8. Discovery and Security for the Web of Things

- Discovery and Security for the Web of Things
  - Motivation
  - Interoperability Challenge
  - Discovery
    - Zero-Configuration
    - Hacks?
  - The WoT model
  - Security
    - State of security
    - Practices
    - OAuth

1. Motivation

2. Interoperability Challenge

Let us imagine that I had created a new webpage called “Remote”, a magic of sort remote, that I needed to communicate with every kind of Web of Things device.

Well.. it is kind of a challenge. From a web of things device, we expect it to be observe- and controllable using traditionally web-technologies, such as HTTP request, web sockets, perhaps MQTT, and that it there under adheres to the REST guidelines.

To do all of this, I would need to know not only the URL of all different devices and their API.. and you know what.. what seems like quite a challenge.

And well, having the web of things, even though they are built as described before, talk together, is difficult, if they dont have a shared understand and practice of how to work.

3. Discovery

Okay so let’s first discuss how our remote could go about discover all devices it can control. When I think remote, its a point and shoot device, which means that it control local things. So lets just start by discussing discovery locally, and from that see if we also will be able to do it remotely.

Well, being that the web of things is built on web technologies, web are sadly in no luck. The web was not meant for discovery as we would like. On the web, you are instead expected to know your destination.

But Magnus, what about the search engines? You ask. Well, it is not really, that they have “discovered” anything, they have simply crawled websites and followed links; which is a very brute force way of index stuff. As I said, the web is not built for discovery.

What we would like, for our remote is to be able to scan the network and find devices it can control. Now that we have already concluded, that there isn’t any web technologies which are able to do so, what do we do?

3.1. Zero-Configuration

Enter *zero configuration technologies*. It turns out that “discovering new devices on the local network_ it not a new problem. Think about wireless printers?

There are a bunch zero-configuration technologies:

- `DHCP` *Dynamic Host Configuration Protocol is used to contact the server and get an IP address*
- `UPnP` *Universal Plug and Play devices seamlessly discovery each other and communicates*
- `Bonjour` *and Bonjour is apples take on it*

So these seem very very smart! Can we bring them to the web? Well no, we cant. They mainly work on UDP and the web relies on TCP connections.

3.2. Hacks?

Oh well, there is always a way! I guess?

Yes of course, we could make some hacks.

- `Extension` we could write a plugin native code for every platform .. but well, that’s a lot
- `BLE` We could have every device have a Bluetooth LE tag which could deeplink and such
- `QRCode` or something similar with QR Codes
- `Route registration` or we could have every device registre with the router, and then from the remote connect to the router and get the list.. but how do we connect with the router then?

So yeah … these are all hacks. In conclusion on the web, we are currently not able to discover things locally, and therefor not remotely either.

4. The WoT model

But! Lets assume that we can discover devices and connect to them! Now we need to discover the possibilities of the device. We need in short: a way to index a device.

Our way pf index, should be very diverse, so that I can tolerate all kinds of web of thing products: from simple tags on a milk carton, to large and complex systems, and gateways.

One way to do so, is exposing a model of the product when connected to it. This model called “The Web Thing Model” is a product of a W3C Working Group Proposal, and aims to make web of things indexable by following the simple rules of the model.

The model is given as a JSON representation, where it is split into for different parts:

- ***The Web Thing Model***
- `/model: name, description, tags; actions and properties` *The first part is some information of the device it self. Located at /model on the device, it gives the name, description and tags of the device. It also list the actions and the properties of it and where to find them.*
- `/actions` *second part is the actions, located at /actions. Here all the actions and where to find them is listed. The information given should also be enough the be able to operate the actions. This is in short, the actuators and the use of them.*
- `/properties` *third part is the properties of the device, typically defined as the sensor readings, located at, yes you guessed it /propperties*
- `/things` *as the fourth and last part of the model, we find a description of the things we are able to control using the device, it could also be those things which is gateawayed.*

At every action, or property or such, the model contains a link to further index it, and we see that the /model path actually contains all the needed information.

- *Auto generated UI from model*

Now we have this general way of writing models, we can write code which understands it, and could fx. Create dynamic UIs for our remote, and it would call the propper endpoints at the device to read values and perform actions.

