Desarrollo de una arquitectura de ciberseguridad en redes IoT, aplicada a un ecosistema Zigbee basado en SDN

Reinaldo Scappini[1], Diego Bolatti[1], Sergio Gramajo[1], Jorge Roa[1], Raul Montiel[1]
[1]Universidad Tecnológica Nacional Facultad Regional Resistencia – Chaco - Argentina

{rscappini – dbolatti - sergiogramajo –roajorge - raulmontiel}@gfe.frre.utn.edu.ar

Resumen. Este trabajo propone el desarrollo de arquitectura de ciberseguridad para sistemas basados en IoT, mostrando un ejemplo aplicado a un entorno Zigbee. Para ello se presenta una arquitectura de ciberseguridad innovadora basada en SDN para proteger de manera efectiva las redes IoT. La propuesta centraliza la gestión de políticas de seguridad en un controlador SDN, permitiendo un control granular del tráfico a través de conmutadores OpenFlow. Al aprovechar parámetros de los dispositivos IoT, como identificadores únicos y niveles de batería, se establecen políticas de acceso y priorización personalizadas. La arquitectura se valida en un entorno real utilizando una red Zigbee, demostrando su eficacia en la detección y mitigación de amenazas. Los resultados obtenidos respaldan la viabilidad de esta solución para asegurar la creciente diversidad de dispositivos IoT y garantizar la privacidad de los datos.

Palabras clave: IoT, Ciberseguridad, SDN, Zigbee.

1 Introducción

Algunos dispositivos de IoT tienen protocolos diferentes a los protocolos TCP/IP/protocolos de control de transmisión TCP. Los protocolos propios de un ecosistema IoT, pueden utilizarse para controlar el modo en que los dispositivos IoT se comunican entre sí. Así, deben existir capacidades de filtrado de protocolos específicos del ecosistema IoT para detectar cargas útiles maliciosas que podrían esconderse en sus protocolos. Según la Recomendación UIT-T X.1361 "Marco de seguridad para la Internet de las cosas basado en el modelo de pasarela" [1] , en su apartado 7 establece; "Tendrá que haber una capacidad de cortafuego en la pasarela (gateway) para controlar el tráfico destinado a terminar en el dispositivo". El gateway debería ejercer una función de filtrado de datos concretos destinados a finalizar en ese dispositivo de modo que se utilicen al máximo los recursos computacionales disponibles y limitados. El gateway participa como un elemento único en la arquitectura funcional. A menudo es el primer punto de seguridad fiable en un sistema IoT porque los puntos extremos son más vulnerables a la manipulación física. Desempeña un papel en la IoT que justifica su distinción como activo de seguridad especial aparte de la red, debe tener en cuenta las limitaciones de los nodos de sensor y s menudo puede realizar algunas funciones de seguridad por cuenta de puntos extremos limitados como: gestión de claves, negociación criptográfica, prevención de intrusiones, etc. El gateway tendrá capacidades de seguridad muy diversas dependiendo de factores como: la potencia y las capacidades de los puntos extremos, el diseño de servicio, el diseño de red, las ubicaciones físicas y el contexto de utilización".
La arquitectura de ciberseguridad de este trabajo (ver Figura 1), utiliza Redes Definidas por Software (SDN) [2] para proporcionar y administrar aspectos de seguridad en infraestructuras de IoT. En la infraestructura aplicada en este trabajo, el controlador SDN [3] tiene visibilidad sobre su dominio de red, las aplicaciones que se ejecutan en

el controlador pueden gestionar la seguridad de la infraestructura de red IoT subyacente. El Controlador SDN actúa como una autoridad de decisión de políticas de seguridad, y los conmutadores de red y las puertas de enlace de IoT aplican las políticas de seguridad en la infraestructura de red IoT, tal como se menciona en [4]. Este enfoque basado en políticas proporciona la capacidad de lograr una gestión segura de los flujos de red en una infraestructura IoT de manera dinámica, y de enfrentar ataques de seguridad de manera proactiva. Las políticas pueden imponer la creación de canales seguros para datos de dispositivos IoT autenticados específicos a través de puertas de enlace específicas hacia la nube. Esto puede ayudar a lograr una gestión segura de los flujos de datos en la infraestructura de red IoT.

