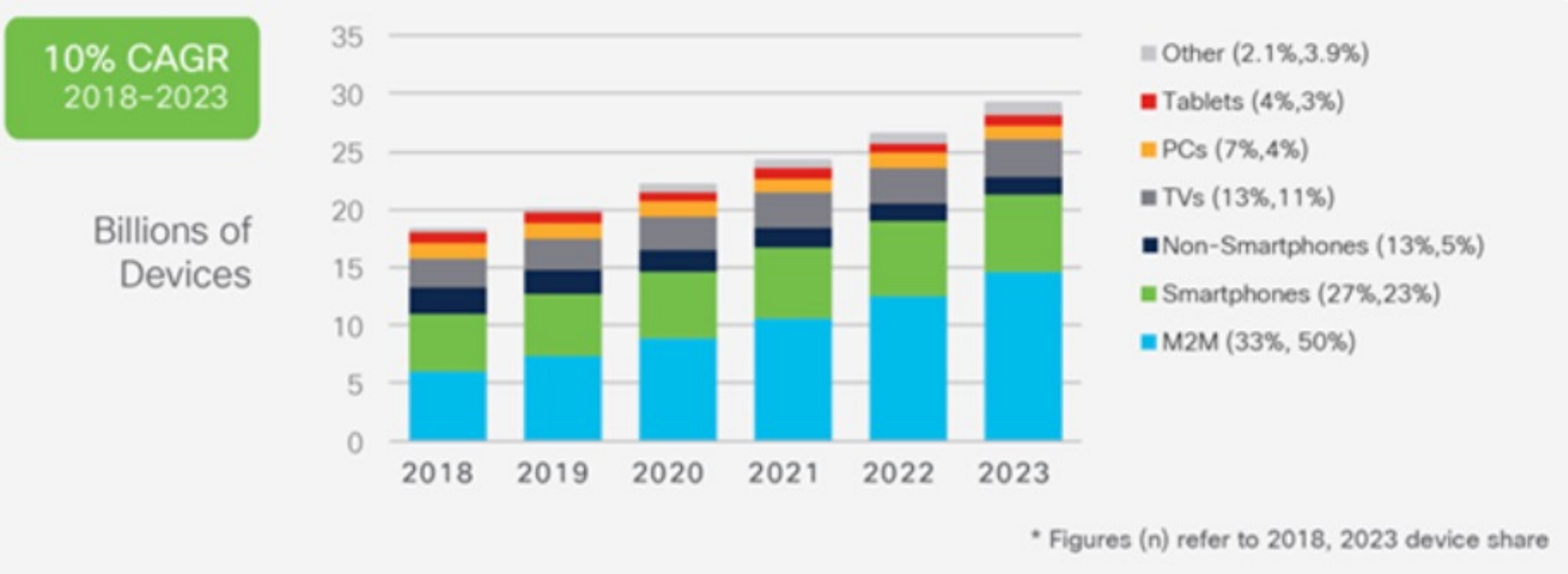
The bridge to possible

Onboarding Securing the Internet of Things

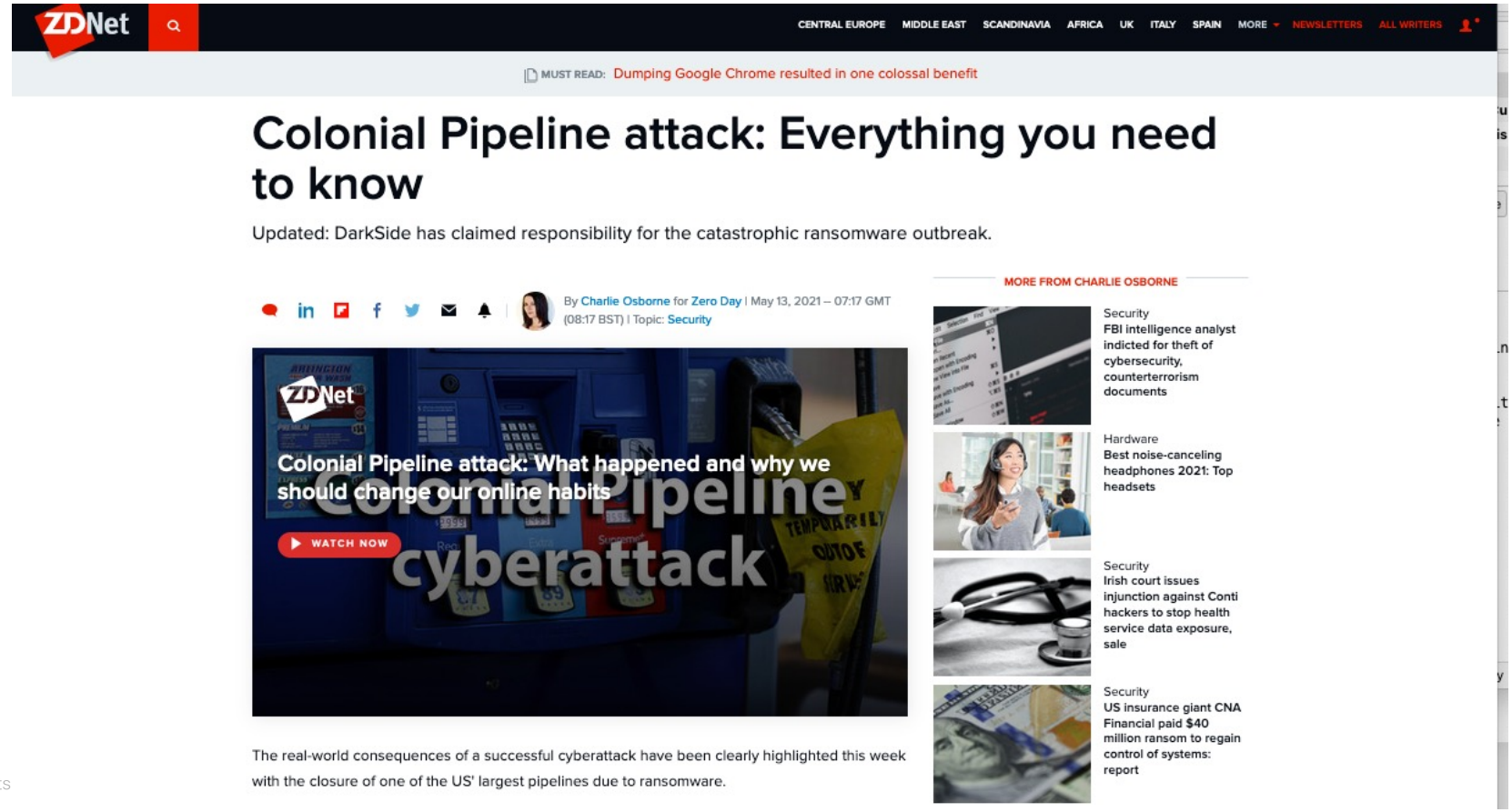
Eliot Lear
Principal Engineer
25 May 2021

The nice thing about Things is that there are so many of them

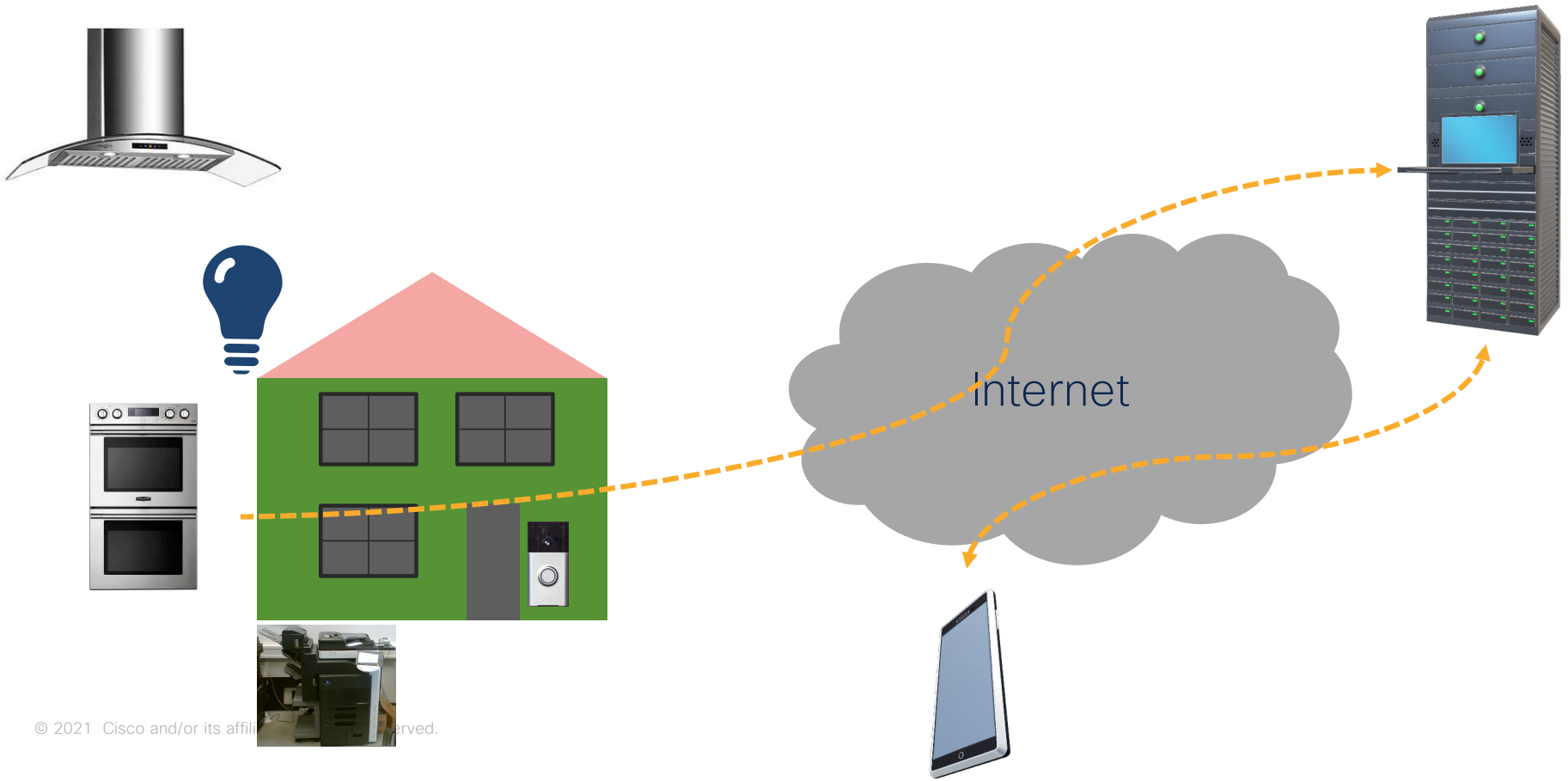


Let’s talk about an oven

For OT to be secure, IT has to be secure



What A House Might Have



The nice thing about definitions of Internet of Things is that there are so many of them

Today's definition of IoT

A device with a CPU, some memory, and a transceiver that impacts detects or effects some observable physical activity beyond itself.

Let’s establish some basic questions

What is this thing?	• What is the device’s identity? Does this particular thing belong on the network? • What type of thing is it?
Who is responsible for it?	• If something breaks, who should be called?
What access does it need?	• With which devices should it communicate?
Is it doing what it should be doing?	• With which devices is it actually communicating? • Is it behaving as designed?

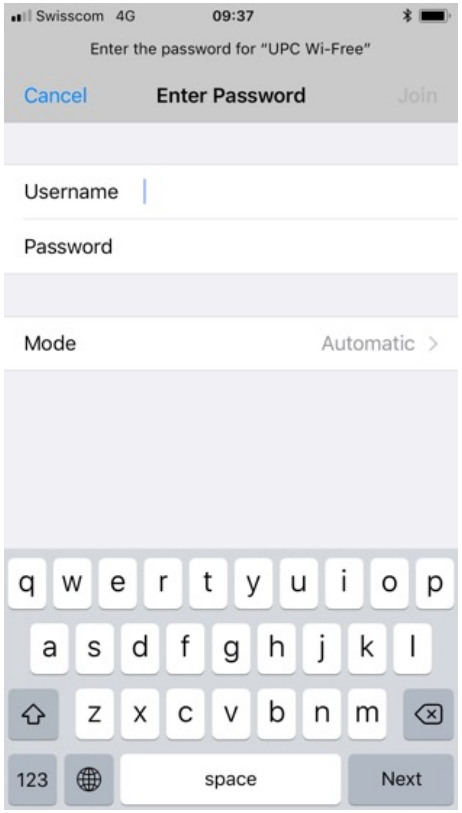
Let’s establish some basic questions

What is this thing?

- What is the device’s identity? Does this particular thing belong on the network?
- What type of thing is it?

The nice thing about connection technologies is that there are so many of them

Why is IoT different?



Basic Requirement for Onboarding: Trust

"Can that network prove to me that I should join it?"

"Is that thing supposed to join **my** network?"

