Demo: Secure M2M Cloud Testbed

Heikki Mahkonen heikki.mahkonen@ericsson.com
Teemu Rinta-aho teemu.rinta-aho@ericsson.com
Tero Kauppinen tero.kauppinen@ericsson.com
Mohit Sethi mohit.m.sethi@ericsson.com
Jimmy Kjällman jimmy.kjallman@ericsson.com
Patrik Salmela patrik.salmela@ericsson.com
Tony Jokikyyny tony.jokikyyny@ericsson.com
NomadicLab
Ericsson Research, Finland

ABSTRACT

This demo presents a Machine-to-Machine (M2M) service platform running on an OpenStack cloud environment. Access to this platform is secured with authentication based on 3GPP standardized Generic Bootstrapping Architecture (GBA). The M2M network in this testbed consists of Raspberry Pi gateways and resource-constrained devices such as Arduinos. These devices have associated sensors, that report physical data such as temperature, and actuators such as LEDs, that respond to actuation commands. In this demo, the sensor data is authenticated and integrity protected end-to-end with public-key elliptic curve digital signature algorithm (ECDSA). Lastly, the demo testbed includes a web-based cloud management interface referred to as the "IoT Portal". This portal allows a system administrator to specify pre-configured Virtual Machine (VM) images with custom applications. As an example, a M2M Web service is installed into the VMs. This web service collects and displays authenticated and verified sensor data and can also be used to send actuator commands.[1]

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: Computer-Communication Networks—*Distributed Systems*

Keywords

Cloud computing; Generic Bootstrapping Architecture; Internet of Things; Machine-to-Machine; OpenStack; Security

1. INTRODUCTION

Internet of Things (IoT) encompasses a wide variety of technologies, objects and protocols. Physical objects that are part of IoT, no longer act as unresponsive nodes but rather understand and react to the environment they reside in. Such objects, referred to as smart objects, form the building blocks of the Internet of Things. In order to ensure smooth and successful deployment of these smart objects, as an important first step, it is necessary to develop experimental prototypes that study the feasibility of different protocols and technologies. To this end, our testbed presents a prototype platform that provides resource-constrained smart objects with secure connectivity to a cloud environment over the Internet. Using a cloud service allows easy and dynamic deployment of services, as well as, facilitates network operators and enterprises with a platform where services can be managed centrally on a large scale.

The rest of this paper is structured as follows. In Section 2, we present relevant background information about the secure M2M and Cloud solution implemented in our testbed. Section 3 then gives a detailed overview of the demonstration. Lastly, Section 4 provides a conclusion.

[1]Demo URL: `http://users.piuha.net/nomadiclab/smct.mov`

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s). Copyright is held by the author/owner(s).
MobiCom'13, September 30–October 4, 2013, Miami, FL, USA.
ACM 978-1-4503-1999-7/13/09.
http://dx.doi.org/10.1145/2500423.2505294 .

2. BACKGROUND

2.1 Machine-to-Machine Network

In the testbed demo, we prototype smart object devices with Arduino[2] development boards. They have associated sensors, that report physical data such as temperature, and actuators such as LEDs, that respond to actuation commands. Since these devices are constrained in the amount of energy available to them, we rely on energy-efficient IEEE 802.15.4 short-range radio boards (Digi XBee[3]) for communication. To enable seamless interaction between users and devices across heterogeneous networks, it is important that these devices are accessible over the Internet. We therefore use a Java OSGi[4] based M2M Gateway (GW) device running on a Raspberry Pi[5] development board to provide Internet connectivity to the smart object network.

2.2 Cloud Environment

We use the OpenStack[6] platform (Grizzly 2013.1 distribution) running on Linux servers with Ubuntu 12.04 as the

[2]`http://www.arduino.cc/`
[3]`http://www.digi.com/xbee/`
[4]`http://www.osgi.org/Main/HomePage`
[5]`http://www.raspberrypi.org`
[6]`http://www.openstack.org/`

cloud backend in our testbed. This backend operates on the KVM[7] hypervisor and utilizes Open vSwitch[8] networking. VM instances for the smart object devices that are part of the M2M network are managed and run by the OpenStack "compute" component. Assignment of IP addresses and connectivity management for the VMs is controlled by the "networking" component of OpenStack. Lastly, the "image service" component is responsible for storing pre-configured VM images which are used when spawning new VMs.

Our M2M service platform includes a web interface for management of the cloud backend. This management interface, referred to as the "IoT Portal", consists of components written in PHP, JavaScript and shell scripts (for issuing OpenStack commands), and runs on an Apache web server. It interfaces with an SQL database for storing data and configuration information. With the IoT Portal, a system administrator can obtain an overview of gateways, registered devices, their corresponding VM instances, IP addresses, the status of the VM instances, and other configuration data. Through its interface, the IoT Portal also allows specification of VM templates to ensure that devices of a certain class (for example, a specific vendor, type or subscriber) can be assigned VMs with different configuration data such as memory, disk space, operating system, and software packages. Moreover, the IoT Portal can also specify if a VM instance for a device class can be shared by several devices in that class, or if a new instance needs to be created for each device.

