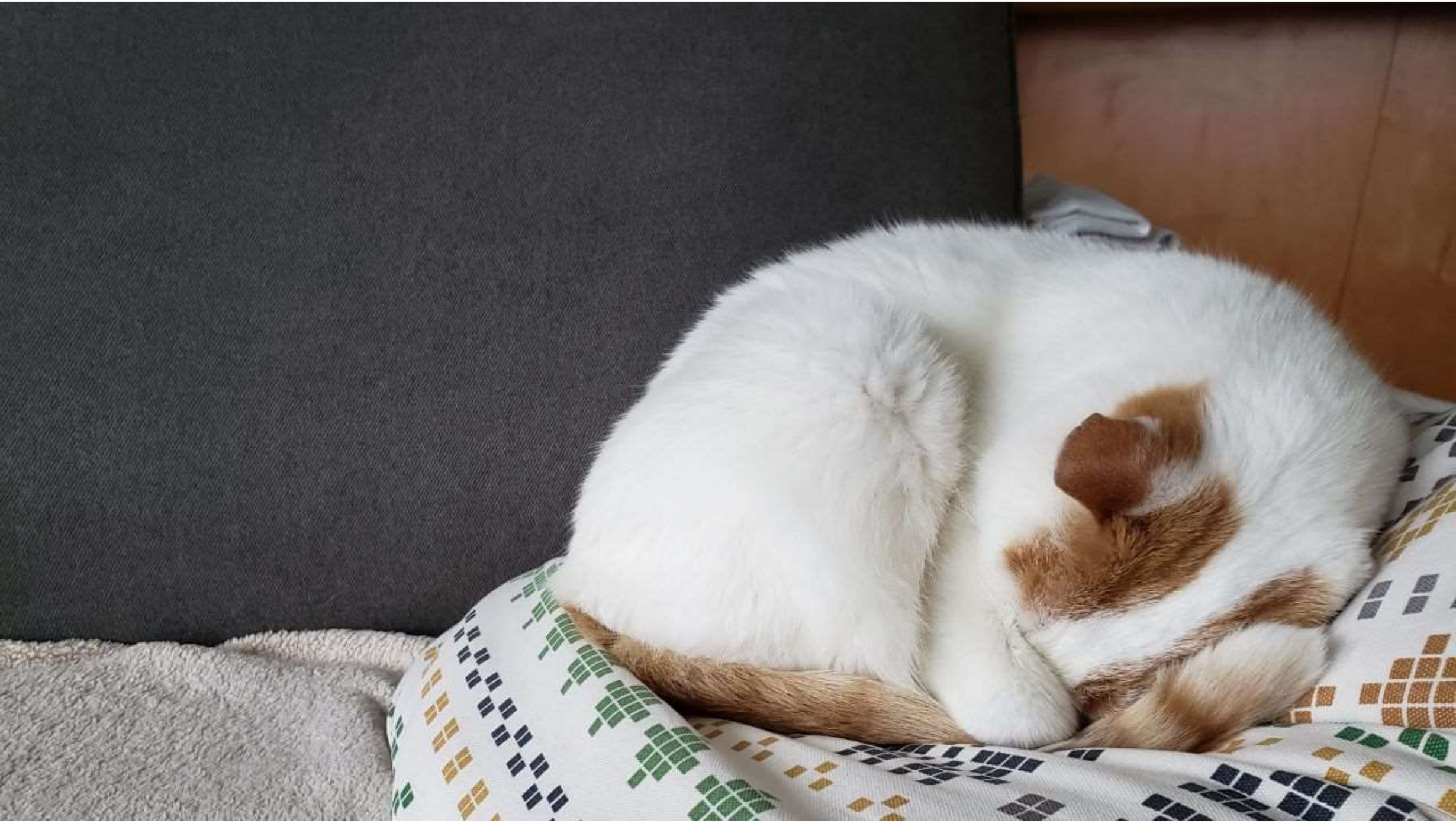
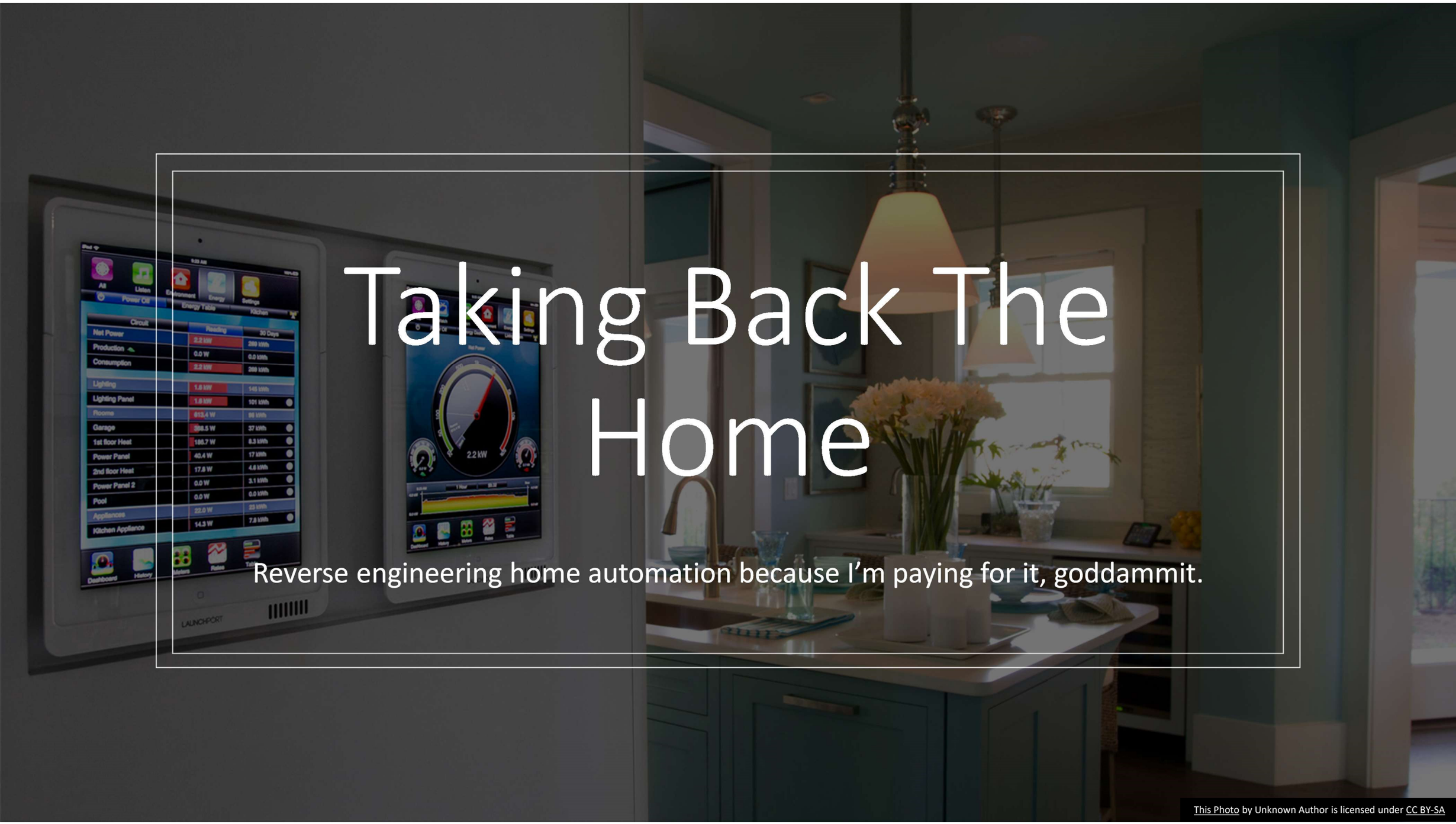
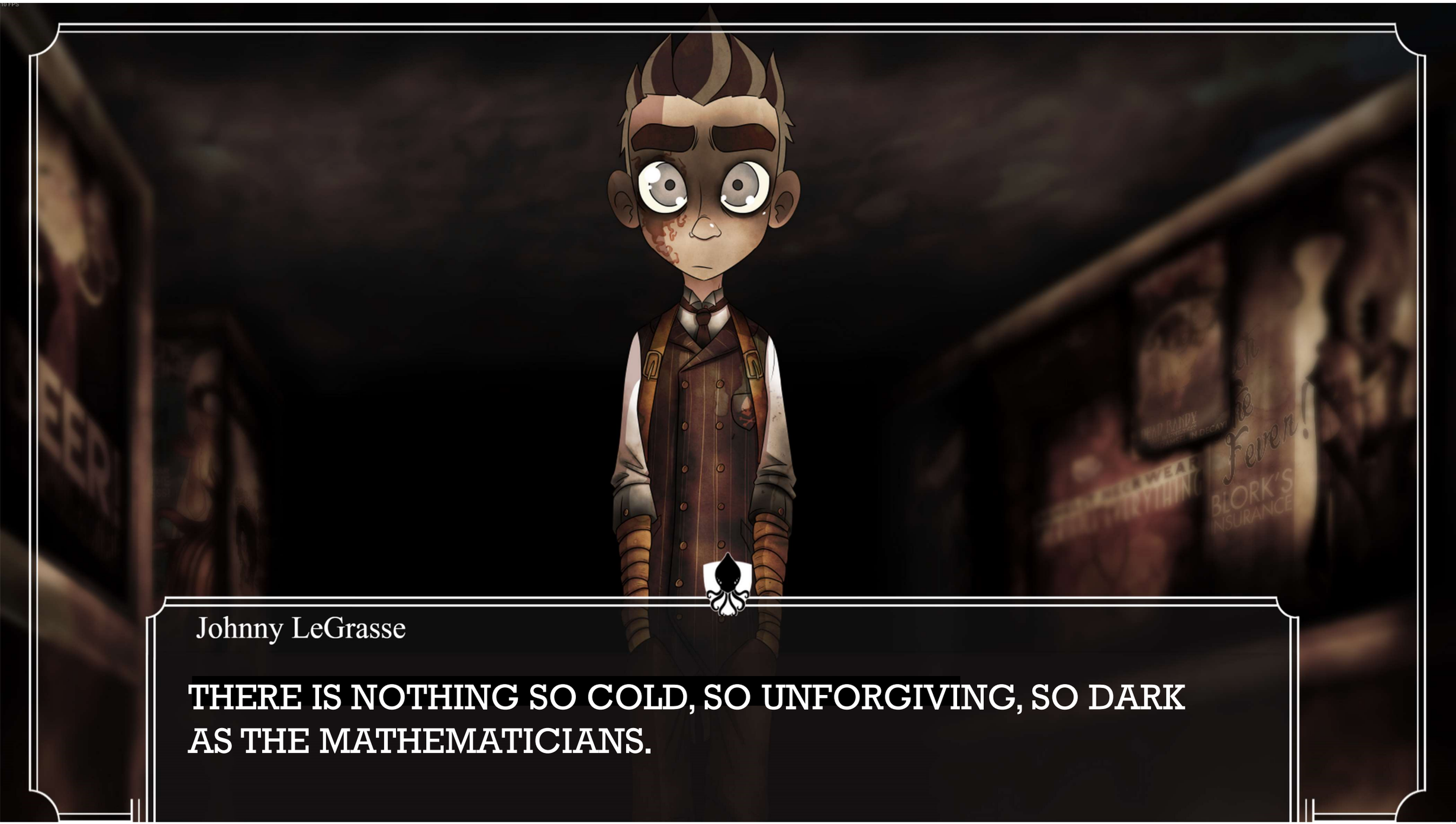


AMAZON AMAZON AMAZON

Shit how did this get in here


Taking Back The Home
Reverse engineering home automation because I’m paying for it, goddammit.
This Photo by Unknown Author is licensed under CC BY-SA


Johnny LeGrasse
THERE IS NOTHING SO COLD, SO UNFORGIVING, SO DARK AS THE MATHEMATICIANS.

