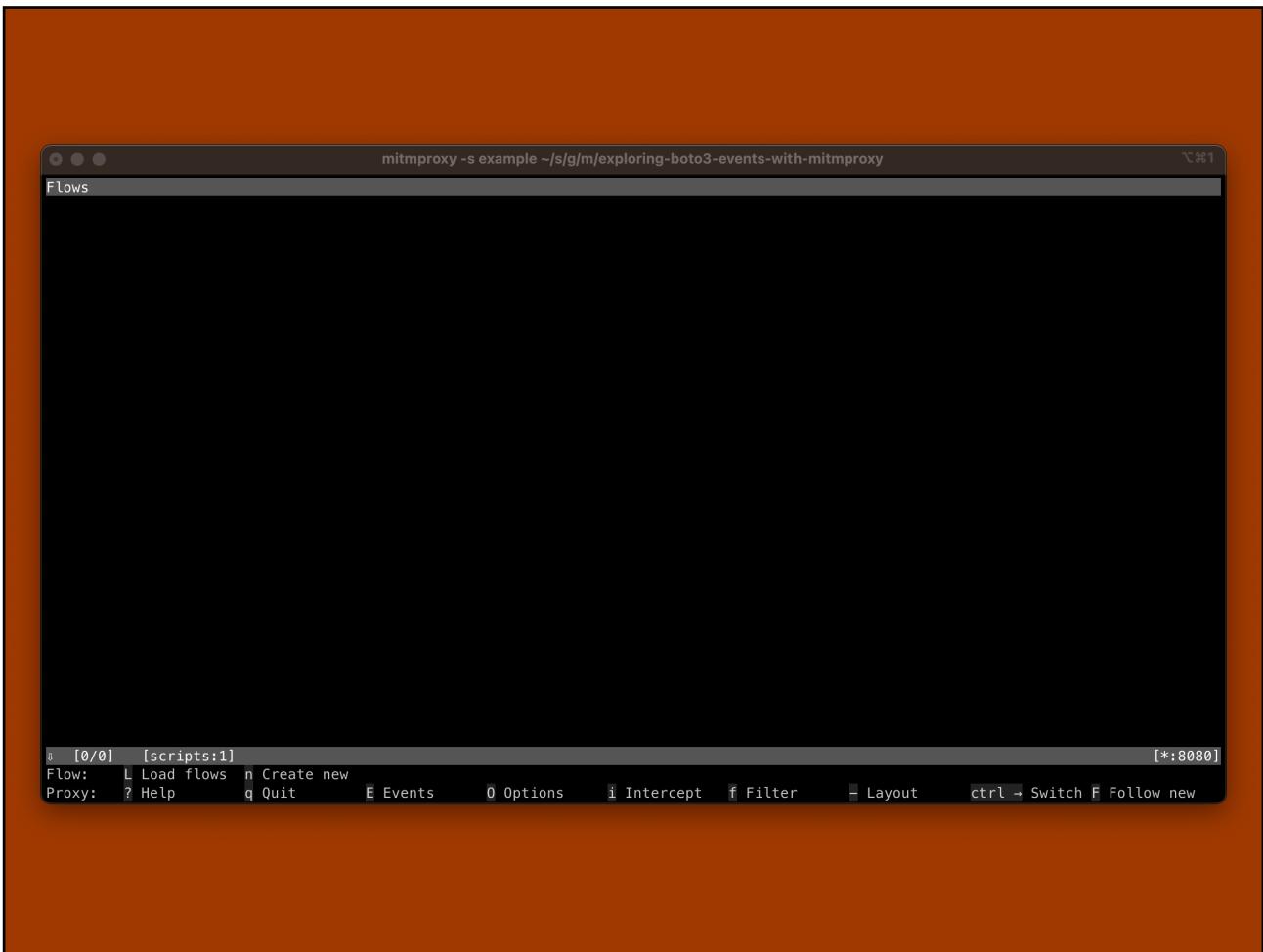
class RateLimitExceededFilter:
    filter: flowfilter.TFilter