The M2M gateway communicates with cloud backend through a REST API. The IoT portal creates a new VM instance for new devices joining the M2M network if it does not already have an instance, or if the device class does not have a shared instance. It is possible for the IoT portal to acquire software and configuration options for a new VM instance from an external source, for example, a device vendor service. In our cloud backend, custom software and configurations can be installed into a VM instance using the Puppet[9] automation software.

2.3 Authentication and Security

Secure authentication and bootstrapping of devices can be non-trivial in smart object networks. Generic Bootstrapping Architecture (GBA) [1] is a 3GPP standard for secure and flexible authentication of a user to application services. GBA relies on the Authentication and Key Agreement (AKA) protocol and utilizes SIM credentials along with the operator infrastructure to setup a shared secret between a user equipment (e.g. mobile phone) and an application server. Since GBA can be used with little or no user interaction, and the fact that it provides strong authentication, we believe that it is a potential candidate protocol for secure authentication and bootstrapping of devices. In our demo testbed, we use GBA for secure authentication and provisioning of gateway devices.

A smart object network that contains resource-constrained sensor devices, may have several intermediaries between the sensor where the data is generated, and the cloud, where the sensor data is stored. These intermediaries may include gateways, firewalls and caching proxies. The sensor data stored in the cloud may also later be accessed by user applications running on different devices. In order to ensure data authenticity and integrity end-to-end in such a multi-hop environment, it is beneficial to have asymmetric public-key based digital signatures which can be used to sign data at its origin (sensors). This data can later be verified for its authenticity and integrity by caching proxies, end-user applications consuming the data or any other intermediary. As shown by Sethi et al. [2], it is possible to use Elliptic Curve Digital Signature Algorithm (ECDSA) to sign data even on 8-bit micro-controllers. In this demo, we use the same library (Relic[10]) and configuration to demonstrate end-to-end data authenticity and integrity from sensors to the cloud with digital signatures.

[7] http://www.linux-kvm.org/
[8] http://openvswitch.org/
[9] http://puppetlabs.com/

3. DEMO OVERVIEW

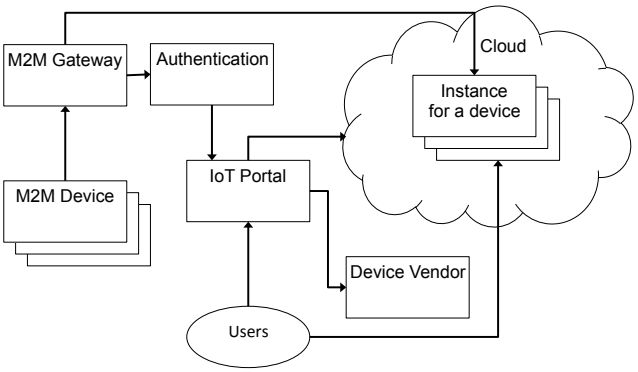

Figure 1: Secure M2M Cloud Testbed

Figure 1 shows the components of the testbed. The M2M network has resource-constrained smart object devices and GWs. These devices use IEEE 802.15.4 radios for communication in the M2M network and are connected to the Internet through GWs. Each GW in the M2M network first authenticates itself to the cloud backend using standard 3GPP Generic Bootstrapping Architecture [1]. The Network Application Function (NAF) service is part of the authentication infrastructure and is located at well-know address (domain name). It is responsible for providing the correct IoT Portal address to an authenticated GW based on the GBA credentials presented by the GW. Henceforth, when an authenticated GW receives an attach request from a resource-constrained device in the M2M network, it requests the IoT Portal to register the new device. The portal may use an existing VM instance or, if needed, may create a new VM instance for this device. Thereafter, the address of this VM instance is communicated to the GW. The IoT portal in this demo uses VM templates when instantiating new VMs. These templates have pre-configured operating system (Ubuntu 12.04 amd64) and Apache web server along with a custom M2M web service. If the device is a sensor, then this web service can be used for collecting as well as viewing sensor data. On the other hand, if the device is an actuator, this web interface can be used to send and view the status of actuation commands.