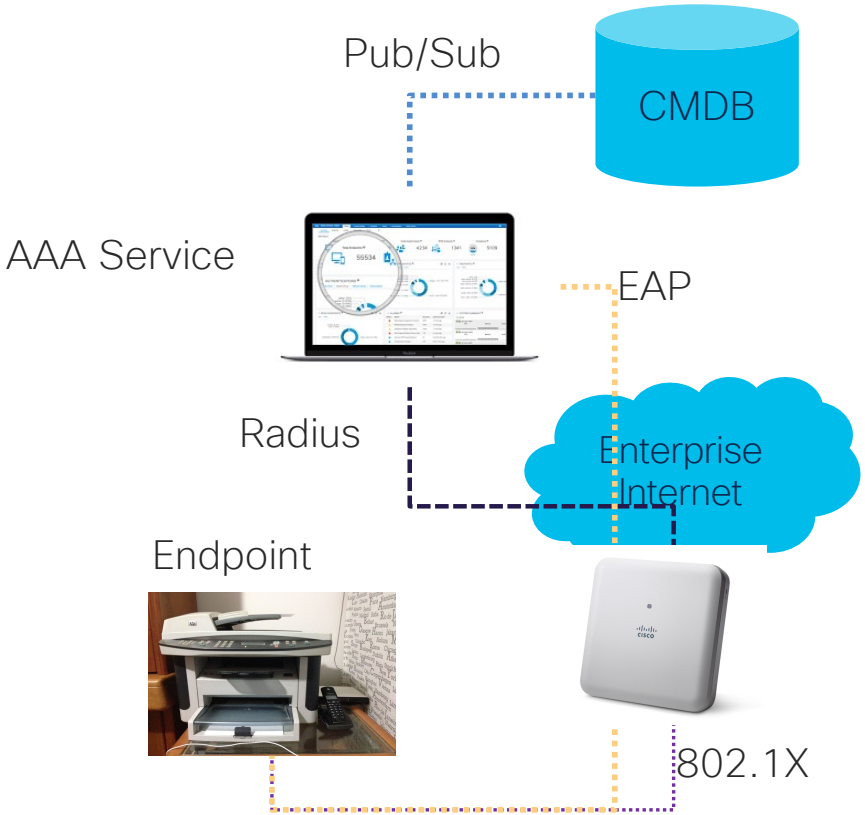
The Easy Version of Trust: a wire!

Threat model assumptions:

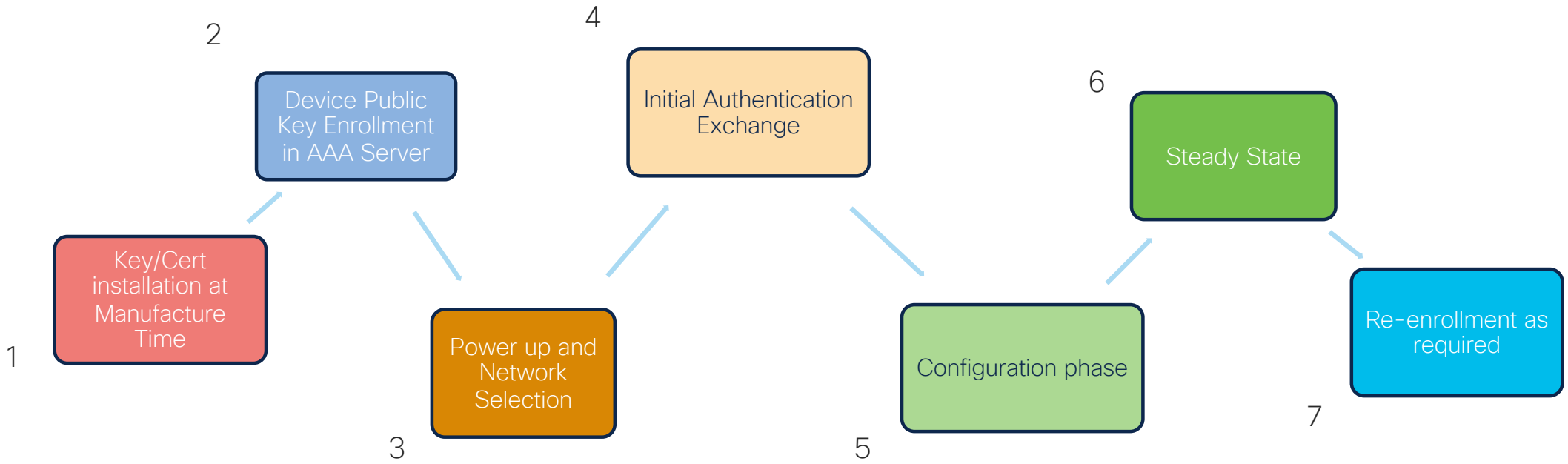
- Physical security
- Supply chain security

What's there now?

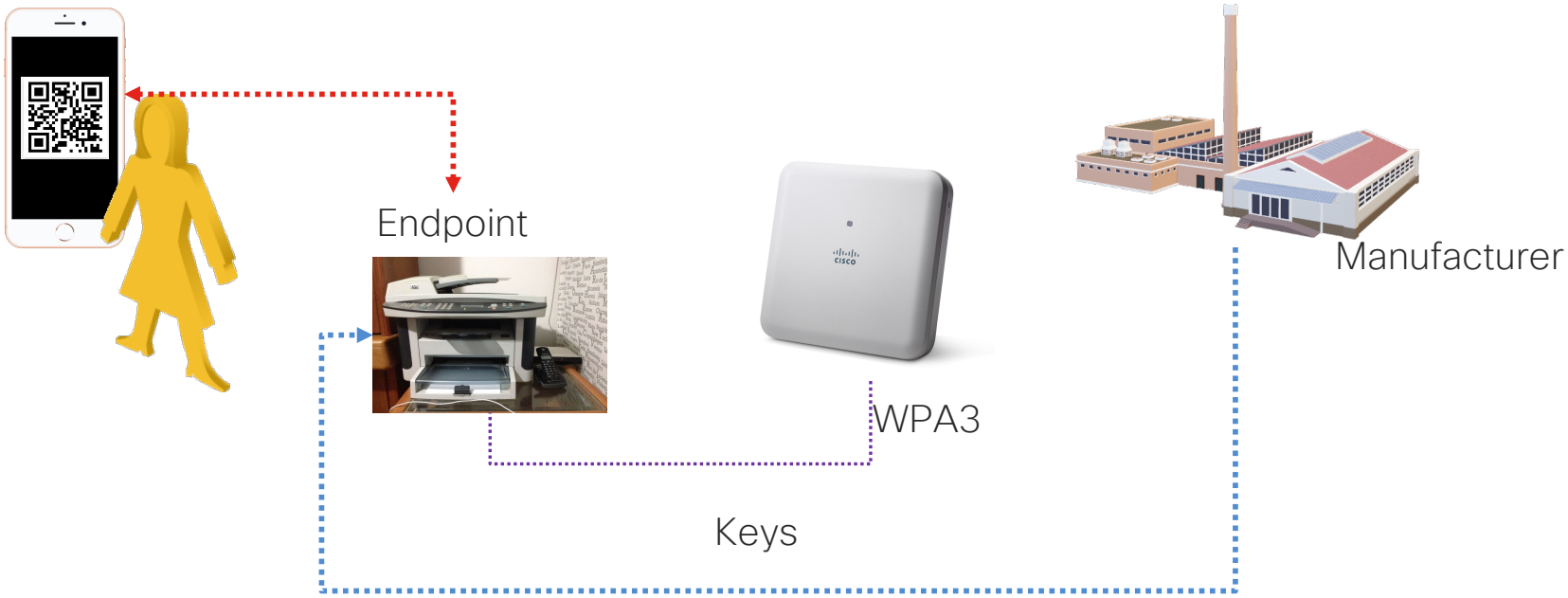
- The IoT Device
- AAA / policy server
- Radius and EAP control channels
- Wireless AP or switch
- An inventory control system
- End goal: steady state with EAP



Generic Onboarding Flow



Wifi Alliance DPP Architecture



Bootstrap Information

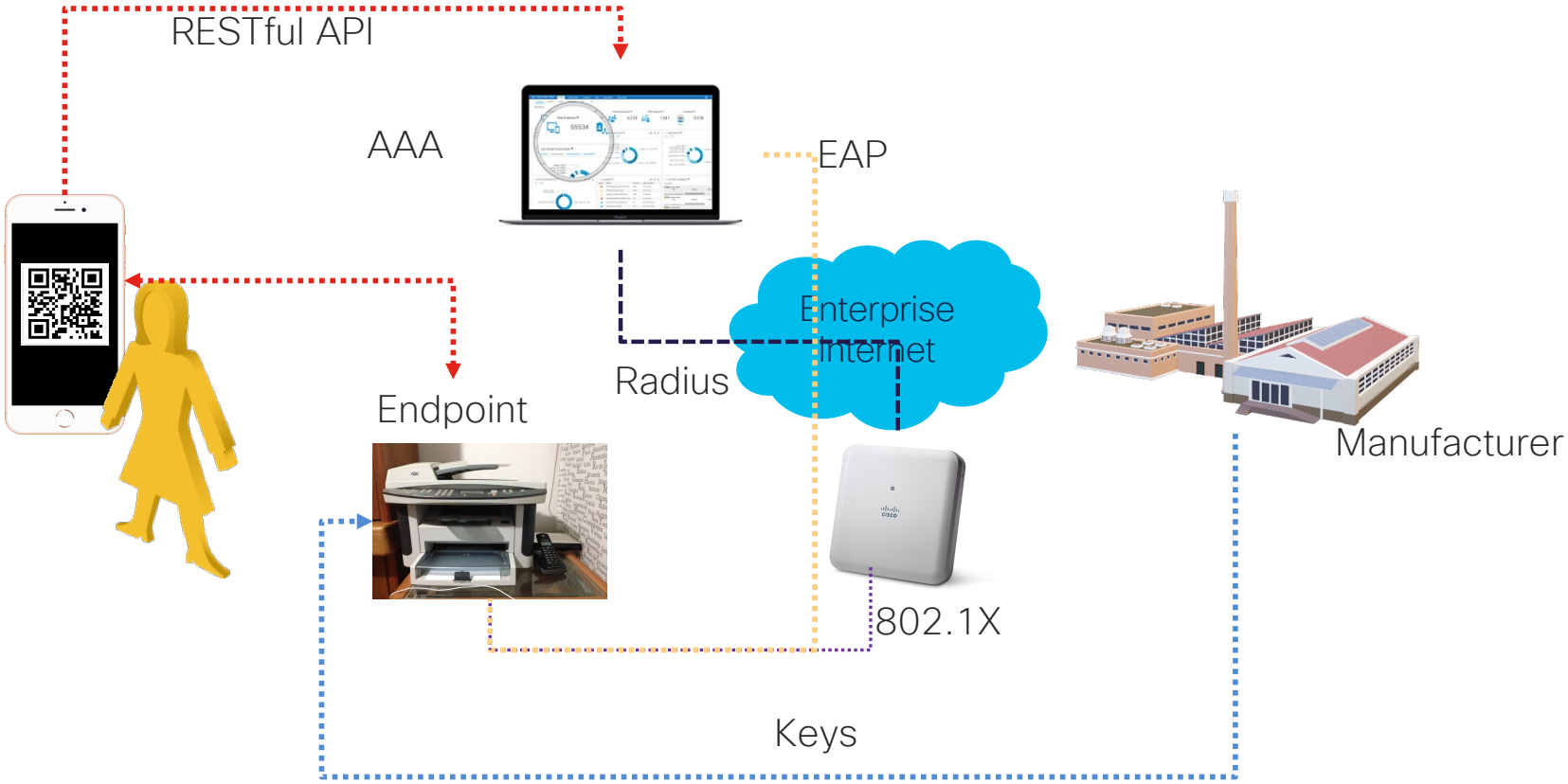
Information

MAC Address

DPP:I:SN=4774LH2b4044;M:010203040506;
K:MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIg
ADURzxmttZoIRIPWGoQMV00XHWCAQIhXr
uVWOz0NjlkIA=;;

PublicKeyInfo

DPP/TEAP architecture (for the future)

