Demo: Do not trust your neighbors! A small IoT platform illustrating a man-in-the-middle attack

Renzo Efrain Navas, Hélène Le Bouder, Nora Cuppens-Boulahia, Frédéric Cuppens, Georgios Papadopoulos

Demo: Do not trust your neighbors! A small IoT platform illustrating a man-in-the-middle attack

Renzo E. Navas, Hélène Le Bouder, Nora Cuppens, Frédéric Cuppens, and Georgios Z. Papadopoulos

IMT Atlantique, UBL, France
{renzo.navas,helene.le-bouder,nora.cuppens,frederic.cuppens,
georgios.papadopoulos}@imt-atlantique.fr

Abstract. This demonstration defines a small IoT wireless network that uses TI CC2538-OpenMote as hardware platform and state-of-the-art IETF network standards such as 6LoWPAN, RPL, and CoAP implemented by ContikiOS. The IoT nodes are controlled from outside the IoT network using end-to-end connectivity provided by IPv6-CoAP messages. We implement a man-in-the-middle attack that disrupts the normal behavior of the system. Our attack leverages on the inherent hierarchical routing topology of RPL-based IoT networks. The demonstration aims at highlighting the need for end-to-end source-authentication and authorization enforcement of information even inside a trusted IoT network. We also provide some insights on how these services can be offered in a IoT-friendly way.

Keywords: IoT · MITM attack · IPv6 · CoAP · RPL · e2e security

1 Introduction and Motivation

Internet of Things (IoT) lack-of-security awareness has been rising in recent years [4]. Mass-media publications label as "IoT" a wide and heterogeneous set of devices: smart-watches, thermostats, surveillance cameras, toasters, refrigerators, light-bulbs, etc. Most of them have more in common with a powerful desktop computer rather than with a 5-dollar system-on-chip. On this paper IoT represents constrained devices as defined on RFC7228 [2]. These devices can not run Linux systems or legacy Internet and security protocols: special solutions suited for their constraints must be used.

For this demonstration we set up an IoT platform reachable by IPv6 from external networks. The platform allows controlling a robot arm from an android tablet using CoAP messages. The defined IoT platform has a compromised node inside, that after behaving as expected for a certain amount of time, executes a Man-In-The-Middle (MITM) attack. The demonstration aims at highlighting the consequences of unrestrictedly trusting nodes of an IoT network, and the need for end-to-end security. Risk of insider attacks is real from the very moment we use devices manufactured or programmed by a party we do not fully trust (i.e. *all*), a mitigation will be running full open-hardware and software solutions.

The rest of the paper is organized as follows: Section 2 presents the platform. Section 3 explains the attack: hypothesis, planning, implementation and execution. Section 4 comments on the feasibility of the attack on real-world settings and on possible solutions. Finally, Section 5 offers a brief conclusion.

2 The Platform

The platform tries to illustrate three concepts associated with the IoT: **heterogeneity**, **connectivity** and **interoperability**. Heterogeneous devices: android devices, WiFi access points, powerful PCs, and constrained IoT nodes; and heterogeneous networks: WiFi, Ethernet, USB-SLIP, IEEE 802.15.4. Connectivity end-to-end is assured by IPv6. Application-layer interoperability is guaranteed by the Constrained Application Protocol (CoAP), the equivalent of HTTP for IoT. A diagram of the platform can be seen on Fig. 1.

The use case involves an Android device that sends IPv6-CoAP packets to an IoT node who controls a robot-arm which serves beverages. The IPv6 packet travels through 4 different layer-2 technologies, but layer-3 (and up) remains unmodified end-to-end. Inside the IoT network the packet is routed by intermediate IoT nodes, leveraging on the RPL routing protocol, until it finally arrives to the destination node which drives the arm according to the CoAP message.