The demo message sequence shown in Figure 2 is as follows:

[10] http://code.google.com/p/relic-toolkit/

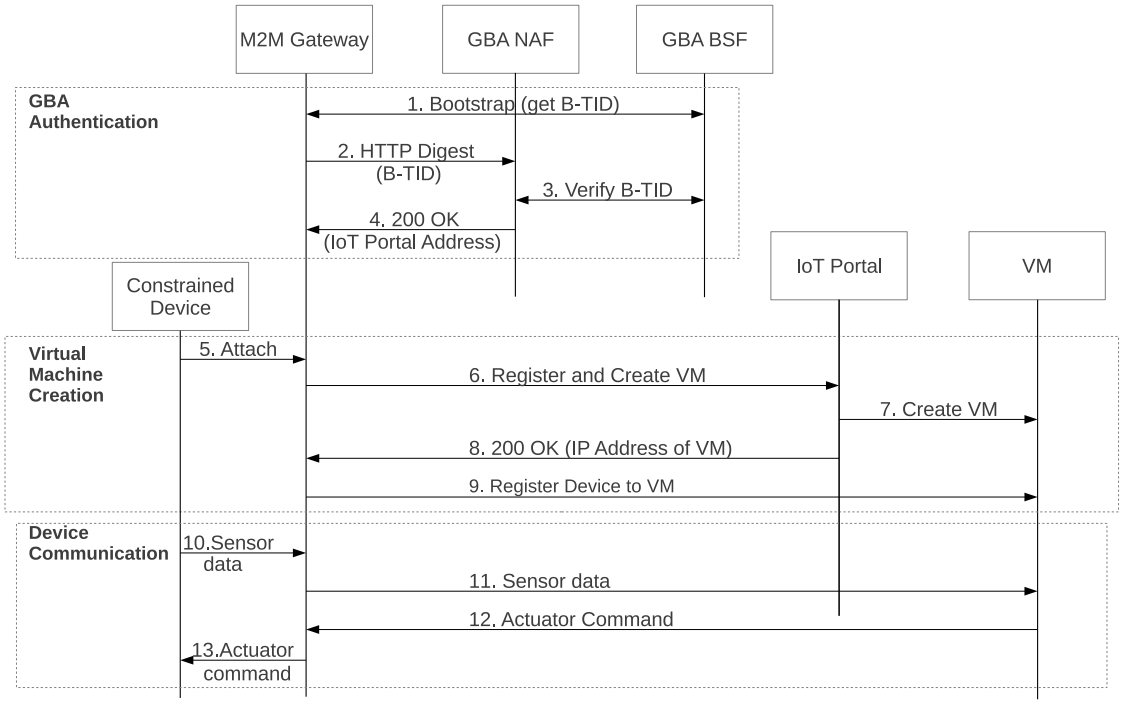

Figure 2: Testbed demo sequence

1. The GW performs GBA [1] bootstrapping with the Booststrapping Server Function (BSF) and both generate a key K_s. The GW also obtains the bootstrapping context identifier (B-TID).
2. Using the B-TID, the GW performs HTTP Digest authentication with the NAF.
3. The NAF requests the BSF for a key matching the received B-TID. Using the B-TID, the BSF generates a key K_sNAF, which is then communicated to the NAF.
4. The NAF, on verifying the digest, replies with an OK message that may contain configuration data, such as the IoT Portal address. Also, the NAF and the GW now share a secret key, K_sNAF which can be used to securely transfer the configuration data.
5. Device sends an attach request to the GW.
6. GW request a new VM for the newly joined device.

7–8. A new VM is created and its IP address is communicated to the GW.

9. The GW registers this new device to the M2M web service in the VM.

10–11. If the device is a sensor, it can start sending signed data to the GW which is forwarded and stored in the VM. A user can use the web service in the VM to view collected and verified data.

12–13. If however, the device is an actuator, the gateway would forward actuation commands to the device from the web service and return any status updates from the device to the web service.

3.1 Demo Requirements

Our demo requires a demo stand with a table to hold a laptop, a display, and a few M2M devices. The demo requires electricity and Internet access. The minimum required setup time is 1 hour.

4. CONCLUSIONS

We demonstrate a system where devices are automatically and securely provisioned and connected to a cloud backend. We note that there are many benefits of placing M2M services into the cloud. Firstly, potentially billions of objects can be managed centrally, removing the need to have local server deployments for services and management. Secondly, it enables dynamic configuration of service and networking resources, which can be instantiated and scaled as per need basis. Thirdly, data published by these smart object can be accessed from anywhere on the Internet and they can also be managed remotely. Fourthly, the solution is completely horizontal, that is, the same platform can be used by any application and user. Services can reside in separate secure domains within the cloud. Lastly, the solution provides lower OPEX and CAPEX by enabling faster deployment of M2M services with minimal configuration and maintenance.

5. ACKNOWLEDGMENTS

This work was partly done in the Internet of Things program[11] of Tivit (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT), funded by Tekes.

6. REFERENCES

[1] 3GPP. TS 33.220 - Generic Bootstrapping Architecture (GBA), 2007.

[2] M. Sethi, J. Arkko, and A. Keranen. End-to-end security for sleepy smart object networks. In *LCN Workshops 2012*, pages 964–972. IEEE, 2012.

[11] `http://www.internetofthings.fi/`