Self-descriptions for Interoperability and Security

Using WoT TD and MUD with RIOT

RIOT Summit 2021
Hugo Damer, Jan Romann, University of Bremen
About us

- Student computer science project at University of Bremen
- Supervised by Prof. Dr. Carsten Bormann, Prof. Dr. Ute Bormann and Dr. Olaf Bergmann
- Goal: Improve IoT security and usability
  - Return improvements to the open source community
- Results from the bachelor phase
Motivation
Two main problems:

Security and Interoperability
Problem 1: Security

Devices often are...

... poorly secured

... untrustworthy

... receiving only limited support

Source: https://xkcd.com/1966/
Problem 2: Interoperability

Devices are often ...

... only compatible with their own ecosystem

... using closed/proprietary standards

... not able to interact with each other

Source: https://unsplash.com/photos/GbAEJUJKJ88
Enter: Self-descriptions

- Open standards
- Web of Things Thing Description (WoT TD)
  - W3C Recommendation
  - Describe what a device can do
  - Expose capabilities
- Manufacturer Usage Description (MUD)
  - RFC 8520
  - Describe what a device needs to fulfill its function
  - Reduce attack surface
Standards
WoT TD

- TD as “Entry Point” (cf. index.html)
- Serialized as JSON
- Consumers interact with Thing based on TD

See: Sebastian Kaebisch, Takuki Kamiya, Michael McCool, Victor Charpenay and Matthias Kovatsch, »Web of Things (WoT) Thing Description«, W3C Recommendation, Apr. 2020, URL: https://www.w3.org/TR/2020/REC-wot-thing-description-20200409/
TD Structure

See: Sebastian Kaebisch, Takuki Kamiya, Michael McCool, Victor Charpenay and Matthias Kovatsch. »Web of Things (WoT) Thing Description«, W3C Recommendation, Apr. 2020, URL: https://www.w3.org/TR/2020/REC-wot-thing-description-20200409/
Interaction affordances

- Properties
- Actions
- Events

- Expose the capabilities of the Thing
- Mandatory “forms” field for protocol bindings
  - Map affordances to protocols and resources

See: Sebastian Kaebisch, Takuki Kamiya, Michael McCool, Victor Charpenay and Matthias Kovatsch, »Web of Things (WoT) Thing Description«, W3C Recommendation, Apr. 2020, URL: https://www.w3.org/TR/2020/REC-wot-thing-description-20200409/
Interaction affordances: Properties

```json
{
  ...,  
  "properties": {
    "temperature": {
      "type": "number",
      "unit": "C",
      "forms": [
        {
          "href": "coap://example.org/temperature"
        }
      ]
    }
  }
}
```

Schema information

Protocol bindings

See: Sebastian Kaebisch, Takuki Kamiya, Michael McCool, Victor Charpenay and Matthias Kovatsch. »Web of Things (WoT) Thing Description«, W3C Recommendation, Apr. 2020, URL: https://www.w3.org/TR/2020/REC-wot-thing-description-20200409/
Actions

See: Sebastian Kaebisch, Takuki Kamiya, Michael McCool, Victor Charpenay and Matthias Kovatsch, »Web of Things (WoT) Thing Description«, W3C Recommendation, Apr. 2020, URL: https://www.w3.org/TR/2020/REC-wot-thing-description-20200409/
Obtaining TDs

- /.well-known/wot-thing-description
- /.well-known/core
- CoRE Link Format
  - Content-Type application/td+json

→ Enables multicast discovery

WoT in RIOT

- Upcoming RIOT-Module (still WIP)
  - Collaboration with Philipp Blum (@Citrullin)
- Serialization of TD as JSON at runtime
- CoAP support using Gcoap
- Code generation at compile time
Example Setup

Weather Station

Request Properties
(periodically)

Request TDs/
Device Discovery
(Multicast)

Router

Temperature Sensor

Humidity Sensor

Icon Packs used: Font Awesome (https://github.com/FortAwesome/Font-Awesome, License: CC BY 4.0), Gnome Symbolic Icons (https://gitlab.gnome.org/Archive/gnome-icon-theme-symbolic/-/tree/master, License: CC-BY-SA 3.0)
Weatherstation Logic
Weather Station UI

### Monitor

**Humidity**
- NAMIB-DHT (innen)

**Temperature**
- NAMIB-DHT (innen)

### Control-Panel

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAMIB-Water Sensor Node</td>
<td>A sensor node designed to be used in the undergraduate project NAMIB</td>
</tr>
<tr>
<td>NAMIB-DHT (innen)</td>
<td>Ein Sensor-Node für den Einsatz im Bachelorprojekt NAMIB</td>
</tr>
</tbody>
</table>
Manufacturer Usage Description (MUD)

- Specified in RFC 8520
- Manufacturer-provided device description of which network access is necessary
- Serialised as a JSON file (MUD-File) following a YANG data model
- Mainly consists of Access Control Lists (ACLs) aside from meta information

MUD File Example

```
{
  "ietf-mud:mud": { 
    ...
  },
  "ietf-access-control-list:acls": { 
    ...
  }
}
```

MUD File Example - Metadata

```json
{
    "ietf-mud:mud": {
        "mud-version": 1,
        "mud-url": "https://lighting.example.com/lightbulb2000",
        "last-update": "2019-01-28T11:20:51+01:00",
        "cache-validity": 48,
        "is-supported": true,
        "systeminfo": "The BMS Example Light Bulb",
        ...
    },
    ...
}
```

{...
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "mud-76100-v6to",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [ ... ]
        }
      }
    ]
  }
}
{ "aces": [ { "name": "cl0-todev", "matches": { "ipv6": { "ietf-acldns:src-dnsname": "test.example.com", "protocol": 6 }, "tcp": { "ietf-mud:direction-initiated": "from-device", "source-port": { "operator": "eq", "port": 443 } } }, "actions": { "forwarding": "accept" } } ] }
MUD Architecture


MUD Controller

- We wrote our own MUD manager implementation
- Consists of...
  - ...Controller
    - runs on e.g. Raspberry Pis
  - ...Enforcer
    - runs on home routers (OpenWRT)
MUD Controller
MUD Controller
How to deal with devices that do not support MUD?

- Legacy-Service component
  - Can analyse network traffic of non-MUD devices
  - MUD recommendations based on device behavior
  - Based on crowdsourced repository of MUD files
    - https://gitlab.freedesktop.org/sw0rd/MUD-Files
MUD in RIOT

RIOT can already communicate its own MUD-URL!

- Implementation of DHCPv6 MUD option
- Client can request non-temporary addresses
  - MUD-Controller works with leases
- Open PR for stateless DHCPv6

How does the RIOT device know its own MUD-URL?

- Configuration of MUD URL via Makefile or KConfig
Insights
Insights: WoT

- Discovering and using TDs works
- However: No support for composition yet
- JSON not ideal for constrained devices → CBOR
Insights: MUD

- DHCP not ideal for emitting MUD URLs in the IoT
  - SLAAC makes DHCPv6 less relevant
  - Possible alternative: CoRE resource directories
- Keeping track of IP addresses for DNS entries is difficult
  - Especially when dealing with load balancers
- Isolating devices in home networks is difficult
  - VLANs not supported by hardware
Conclusion

- Self-descriptions are very promising
- Can make the IoT more secure and easier to use
- However:
  - Some weaknesses
  - Constrained devices require a bit more effort

→ There is still a lot to do
Fork Us on Github

- https://github.com/namib-project

We're looking forward to your feedback and contributions!