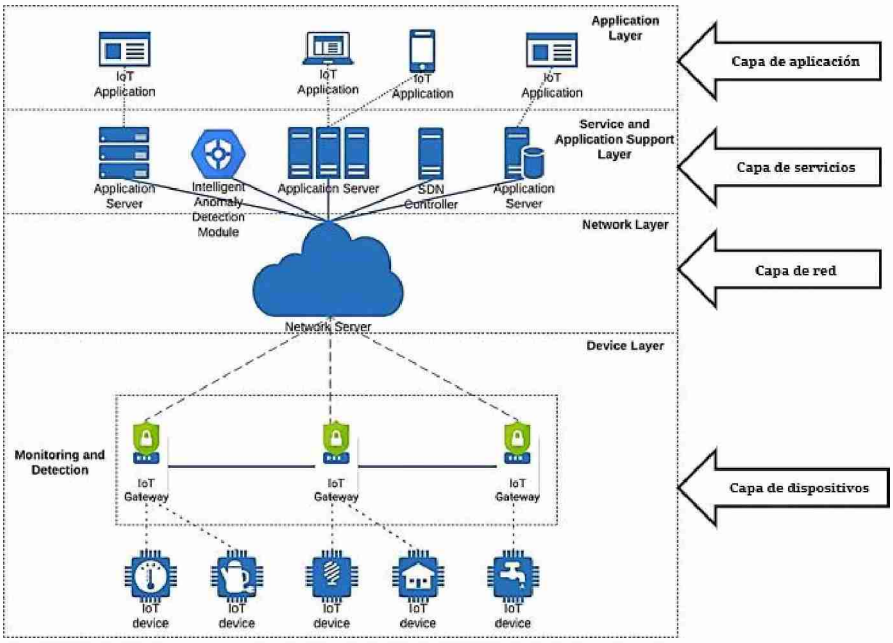

Figura 1 arquitectura de ciberseguridad red IoT (extraido de [5])

Para cubrir los conocimientos mostrados en este trabajo, este artículo se estructura de la siguiente manera. En la sección 2 se describe la infraestructura por capas; en la sección 3 el modelo general de la aplicación en un ecosistema Zigbee [6] y por último las conclusiones.

2 Infraestructura de Estudio

Se puede pensar una infraestructura IoT como compuesta por cuatro capas de arquitectura, a saber: Capa de Aplicación, Capa de servicios Capa de Red, y Capas de dispositivos (Figura 1).

- La Capa de Aplicación proporciona la interfaz para que terceros desarrollen y ejecuten sus aplicaciones para el almacenamiento y procesamiento adicional de los datos de los dispositivos.
- La Capa de Servicios ayuda a realizar acciones dirigidas por aplicaciones de terceros que se ejecutan en la Capa de Aplicación.
- La Capa de Red procesa y enruta los datos a través de la infraestructura de red. Un procesamiento adicional de los datos ocurre en la Capa de Servicios.
- La Capa de dispositivos actúa como una interfaz con el mundo físico y consiste en actuadores y sensores. Esta capa transfiere los datos en bruto a la Capa de Red.

Las características de la tecnología SDN que la hacen una plataforma adecuada para asegurar la infraestructura IoT son las siguientes:

Separación del Plano de Control del Plano de Datos: Esto es útil para diseñar nuestra arquitectura de ciberseguridad basada en políticas en el Controlador SDN en el plano de control y para imponer las políticas de seguridad en los conmutadores de red y dispositivos IoT en el plano de datos. El controlador SDN se comunica con los conmutadores en el plano de datos utilizando interfaces y protocolos abiertos y estandarizados (OpenFlow) [7], lo cual es útil para asegurar las comunicaciones entre la autoridad de decisión de políticas en el Controlador y los mecanismos de cumplimiento en los dispositivos IoT y conmutadores.

Vista del Dominio de Red: El Controlador SDN tiene visibilidad sobre todo el dominio de red bajo su jurisdicción. Esto puede ser utilizado por nuestra arquitectura para lograr una gestión segura de los dispositivos IoT y los flujos en la infraestructura de red. El Controlador mantiene una base de datos de información topológica que registra información sobre todos los dispositivos de reenvío conectados al Controlador. Esto será útil en la especificación de políticas de seguridad basadas en rutas en nuestra arquitectura.

Aplicaciones Northbound de SDN [8]: SDN proporciona una API northbound flexible que nos permite desarrollar aplicaciones seguras o utilizar aplicaciones de terceros para monitorear y controlar de manera segura el comportamiento de los dispositivos IoT y nodos de red en el dominio de la red SDN. Nuestra arquitectura de ciberseguridad tiene una aplicación segura que se ejecuta sobre el Controlador (desarrollada utilizando su API northbound) que proporciona servicios de seguridad en la infraestructura IoT.