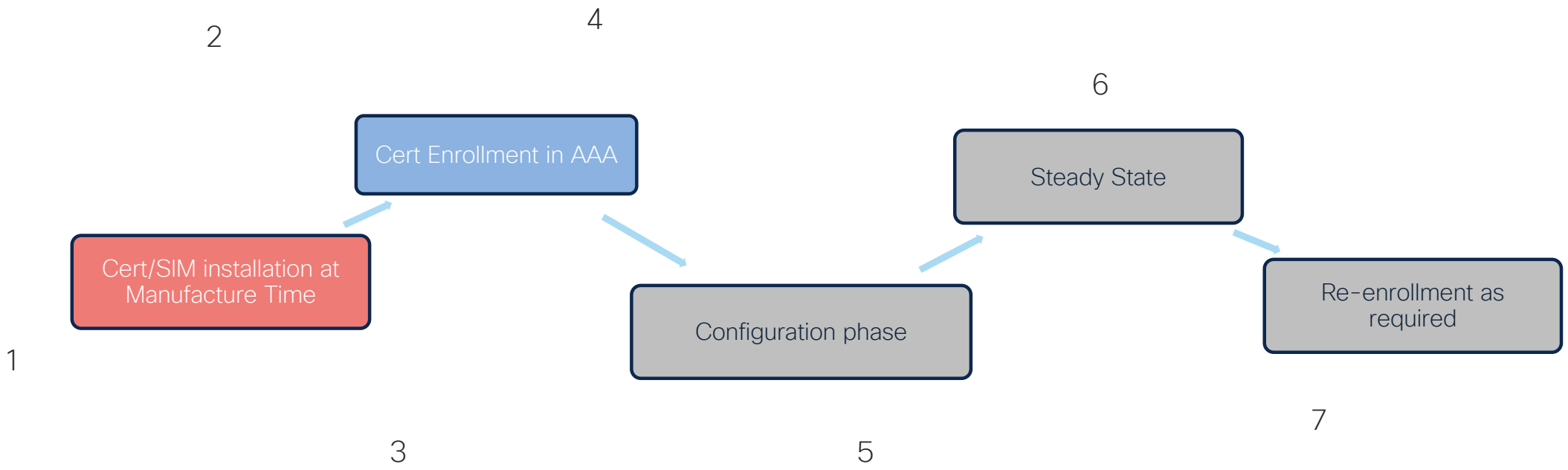


DPP Advantages and Disadvantages

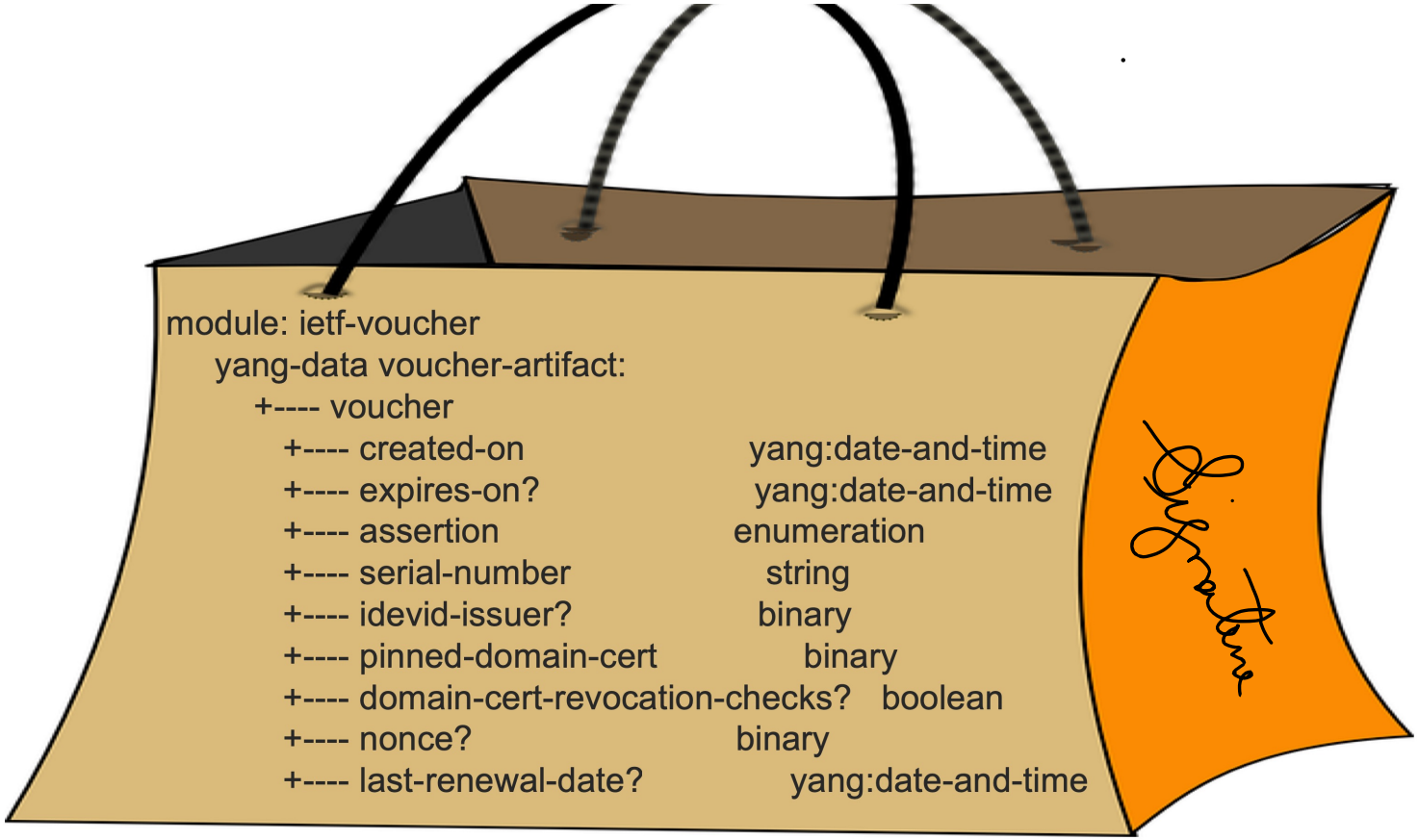
Advantages
One step provisioning with an industry standard QR code
Can work with or without Internet connectivity
Ownership transfers are a matter of resetting the device and reusing the QR code
No PKI required

Disadvantages
Some chip set limitations in phones and tablets to process L2 packets
Still one step, not zero step provision (but this can be further developed)
No native anti-counterfeit capabilities
Primarily developed for wireless technologies

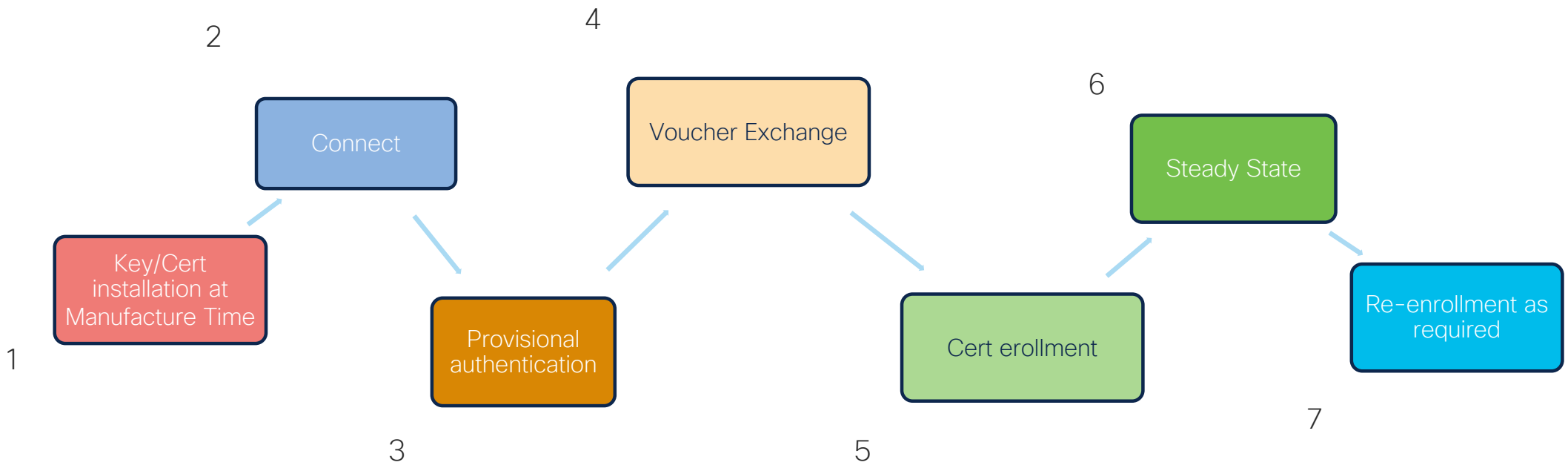
Pre-Provisioned/SIM/e-SIM Onboarding Flow



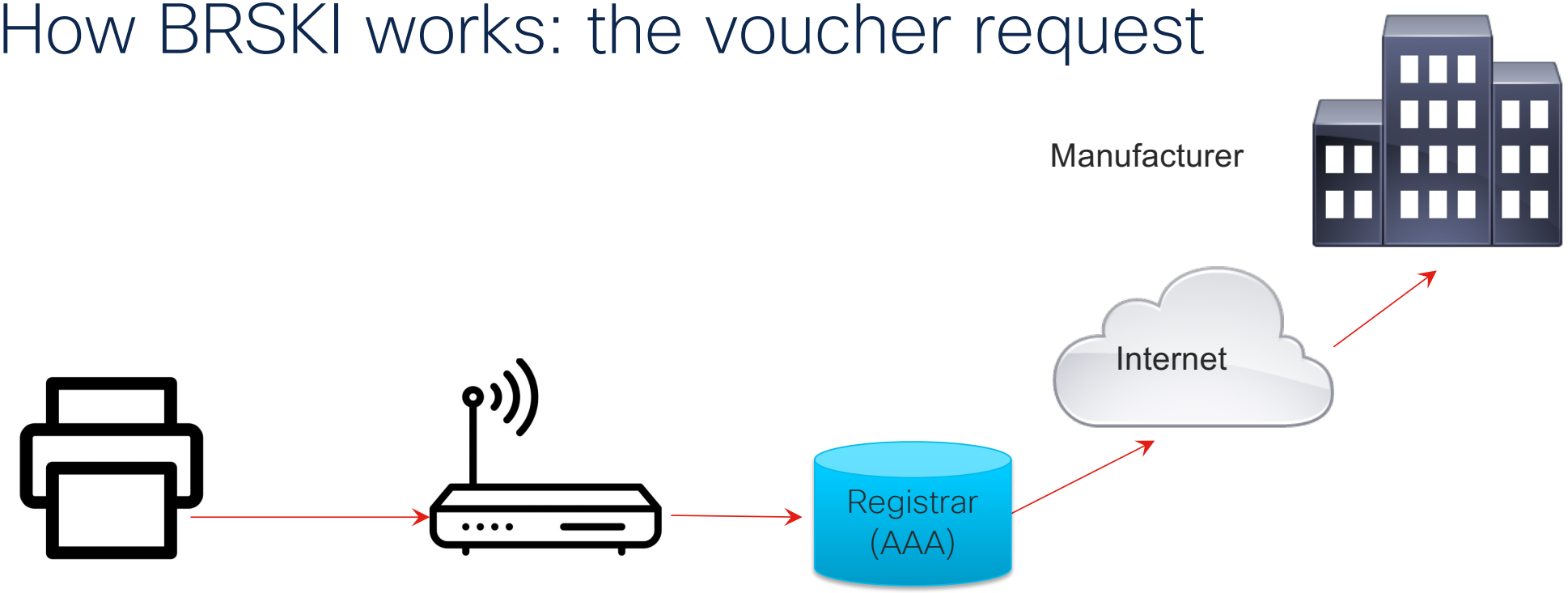
Bootstrapping Remote Key Infrastructure (BRSKI): It centers around a voucher (RFC 8366)