Being a politician is a poor profession. Being a public servant is a noble one.

(Herbert Hoover)

Contrary to popular belief

I can talk about something other than AWS

Today’s talk is going to be a story of one particular trench

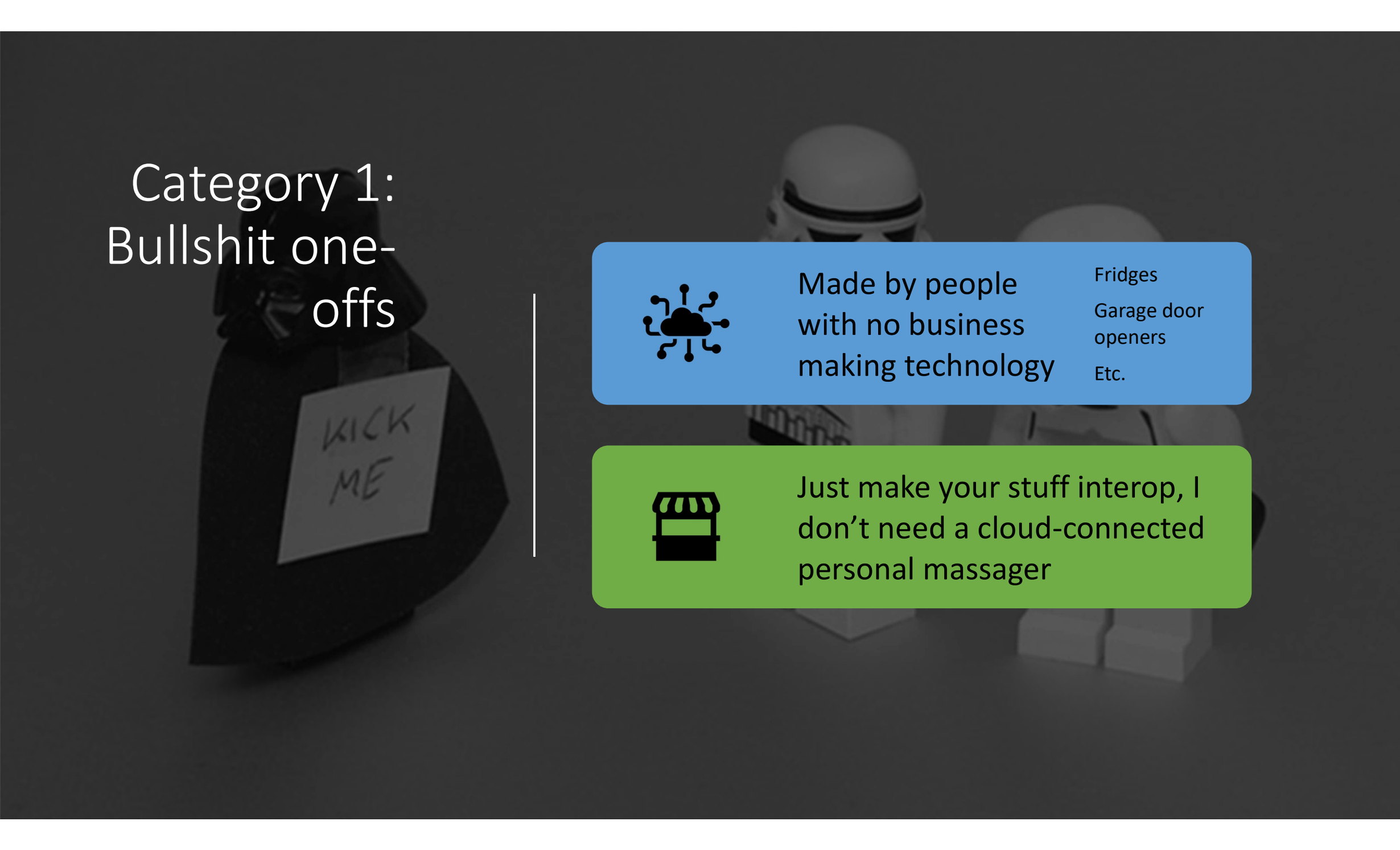
Home automation

Home Automation or Smart Home

- Who has some amount of it


“Home Automation”
I’m going to split it into 3 categories and 2 classes.

Category 1: Bullshit one-offs



Made by people with no business making technology

- Fridges
- Garage door openers
- Etc.

Just make your stuff interop, I don’t need a cloud-connected personal massager

Category 2: Good one-offs without a diverse ecosystem

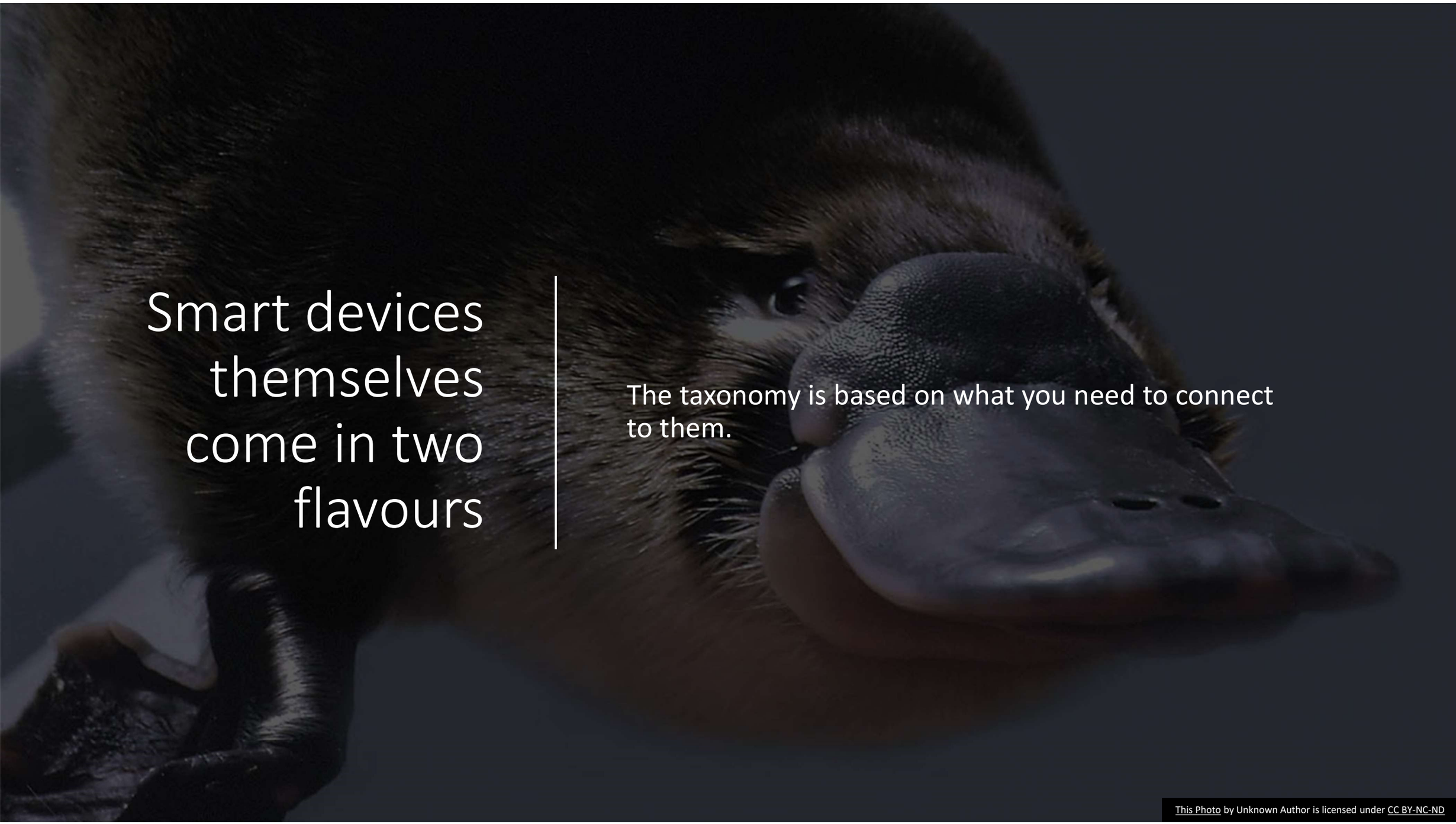
Alexa, Google Home, Apple

Where any one device is worth it, but it wouldn't be that much more worth it with them all.

