Implementing a Signing Forms mechanism in an open XMPP Server to reduce successful network attacks

Marcelo Aros
Facultad de Ingeniería
Universidad Andres Bello
Viña del Mar, Chile
marcelo.aros@unab.cl

Romina Torres
Facultad de Ingeniería
Universidad Andres Bello
Viña del Mar, Chile
romina.torres@unab.cl

Abstract

Servers implementing the eXtensible Messaging and Presence Protocol (XMPP) enable the near-real-time exchange of structured yet extensible data between any two or more network entities. In this paper we propose to address network attacks, a security issue, by implementing signing forms over an in-band registration mechanism. We validate our proposal implementing this mechanism over *OpenFire*, which is a free, community-supported and open source XMPP server. Our experiments showed that the number of successful attacks was reduced to 0 when our proposal is implemented.

1 Introducción

XMPP es un protocolo abierto para la comunicación en tiempo real. Existen múltiples ejemplos de aplicaciones que utilizan este protocolo, incluyendo mensajería instantánea, presencia y colaboración [5][10]. Las especificaciones centrales fueron desarrolladas por la Internet Engineering Task Force (IETF) pero es extensible por la XMPP Standards Foundation, quienes estandarizan protocolos publicando XEPs [7], los cuales extienden la funcionalidad de XMPP.

Ahora bien, las redes de Internet de las cosas (IoT) pueden ser implementadas utilizando diversos protocolos de comunicaciones. Aunque a nivel de usuario, una característica relevante es la facilidad que provee el servidor para adicionar nuevas "cosas" (tipo Plug and Play), tales como sensores, controladores, actuadores o concentradores, tal como se aprecia en la Tabla 1, diversas características técnicas de los protocolos utilizados para IoT deben ser considerados al momento de realizar una elección para implementar una red IoT.

Table 1: Comparativa de los cuatro protocolos mas utilizados en IoT, donde el símbolo "✗" significa que no cumple la característica, el signo "✓" significa que la cumple, "(✓)" que la cumple parcialmente, y finalmente "✓✓" que incluso hay más de un componente que realiza la función.

Característica	**HTTP [3]**	**CoAP [11]**	**MQTT [1]**	**XMPP [5][10]**
Request/ Response	✓	✓	✗	✓
Publish/ Subscribe	✗	✗	✓	✓
Multicast	✗	✓	✗	✓
Events or Push	✓	✓	✓	✓
Bypasses firewall	✗	(✓)	✓	✓
Federation	✗	✗	✗	✓✓
Authentication	✓	✓	✓	✓
Network Identity	(✓)	(✓)	✗	✓
Authorization	✗	✗	✗	✓✓
Encryption	✓	✓	✓	✓
End-to-end encryption	✗	✗	✗	✓✓
Compression	✓	✗	✗	✓
Streaming	✓	✗	✓	✓
Reliable messaging	✗	✗	✓✓	✗
Message Queues	✗	✗	✗	✗

Lamentablemente, a nivel de seguridad, aún existen desafíos importantes. CoAP, por ejemplo, es un protocolo de transferencia RESTful para nodos

IN: Proceedings of the IV School of Systems and Networks (SSN 2018), Valdivia, Chile, October 29-31, 2018. Published at http://ceur-ws.org

y redes con restricciones, el cual tiene una pobre capa de seguridad con un *Bypass de Firewall* limitado [11]. MQTT tiene serias vulnerabilidades conocidas, tanto así que incluso gobiernos han prohibido su uso [6]. Tal como se aprecia en la Tabla 1, XMPP es más robusto en términos de seguridad. Permite delegar automáticamente y de manera segura permisos a los clientes - cosas - para registrar una identidad en la red XMPP-IoT dado que utiliza OAuth 1.0 [4] con dos XEPs específicas. La XEP-0077 (`https://xmpp.org/extensions/xep-0077.html`) [9] que describe el mecanismo para registrarse en banda en un servidor XMPP. La XEP-00348 (`https://xmpp.org/extensions/xep-0348.html`)[12] que lo hace tanto para clientes como para el servidor. Ahora bien, dado que esto le da protección a la red IoT contra bots maliciosos mediante resolución de CAPTCHAS (`https://xmpp.org/extensions/xep-0158.html`) [9][8], (opción que puede ser habilitada o no), lamentablemente, las "cosas" no pueden registrarse de forma automatizada en la red IoT. Lamentablemente, si el registro en banda es activado pero sin habilitar la validación por CAPTCHAS, se abre una brecha en la seguridad de la red XMPP-IoT. Debido a esto, es habitual que se deje la red XMPP-IoT con el XEP-0077 activado pero con el XEP-0158 deshabilitado, es decir, que cualquiera que pueda visualizar a nivel de red el servidor puede registrar identidades en el servidor. Esto abre una brecha en la seguridad de la misma red XMPP-IoT.

