Week 3 Notes and Resources

KEY TOPICS

Communications, Security, and Computing at the Edge

This week we focused on protocols used to communicate with and between devices, IoT security, and moving compute to the "edge"

MQTT/HTTP/Websockets

As Wes desribed, the AWS IoT Message Broker, supports MQTT, HTTPS and MQTT over WebSockets. Both MQTT aqnd HTTPS support X.509 client certificate authentication. HTTP and MQTT over WebSockets both support SigV4 Authentication. See the above linked documentation for details about the AWS Iot Message Broker

MQTT

MQTT is a widely adopted, lightweight messaging protocol designed for constrained devices. For more information, see MQTT. Although the AWS IoT message broker implementation is based on MQTT version 3.1.1, it deviates from the specification. See the current AWS IoT MQTT documentation for details.

HTTP

The message broker supports clients connecting with the HTTP protocol using a REST API. Clients can publish by sending a POST messages. See the current AWS IoT HTTP documentation for details.

MQTT over WebSocket Protocol

AWS IoT supports MQTT over the WebSocket protocol to enable browser-based and remote applications to send and receive data from AWS IoT-connected devices using AWS credentials. AWS credentials are specified using AWS Signature Version 4. WebSocket support is available on TCP port 443, which allows messages to pass through most firewalls and web proxies.
A WebSocket connection is initiated on a client by sending an HTTP GET request.

See the latest AWS IoT MQTT over WebSocket Protocol documentation for details.

AWS IoT Security and Identity

Each connected device must have a credential to access the message broker or the Device Shadow service. All traffic to and from AWS IoT must be encrypted over Transport Layer Security (TLS). Device credentials must be kept safe in order to send data securely to the message broker. AWS Cloud security mechanisms protect data as it moves between AWS IoT and other devices or AWS services.

The AWS IoT message broker and Device Shadow service encrypt all communication with TLS version 1.2. TLS is used to ensure the confidentiality of the application protocols (MQTT, HTTP) supported by AWS IoT. TLS is available in a number of programming languages and operating systems.
For MQTT, TLS encrypts the connection between the device and the broker. TLS client authentication is used by AWS IoT to identify devices. For HTTP, TLS encrypts the connection between the device and the broker. Authentication is delegated to AWS Signature Version 4.

See the AWS IoT Security and Identity section of the AWS IoT documentation for details.
As noted in the Communications section above, AWS IoT uses X.509 certificate based authentication. Details of X.509 certificates are outside the scope of this class. A general introduction can be found on Wikipedia as well as articles on Public or asymmetric cryptography.

Computing/IoT at the Edge

Amazon FreeRTOS

Amazon FreeRTOS is an operating system for microcontrollers that makes small, low-power edge devices easy to program, deploy, secure, connect, and manage. Amazon FreeRTOS extends the FreeRTOS kernel, a popular open source operating system for microcontrollers, with software libraries that make it easy to securely connect your small, low-power devices to AWS cloud services like AWS IoT Core or to more powerful edge devices running AWS IoT Greengrass.

AWS IoT Greengrass

AWS IoT Greengrass seamlessly extends AWS to edge devices so they can act locally on the data they generate, while still using the cloud for management, analytics, and durable storage. With AWS IoT Greengrass, connected devices can run AWS Lambda functions, execute predictions based on machine learning models, keep device data in sync, and communicate with other devices securely – even when not connected to the Internet.

WHAT YOU ACCOMPLISHED THIS WEEK

- You learned how to commumnicate with your IoT Devices.
- You created a web-based application to communicate with your "cars"
- You created a Greengrass core and let your cars communicate without having to communicate directly with AWS IoT Core.