Begrudgingly: Logitech

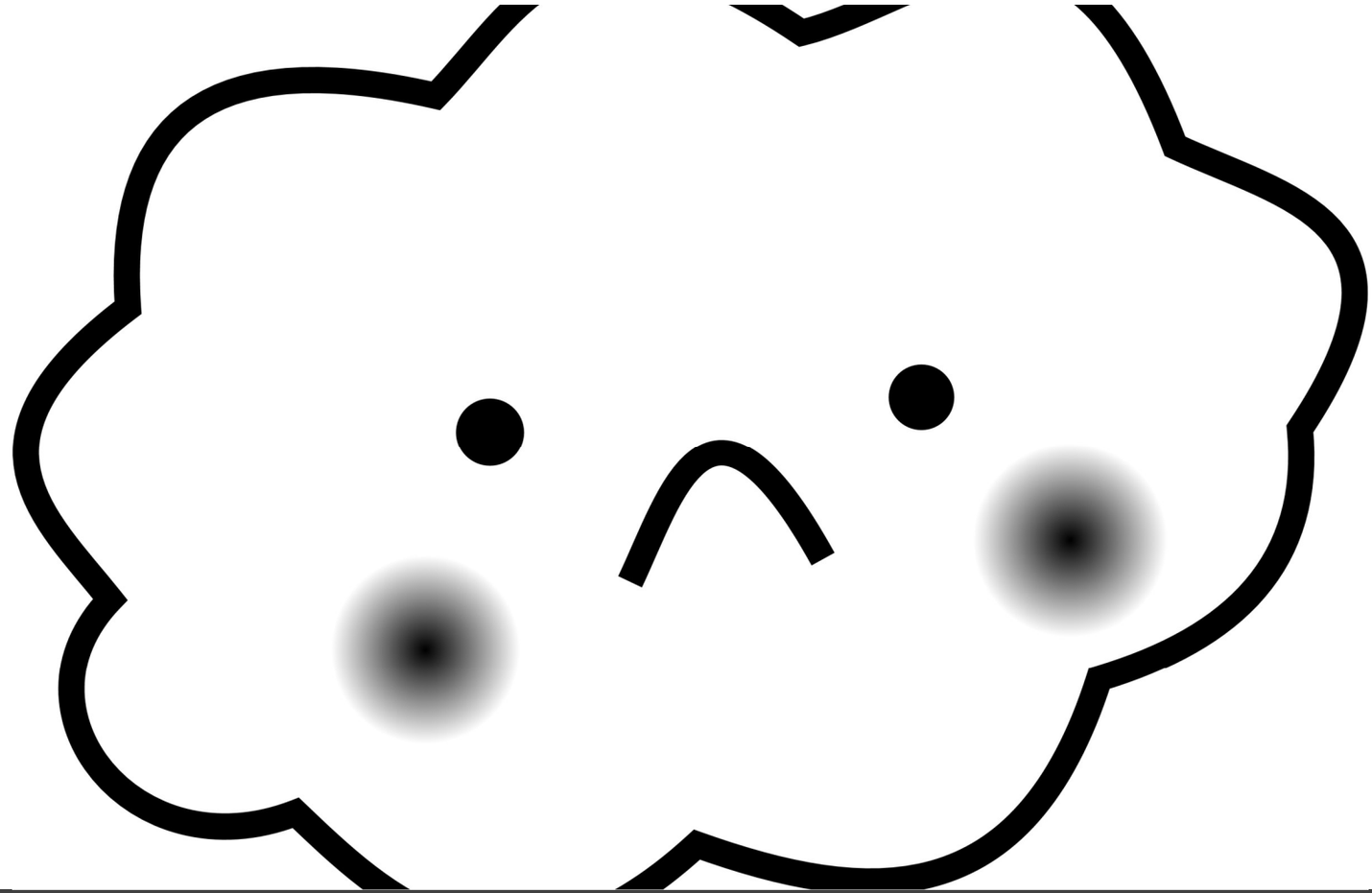
Category 3: Smart device ecosystems

Where any one device is pointless, but get half a dozen, and you've really got something.



Smart devices themselves come in two flavours

The taxonomy is based on what you need to connect to them.



Proprietary cloud connected

- These ones require a third-party cloud service to function.