Bootstrapping Remote Key Infrastructure

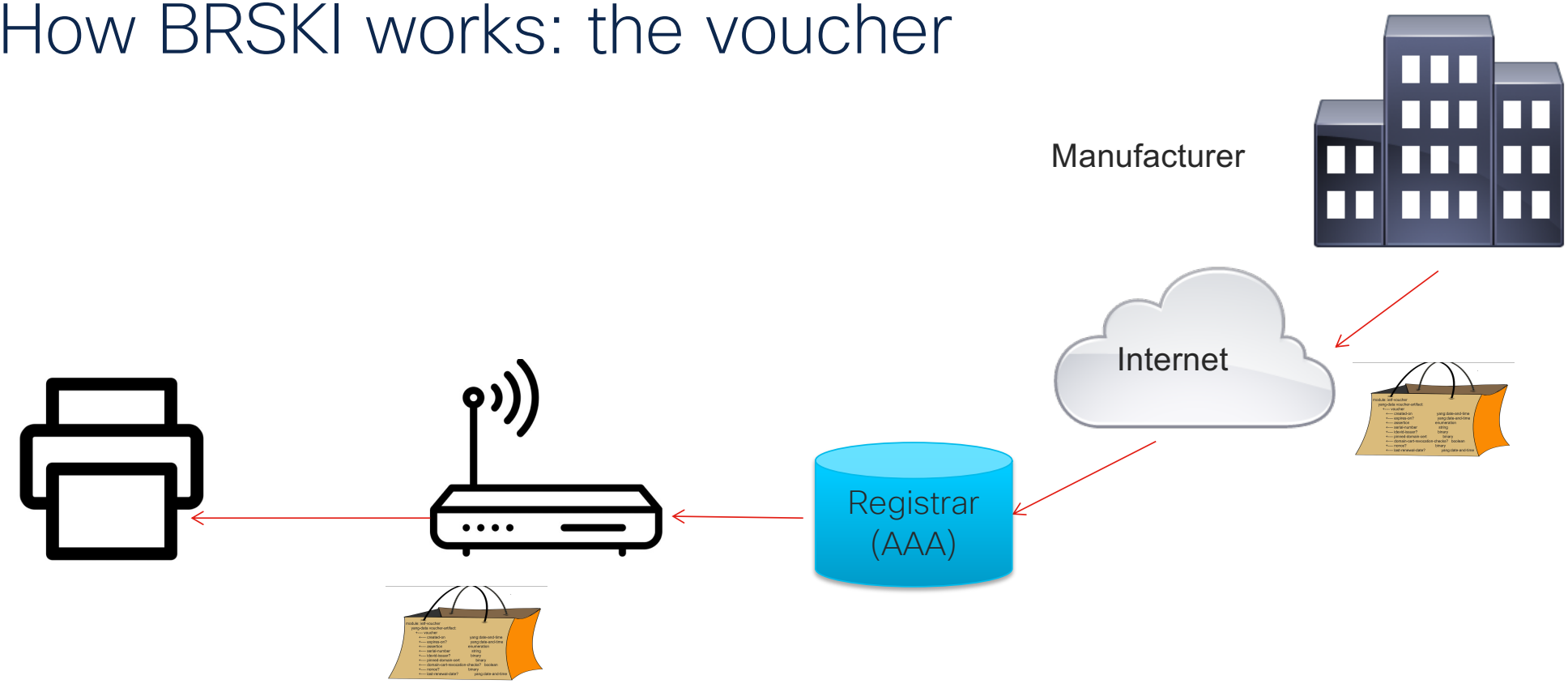


How BRSKI works: the voucher request



22

How BRSKI works: the voucher



Next Step: provision a deployment certificate

BRSKI Advantages and Disadvantages

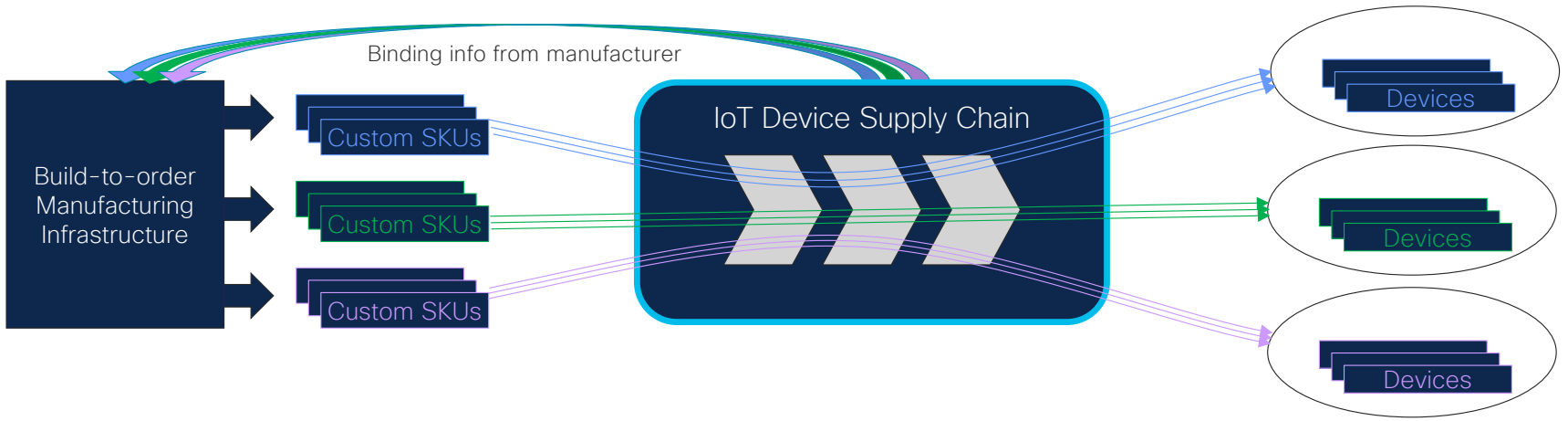
Advantages
Can be **zero** step provisioning
Has basic anti-counterfeiting measures
Leverages existing enterprise infrastructure

Disadvantages
Requires a manufacturer service and Internet accessibility
Ownership transfer required
Requires certificate infrastructure
Needs more work to support wireless

FDO delivers Binding at End of the Supply Chain

Onboard without FDO
IoT device software and security customization happens during manufacturing

Result:
Complicated build-to-order manufacturing infrastructure, many SKUs, small lot sizes, long lead times, higher cost

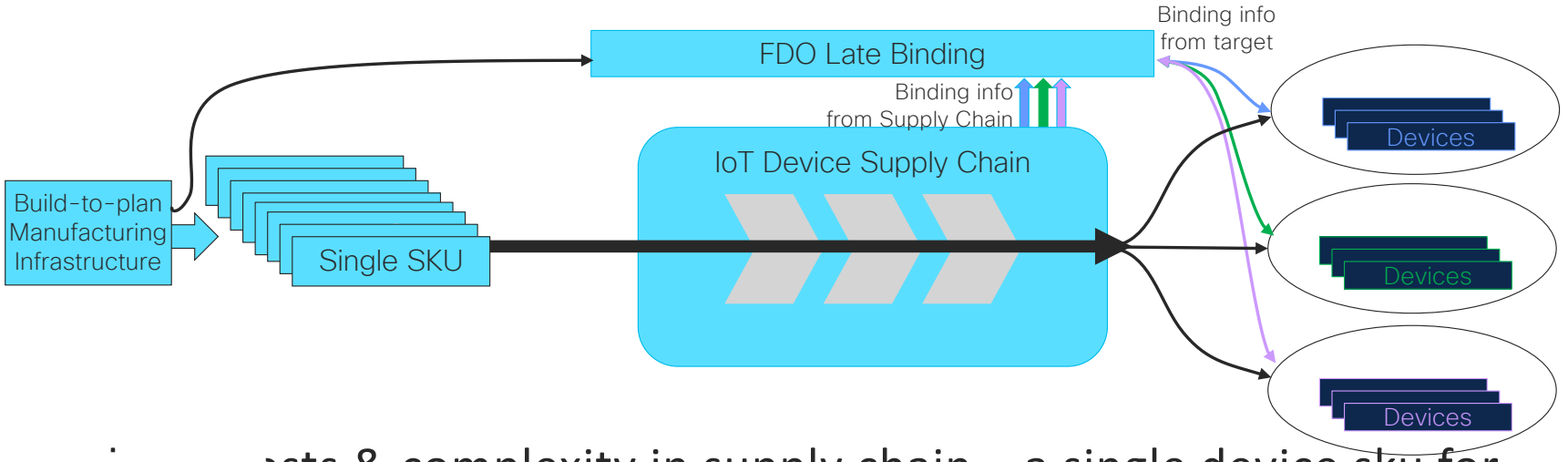


Onboard with FDO
IoT device software and security customization happens at the end of the supply chain

Benefits:

Simplified build-to-plan manufacturing infrastructure, fewer SKUs, large lot sizes, enable stocking distributors, low customization cost

Result: *Increased supply chain volume and velocity*



osts & complexity in supply chain – a single device sku for all customers

FDO Advantages and Disadvantages

Advantages
Can be **zero** step provisioning
Has basic anti-counterfeiting measures
Leverages existing enterprise infrastructure
Creates late binding for SKU management

Disadvantages
Requires a manufacturer-referenced rendezvous service
Ownership transfer required
Doesn't address network-layer onboarding

Let’s establish some basic questions

Who is responsible for it?

- If something breaks, who should be called?

Let’s establish some basic questions

What access does it need?

- With which devices should it communicate?

A Common Threat: Printers

Study cites multi-function printers as some of the most dangerous members of the IoT family

What Sort of Access Do These Printers Require?

From	To	Protocol	Source Port	Destination Port(s)
Printer	xmpp009.hpeprint.com	TCP		80, 443, 5222,5223
Printer	DNS Server	UDP		53
Printer	chat.hpeprint.com	TCP		80,443
Printer	224.0.0.251/32	UDP		5353
Printer	220.0.0.252/32	UDP		5355
Printer	h10141.www1.hp.com	TCP		80
Printer	Local Networks	UDP	5353	
Printer	Local Networks	TCP	80	

What's the scope of the problem?










The nice thing about the number of types of things is that there are so many of them

Learned and Declared Models

	What is it?	Benefits	Drawbacks
Learned	Cisco-provided Expertise + your deployment knowledge	• Required for “brownfield” deployments for years to come • No ecosystem requirements	• Requires relearning from time to time • Can be compute intensive
Declared	Manufacturer-provided expertise plus your deployment knowledge	• Authoritative source of vendor information • Combines policy and classification	• Ecosystem must adopt these approaches

Good news! Use both!

Declared Approach: Assumptions and Assertions

Assumptions	Assertions
A Thing has a single use or a small number of uses.	Because a Thing has a single or a small number of intended uses, all other uses must be unintended.
Things are tightly constrained. Very little CPU, memory, and battery.	Any intended use can be clearly identified.
Network administrators are the ultimate arbiters of how their networks will be used	Manufacturers are in a generally good position to provide guidance to administrators.
Even those Things that can protect themselves today may not be able to do so tomorrow	A mechanism is needed to protect devices that may have vulnerabilities.

Translating intent into config

Any intended use can be clearly identified by the manufacturer

access-list 10 permit host controller.mfg.example.com

All other uses can be warned against in a statement by the manufacturer

access-list 10 deny any any

Introducing Manufacturer Usage Descriptions (MUD)

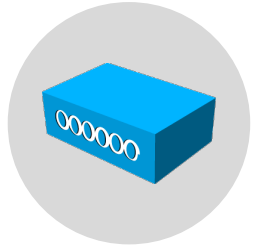
A URL:

https://manufacturer.example.com/mydevice.json

A MUD File:

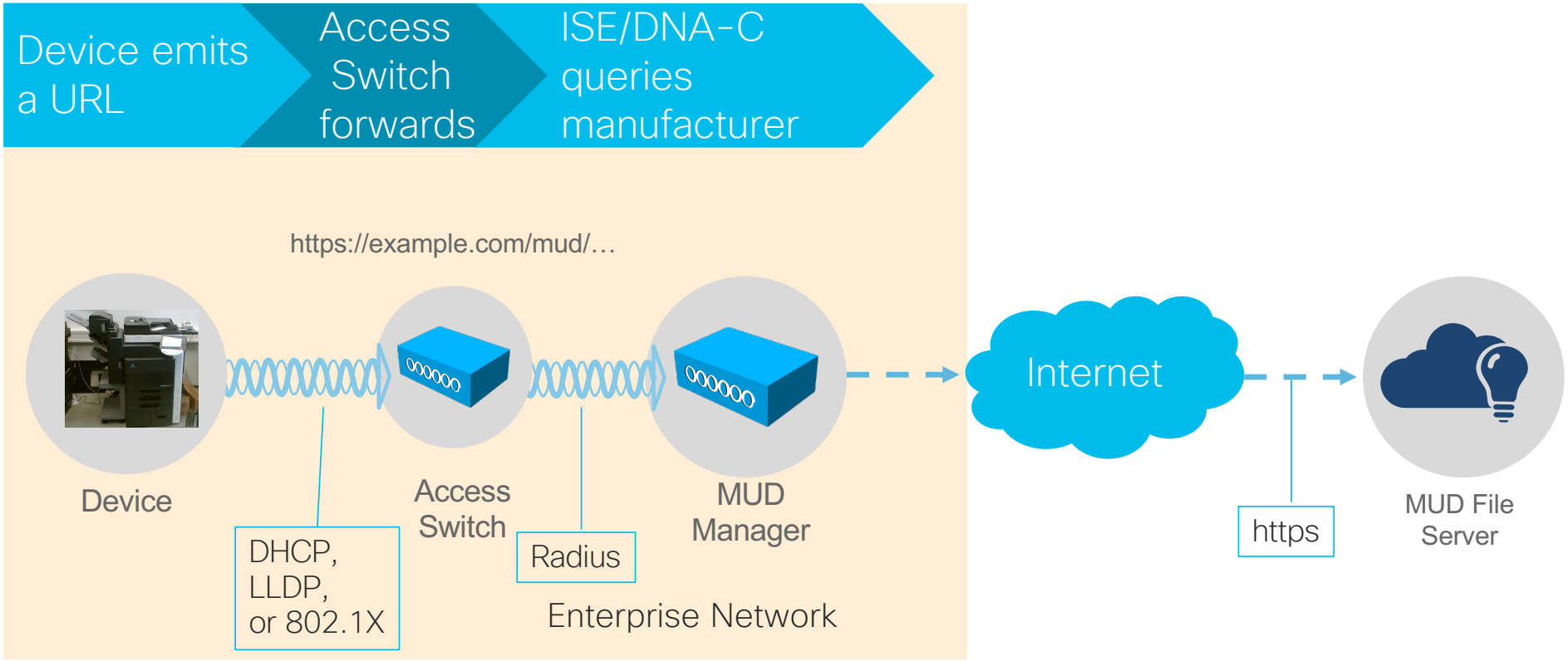
```
...
"ace": [   {
          "name": "cl0-todev",
          "matches": {
            "ietf-mud:mud": {
             "my-controller": [
              null
             ]
            } },
          "actions": {
            "forwarding": "accept"
          } }  ]
...
```