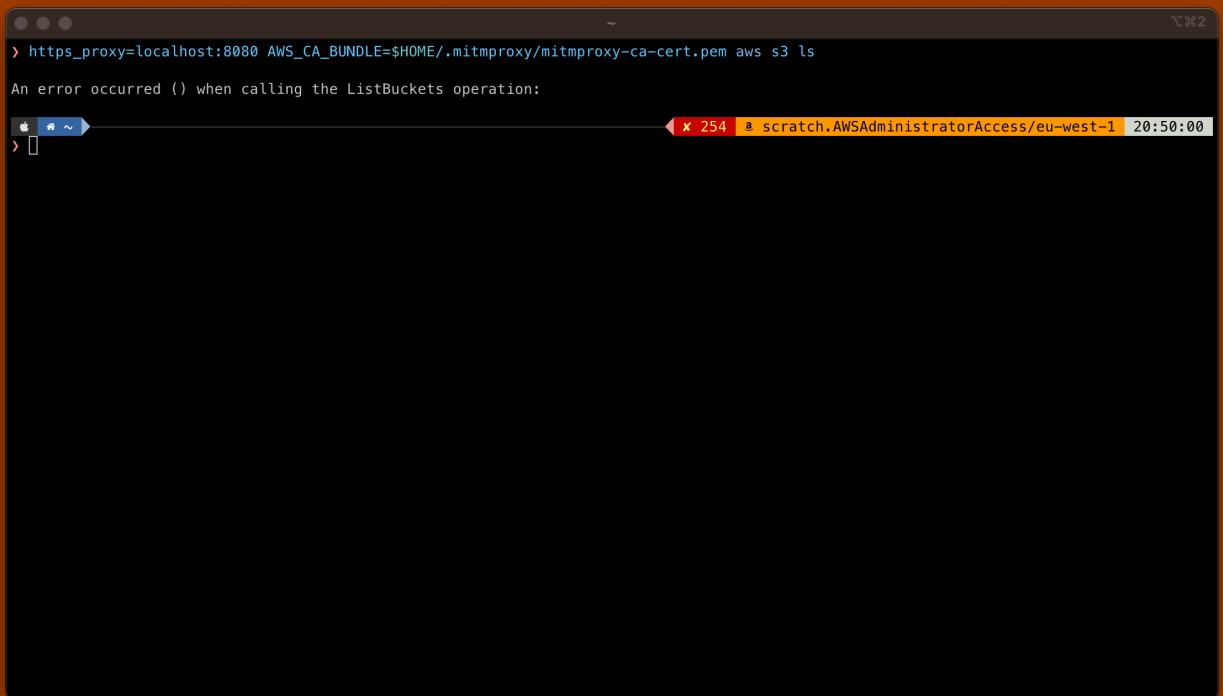
    def __init__(self):
        self.filter = flowfilter.parse("~d s3.eu-west-1.amazonaws.com & ~s")

    def response(self, flow: http.HTTPFlow) -> None:
        if flowfilter.match(self.filter, flow):
            flow.response.status_code = 429
            flow.response.reason = "Too Many Requests"

addons = [RateLimitExceededFilter()]
```

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



A screenshot of a macOS terminal window titled 'Terminal'. The command entered is 'https_proxy=localhost:8080 AWS_CA_BUNDLE=\$HOME/.mitmproxy/mitmproxy-ca-cert.pem aws s3 ls'. The response shows an error message: 'An error occurred () when calling the ListBuckets operation:'. The status bar at the bottom right indicates the session ID 'x 254' and the profile 'scratch.AWSAdministratorAccess/eu-west-1'.

Here I'm testing using the aws cli to list buckets.



```
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 L Layout
```

Here you can see the request was intercepted by mitmproxy and the response was automatically rewritten

Interestingly the aws cli doesn't retry automatically!

```
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: +wB3Wljx8bmius4XI+5wE7U9xIx0gaw1KL48FfsiQLY3S89IyLFB3dp3Hi9iWPGVnn0NKHDR0=		
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 n Next flow p Prev flow
Proxy: ? Help q Back E Events o Options i Intercept f Filter w Save flows z Clear list l 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: **s** 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 **L** Layout

```
import boto3

def print_event(event_name: str, attempts: int,
response, **kwargs):
    print(
        event_name,
        attempts,
        response[1]["ResponseMetadata"]
["HTTPStatusCode"],
    )

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

Here you can see all the retry attempts in mitmproxy

```
~/s/g/m/exploring-boto3-events-with-mitmproxy
> AWS_CA_BUNDLE=$HOME/.mitmproxy/mitmproxy-ca.pem https_proxy=localhost:8080 python examples/intercepting.py
needs-retry.s3.ListBuckets 1 429
needs-retry.s3.ListBuckets 2 429
needs-retry.s3.ListBuckets 3 429
needs-retry.s3.ListBuckets 4 429
needs-retry.s3.ListBuckets 5 429
An error occurred () when calling the ListBuckets operation (reached max retries: 4):
...
8s  exploring-boto3-events-with-mitmproxy  scratch.AWSAdministratorAccess/eu-west-1  impure  23:10:17
>
```

And here you can see the output
The retry event count always
starts on 1 for the first normal
request 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

Metrics!

```
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!")