5. Security

So now, we have discussed the discovery of not only the things itself but also what they are capable of. The latter thing, exposed a lot of things about a device, but does that really seem secure?

5.1. State of security

Lets first talk about the state of security in, at least, the Internet of Things today. Or, maybe the lack of.

Through the years we have seen many botnets of CCTV, carrying out DDOS attacks on systems.

It turned out, that these CCTV were right open to hacking, and with an intrusion attack, hackers were able to install software on them and remote tell thousands of CCTV cameras to direct traffic at specific targets.

A funny thing about cameras is, there are one the internet, websites where you can go to, and browse privat and open cameras. These are security cameras which have not been secured in any way, and typically was just scraped by brute forcing ports on IP addresses.

Imagine what you can do with such data? I mean, this is some seriously surveillance, and the owners properly dont even know about it!

It turns out, that a lot of devices has poor security, often do to:

- *unencrypted traffic*
- *firmwares not updated*

5.2. Practices

Security is a never ending battle. Its like eating healthy. Actually, we can compare our health, to the security of a system. So lets talk about some measures we can take, to create secure web of thing devices.

- ***Encrypted communicated***
- `share keys securely`

For starter, we could begin to encrypt the communication that happens back and fourth with devices and servers. This is really basic, and it is a battle tested way of increasing the security, proven mathematically, where one could use something like RSA which is pretty simple to implement.

But for it to be secure, the exchange of keys also need to be done securely. And well, this is a topic in it self.

- ***Authentication***
- `Servers` *talking about authentication, we could start by being sure, that the thing or servers we communicate with, actually is who we think it is.*
- `* HTTPS` *HTTPS is a way to establish a secure SSL connection. This means that a sequence of handshakes and certificates approvals has been performed, and the server and client now can trust each other*
- `Client` *we can also authenticate on the client side, meaning that we can authenticate the user. For some products, this does not make sense, like when we talk about weather stations, public sensor or other things there can be at no harm. But when we begin to talk about about cameras, any device capable of surveillance, or talk about anythings with actuators; we must defiantly need to authenticate the user client side.*
- `* username / password` *such authentication can be done, simply by supplying a username and a password*
- `* OAuth` *but actually having users remember yet another username and password is a risk itself. It can be that they forget about it, it can be that to remember it they make it so simple that it is easy to hack. This is where OAuth comes into the game, and allows users to login using one of their other accounts from a big company like Google, Facebook, Apple or Twitter among many more.*
- ***Access Control***
- `User Roles` *in systems here there is client side authentication and the user has to login, the user will often also be given a role in the system. This role, defines what level of control the user will have in the system. Here roles could be something like user or administrator.*
- ***Software Updates***
- *iOS App Store*

And the last measure for security, is to not only perform software updates, but we also need the software to be update securely. First of all, if software doesn’t get update, the device can really by vulnerable, since a lot of software update only contains bug fixes. But what if the software update mechanism is comprised? Well, then anybody would be able to load all kinds of bad software onto our devices. And of course, this is a great treat which we can fix.

Apple has taken a lot of hits for their App Store and its strict rules, but one things is for sure. There heavy use of certificates and private keys when deploy applications and updates to it, have proven to big a great guard in secure software updates.

5.3. OAuth

So, we got a little time left, so lets just briefly discuss in a bit more detail what OAuth is.

We have all seen them, “Sign in with google” or “Sign in with Facebook”, and by clicking on them, we get redirected to the particular service to grant the login and perhaps give some information.

Being that the application we login to, is registreret with the different authentication services, they know us and are able to redirect back to us. When we sign up the such a service, we get an *client id* and a *client secrete*, which is why the service can regocnize and trust us.

In modern OAuth, users can also use the services to grant access to their data. This is done, so the user, using OAuth 2.0 can grant an application temporary acces to a resource. When they do so, the application receives a token, which it can contact the service with to get the information. It could be f.x a contact list or such.

OAuth can also to misused, if a bad actor tries make the user think it has pressed a “login with Facebook” button, but now the user is at a fake login screen, where the users credentials is tried to be obtained. This is called phishing.

So yeah, the web of things face great problems when talking about security, but being that it is built upon the web, we have already great and battle tested solutions. One only need to use them.