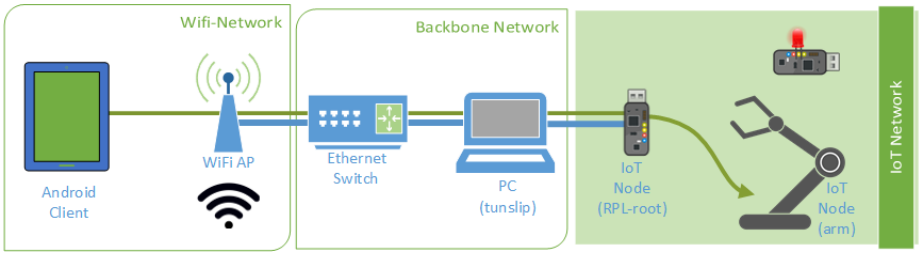

Fig. 1. Diagram of the platform. An heterogeneous platform for end-to-end IPv6-CoAP interoperability with an IoT-node. The green arrow represents the IPv6 message.

The hardware and software components used are the following:

- Android Client: Galaxy Tab A 10.1 (2016). SW: Samsung stock android 7.1.
- WiFi AP + Switch: Router Linksys E900. SW: Custom Firmware tomato-E900-NVRAM64K-1.28.RT-N5x-MIPSR2-140-Max (Needed to add custom IPv6 routing table rules)
- PC Lenovo ThinkPad T460 Intel i7-6600U CPU @ 2.60GHz (x86_64). SW: Ubuntu 17.10 64-bits and binary *tunslip*.
- IoT Nodes: OpenMote-CC2538 Rev.A1 board (SoC: TI CC2538SF53, 256 KB Flash, 32KB RAM). SW: ContikiOS 2.7.

– Robot Arm: RobotGeek Snapper Arduino Robotic Arm[1], five 13kg-cm servo-motors (16 Volts powered). And controller by an Ardhuino-uno-based device.

A real world picture of the platform can be seen on Fig. 2.

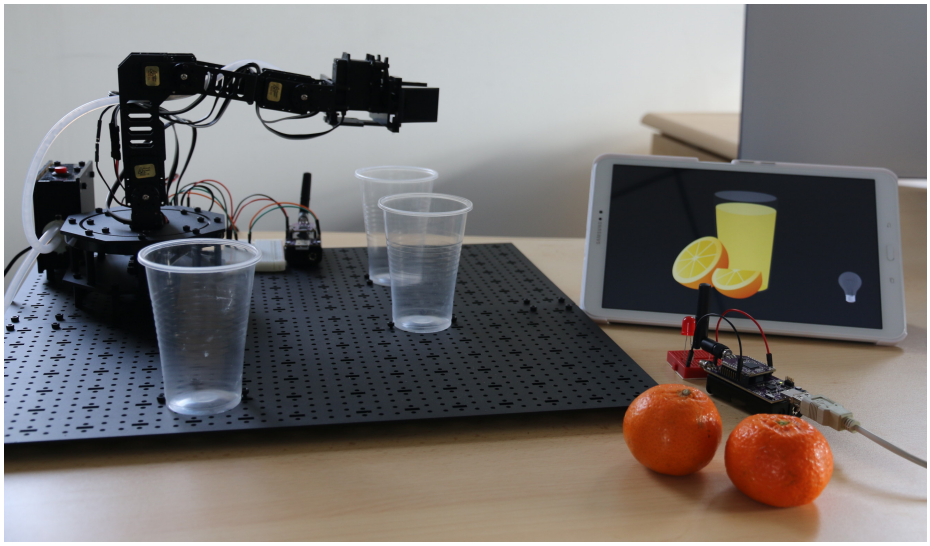

Fig. 2. Real world platform: The IoT-driven water-pouring robot arm. We can see the robot arm, two OpenMotes and the Android tablet.

3 The Attack

3.1 Hypothesis

The following hypothesis are needed to execute our attack:

Internal IoT attacker: We assume an internal attacker, one of the IoT nodes has been compromised or has always been malicious but on a latent state.