Open device connectivity

Zwave or WiFi Bluetooth or LORA devices that will talk to anything

Lastly, this all breaks out into two large functional groups

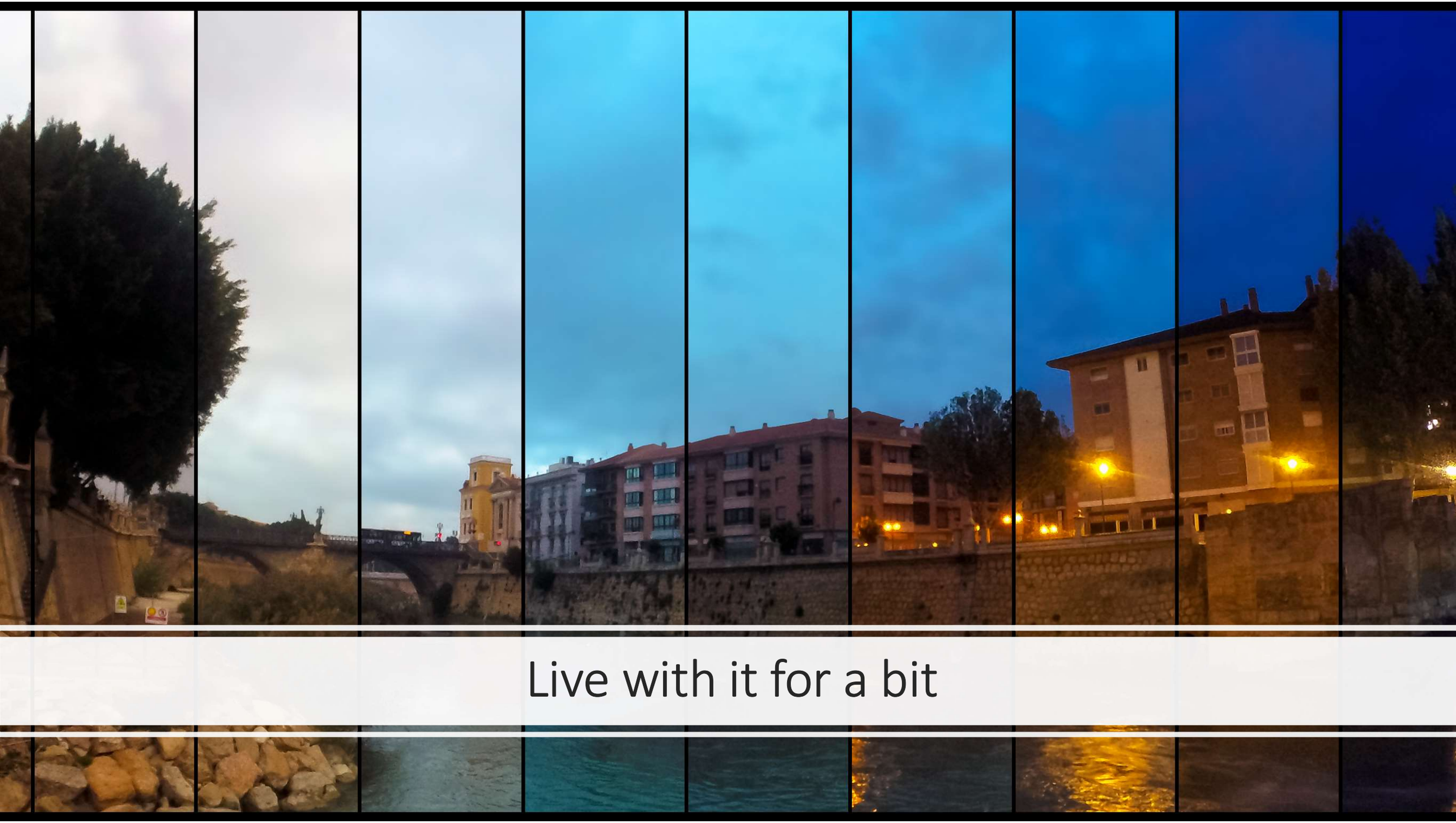
Open and accessible

… Not that

In 60 seconds or less

My story

Get a smart home system with device ecosystem

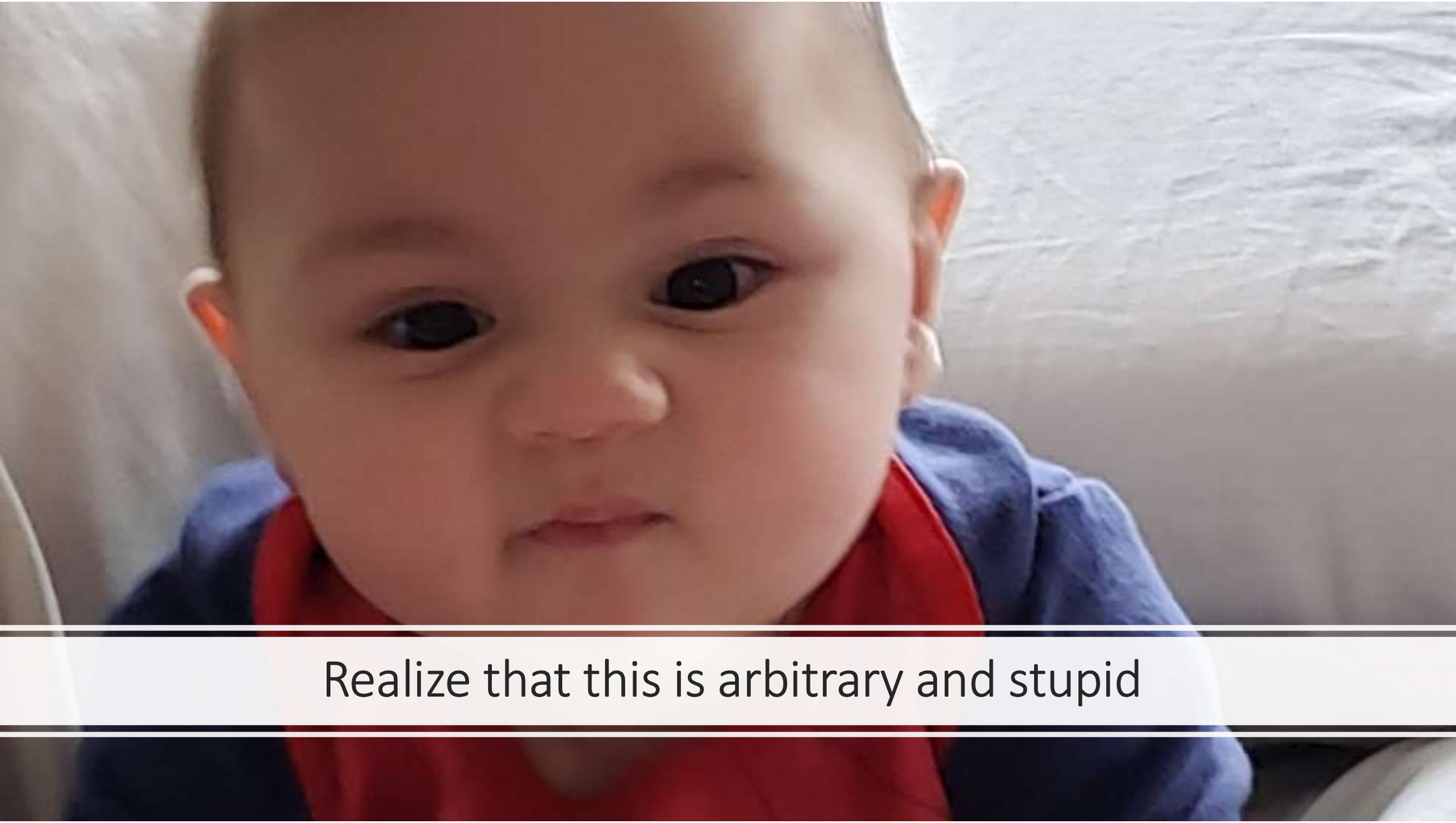


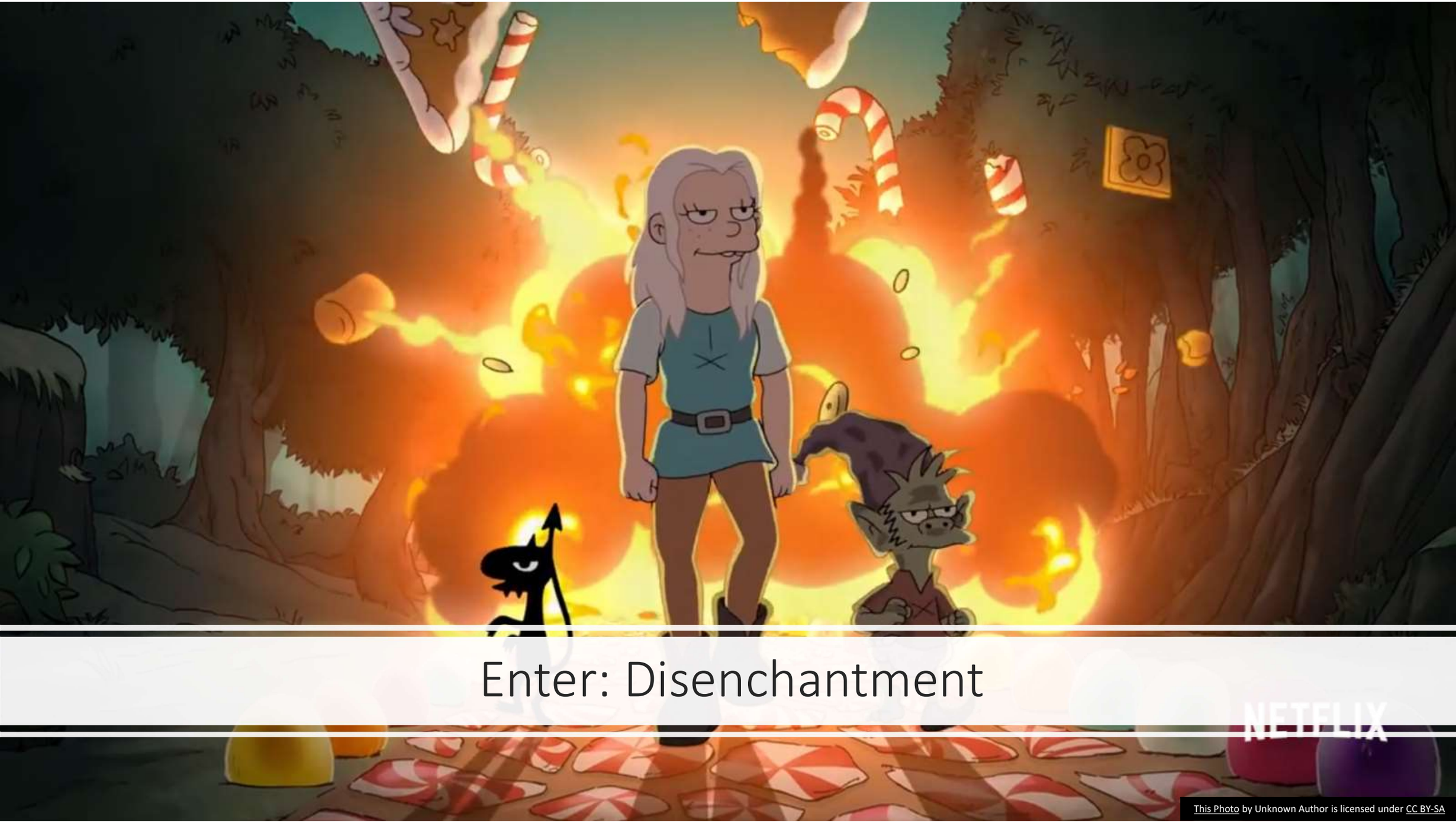
Live with it for a bit


Want something that it doesn’t do

Try to do the thing


Realize that no matter how easy it should be you can’t do thing


Realize that this is arbitrary and stupid

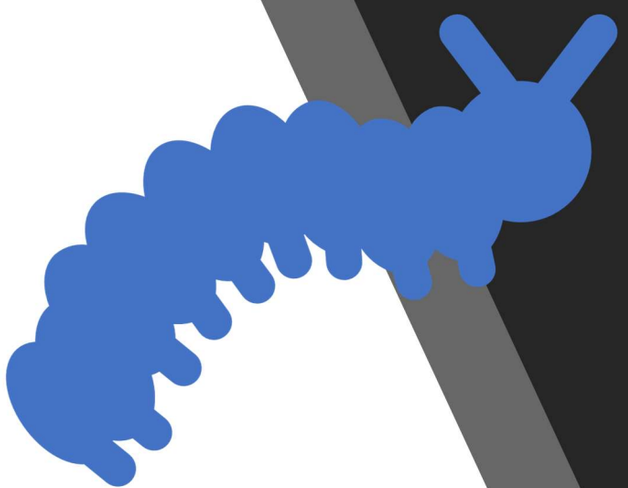


Enter: Disenchantment

A Greek Tragedy in 5 Acts

Let the reverse engineering begin!

Prologue:
The motivation



I have a Vivint system at home

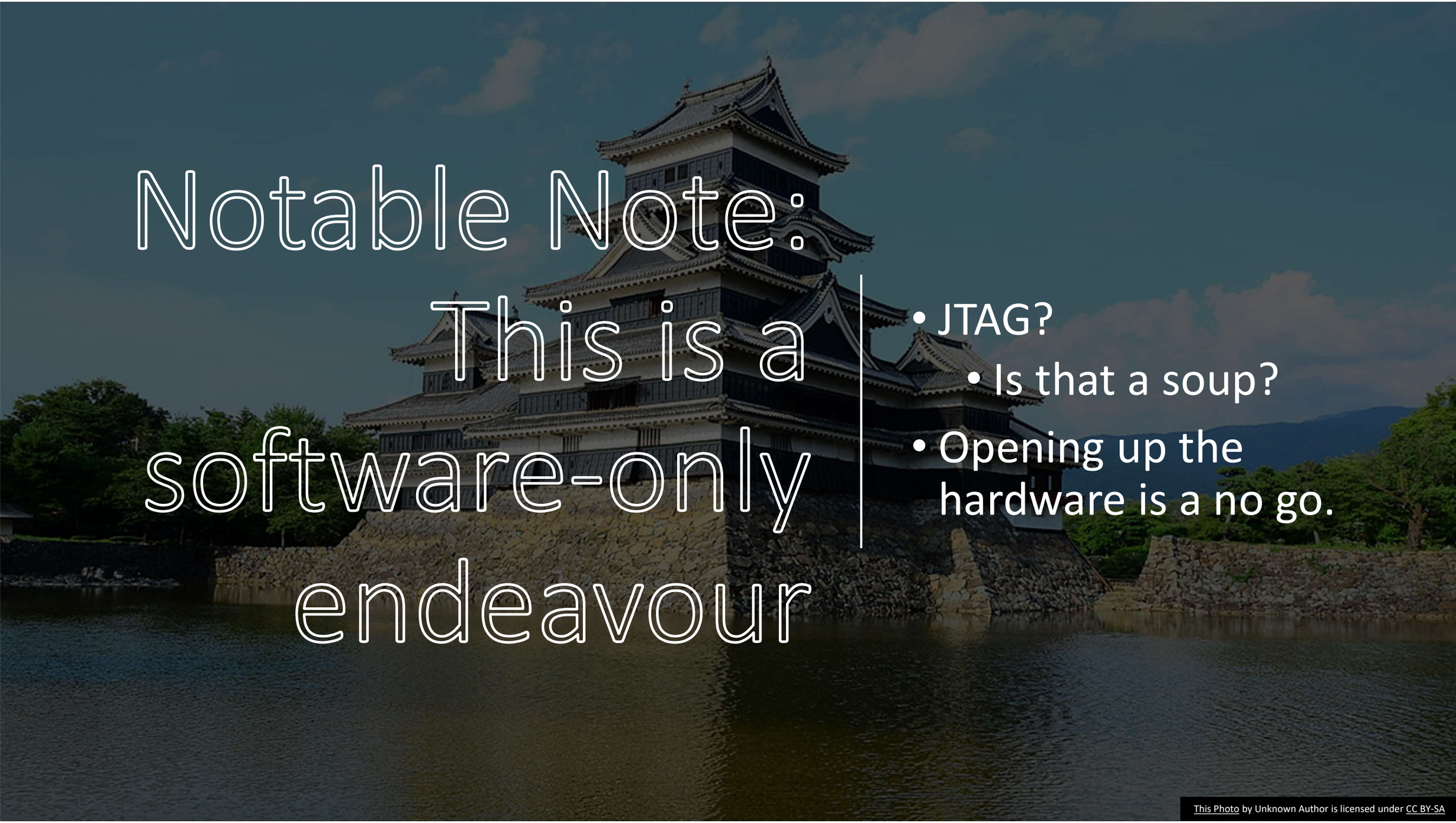
I wanted to track temperature and humidity

And stream the video to long-term storage, not just clips.

It can’t do that out of the box.

Prologue:
The ultimate goal

Be able to have the flexibility of integration, bridging, and longevity of support that I get from an open system, with the phone support and guaranteed compatibility of the Vivint ecosystem.



