Securing openHAB Smart Home through User Authentication and Authorization

Jesús Antonio Soto Velázquez

UNIVERSITY OF TARTU

Institute of Computer Science
Computer Science Curriculum

June 2018

Overview

Introduction: Internet of Things

Internet of Things. Dynamic and heterogenous environment where *sensing* devices interact with each other.

- Devices: smartphones, smart watches, smoke alarm, security camera, heartbeat and temperature monitor, etc.
- Applications: Smart home, smart city, smart healthcare, intelligent transportation, etc.
- Challenges: interoperability, architecture, device naming, usability, security and privacy.

A Smart Home Application: OpenHAB

“a vendor and technology agnostic open source automation software for your home” (openHAB Community, 2010)

- May be deployed on very modest devices (e.g., Raspberry Pi)
- Mostly works without an Internet connection.
- Uses thing-specific *bindings* to connect to devices.
- Made up of individual projects, e.g. Eclipse SmartHome.
- Written in Java under the OSGi architecture.

Objectives

1. To analyze the security mechanisms in use for the openHAB automation software.
2. To implement an authenticator based on the JSON Web Token for Eclipse Smarthome.
3. To propose a fine-grained, yet usable authorization model for openHAB to manage usage permissions.

Motivation

- Developments of IoT applications focus mostly on functional requirements, leaving usability, performance, security, etc., for later.
- Consequences in smart home range from mere user discomfort to identity theft, or worse.
- Lack of security deters adoption of IoT applications, halts progress.

Security Challenges in OpenHAB

- Intranet of Things
- Bindings
- Access Control

Intranet of Things

- Aims to contain private data locally.
- Puts a limitation on use cases.
- Secure remote access: a challenge.
- Assumes security of the private network.

Security of Bindings

Binding. Logical modules to support inter-device interaction inside openHAB.

- Each binding is defined for things that use a certain protocol, e.g.: KNX, Z-Wave, ZigBee, Panasonic TV, etc.
- Some bindings use an API for remote connection to the vendor.
- Remote communication may be done under HTTPS if specified by binding.

Access Control

- Whoever gains access to private network gains access to smart home management.
- Not trivial: previous attempts have failed.
- To be implemented inside the core framework: Eclipse SmartHome.
- Main challenge: single *authentication context* recognizable in all end points (servlets).

Misuse Cases

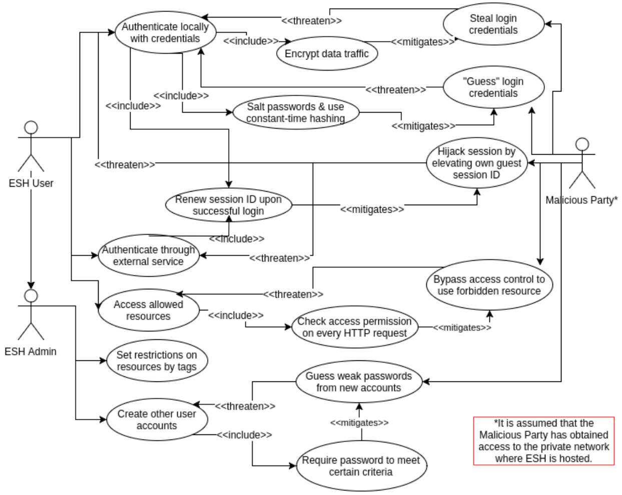
Authenticate locally with credentials
<<threaten>>
Steal login credentials
<<include>>
Encrypt data traffic
<<mitigates>>
"Guess" login credentials
<<threaten>>
<<include>>
<<include>>
Salt passwords & use constant-time hashing
<<mitigates>>
ESH User
<<threaten>>
Hijack session by elevating own guest session ID
Malicious Party*
Renew session ID upon successful login
<<mitigates>>
<<include>>
Authenticate through external service
<<threaten>>
Bypass access control to use forbidden resource
ESH Admin
Access allowed resources
<<threaten>>
<<include>>
Check access permission on every HTTP request
<<mitigates>>
Set restrictions on resources by tags
Guess weak passwords from new accounts
Create other user accounts
<<threaten>>
<<mitigates>>
<<include>>
Require password to meet certain criteria
*It is assumed that the Malicious Party has obtained access to the private network where ESH is hosted.

Figure 1: Misuse Cases for Access Control in ESH.

