Discovery

Michael McCool, Intel

8 June 2020

Discovery Requirements

- Capabilities
  - Support both local and global/remote discovery (unconstrained by network domain)
  - Support "localizable" discovery (constrainable by location)
  - Support some form of "semantic query"
  - Support both
    - Directory services for searching large repositories of Things
    - Peer-to-peer (self-identifying) discovery
- Privacy-Preserving Architecture
  - Respect device and information Lifecycle
  - Distribute TDs only to authenticated and authorized users
  - Don't leak private data to unauth. users
  - Don't leak information that can be used to INFER private information to unauth. users
- Alignment with existing standards
  - E.g. IETF CoRE Resource Directories, CoRE Link Format, DID, …
  - Align with WoT Scripting API

Two-Phase Architecture

1. Introduction
   - “First Contact” Protocol
     - Answers the question: how to start?
   - Open
     - Can be accessed with no or limited access controls
   - Lightweight
     - Does not use significant resources on responder
     - Resistant to Denial of Service attacks
   - Provides intentionally limited information
     - Avoid leaking any metadata that can be used to infer private data
     - This includes types of devices, device ids, owners, timestamps, etc.
2. Exploration
   - Authentication and authorization required
   - Supports more complex search capabilities
   - Provides access to rich metadata
   - Access controls can limit data returned

Introduction

- “First Contact” protocol
  - Output: Address of exploration service, for example, a directory service
  - Need not be broadcast; could use well-known network services (eg DNS, DHCP, etc)
- Address should not leak any other metadata, e.g. type of devices
- Can have multiple mechanisms for introduction
  - Local: QR code, mDNS, DNS-SD, DHCP, Bluetooth beacons (Eddystone), etc.
  - Global: Search engine, well-known global repositories, company repositories, cities, etc.
  - Self: Well-known addresses, eg “.well-known/td”
- Existing mechanisms that have lists of typed links can also be used here:
  - CoRE RD, DID Documents, DNS, etc.
  - Use these to find directories rather than to distribute metadata directly
- *May* in some cases point directly at a Thing Description
  - Degenerate case: like a “directory” has only one TD
  - Still requires authentication in principle to access content

Exploration

- Authentication required, and then…
- Queryable Directory service
  - Lightweight: specific query parameters, eg. location, keywords
  - Full: (sub-)SPARQL semantic query AND/OR GraphQL AND/OR by-example
- Gateway: registration sub-API, timeouts, etc.
- Self: same query API, but no public registration API
- Mutable IDs → need way to notify registered users of changes

Privacy Issue

- Privacy
  - Two-phase approach ***not sufficient*** to preserve privacy in all contexts
  - Privacy preservation also depends on the design of API
  - API needs to hide data that can be used to infer private information, such as location of device doing the discovery
- Third-party code context (eg browser):
  - If discovery API follows two-phase structure, where Introduction returns list of directories, then even without authenticating the list of directories visible can possibly be used to infer location
  - This is especially true if the discovery mechanism can be constrained to particular Introduction mechanisms.
  - May also be a problem in proposed non-browser contexts, eg. “Edge Workers”

Resources

- Repository: https://github.com/w3c/wot-discovery
- Prior work: https://github.com/w3c/wot-discovery/blob/master/prior.md
- Background: https://github.com/w3c/wot-discovery/blob/master/background.md
- Requirements: https://github.com/w3c/wot-discovery/blob/master/requirements.md
- Design: https://github.com/w3c/wot-discovery/blob/master/design.md
- Proposals: https://github.com/w3c/wot-discovery/tree/master/proposals

Current Scripting API – Discussion Points

- Allows discovery mechanism to be specified
- Returns discovered TDs
- Assumes authentication/authorization is handled out-of-band

Discussion Points:

- Not incompatible with two-phase approach
  - Authentication/authorization needs to be set up outside script
- Query format needs to be better standardized
  - What query forms are supported? What are the parameters?
  - How can I search for Things in a particular physical location (that I am not necessarily at?)
- Options to select mechanisms may be a privacy risk
  - If I can discover a thing via Bluetooth, I know it is within a few meters
  - Perhaps the mechanisms and desired location should also be specified out of band
- Alternatives:
  - “Discovery sub-API” could be factored out and run during setup, like security config
  - Things to be consumed by a script perhaps managed declaratively using dependences

Discussion Points

- Query Language
  - JSON Path?
  - X Path?
  - Custom?
  - SPARQL?
- Security Schemes
- Supported Introduction Protocols
  - Anything that gives an address/URL will do
  - Some need actions to support, eg service name registrations for DNS-SD
  - Also need CoRE-RD and DID integration – as Introductions