The MUD Manager:

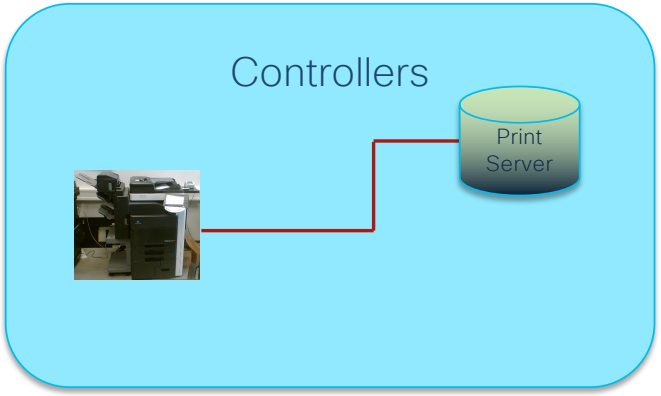


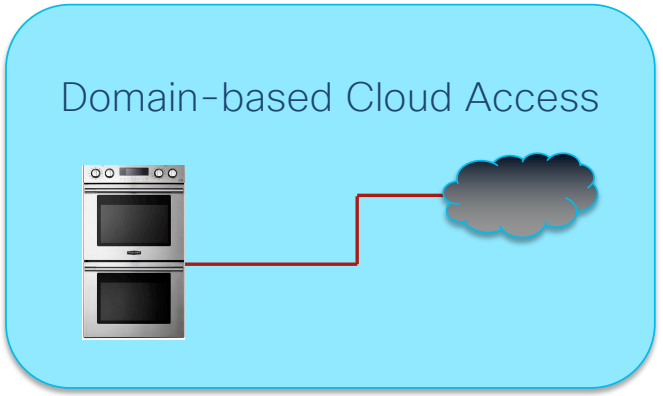
The MUD File Server:

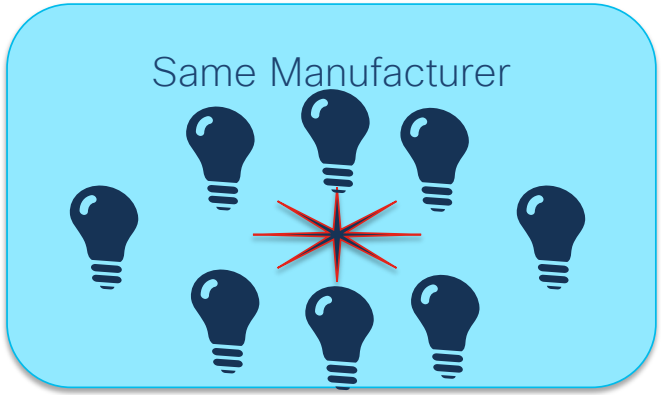
Expressing Manufacturer Usage Descriptions



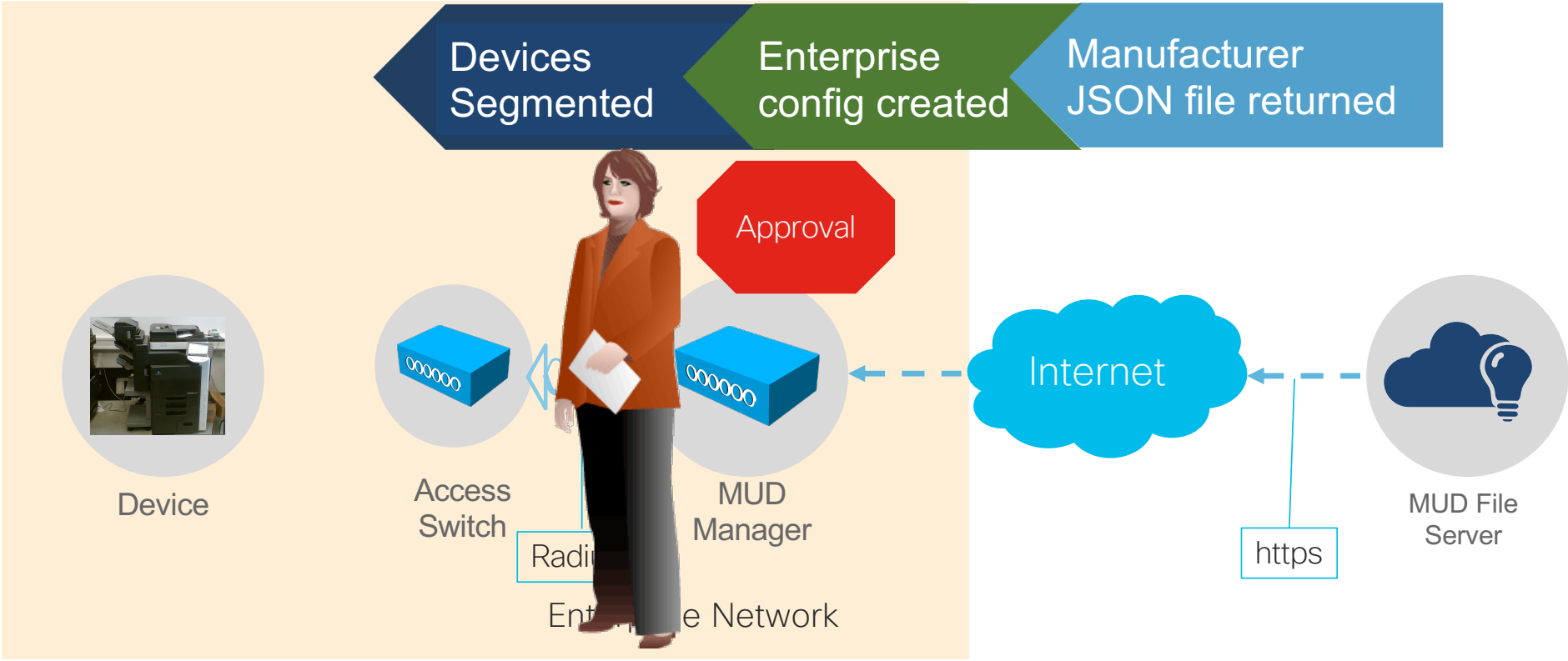
What Classes of Endpoints MUD provides access to



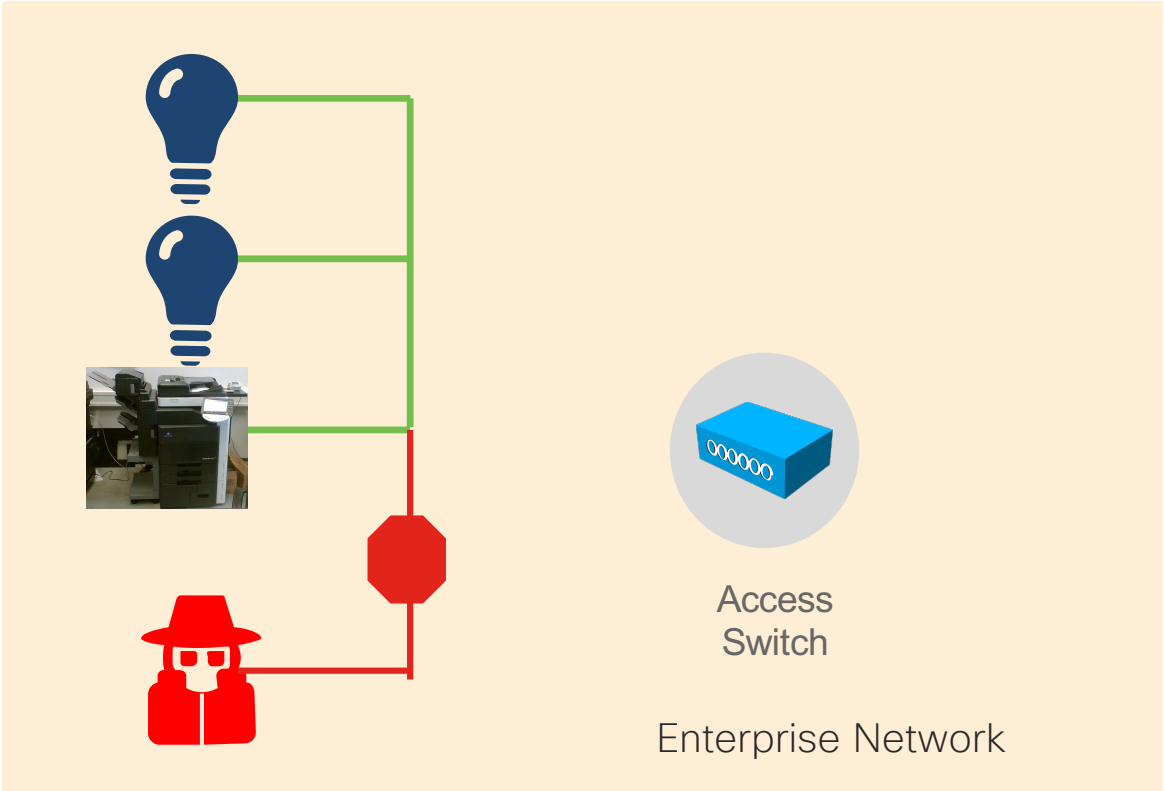




Expressing Manufacturer Usage Descriptions

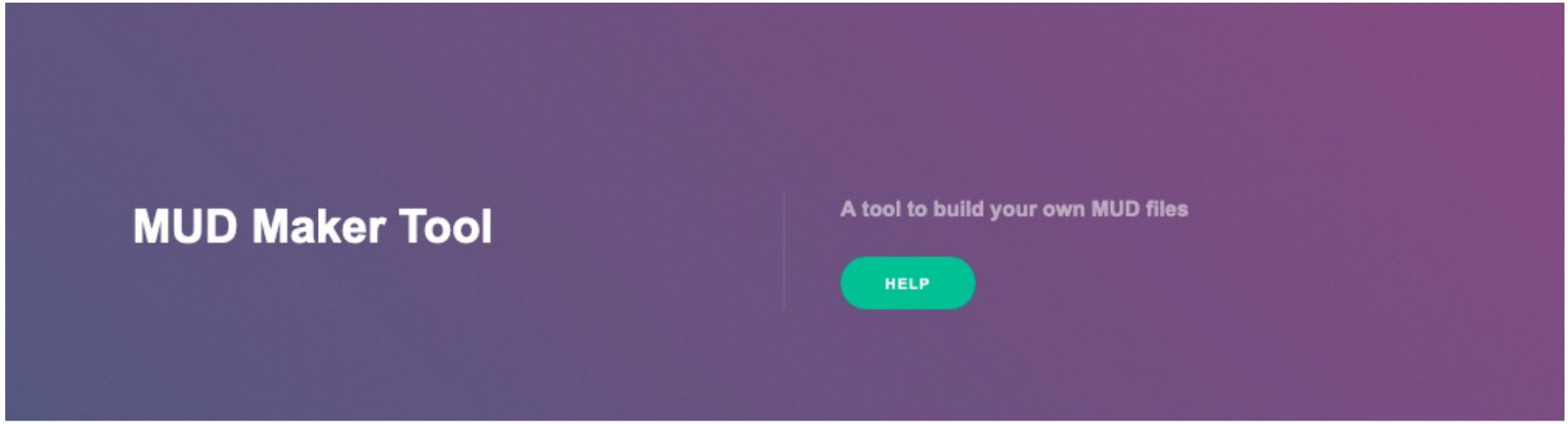


Results: Micro-segmentation of that printer



- Visibility of what's on the network
- Access limited to devices based on manufacturer recommendations
- Policy choices easily identified by MUD file
- Hacked devices can't probe for holes
- An additional layer of security
  - BUT- manufacturers should still **always** secure their devices