Proposed Security Mechanisms

- JSON Web Token-based Authenticator
- RBAC-inspired Authorization Model

JSON Web Token Structure

1. Header: type of token (JWT), hashing algorithm (e.g., HMAC SHA256, RSA).
2. Payload: claims about the user (e.g., username, roles).
3. Signature: Using a secret key creates a signature of the previous parts.

Token-based Authentication Procedure

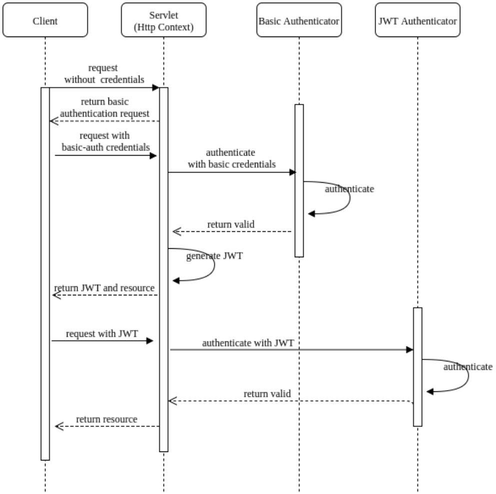

Figure 2: Addition of authenticators into the architecture.

Architectural Implications

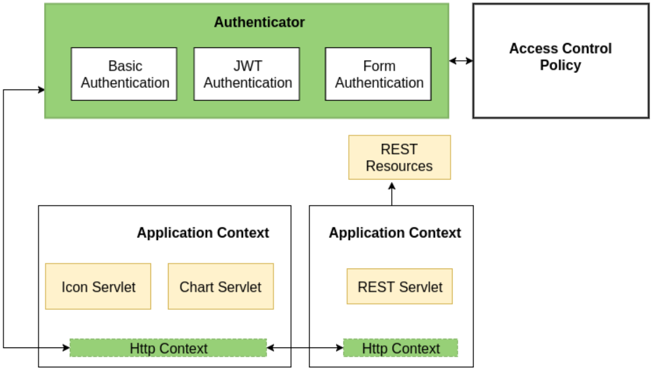

Figure 3: Addition of authenticators into the architecture.

Implementation of Authenticators

Some remarks:

- Prepared to work for a *shared* authentication context.
- *JWT* authentication when token is provided in request (cookie or HTTP header).
- Nimbus used as JWT library.
- RSA key pair is created during runtime and stored in memory.
- *Basic* authentication (username:password) as fallback authentication mechanism.
- Registration to OSGi 4.2 runtime is done differently than in OSGi 6 runtime.

Proposed Authorization Model

- As a smart home application, an authorization model should be fine grained and usable.
- Tasks that may be done on openHAB are grouped as *capability sets*.
- Registered users are assigned one or more capability sets to reflect their privilege in the system.
- Permissions range from: REST endpoints, sitemaps, traditional servlets, automation rules, third-party add-ons, system settings, etc.

User-Assigned Capability Sets

Table 1: Sample relation of user and capability sets

User	**Capability Sets**
Marian	(speakers-quiet, lights-on, doors-close, sitemaps-paper)
Erika	(speakers-playback, lights-all, doors-all, sitemaps-all)

Table 2: Sample listings of operations involved for each capability set

Capability Set	**Involved Operations**
speakers-playback	yamahareceiver.internal.state. NavigationControlState.getCurrentItemName() ZoneControlState.volume

things-all	rest.core.internal.thing.ThingResource.getAll()
	rest.core.internal.thing.ThingResource.create()

Evaluation

Authenticator was evaluated in two OSGi runtimes: Karaf (Apache Felix), and Eclipse SmartHome (Eclipse Equinox).

1. Karaf deployment supported OSGi R6.
   Servlet and context registration was made through Whiteboard pattern.

2. ESH deployment supported up to OSGi R4.2.
   Servlet and context registration was made through `Http Service` and `Http Tracker`.

Conclusion

Contributions:

- Analysis of the security mechanisms in use for the openHAB automation software.
- Implementation of a client authenticator based on the JSON Web Token for Eclipse Smarthome.
- Proposal of a fine-grained, yet usable authorization model for openHAB to manage usage permissions.

Future Research Directions

- Incorporation of form-based authentication in the OSGi architecture.
- Expiration and renewal of JWT issued by Eclipse SmartHome.
- Encryption of JWT.
- Unique generation and storage RSA key pair.
- Local storage of user credentials (e.g., LDAP).
- Integration with external authentication providers (e.g., OAuth).
- Design access for first-time users.
- Implementation of the proposed authorization model.