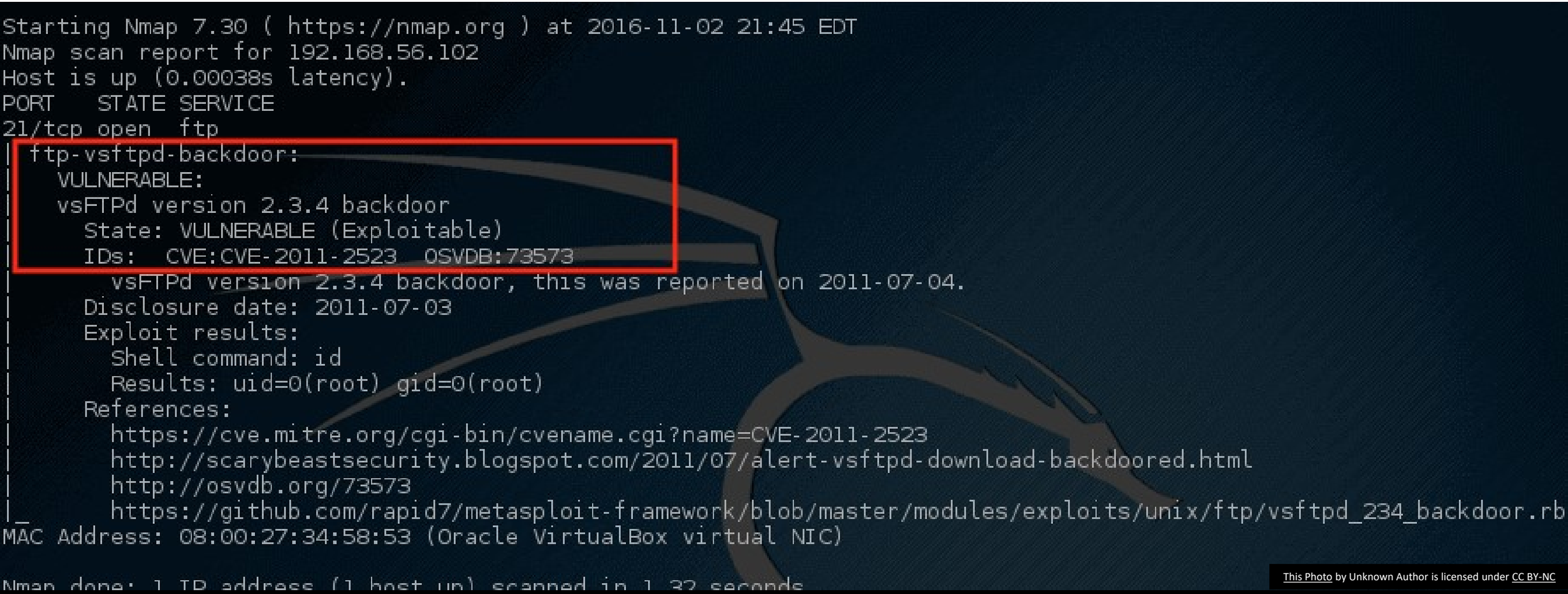
Notable Note: This is a software-only endeavour

- JTAG?
  - Is that a soup?
- Opening up the hardware is a no go.

So, how we gonna do this

- Python module
- Reverse engineer the API calls to the Vivint servers
- Provide a nice abstraction to devices


Charlotte LeStrange
SO SCIENCE IS MAGIC PLUS UNDERWEAR, HUH?

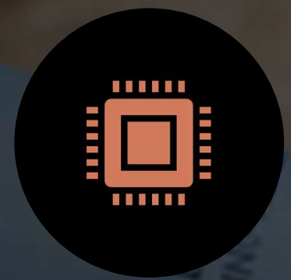


Act 1: The network

Nmap and tcpdump all the things

Results

The panel has a bunch of listening services



The panel is the aggregator of other devices

It talks HTTPS and OpenVPN

Clients will attempt peer-to-peer with the panel for video

Network history

Cameras used to be on my WiFi

Suddenly, they're not there

And a new unnamed SSID is there

- A firmware update moved all of the WiFi devices (cameras) to a network provided by the panel

Act 2: Browser Apps

Dev tools are great.

This let me get really far in understanding how the API worked.

- Reasonably RESTful
- All JSON
- Completely mangled, minified keys and values in the JSON
- Integers everywhere

Authentication notes

There’s probably a third party in there somewhere

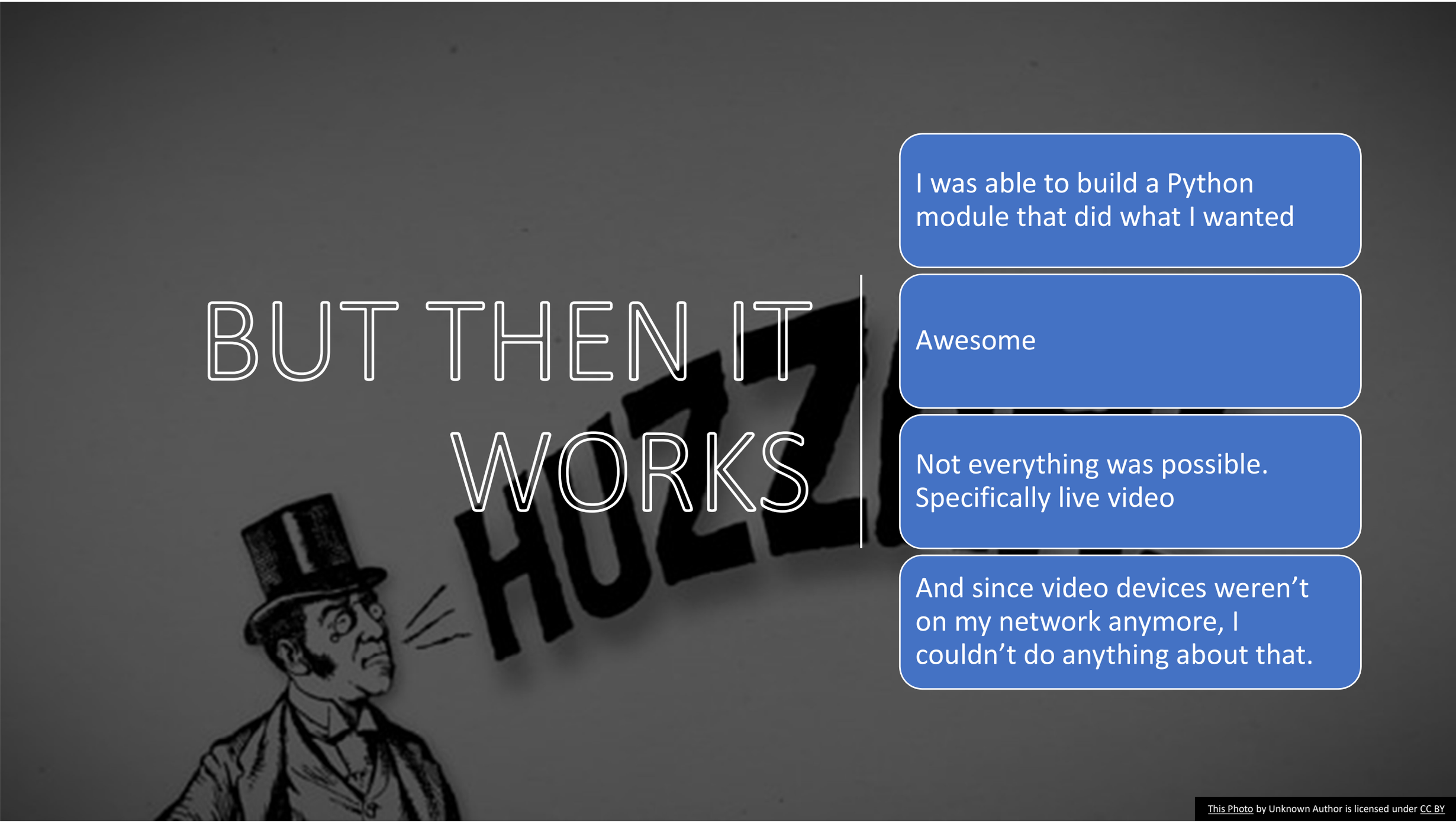
It’s probably eventually OAUTH… eventually

This one was PingFederate, and some HTML/JS garbage

Parsing out hard-coded tokens from JavaScript sources

- Highly fragile

BUT THEN IT WORKS

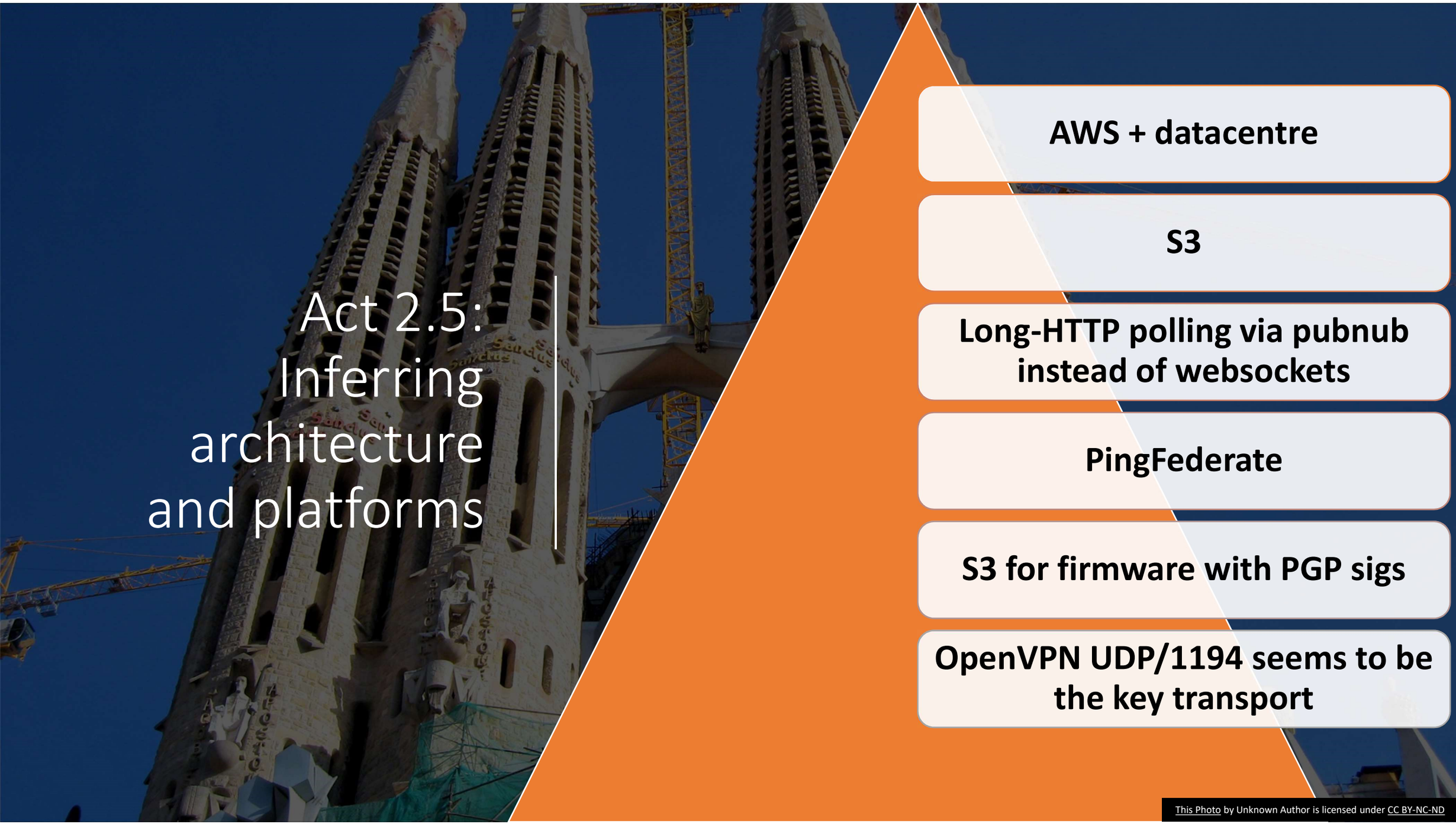


I was able to build a Python module that did what I wanted

Awesome

Not everything was possible. Specifically live video

And since video devices weren't on my network anymore, I couldn't do anything about that.