Let’s make a MUD file and see what that means



Please enter host and model the intended MUD-URL for this device:

https:// lighting.molex.com / (model name here->) lightcontroller

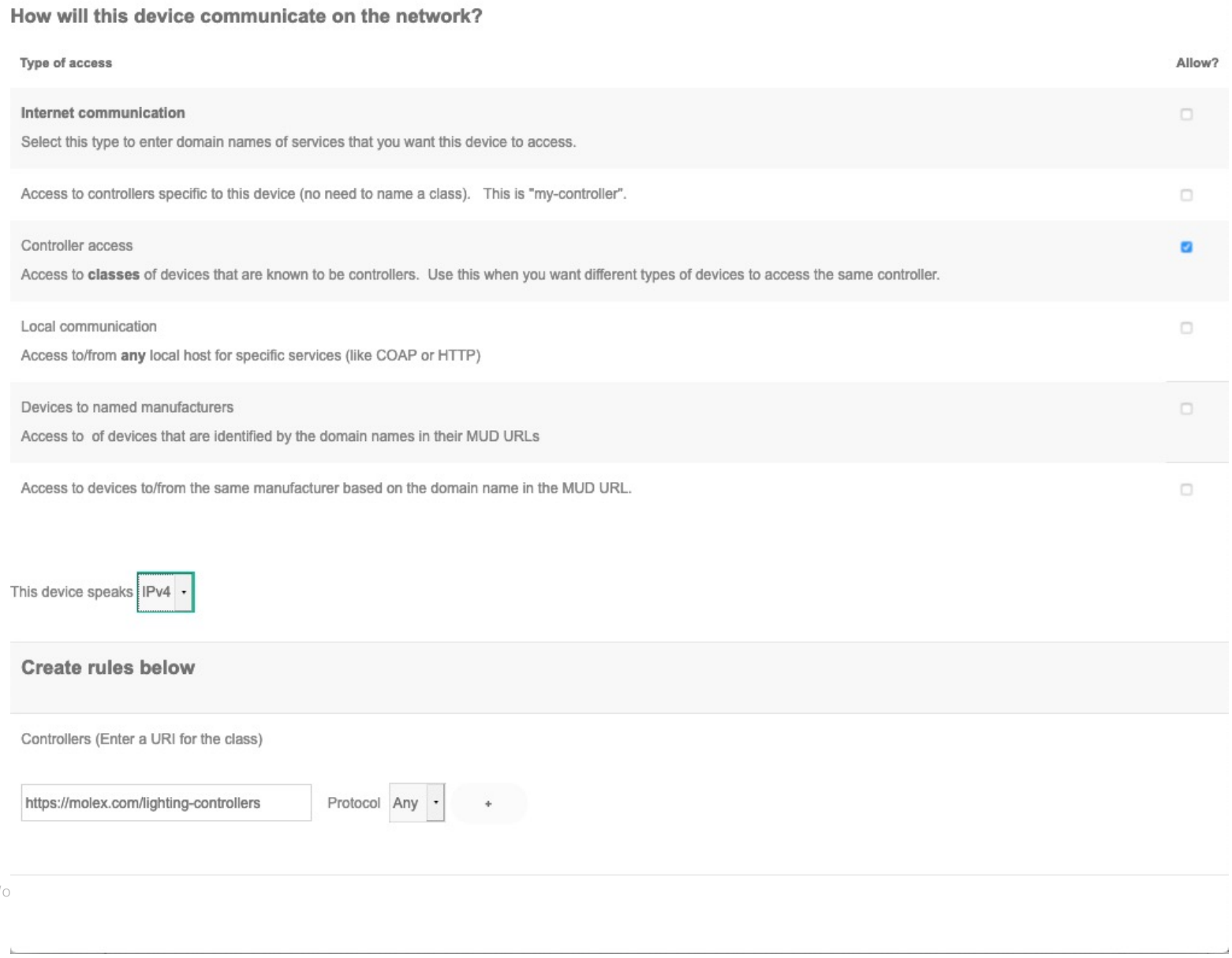
Manufacturer Name Molex

Please provide a URL to documentation about this device:

https://molex.com

Please enter a short description for this device:

Molex Luminaire



Your MUD file is ready!

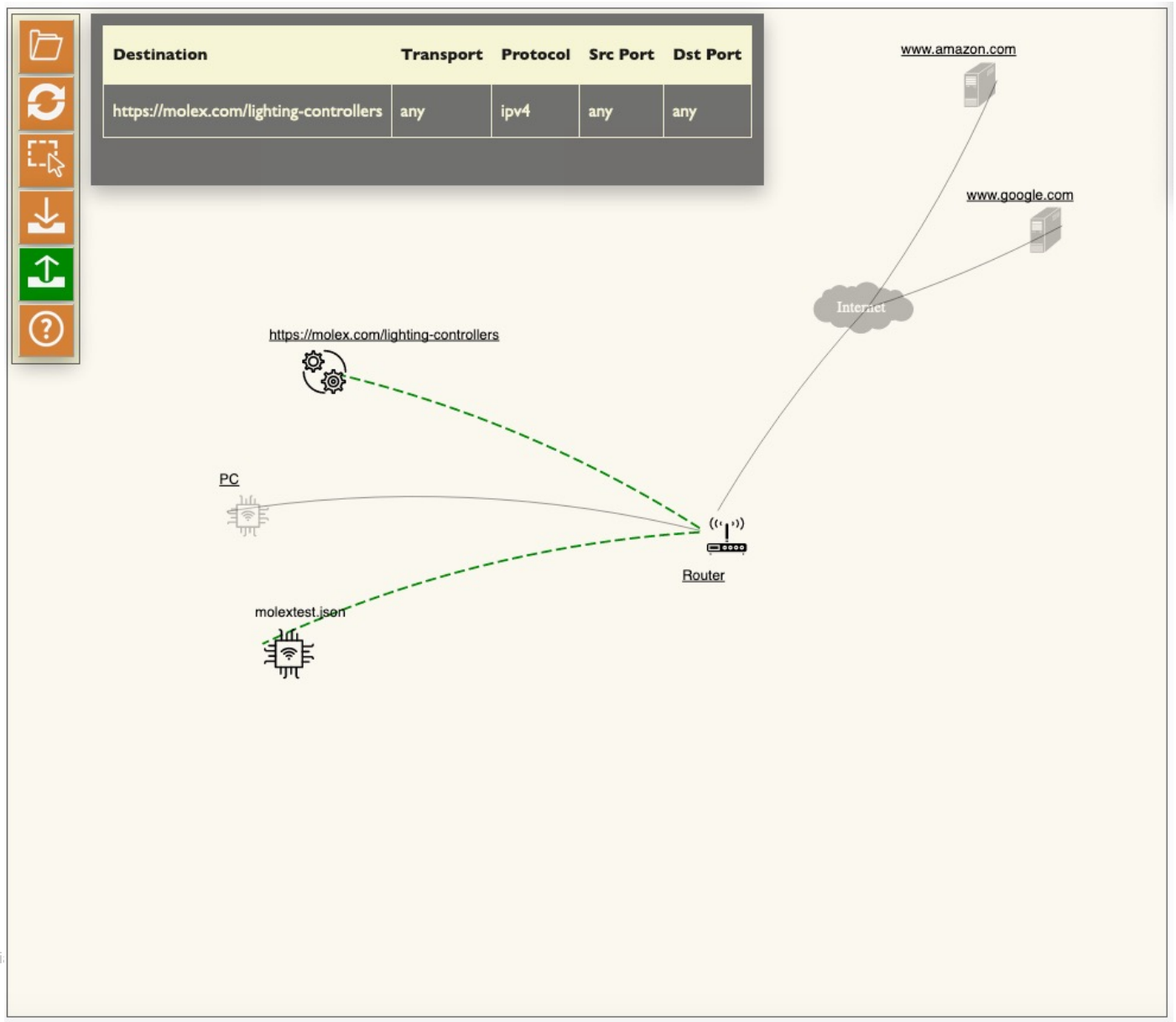
Congratulations! You've just created a MUD file. Simply Cut and paste beween the lines and stick into a file. Your next steps are to sign the file and place it in the location that its corresponding MUD URL will find. To sign the files, do the following:

- Get a certificate with which to sign documents/email.
- Use OpenSSL as follows:
  openssl cms -sign -signer YourCertificate.pem -inkey YourKey.pem -in YourMUDfile.json -binary -outform DER -certfile intermediate-certs.pem -out YourSignature.p7s
- Place the signature file and the MUD file on your web server (it should match the MUD-URL)

Would you like to download this file? Download

Visualize this device in a network? Visualize

```
{
  "ietf-mud:mud": {
    "mud-version": 1,
    "mud-url": "https://lighting.molex.com/lightcontroller",
    "last-update": "2019-10-14T14:09:55+00:00",
    "cache-validity": 48,
    "is-supported": true,
    "systeminfo": "Molex Luminaire",
    "mfg-name": "Molex",
    "documentation": "https://molex.com",
    "model-name": "lightcontroller",
    "from-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "mud-37278-v4fr"
          }
        ]
      }
    },
    "to-device-policy": {
      "access-lists": {
        "access-list": [
          {
            "name": "mud-37278-v4to"
          }
        ]
      }
    }
  },
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "mud-37278-v4to",
        "type": "ipv4-acl-type",
        "aces": {
          "ace": [
            {
              "name": "ent0-todev",
              "matches": {
                "ietf-mud:mud": {
                  "controller": "https://molex.com/lighting-controllers"
                }
```



Let’s establish some basic questions

Is it doing what it should be doing?

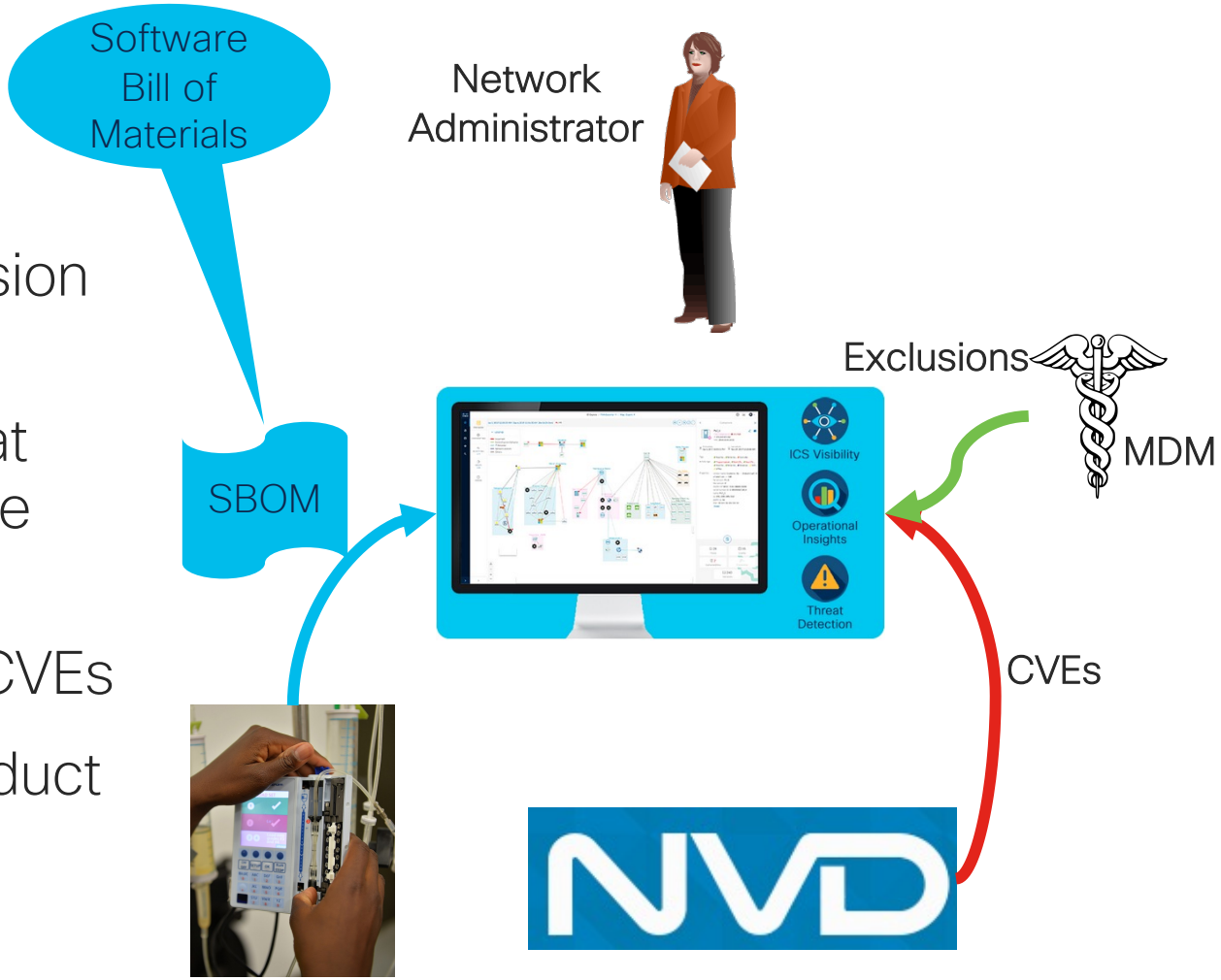
- With which devices is it actually communicating?
- Is it behaving as designed?

What happens when things break?