2.2 Arquitectura Utilizada

Nuestra propuesta de arquitectura de ciberseguridad basada en SDN utiliza políticas para controlar y gestionar dispositivos IoT, servicios y entidades de red (conmutadores, nodos y puertas de enlace); y además tiene como característica establecer una primera línea de defensa al tener los conmutadores en la capa más baja de la arquitectura. Como se puede ver en Figura 1, El elemento decisor de la arquitectura de ciberseguridad es el controlador SDN (capa de servicios), donde residen y se evalúan las políticas de seguridad. Los actuadores y sensores IoT son los dispositivos finales y se conectan a los Nodos IoT. Estos Nodos IoT están conectados a las Puertas de Enlace IoT, ya sea a través de redes cableadas o inalámbricas. Las Puertas de Enlace IoT contienen conmutadores OpenFlow y están conectadas al Controlador SDN. En algunos casos, los conmutadores OpenFlow pueden actuar como Puertas de Enlace/Nodos IoT en sí mismo. Nuestra arquitectura considera los conmutadores OpenFlow como Puertas de Enlace IoT y punto donde se efectúan funciones de monitoreo y detección. Básicamente los elementos de la capa de dispositivos están conectados a los conmutadores OpenFlow.

Los dispositivos IoT son de naturaleza heterogénea y pueden usar diferentes protocolos de red, mecanismos de autenticación y pueden tener diferentes plataformas de operación y aplicación. Además, puede haber un gran número de dispositivos IoT. Por lo tanto, se necesita una solución escalable que reconozca las capacidades individuales de los dispositivos IoT conectados.

En el contexto de OpenFlow, un "datapath" se refiere al componente lógico en un switch o router que maneja y procesa los paquetes de datos según las reglas definidas por un controlador OpenFlow.

El concepto central es arbitrar un mecanismo de acceso y validación a los dispositivos IoT, que les asigne un canal seguro implementado por un “Datapath” en el conmutador OpenFlow. Con OpenFlow, una parte del datapath reside en el mismo switch, pero es el controlador SDN el que realiza las decisiones de encaminamiento de alto nivel.

El datapath se compone de varios elementos clave:

- Flow Table (Tabla de Flujos): Una o más tablas que contienen las reglas para manejar los paquetes. Cada entrada en la tabla de flujos especifica un conjunto de coincidencias (match fields), acciones (actions) y contadores (counters).

- Match Fields (Campos de Coincidencia): Campos específicos de los paquetes (como direcciones IP, direcciones MAC, puertos, etc.) que se utilizan para identificar a qué flujo pertenece un paquete.
- Actions (Acciones): Las operaciones que se realizan en los paquetes que coinciden con una entrada de flujo. Esto puede incluir acciones como reenviar el paquete a un puerto específico, modificar los campos del paquete, enviar el paquete al controlador, o descartar el paquete.
- Counters (Contadores): Registros que mantienen información sobre el número de paquetes y bytes que han coincidido con cada entrada de flujo, proporcionando estadísticas útiles para la administración de la red.

El datapath, en esencia, es el componente dentro del switch que se comunica con el controlador OpenFlow, ejecuta las decisiones de enrutamiento y reenvío basadas en las reglas definidas por el controlador y gestiona el tráfico de la red. Es el núcleo de la infraestructura de red definida por software (SDN), donde la lógica de control está separada del hardware de reenvío y centralizada en el controlador OpenFlow. En el caso específico de este trabajo, se utilizan los parámetros y atributos correspondientes a los sensores como **Match Fields** y ejecutar las **Actions** en consecuencia.

3 Aplicación en un ecosistema Zigbee

Si bien el modelo estudiado tiene la suficiente generalidad para ser aplicado a diversos ecosistemas IoT, en este trabajo ofrecemos como ejemplo su aplicación en un ecosistema Zigbee. En una red Zigbee, varios parámetros de los sensores pueden ser útiles para un controlador SDN conectado al Coordinador/router Zigbee; Algunos de estos parámetros incluyen:

- ID del Sensor: Identificación única de cada sensor en la red Zigbee.
- Estado del Sensor: Información sobre si el sensor está activo, inactivo o en modo de bajo consumo.
- Tipo de Datos: Tipo de datos que el sensor está recolectando, como temperatura, humedad, movimiento, etc.
- Intervalo de Muestreo: Frecuencia con la que el sensor recolecta datos.
- Potencia de la Señal (RSSI): Indicador de la fuerza de la señal recibida, útil para evaluar la calidad de la comunicación y el alcance de los sensores.
- Latencia de Comunicación: Tiempo que tarda en transmitirse la información desde el sensor hasta el controlador.
- Uso de la Batería: Nivel de batería del sensor, importante para la gestión de la energía en la red.
- Tráfico de Datos: Cantidad de datos que el sensor está enviando, lo cual puede afectar la carga de la red.
- Ubicación del Sensor: Información de la ubicación física del sensor, si está disponible, para optimizar la gestión y el enrutamiento de datos.
- Estado de la Red: Información sobre la conectividad del sensor con la red Zigbee, incluyendo posibles problemas de conexión o interferencias.

Estos parámetros permiten al controlador SDN tener una visión holística del estado de la red y tomar decisiones informadas sobre la gestión de recursos, el balanceo de carga, la optimización del rendimiento y la prolongación de la vida útil de los sensores.

A modo de ejemplo se muestran dos contextos de una red con tecnología SDN y controlador con reglas de flujo basadas en los parámetros de los sensores de una red Zigbee

3.1 Contexto 1

Supongamos que tenemos una red Zigbee con varios sensores de temperatura y humedad distribuidos en un edificio. El controlador SDN quiere optimizar la red para priorizar el tráfico de los sensores de temperatura cuando la temperatura ambiente supera un umbral específico, ya que es crucial para la gestión del sistema de climatización del edificio.

Regla de Flujo

Condiciones:

- Tipo de datos: Temperatura
- Nivel de batería: Mayor al 20%
- Potencia de la señal (RSSI): Mayor a -70 dBm
- Temperatura medida: Mayor a 25°C

Acciones:

- Priorizar el tráfico de estos sensores.
- Reenviar los datos de estos sensores a un servidor específico para procesamiento inmediato.
- Reducir el intervalo de muestreo a cada 5 segundos para obtener datos más frecuentes.

Ejemplo de Regla de Flujo (Pseudocódigo)

```
{
  "flow_rule": {
    "match_fields": {
      "sensor_type": "temperature",
      "battery_level": ">20%",
      "rssi": ">-70dBm",
      "temperature": ">25°C"
    },
    "actions": [
      {
        "action_type": "set_priority",
        "priority_level": "high"
      },
      {
        "action_type": "forward",
        "destination": "processing_server_1"
      },
      {
        "action_type": "set_sampling_interval",
```

```
        "interval": "5s"
      }
    ]
  }
}
```

Explicación

- **match_fields**: Define los criterios para que un paquete de datos coincida con esta regla de flujo. En este caso, el sensor debe ser de tipo temperatura, tener un nivel de batería superior al 20%, una potencia de señal mayor a -70 dBm y una temperatura medida superior a 25°C.
- **actions**: Define las acciones que se tomarán si un paquete de datos coincide con los campos especificados. Aquí, se establece una prioridad alta para el tráfico de estos sensores, se reenvían los datos a un servidor de procesamiento específico y se ajusta el intervalo de muestreo a 5 segundos.

Este es un ejemplo simplificado, pero ilustra cómo se pueden usar los parámetros de los sensores Zigbee en la toma de decisiones y la optimización de una red SDN.

3.2 Contexto 2

Se muestra a continuación, un ejemplo de una regla de flujo para un controlador ONOS (Open Network Operating System) basada en el identificador de un dispositivo Zigbee. Supongamos que tenemos un sensor Zigbee con el identificador sensor-01. Queremos crear una regla de flujo que priorice el tráfico de este sensor y reenvíe sus datos a un servidor específico para análisis.

Regla de Flujo en ONOS

- Identificador del Dispositivo: sensor-01
- Acciones:
    - Establecer una alta prioridad para el tráfico de sensor-01.
    - Reenviar el tráfico a un puerto específico donde se encuentra el servidor de análisis.

Ejemplo de Regla de Flujo (Pseudocódigo ONOS)

```
{"priority": 40000, "timeout": 0, "isPermanent": true,
"deviceId": "of:0000000000000001", "treatment": {
"instructions": [ { "type": "OUTPUT", "port": "3" } ]
}, "selector": { "criteria": [ { "type": "ETH_TYPE",
"ethType": "0x0800" }, { "type": "IPV4_SRC", "ip":
"192.168.1.101/32" } ] } }
```

Desglose del Pseudocódigo

- priority: Establece la prioridad de la regla. En este caso, 40000 es una prioridad alta.
- timeout: Establece el tiempo de expiración de la regla. 0 significa que la regla es permanente.
- isPermanent: Indica si la regla es permanente (true).