Act 2.5: Inferring architecture and platforms

- **AWS + datacentre**
- **S3**
- **Long-HTTP polling via pubnub instead of websockets**
- **PingFederate**
- **S3 for firmware with PGP sigs**
- **OpenVPN UDP/1194 seems to be the key transport**

Act 3: Community and dwindling of motivation

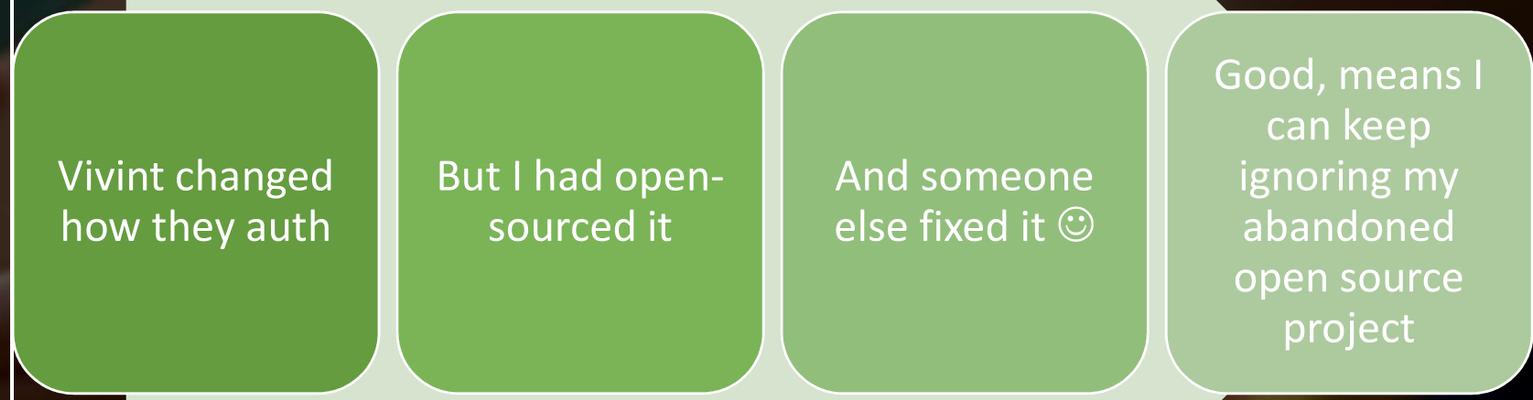
Stuck it on GitHub, then forgot about it