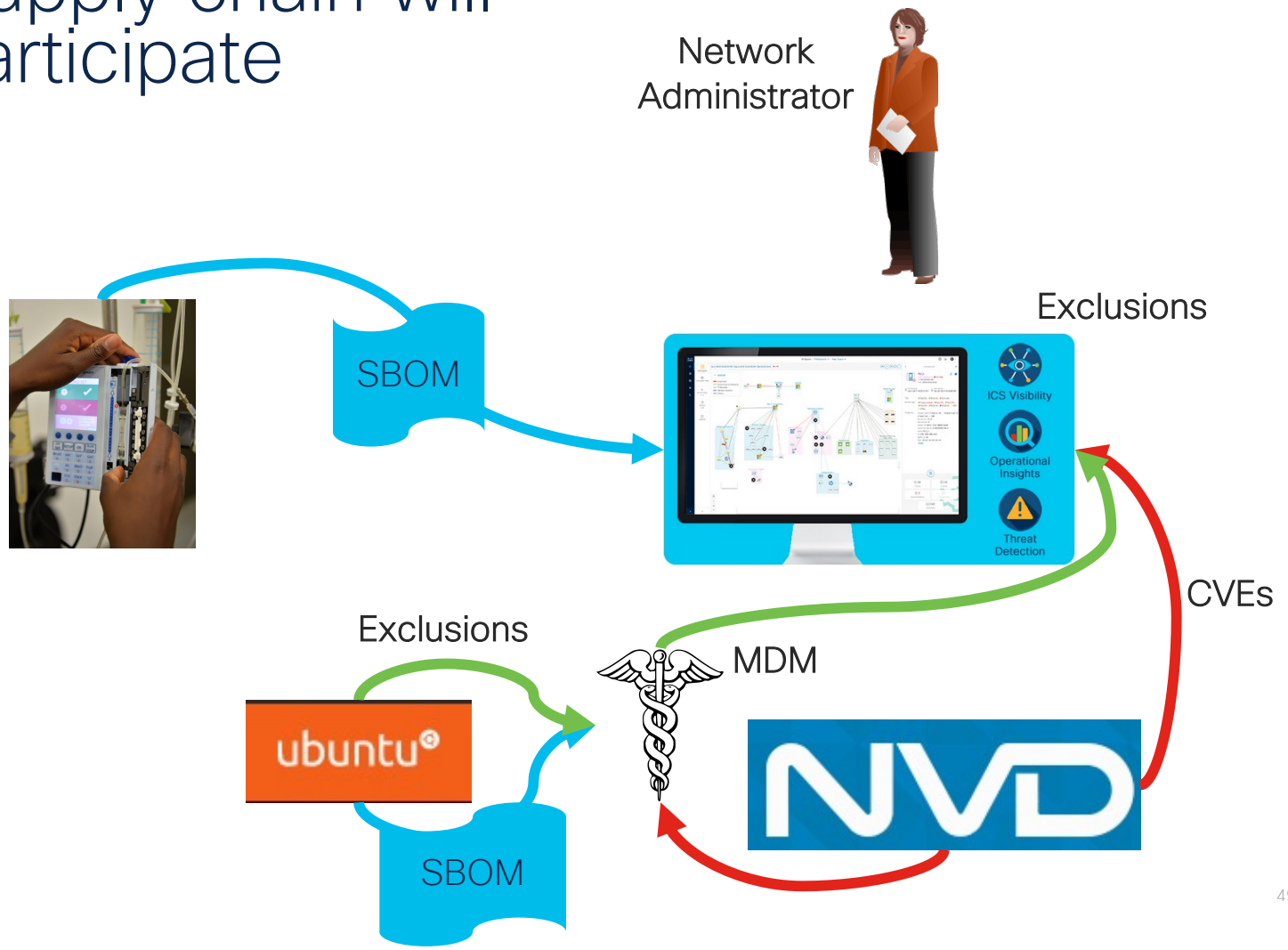
Knowing one has a problem is the first step to fixing it.

What do we need?

- A list of all software in the device, complete with version information
- A list of CVEs to know what software may be vulnerable
- A **name** for packages that matches to what is listed CVEs
- A list of packages in a product that have been fixed
- A way to retrieve **all** of this



The whole supply chain will **eventually** participate



Software inventory: Linux Foundation SPDX

```
## 2.4 Primary Component (described by the SBOM)
        PackageName: INFUSION
        SPDXID: SPDXRef-INFUSION
        PackageComment: <text>PURL is pkg:supplier/ACME/INFUSION@1.0</text>
        ExternalRef: PACKAGE-MANAGER purl pkg:supplier/ACME/INFUSION@1.0
        PackageVersion: 1.0
        PackageSupplier: Organization: ACME
        Relationship: SPDXRef-DOCUMENT DESCRIBES SPDXRef-INFUSION
        Relationship: SPDXRef-INFUSION CONTAINS NONE
        PackageDownloadLocation: NOASSERTION
        FilesAnalyzed: false
        PackageLicenseConcluded: NOASSERTION
        PackageLicenseDeclared: NOASSERTION
        PackageCopyrightText: NOASSERTION
```

Names

Very similar to what dpkg uses in Ubuntu/Debian.

SPDX Good and Bad

Good

- Very simple to extend
- YAML-esque
- Easy for humans to read
- Lots of fields and relationships defined
- Gives you a BOM
- Tooling available: https://spdx.dev/resources/tools/

The Bad

- Doesn't tell you what is fixed or not
- They haven't thought through how they want secure the SPDX file itself
- Lots of discussion of GPG key usage
  - This is fine for self-contained and managed linux distributions
  - It's lousy for anything else

CycloneDX

```
<?xml version="1.0" encoding="UTF-8"?>
<bom xmlns="http://cyclonedx.org/schema/bom/1.2" serialNumber="urn:uuid:3e671687-395b-41f5-a30f-a58921a69b79"
version="1">
  <components>
    <component type="library">
      <publisher>Apache</publisher>
      <group>org.apache.tomcat</group>
      <name>tomcat-catalina</name>
      <version>9.0.14</version>
      <hashes>
        <hash alg="MD5">3942447fac867ae5cdb3229b658f4d48</hash>
        <hash alg="SHA-1">e6b1000b94e835ffd37f4c6dcbdad43f4b48a02a</hash>
        <hash alg="SHA-256">f498a8ff2dd007e29c2074f5e4b01a9a01775c3ff3aeaf6906ea503bc5791b7b</hash>
        <hash alg="SHA-
512">e8f33e424f3f4ed6db76a482fde1a5298970e442c531729119e37991884...ffab4f94...7b7ee11fccd074eeda0634d71697d6f88a460dce0a
c8d627a29f7d1282</hash>
      </hashes>
      <licenses>
        <license>
          <id>Apache-2.0</id>
        </license>
      </licenses>
      <purl>pkg:maven/org.apache.tomcat/tomcat-catalina@9.0.14</purl>
    </component>
      <!-- More components here -->
  </components>
</bom>
```

Names

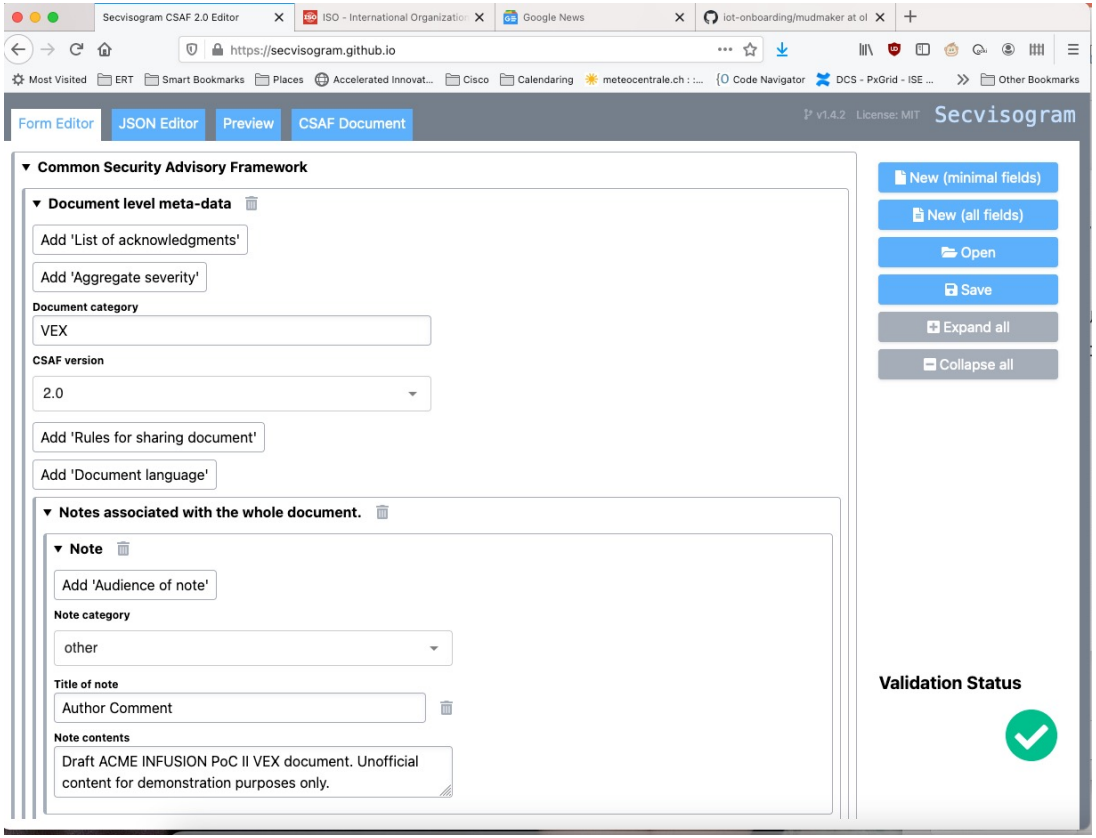
Vulnerability EXchange Format (VEX)

- A concept, not a standard
- Probably the most important part for our customers
  - Answers: is this product/package exploitable?
  - Would save us a lot of pain explaining the answer
- Requires trust of the person saying, “yes, I fixed this” or “no, we’re not vulnerable”

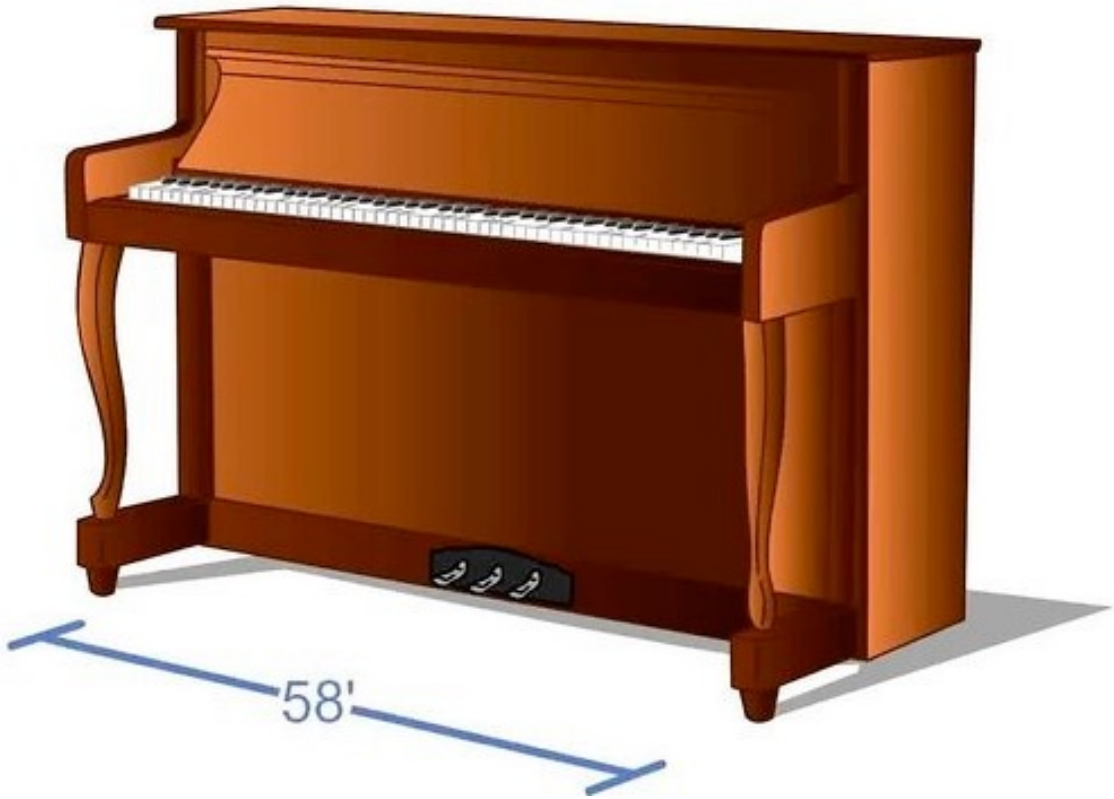
Example: OASIS Common Security Advisory Framework (CSAF)

- Issued by vendors based on product vulnerabilities
- Describes what you would see in a PSIRT advisory
  (this is no accident, thanks Omar)
- Not designed to bind to an SBOM
  (We're still **vex**ed)

There’s tooling!



Great! We have all the doc we need. How do we move it?