- deviceId: El ID del dispositivo OpenFlow en ONOS donde se aplicará la regla. En este caso, of:0000000000000001 es un ejemplo.
- treatment: Define las acciones a realizar en los paquetes que coinciden con los criterios de selección.
    - **instructions**: Especifica las instrucciones a ejecutar. En este caso, reenviar el tráfico al puerto 3.
- **selector**: Define los criterios para seleccionar los paquetes a los que se aplicará la regla.
    - **criteria**: Una lista de criterios. En este caso:
        - **ETH_TYPE**: Filtra los paquetes Ethernet de tipo IPv4 (0x0800).
        - **IPV4_SRC**: Filtra los paquetes cuyo origen IPv4 es 192.168.1.101, que corresponde a sensor-01.

Adaptación a un Dispositivo Zigbee

Para un dispositivo Zigbee, se podría necesitar adaptar el selector a los identificadores y protocolos específicos de Zigbee. Por ejemplo, si Zigbee usa una dirección MAC o un identificador específico en la capa de enlace, podrías ajustar los criterios del selector para que coincidan con esos parámetros.

Ejemplo Adaptado para Zigbee

```
{"priority": 40000, "timeout": 0, "isPermanent": true,
"deviceId": "of:0000000000000001", "treatment": {
"instructions": [ { "type": "OUTPUT", "port": "3" } ]
}, "selector": { "criteria": [ { "type": "MAC_SRC",
"mac": "00:0d:6f:00:01:23:45:67" } ] } }
```

En este caso, 00:0d:6f:00:01:23:45:67 es la dirección MAC del sensor Zigbee sensor-01.

En este ejemplo se muestra cómo crear una regla de flujo en ONOS basada en el identificador de un dispositivo Zigbee, priorizando su tráfico y reenviándolo a un servidor de análisis específico. De igual modo a lo explicado en los párrafos anteriores, se puede implementar un sistema de validación y acceso mediante un token gestionado por un protocolo de seguridad estándar abierto [9],[10] y [11]

Crear un sistema de acceso seguro para la conexión de un sensor Zigbee utilizando un token gestionado mediante un controlador SDN como ONOS implica varios pasos.

A continuación, un posible enfoque:

Paso 1: Configuración del Sensor Zigbee

- Asignación de un Token de Autenticación: Cada sensor Zigbee debe tener un token único que se utilizará para autenticar su conexión.
- Implementación de Mecanismos de Seguridad: Asegúrate de que el sensor tenga habilitados mecanismos de cifrado y autenticación de Zigbee.

Paso 2: Gestión de Tokens

- Generación y Distribución de Tokens: Un servidor de gestión de tokens (Token Management Server) se encargará de generar y distribuir tokens a los sensores Zigbee de forma segura.
- Almacenamiento Seguro de Tokens: Los tokens deben almacenarse de manera segura tanto en los sensores como en el servidor.

Paso 3: Controlador SDN (ONOS)

- Integración del Controlador SDN con el Servidor de Gestión de Tokens: El controlador ONOS debe ser capaz de comunicarse con el servidor de gestión de tokens para verificar la autenticidad de los tokens.

- Configuración de Reglas de Flujo Basadas en Tokens: El controlador SDN ONOS debe configurar reglas de flujo para permitir o denegar el tráfico de los sensores basándose en la verificación del token.

Proceso de Conexión Segura

Paso 1: Solicitud de Conexión

El sensor Zigbee intenta conectarse a la red y envía su token de autenticación al controlador SDN ONOS.

Paso 2: Verificación del Token

El controlador ONOS recibe la solicitud y el token, y verifica el token con el servidor de gestión de tokens.

Si el token es válido, el controlador ONOS permite la conexión del sensor estableciendo reglas de flujo adecuadas.

Paso 3: Establecimiento de Reglas de Flujo

El controlador ONOS configura reglas de flujo específicas para permitir el tráfico de datos del sensor Zigbee a través de la red.