Routing tree-like topology: The attack relies on RPL (IPv6 Routing Protocol for Low-Power and Lossy Networks) routing tree-like hierarchy. The goal is for the compromised node to be conveniently placed on the RPL topology, so it becomes a legitimate router of most of the packets on the network. This can be achieved by exploiting well-known RPL vulnerabilities [5] (e.g. rank attack, version number attack). *Note*: For single-hop networks this attack is not possible as is: the malicious node will have to illegitimately intercept and forward packets.

[1] https://www.robotgeek.com/robotgeek-snapper-robotic-arm

3.2 Plan

The following are the ordered steps needed to execute the attack:

1. **Insider IoT-node compromise.** IoT-node has been malicious from the beginning of its life-cycle. A modified version of ContikiOS was flashed on one of the nodes, the node behaves as a regular node until the malicious-mode is activated. The content of the custom code is explained on Subsection 3.3. This kind of insider attack is a realistic threat: normally enterprises and users buy, configure, and use IoT nodes with pre-loaded closed-source firmware.
2. **Malicious-mode activation**. Activation is done by an external agent sending an HTTP request to a specific resource of the compromised node. The node is reachable by IPv6. Automatic activation is realistic also e.g. the compromised node activates itself after 72hs of use, or the 24th. Nov. 2020.
3. **RPL-attack.** The malicious node modifies the RPL-topology to be placed close to the root and legitimately route most of the packets. For simplicity of the implementation, the RPL root node is the compromised node, so no RPL attack was needed. An extension of the demonstration would be performing a RPL-attack.
4. **MITM: In-transit CoAP message modification**. Once the compromised node is placed on a privileged position on the routing hierarchy, it targets IP packets for a fixed destination inside the IoT network. In this demonstration, legitimate CoAP messages that control the robot arm are targeted, and its content is modified. Once in this position on the IoT network several attacks could be performed e.g. Black-hole attack (disrupting routing of the packets), information leakage (e.g. sending a copy of the IoT internal messages to an outsider); Layer-2 security does not prevent any of these attacks, as the compromised node is an insider.

The attack schema can be seen on Fig. 3. To activate the malicious node a Samsung Galaxy S8 cellphone with Android 7.1 is used, but any IPv6 capable device with an HTTP client could have been used.

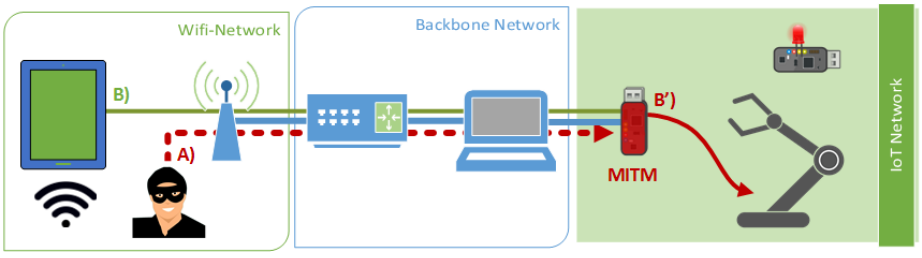

Fig. 3. Attack schema. In *A* the message to activate the malicious mode of the compromised node (MITM). In *B* a legitimate message for the IoT-arm, modified by MITM and routed in *B'*

3.3 Implementation details and Execution

The most relevant part of the code modifications to present is the ContikiOS modification to target and modify specific IPv6 packets. This modification is done inserting code on line 1187 of the Contiki 2.7 file `contiki/core/net/uip6.c` [3] , this is inside the function `uip_process(uint8_t flag)` which does the IPv6 packet processing, and the node is about to forward a IPv6 packet with the line `goto send`. Before sending the packet, it checks if the malicious state is activated, if so it targets a specific IPv6 destination address and the UDP-CoAP port, then it modifies the CoAP message content, and recalculate the UDP checksum. On Fig. 4 a debug console of the malicious node modifying a message in transit is shown.

```
[*******************************    ^O^   *******************************]

                We are now an Evil Node, waiting for Specific
                  CoAP messsage to moddify it in transit

[*******************************   ^---^  *******************************]

Attack_is_on, is this the message we want?: 1

        [**************    We attack!!  ^O^   **************]

        UDP before checksum: 0xdc26
        We modify the message in transit, need to recalculate checksum
        UDP we set to zero checksum: 0x0000
        UDP after checksum: 0xdc3a
Buff:
  0000  60 01 c7 26 00 13 11 3f aa aa 00 00 00 00 00 00  `..&...?........
  0010  00 00 00 00 00 00 00 01 aa aa 00 00 00 00 00 00  ................
  0020  02 12 4b 00 04 30 53 e5 9d 99 16 33 00 13 3a dc  ..K..0S....3..:.
  0030  40 02 54 61 b3 61 72 6d 02 6f 5a                 @.Ta.arm.oZ
        [*****************    END Attack   *****************]
```