2 Propuesta

Existe una serie de servidores que implementan XMPP. Existen de uso libre o de pago, de código abierto y desarrollado en diversos lenguajes de programación, documentación, con una comunidad existente en torno al servidor, que corren sobre diferentes sistemas operativos y con diversas XEPs implementadas. *IoT Broker* (`https://waher.se/Broker.md`), servidor XMPP de uso no libre, no es open-source y no hay una comunidad incipiente en torno al proyecto. *AstraChat Isode M-Link*, su uso es de pago. *ejabberd*, *Tigase* y *Openfire* son de uso libre. Los primeros dos poseen una comunidad también incipiente en torno al desarrollo de nuevas características. *Openfire*, por otro lado es de uso libre, open-source, está codificado en Java, funciona bajo los Sistemas Operativos Linux, macOS, Solaris y Windows y tiene una gran comunidad activa. La misma organización ofrece *SMACK*, el cliente.

En este trabajo proponemos desarrollar e implementar el mecanismo descrito en la especificación **XEP-0348: Signing Forms** sobre la **XEP-0077: In-band Registration**, tanto en el servidor como en el cliente, para que, de este modo, se pueda asignar el permiso de crear identidades en la red XMPP-IoT sólo a ciertos clientes. Con esto proponemos reducir a cero la creación de identidades por usuarios o bot maliciosos. En particular para la validación realizada mediante experimentos implementaremos nuestra propuesta en el cliente *SMACK* y en el servidor *Openfire* tal como se aprecia en las Figuras 1 y 2.

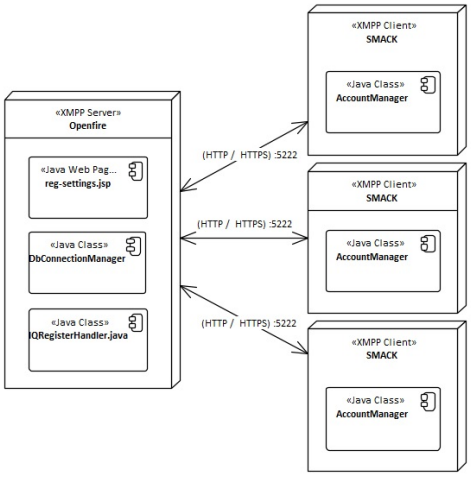


Figure 1: Diagrama de despliegue

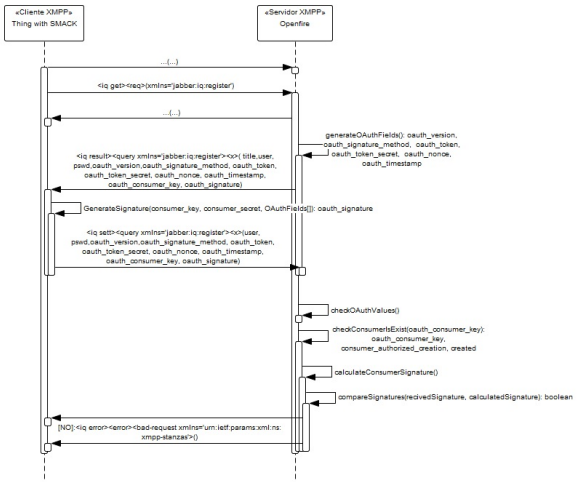


Figure 2: Diagrama de secuencia de In-Band Registration

3 Validación

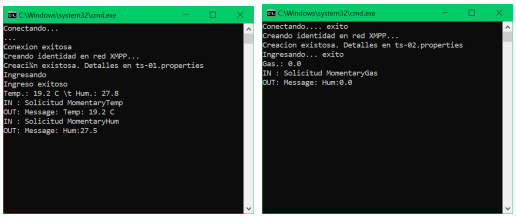
Tal como se ve en la Figura 1 el experimento consta un servidor XMPP *Openfire* modificado con varios clientes XMPP *SMACK* modificados, los cuales son:

- un Cliente XMPP Thing-Sensor (TS–1) que posee un sensor de humedad y temperatura

- un Cliente XMPP Thing-Sensor (TS–2) que posee un sensor de gas,
- un Cliente XMPP Thing-Controlador (TC–1)

Ya realizada las modificaciones, el procedimiento es el siguiente:

1. Ingresar a la consola de administración web de *Openfire*.
2. Ingresar a la sección de registro.
3. Generar Credenciales de Consumidor, es decir, *ConsumerKey*, *Consumer Secret*.
4. Ingresar 3 como cantidad de creación de identidades permitida al *ConsumerKey*.



(a) Thing-Sensor 1 (b) Thing-Sensor 2

Figure 3: Retorno por consola de los clientes.

Para los clientes ingresamos el *ConsumerKey* y el *ConsumerSecret*, para que de este modo, se realice la conexión con el servidor y exista el intercambio de Stanzas con dataforms [2] siendo exitosa la verificación de Signatures, y de este modo, se le delegue al cliente la facultad de crear identidades en el servidor XMPP Openfire. Así, pudimos realizar consultas de temperatura y humedad al cliente TS–1 y de gas al TS–2. Realizado esto, se creó y ejecutó el Bot malicioso en ambos servidores (el modificado con nuestra propuesta y el sin modificar). Los resultados se pueden apreciar en las siguientes Tablas:

Table 2: Test de penetración a servidor Openfire sin implementación

Tiempo (seg.)	Intentos de Creación por segundo	Identidades creadas exitosamente
50	2	100
100	2	200
300	3	900

4 Conclusiones y Trabajos Futuros

Este trabajo brinda un aporte a la comunidad de Internet de las Cosas dado que extiende un servidor libre, de código abierto y apoyado por una comunidad que lo extiende en el tiempo. Tal como se vio en el experimento, dada nuestra implementación es posible reducir el número de ataques de red en el servidor XMPP *OpenFire* que podría sufrir dada la implementación del Inband registration de la XEP-0077 sin la implementación de signing forms.

Table 3: Test de penetración a servidor Openfire implementado el XEP-0348

Tiempo (seg.)	Intentos de Creación por segundo	Identidades creadas exitosamente
50	2	0
100	2	0
300	3	0

Si bien, la brecha de seguridad presente en el XEP-0077 es abordada con la implementación del XEP-0348, quedan varios desafíos, uno de ellos es implementar un Plugin en Openfire para aprovisionamiento. Además, un Registro de Cosas que registre que permisos tiene una “cosa–cosa” y “humano-cosa”.

References

[1] **Banks, A. & Gupta, R.** (**2014**). Mqtt version 3.1. 1. *OASIS standard*, 29.

[2] **Eatmon, R., Hildebrand, J., Miller, J., Muldowney, T., & Saint-Andre, P.** (**2006**). Jep-0004: Data forms. *online] Jan*, 5.

[3] **Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T.** (**1999**). Hypertext transfer protocol–http/1.1. Technical report.

[4] **Hammer-Lahav, E.** (**2010**). The oauth 1.0 protocol (2010). *URL https://tools.ietf.org/pdf/rfc5849.pdf*.

[5] **Internet Engineering Task Force (IETF), P. S.-A.** (**2011**). Rfc6120 extensible messaging and presence protocol (xmpp): Core.

[6] **Lundgren, L.** (**2016**). LightWeight Protocol! Serious Equipment! Critical Implications! [Online; accessed 19-July-2018].

[7] **P. Saint-Andre D. Cridland** (**2015**). XEP-0001: XMPP Extension Protocols. [Online; accessed 20-July-2018].

[8] **Paterson, I. & Saint-Andre, P.** (**2008**). Captcha forms.

[9] **Saint-Andre, P.** (**2004**). Xep-0077: in-band registration. *Jabber Software Foundation*.

[10] **Saint-Andre, P.** (**2011**). Extensible messaging and presence protocol (xmpp): Core.

[11] **Shelby, Z., Hartke, K., & Bormann, C.** (**2014**). The constrained application protocol (coap).

[12] **Waher, P.** (**2017**). Xep-0348: Signing forms.