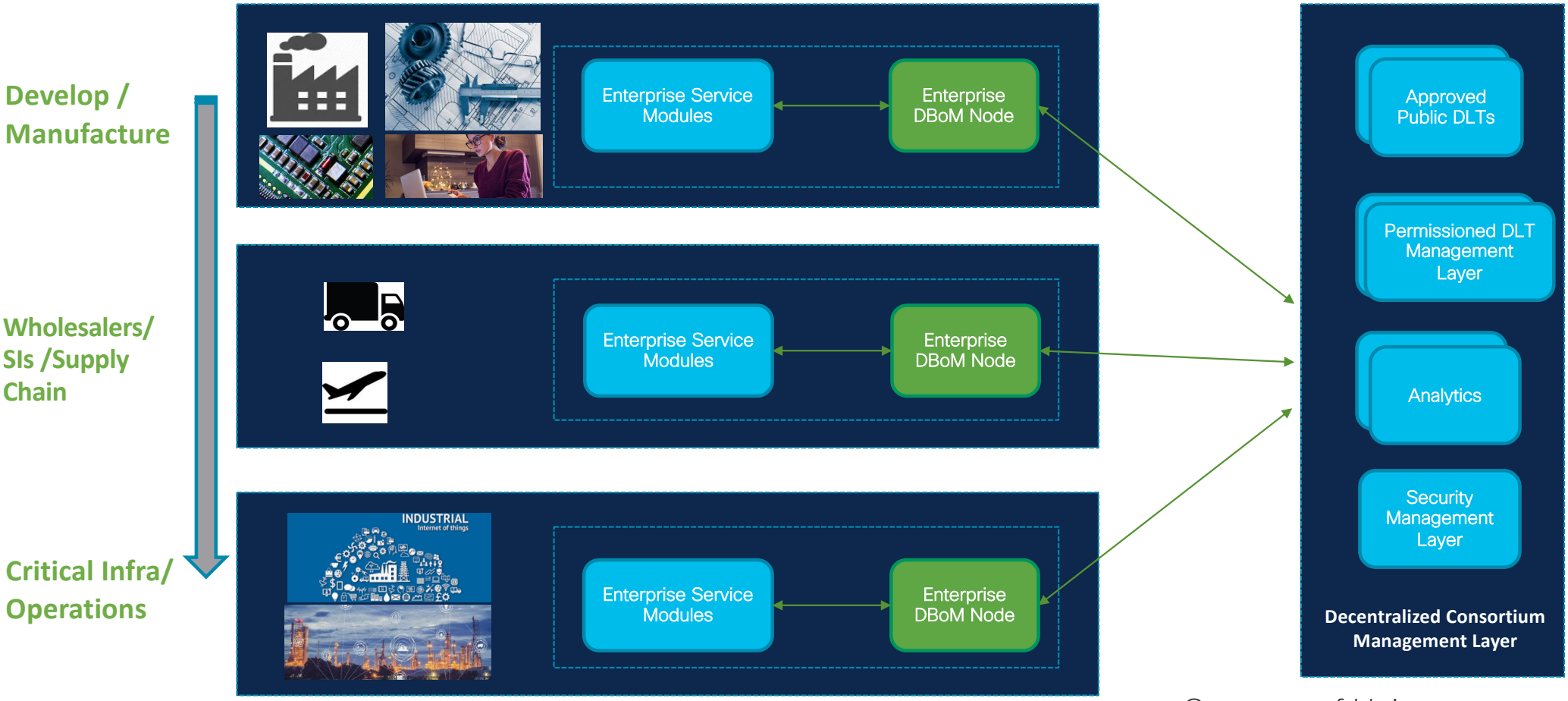


How do we discover what the Thing is and bind it to a device?

draft-ietf-opsawg-sbom-access

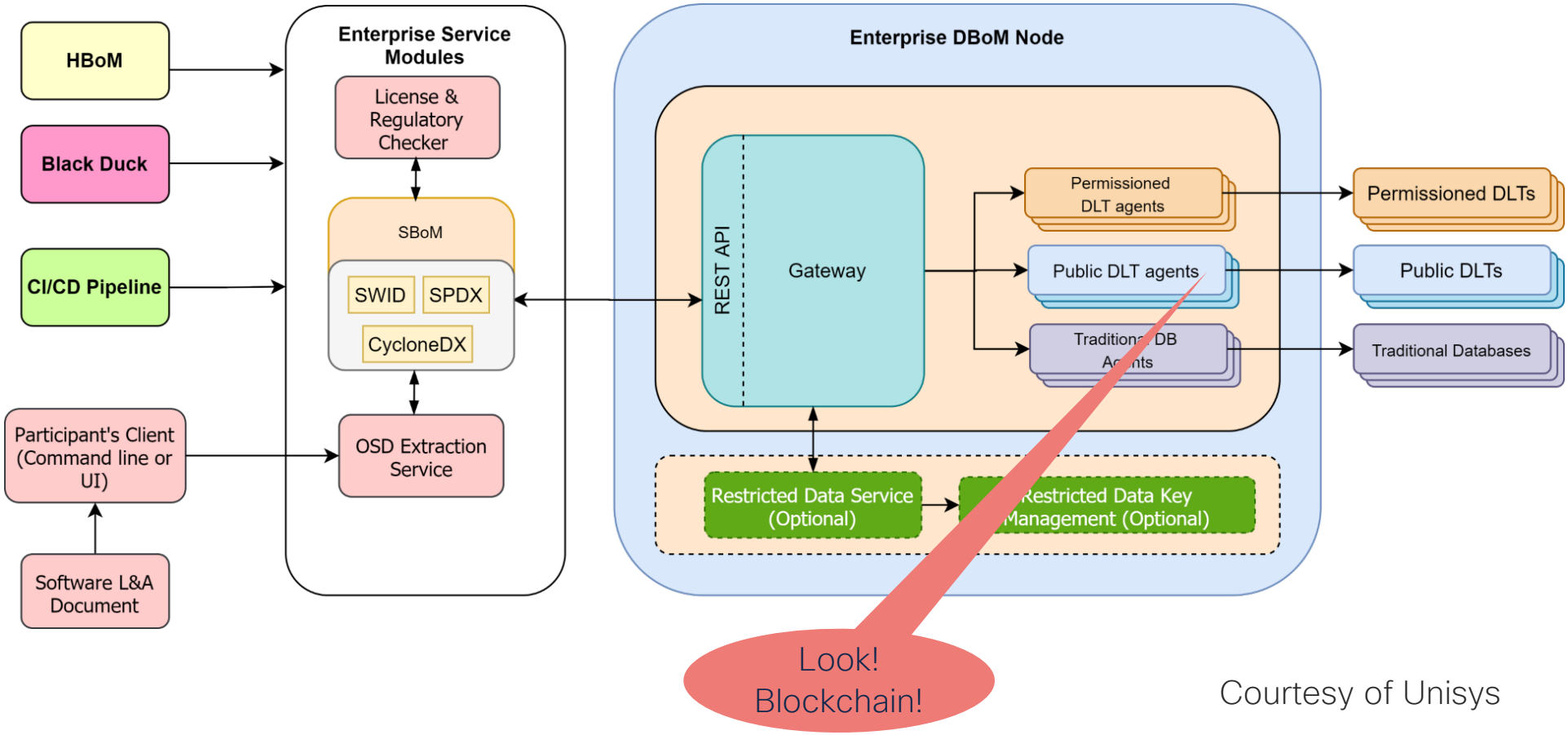
- Key Questions
  - Where does the SBOM reside?
    - On a web server?
    - On the device itself?
  - Do I need permissions?
  - What's the format of the SBOM?
- Answers are all URIs?

Digital Bills of Materials



Courtesy of Unisys

Digital Bills of Materials



Courtesy of Unisys

DBOMs versus direct retrieval

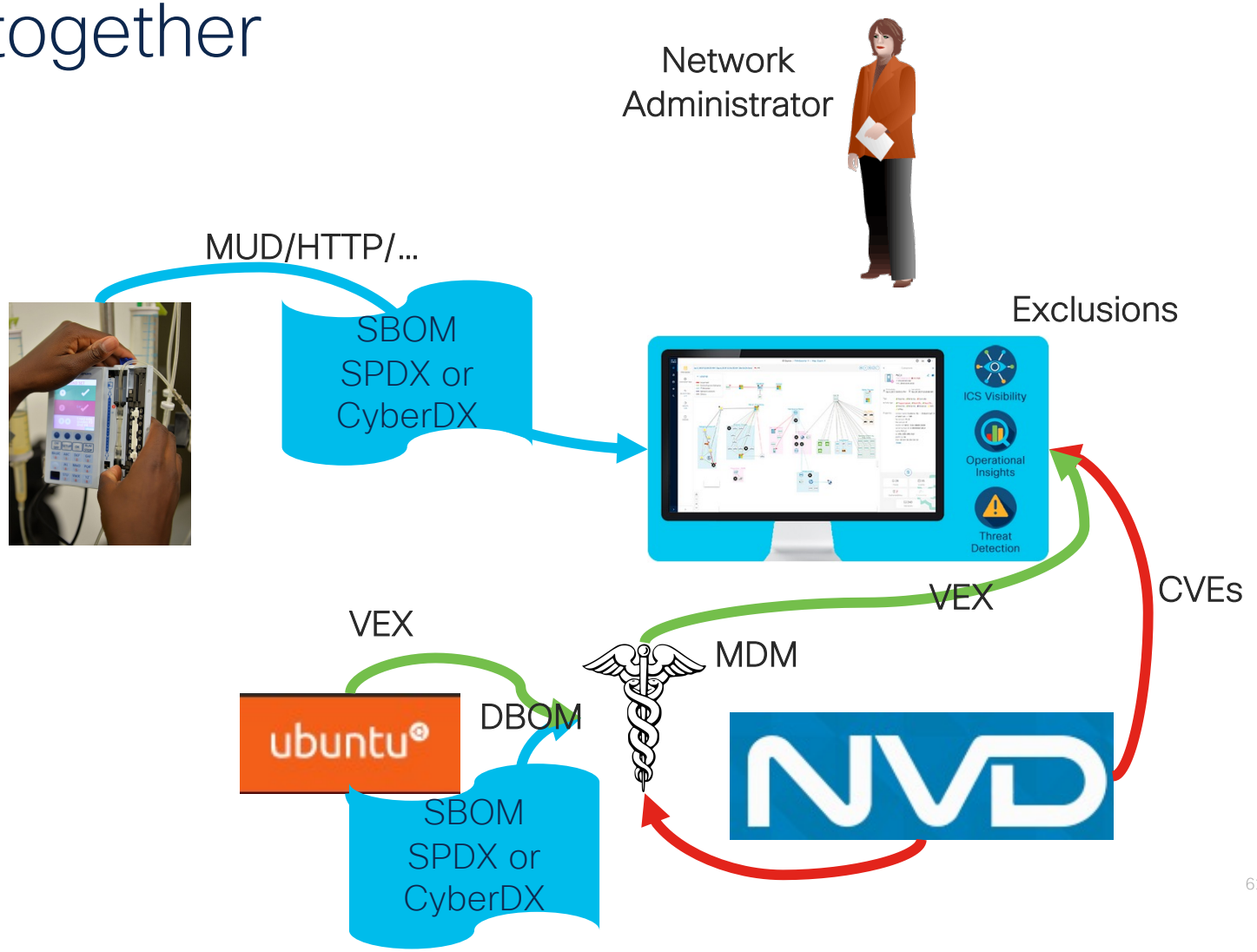
DBOMs are good for…

- Supply chain management
- Private consortia
- Random document exchange
  - DBOM doesn't care about content

MUD discovery is good for…

- Binding the device to a device type
- Learning a specific location of an SBOM
- Ad hoc discovery

Putting it all together



These are very early days

- NTIA healthcare POC taking place now
- SBOM formats evolving
- DBOM is nascent
- MUD extension just adopted by IETF opsawg but on a “slow roll”
- Name standards still challenging
- VEX is still **vex**ing

Roles

End User	Manufacturer	ISPs	Governments
Assume Things are going to break	Keep devices up to date	Identify threats	Lead by example, as end users
Keep an inventory of Things	Follow industry best practices	Facilitate consumer best practices	Set minimum standards for government use of Things
Follow best practices	Protect their backend systems!!!		Set policies that protect the infrastructure

The nice thing about best practices is that there are so many of them

The future is pretty bright, if we can survive the present

- We can secure the processor
  - Intel SGX
  - ARM TrustZone
- We can build attestation
  - IETF Remote ATteStation (RATS)
- We can deliver secure software updates
  - Secure Update of IoT (SUIT)

The nice thing about references is that there are so many of them

- RFC 8520: Manufacturer Usage Descriptions
- NIST 1800-15: Parts A-C on Manufacturer Usage Descriptions and DDOS prevention
- https://www.mudmaker.org
- Cisco IoT Onboarding WP: https://www.cisco.com/c/en/us/solutions/collateral/internet-of-things/white-paper-c11-743623.html
- MUD Manager https://github.com/CiscoDevNet/MUD-Manager
- https://github.com/usnistgov/nist-mud
- osmud.org
- Common Security Advisory Format - https://github.com/oasis-tcs/csaf
- CycloneDX - https://cyclonedx.org/
- SPDX - https://spdx.org
- Digital Bill of Materials - https://github.com/DBOMproject/
- Bootstrapping Remote Secure Key Infrastructure - https://datatracker.ietf.org/doc/rfc8995/
- Device Provisioning Protocol (EasyConnect) - https://www.wi-fi.org/file/device-provisioning-protocol-draft-specification
- Architectural Considerations for IoT Device Security In The Home - https://www.ripe.net/publications/docs/ripe-759
- Fido Device Onboarding - https://fidoalliance.org/intro-to-fido-device-onboard/

Thanks! Questions?

The bridge to possible