Fig. 4. Inside the compromised node while preforming the attack.

The reader can view a short video of the demonstration platform in action on [1], or on this alternative url[2].

4 Reflections and future work

The presented MITM attack is possible because the information intended for the IoT node is not protected end to end. To prevent this attack, at minimum source-authentication (integrity) of the information is needed, this enables to detect messages modified by a third party. Even if the attack on this demonstration seems trivial, it exemplifies what can happen in more complex Cloud-IoT

[2] Demo video: https://youtu.be/Zhrk5-IGKKE

architectures where a false sense of security can be given: we can have strong cryptographic HTTPS-TLS protection from the cloud to an IoT gateway, but inside the IoT we rely on whatever security is offered at Layer 2 (IEEE 802.15.4, Sigfox, LoRaWAN Network Session Key); on such setting this insider IoT attack is still possible.

Security services need to be guaranteed end-to-end. This can be achieved at three different layers of abstraction on the TCP/IP model. *IPsec* at the Internet layer; Datagram Transport Layer Security (*DTLS*) or *TLS* at the Transport layer; And finally, at the Application layer where most current IoT-oriented standardization efforts are being made e.g. Object Security for Constrained RESTful Environments (*OSCORE*) or CBOR Object Signing and Encryption (*COSE*)[6]. Our current research efforts focus on application layer security, also called *object security*, and particularly COSE. We believe solutions at this layer offer the best flexibility (per-message security services), level of abstraction (agnostic to underlying layers), and good message overhead. A comprehensive solution will involve key-establishment protocols, an authorization-framework, a time-synchronization protocol, etc. all these services could leverage from COSE.

5 Conclusion

This demonstration platform illustrates some benefits and challenges of the IoT: connectivity, application-layer interoperability, and weak -or inexistent- security. We focus on IoT-insider attacks, IoT platforms and protocols should be designed with the principle of least privilege in mind: *do not trust your neighbors!* or rather, trust them only with what they need to be trusted. Fine-grained or capability-based authorization is possible for IoT, and the layered nature of protocols also helps. An IoT neighbor node that routes packets for others should only be able to do that, and in an authenticated way. To avoid the unknown evil men-in-the-middle to succeed **security services should be guaranteed end-to-end**. Object security solutions, by means of its flexibility and lower layers independence, seems to be the most suitable tool that can provide it for the future of the heterogeneous IoT.

References

1. Demo video: Iot man-in-the-middle attack (2018), http://www.industry-of-the-future.org/asset/demo/
2. Bormann, C., Ersue, M., Kernen, A.: Terminology for Constrained-Node Networks. RFC 7228 (May 2014). https://doi.org/10.17487/RFC7228
3. ContikiOS: The contiki 2.7 github repository (2018), https://github.com/contiki-os/contiki/blob/release-2-7/core/net/uip6.c#L1187
4. Granjal, J., et al.: Security for the internet of things: a survey of existing protocols and open research issues. IEEE Communications Surveys & Tutorials (2015)
5. Kamble, A., Malemath, V.S., Patil, D.: Security attacks and secure routing protocols in rpl-based internet of things: Survey. In: ICEI 2017 (2017)
6. Schaad, J.: CBOR Object Signing and Encryption (COSE). RFC 8152 (Jul 2017). https://doi.org/10.17487/RFC8152