Ejemplo de Implementación

Regla de Flujo en ONOS para un Sensor Zigbee Autenticado

```
{ "priority": 40000, "timeout": 3600, "isPermanent":
false, "deviceId": "of:0000000000000001", "treatment":
{ "instructions": [ { "type": "OUTPUT", "port": "3" } ]
}, "selector": { "criteria": [ { "type": "MAC_SRC",
"mac": "00:0d:6f:00:01:23:45:67" } ] } }
```

Implementación Detallada

1. Servidor de Gestión de Tokens
   - Generación de Tokens: El servidor debe tener una API para generar y distribuir tokens a los sensores Zigbee.
   - Verificación de Tokens: Una API para verificar tokens recibidos desde el controlador SDN.

2. Controlador SDN (ONOS)
   - Integración con el Servidor de Gestión de Tokens:
     - Configura el controlador ONOS para interactuar con la API del servidor de gestión de tokens.
     - Usa un módulo de autenticación en ONOS que se comunique con el servidor de gestión de tokens.

3. Sensor Zigbee
   - Envía el Token al Intentar Conectarse: El sensor envía su token como parte del proceso de conexión inicial.
   - Recibe la Confirmación de Conexión: Si el token es válido, el sensor recibe la confirmación de que está conectado a la red.

3.3 Consideraciones de Seguridad

- Cifrado de Comunicación: Utiliza cifrado para las comunicaciones entre el sensor Zigbee y el controlador SDN, así como entre el controlador y el servidor de gestión de tokens.

- Rotación de Tokens: Implementa políticas de rotación de tokens para mejorar la seguridad.
- Autenticación Mutua: Considera usar autenticación mutua para asegurar que tanto el sensor como el controlador se autentiquen entre sí.

Este enfoque proporciona una manera segura y controlada de gestionar las conexiones de sensores Zigbee a través de un controlador SDN ONOS utilizando tokens de autenticación.

4 Conclusiones

Este trabajo muestra el potencial de la inclusión de la tecnología SDN en las redes IoT. Se trata solamente de una muestra, pues es una tarea que se encuentra en desarrollo en el marco del proyecto de investigación mencionado cuyo marco es más amplio y a futuro busca encontrar consensos y establecer criterios para una recomendación de estandarización en la ITU-T donde el grupo está trabajando desde el año 2018.

Referencias

1. Unión Internacional Telecomunicaciones Recomendación UIT-T X.1361 “Marco de seguridad para la Internet de las cosas basado en el modelo de pasarela” URL: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.1361-201809-I!!PDF-S&type=items
2. Open Networking Foundation “Software-Defined Networking (SDN) Definition” URL: https://opennetworking.org/sdn-definition
3. Open Networking Foundation “Open Network Operating System (ONOS®)” URL: https://opennetworking.org/onos/
4. K. K. Karmakar, V. Varadharajan, S. Nepal and U. Tupakula, "SDN Enabled Secure IoT Architecture," 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Arlington, VA, USA, 2019, pp. 581-585.
5. Bolatti D. Gramajo S. Scappini R. et al. - Technical Report ITU-T YSTR-IADIoT, "Intelligent Anomaly Detection System for IoT". SG20-TD1191 Study Group 20 URL https://www.itu.int/ITU-T/workprog/wp_item.aspx?isn=17917 - Aprobado Julio 2024 (en prensa)
6. Connectivity Standards Alliance “Zigbee La Solución completa para todos los dispositivos inteligentes” URL: https://csa-iot.org/es/todas-las-soluciones/ZigBee//
7. Scappini, Reinaldo et al. “Trabajando con SDN y OPENFLOW” URL: https://ria.utn.edu.ar/handle/20.500.12272/4435
8. VASCONCELOS, Cesar Rocha, et al. Enabling high-level network programming: A northbound API for Software-Defined Networks. En 2017 International Conference on Information Networking (ICOIN). IEEE, 2017. p. 662-667. URL: https://ieeexplore.ieee.org/abstract/document/7899569/
9. Bertolín,J. “Identificación, análisis y evaluación de la seguridad en las Comunicaciones con tecnología ZigBee” URL: https://www.redeweb.com/_txt/682/114.pdf
10. S. Sciancalepore, G. Piro, D. Caldarola, G. Boggia and G. Bianchi, "OAuth-IoT: An access control framework for the Internet of Things based on open standards," 2017 IEEE Symposium on Computers and Communications (ISCC), Heraklion, Greece, 2017, pp. 676-681, doi: 10.1109/ISCC.2017.8024606.
11. Cipriano, M. et al. “Criptografía Liviana para aplicar en IoT e IIoT” XXIV Workshop de Investigadores en Ciencias de la Computación (WICC 2022, Mendoza) pag. 604 – 607 URL: https://sedici.unlp.edu.ar/handle/10915/144941