- https://github.com/Riebart/vivint.py

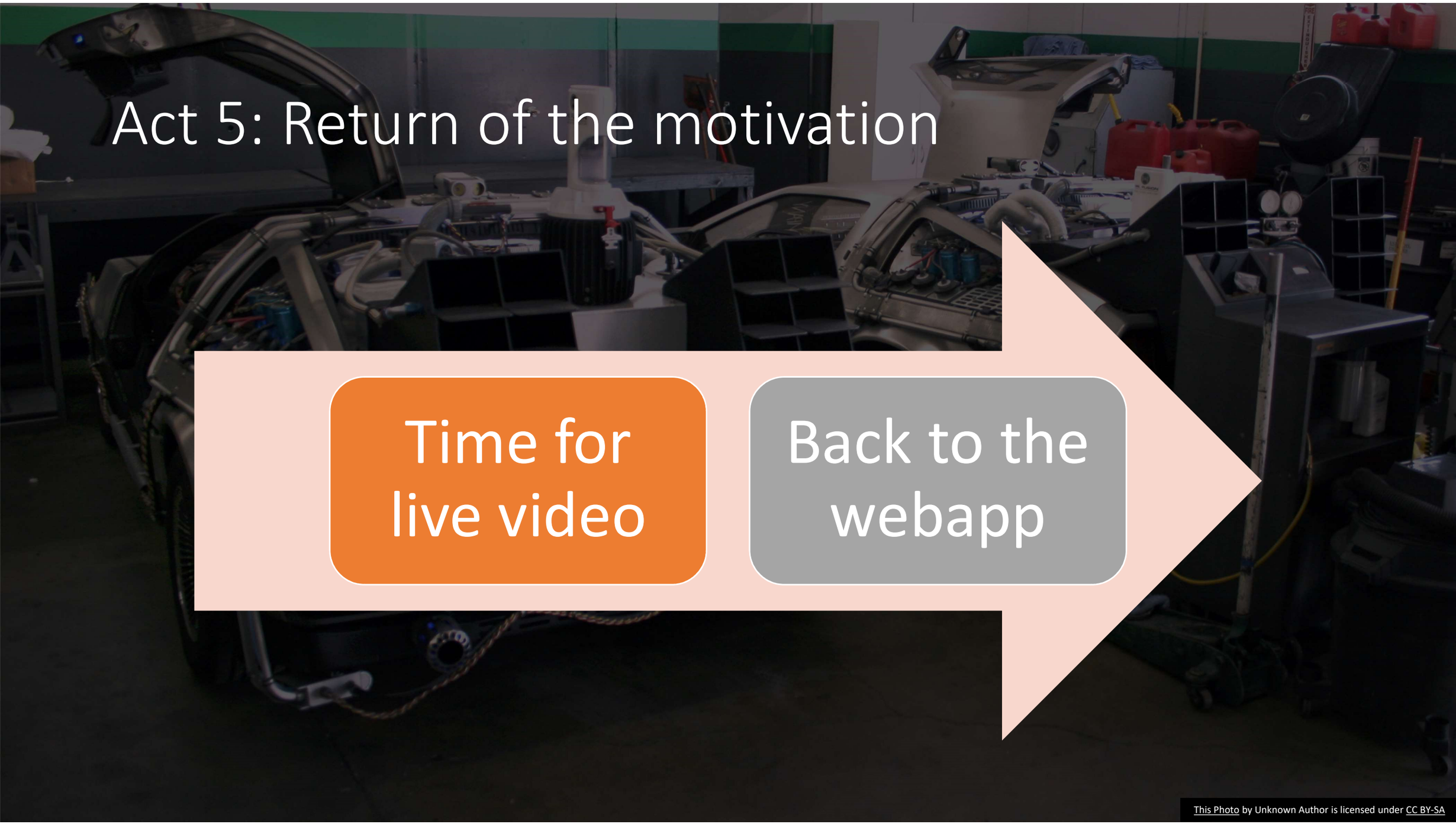
Stopped keeping up with the community

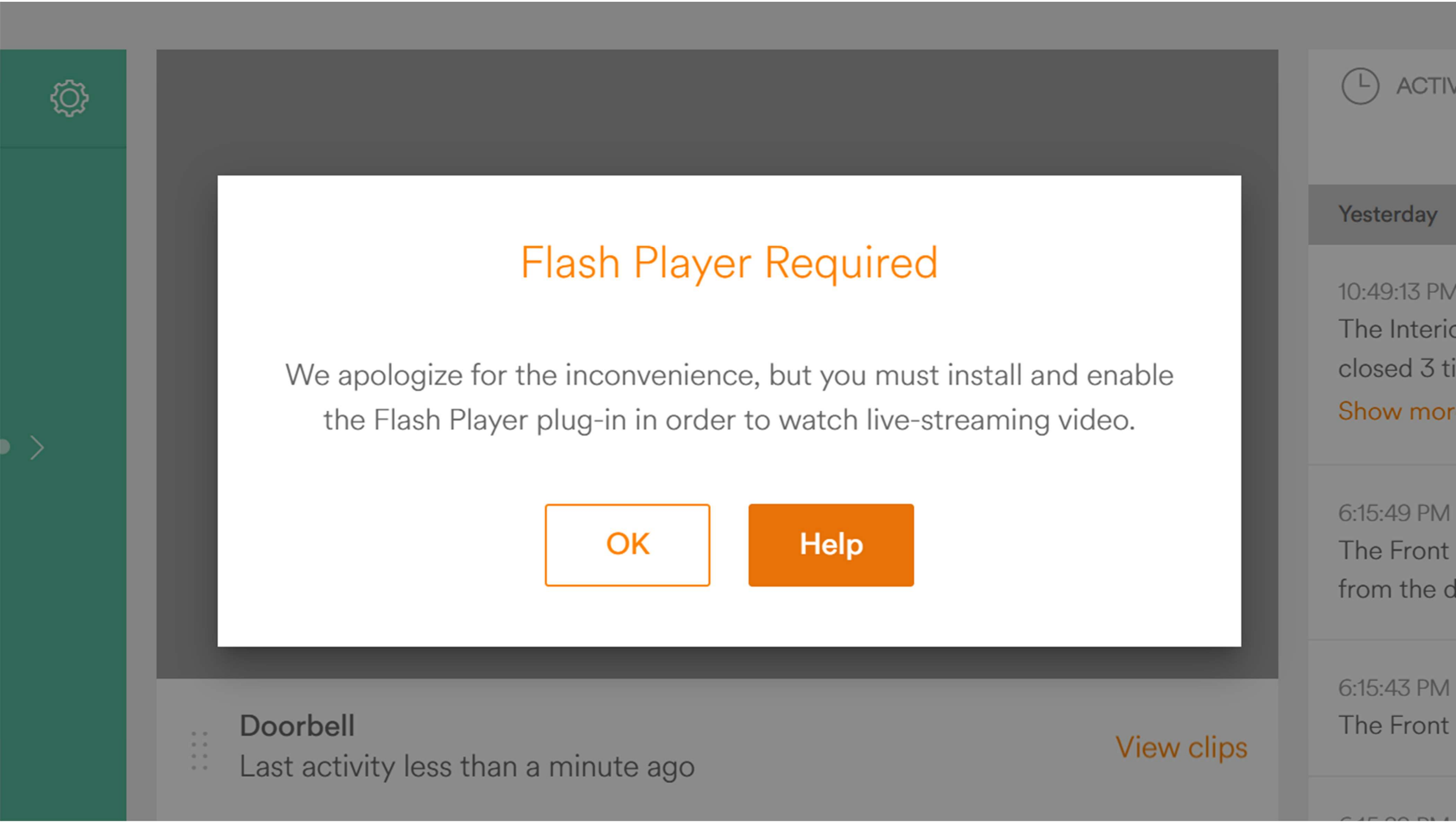
Because it met my needs

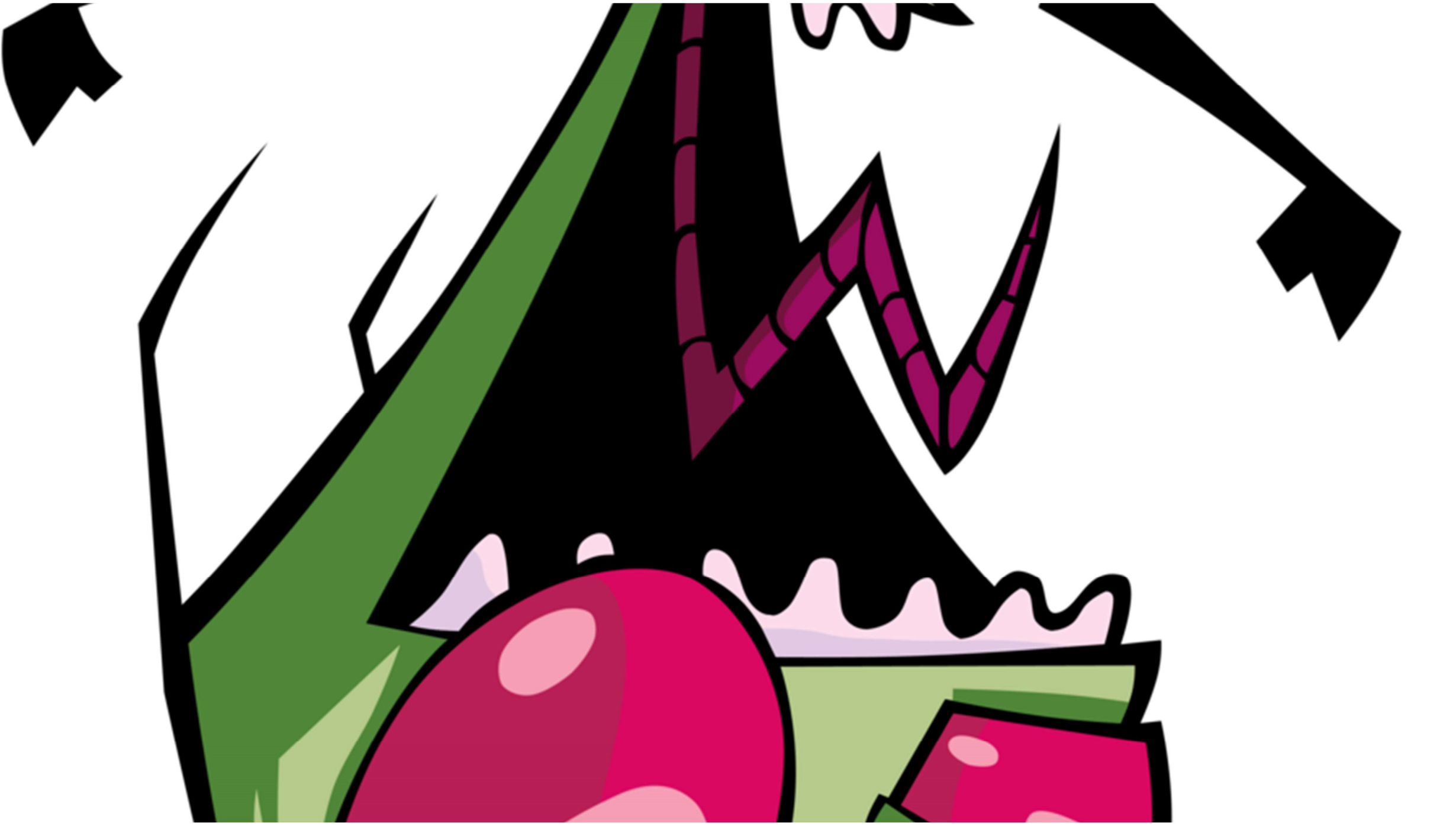
Act 4: Then it breaks (the re-return?)



Act 5: Return of the motivation




Flash Player Required
We apologize for the inconvenience, but you must install and enable the Flash Player plug-in in order to watch live-streaming video.
OK
Help
Yesterday
10:49:13 PM
6:15:49 PM
6:15:43 PM
Doorbell
Last activity less than a minute ago
View clips



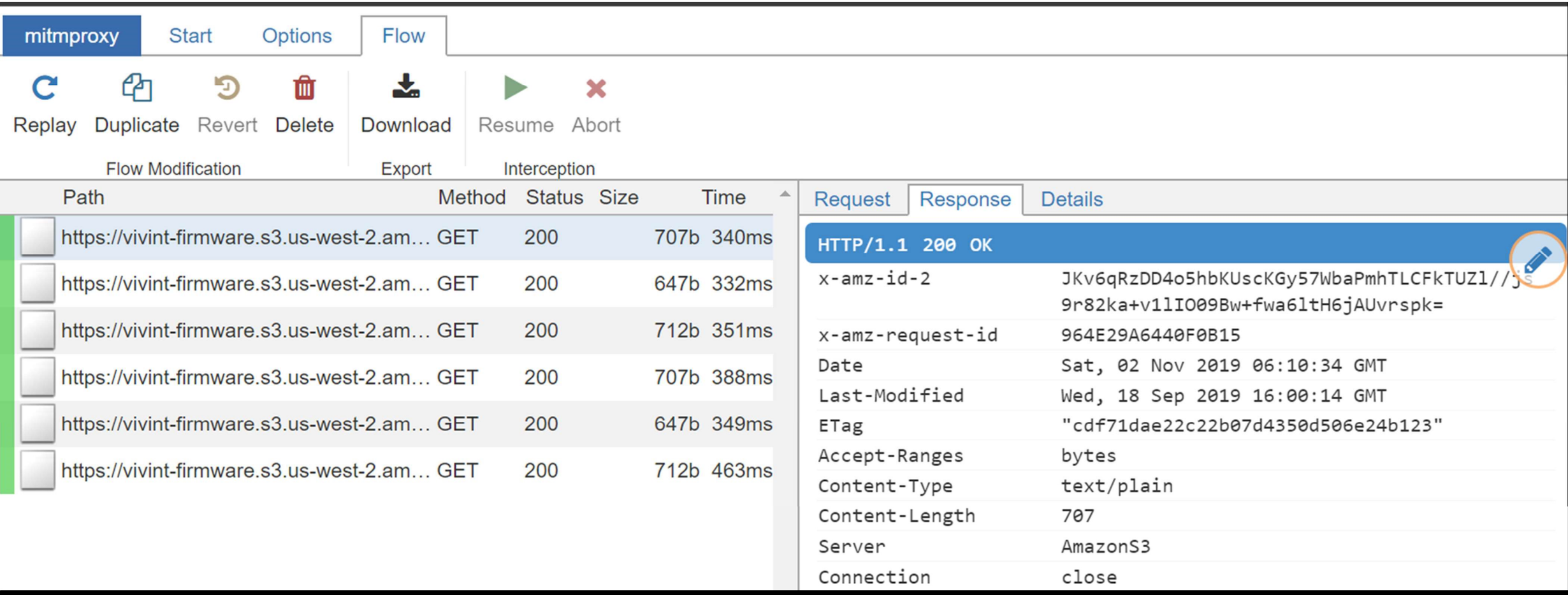
Fukkit, mitmproxy it is

- RPi3B+ as WiFi AP running mitmproxy
- Install CA cert on phone
- Try mobile app…

God dammit, cert pinning

Not everything though.

- The AWS requests all seem to not pin certs


mitmproxy
Start
Options
Flow
Replay
Duplicate
Revert
Delete
Download
Resume
Abort
Flow Modification
Export
Interception
Path
Method
Status
Size
Time
https://vivint-firmware.s3.us-west-2.am… GET 200 707b 340ms
https://vivint-firmware.s3.us-west-2.am… GET 200 647b 332ms
https://vivint-firmware.s3.us-west-2.am… GET 200 712b 351ms
https://vivint-firmware.s3.us-west-2.am… GET 200 707b 388ms
https://vivint-firmware.s3.us-west-2.am… GET 200 647b 349ms
https://vivint-firmware.s3.us-west-2.am… GET 200 712b 463ms
Request
Response
Details
HTTP/1.1 200 OK
x-amz-id-2 JKv6qRzDD4o5hbKUscKGy57WbaPmhTLCFkTUZl//js
9r82ka+v1lIO09Bw+fwa6ltH6jAUvrspk=
x-amz-request-id 964E29A6440F0B15
Date Sat, 02 Nov 2019 06:10:34 GMT
Last-Modified Wed, 18 Sep 2019 16:00:14 GMT
ETag "cdf71dae22c22b07d4350d506e24b123"
Accept-Ranges bytes
Content-Type text/plain
Content-Length 707
Server AmazonS3
Connection close

Wait…

But, the panel!