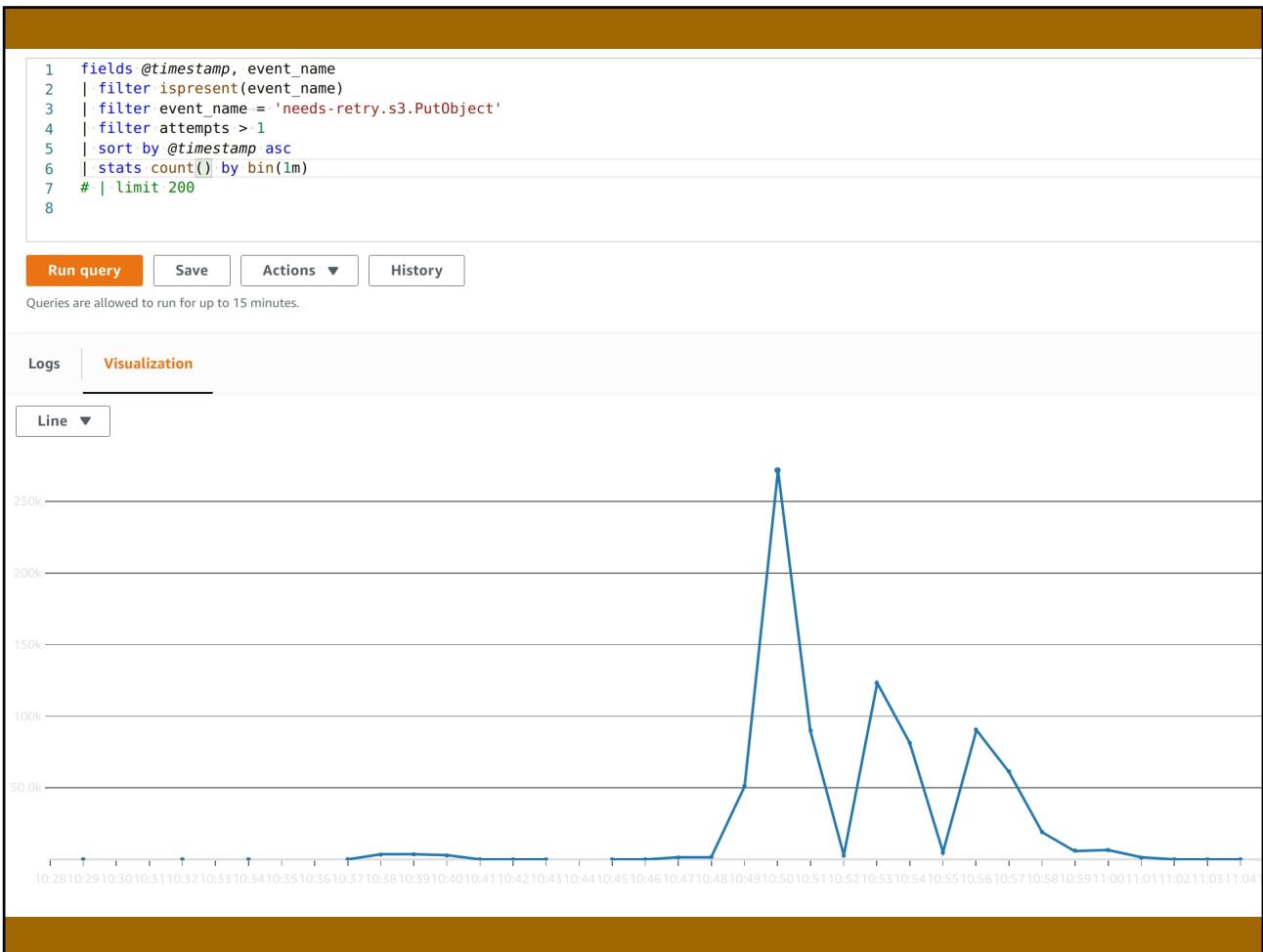
s3 = boto3.client("s3")
s3.meta.events.register("needs-retry.s3.*", handle_retry)
s3.list_buckets()
print("All done!")
```

```
~/s/g/m/exploring-boto3-events-with-mitmproxy
> AWS_CA_BUNDLE=$HOME/.mitmproxy/mitmproxy-ca.pem https_proxy=localhost:8080 python examples/retry_metrics.py
retry event?
nope!
retry event?
needs-retry.s3.ListBuckets|increment|count=1
retry event?
needs-retry.s3.ListBuckets|increment|count=1
retry event?
needs-retry.s3.ListBuckets|increment|count=1
retry event?
needs-retry.s3.ListBuckets|increment|count=1
All done!

... ✘ 5s exploring-boto3-events-with-mitmproxy ✘ scratch.AWSAdministratorAccess/eu-west-1 ✘ impure 23:16:26
> 
```

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](https://github.com/micktwomey/exploring-boto3-events-with-mitmproxy)