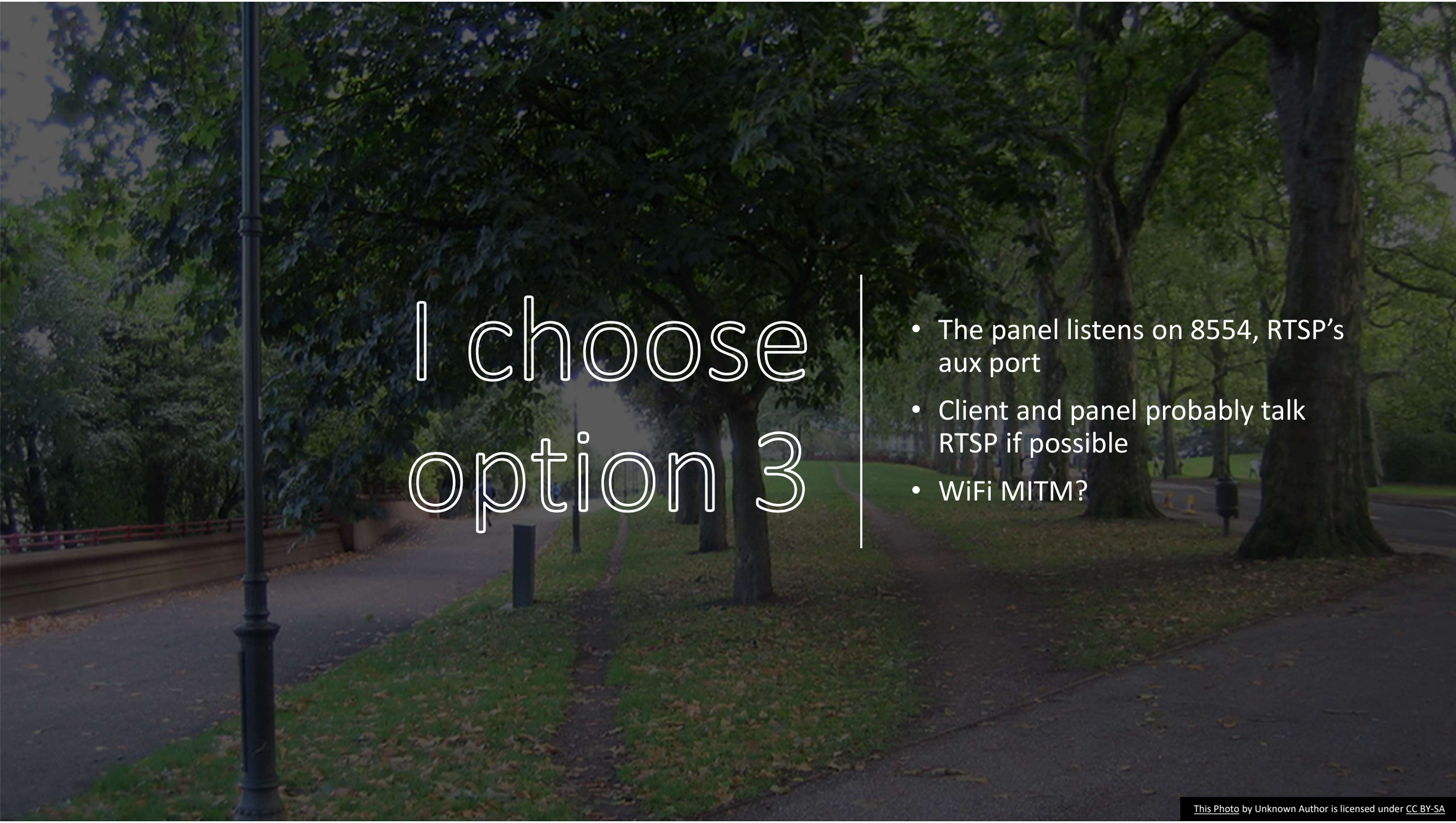
- We can successfully mitm the HTTPS connections to S3
- The PANEL DOES NOT VALIDATE THOSE CERTS

Anyway. That leaves us with 2 options

- Install flash

There is an obvious choice

- Install VirtualBox
- Set up an Android x86 emulator
- Get the app APK from skeezy site
- Sideload the APK into VM
- Set up Frida for runtime debugging
- Futz up the cert-pinning at runtime



Off we go

- 21:18:05.621493 IP 10.0.0.173.59590 > 10.0.0.129.8554: Flags [.], ack 176, win 347, options [nop,nop,TS val 130917697 ec
r 16054423], length 0
E..4..@.@.D.
...
.....!j.ME.&..K...[.....
...A....
21:18:05.622092 IP 10.0.0.173.59[illegible] > 10.0.0.129.8554: Flags [P.], seq 91:378, ack 176, win 347, options [nop,nop,TS val
130917697 ecr 16054423], l[illegible]: RTSP: DESCRIBE rtsp://10.0.0.129:8554/Video-31_SD RTSP/1.0
E..S..@.@.B.
...
.....!j.ME.&..K...[?S.....
...A....DESCRIBE rtsp://10.0.0.1[illegible]/Video-31_SD RTSP/1.0
Accept: application/sdp
CSeq: 3
Authorization: Diges[illegible]ername="user",realm="LIVE555 Streaming Media",nonce="3eb28c0807cef37af75f7a86b8a138c9",uri="rtsp
://10.0.0.129:8554/Video-31_SD",response="6d34d725967a090a339ef09c18a76bae"

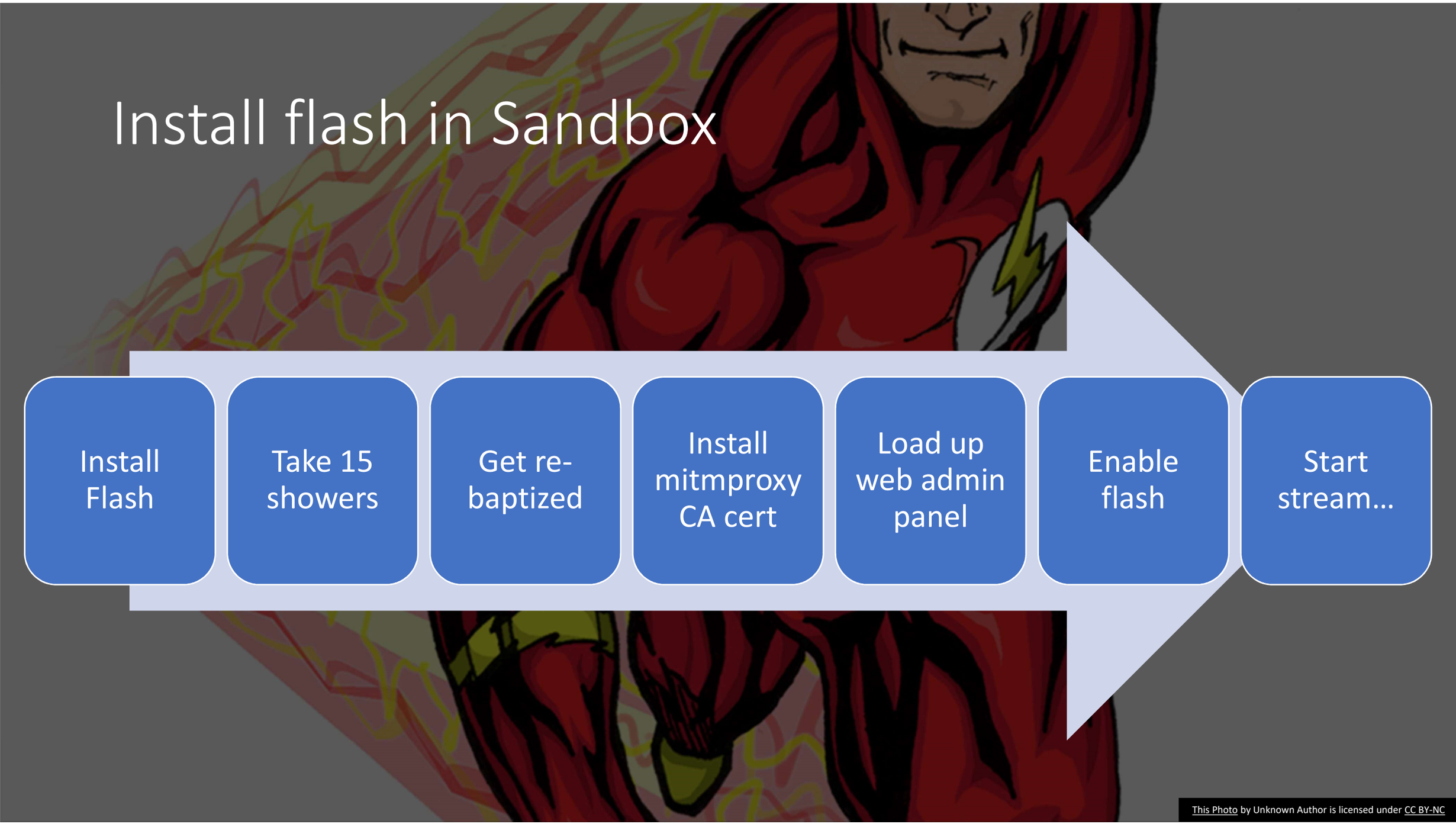
Fuck, nonce digest auth

Welp that's inconvenient

Fuck.
Fine.
Ok

My gut tells me that the password is generated, and not static, let alone low-entropy.

Install flash in Sandbox



SUCCESS!

Watch the API call fly out
once the Flash applet loads up

Back to tcpdump to fetch
some RTSP examples

```
 = md5(("%s:%s:%s" % ("user", "LIVE555 Streaming Media", '
.
675c99c47e82c996383b090576'
 = md5(("%s:%s" % ("DESCRIBE", "rtsp://10.0.0.129:8554/Video-31_SD
.
675c99c47e82c996383b090576'
!
0a67478097972e72937680614c'
(("%s:%s:%s" % (ha1, "b46a4eaaa1dec9cb338e561e85a12f17", ha2)).enc
a4b02ce21079220327cbc79830'
```

DOUBLE SUCCESS

“R6zw7butFpwzqUynA4TzT7o”

As expected, the RTSP password was completely random. >64 bits of entropy.

Act 6: Security

These systems don’t have fine-grained access controls

This means you’re giving very power credentials to your tools

This can be problematic, depending on where you’re storing things.

Credentials, such as RTSP, may never rotate, and may also work for other things.

So, what did we accomplish

We can do stuff

Without the apps

But still depend on the cloud service

We didn’t have to crack open any gear

Everything looks kosher from the service provider perspective

Value judgement time

- Vivint's architecture isn't total horseshit
- It's actually not bad
- The API responses are obtuse, and that's annoying, but at least they're JSON
- The panel itself has a lot of stuff listening, no idea what it all is
  - And the whole "the AWS stuff doesn't validate certs" is horrifying, even with the PGP measures
- The permissions granularity leaves a great deal to be desired
- A useful tool would be a permissions proxy that allowed fine-grained controls

Other notable notes

Halloween. Doorbells. Completely unresponsive.

What does this mean for other things?

If the queues are full of doorbell rings, what if I can't unlock my house?

And by the grace of Mak, I’m finally done.

???