Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN

ESCUELA POLITÉCNICA SUPERIOR DE JAÉN

Trabajo Fin de Grado

ESTUDIO DE TECNOLOGÍAS DE SEGURIDAD EN APLICACIONES DE INTERNET DE LAS COSAS

Alumno: Daniel Casquel Cruz

Tutor: Don Joaquín Cañada Bago

Dpto: Grupo de investigación Ingeniería de sistemas telemáticos

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Joaquín Cañada Bago, tutor del Proyecto Fin de Carrera titulado: Estudio de tecnologías de seguridad en aplicaciones de internet de las cosas, que presenta Daniel Casquel Cruz, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, junio de 2018

El alumno: Los tutores:

Daniel Casquel Cruz Joaquín Cañada Bago

Contenido

1. Introducción

Actualmente nos encontramos sumergidos en una era completamente desconocida para todos, en la cual todos los dispostivos que se utilizan habitualmente (como los distintos electrodomésticos, coches, etc.) se encuentran conectados a internet.

Así mismo, esto puede extrapolarse al ámbito de la investigación, gracias a la proliferación de unos pequeños dispositivos denominados Arduino, los cuales permiten la automatización de pequeños procesos, ya que poseen una notable capacidad de cómputo y una gran resistencia ante situaciones climáticas muy adversas.

La evolución comentada anteriormente ha desencadenado la aparición de una serie de vulnerabilidades mediante las que se pueden aprovechar de distintos agujeros de seguridad que presentan estos dispositivos así como la red (que es insegura por definición), puesto que estos mismos no poseían la capacidad suficiente para protegerse. Todo esto puede dar lugar a graves consecuencias tales como: el apagón del sistema de iluminación de una smart city, alteraciones severas en marcapasos o incluso intervenir en la conducción autónoma de un coche.

Para hacer frente a esta casuística, es necesario adaptar todas las técnicas de seguridad y los avances que hemos realizado en este ámbito a estos pequeños dispositivos para hacer frente a las botnets (red de área local en la cual se encuentran los dispositivos descritos anteriormente y, aprovechándose de una vulverabilidad, conseguir hacerse con el control para realizar ataques como DDoS, phising o click phraud, entre otros).

2. Objetivos

Este proyecto, se caracteriza por ser un proyecto innovador, ya que esta tecnología es emergente, está en pleno auge y no existen muchos estudios sobre ella. Por ello, los principales objetivos de este proyecto son:

- Estudiar la integración de las tecnologías de seguridad de los niveles de red en aplicaciones de IoT.

- Implementar una aplicación IoT (dispositivos, red y plataforma)
- Integrar seguridad del nivel de red en la aplicación
- Integrar protocolos de seguridad en las comunicaciones
- Verificar el funcionamiento del sistema
- Medir las prestaciones del mismo

3. Estado del arte

3.1. Qué es Internet de la Cosas (IoT)

El internet de las cosas consiste en la conexión de un conjunto de dispositivos que forman parte de nuestra vida cotidiana con internet y que no estaban diseñados desde una primera hora con ese objetivo. Algunos ejemplos de esto son: bombillas, termostatos, entre otras cosas.

Este concepto fue introducido por Kevin Ashton en el Auto-ID Center del MIT.

El desarrollo de esta tecnología nos ha permitido obtener una mayor cantidad de datos analizar debido a que es un tipo de red en la cual no hace falta que interactúe con un humano para que realice su actividad correctamente. Pero a pesar de todos los beneficios que nos puede aportar, también posee unas desventajas desde el punto de vista de la seguridad, y en este proyecto vamos a analizar algunas de las técnicas que se podrían utilizar para mitigar estas desventajas.

3.2. Qué es la seguridad

La seguridad en términos generales consiste en el conjunto de medidas que son establecidas para evitar cambios impuestos por el exterior en nuestra estructura.

Desde el punto de vista de la seguridad cibernética, hace referencia a la protección de los distintos equipos que forman parte de una red para evitar cualquier daño hardware, software, de datos o evitar la interrupción de servicios que se provean al exterior. Algunos ataques específicos a evitar son: backdoor, escaladas de

privilegios, la utilización de nuestra red como botnet para realizar DDoS, spam, phishing, entre otros.

3.3. Seguridad en redes locales

Una red local está compuesta por una gran cantidad de nodos que se encuentran interconectados entre sí.

3.3.1. VLAN

Una VLAN es una red virtual de área local. Es utilizada para crear distintas redes lógicas dentro de la misma red física. Gracias a que los distintos elementos que forman parte de la VLAN se pueden configurar por software, facilita tanto el diseño como la implementación de la red. Esta topología de red presenta ventajas tales como:

Permitir a los administradores de la red añadir seguridad adicional a la comunicación

Permite añadir nuevos integrantes a la red de forma más sencilla

Es flexible, ya que permite configurar dispositivos en una red centralizada pese a que se encuentran en distintas posiciones geográficas.

Reduce el tiempo de latencia y el tráfico de la red.

Pero también presenta algunas desventajas:

Gran riesgo con virus si uno de los equipos se encuentra infectado.

Limitaciones de equipos para controlar todos los dispositivos que se encuentran dentro de la red. Muchas veces es necesario tener más routers para controlar la carga de trabajo

Es más efectiva en términos de latencia que una WAN, pero sigue siendo menos eficiente que una LAN.

3.4. Seguridad en internet

Internet desde su creación, se ha caracterizado por ser un conjunto descentralizado de redes de comunicación interconectadas, en la cual nunca se ha priorizado la seguridad, siempre fue la velocidad de la comunicación.

Si queremos centrarnos en términos de seguridad, podemos planteárnosla desde dos puntos de vista distintos: seguridad en la red y seguridad en los distintos equipos que conforman esa red. Cabe destacar que estos dos puntos de vista son capaces de ser puestos en común para conseguir mayor robustez.

3.4.1. VPN

Una red virtual privada nos aporta la posibilidad de conectarnos a una red privada a través de internet. Esto nos permite establecer una serie de usuarios capaces de establecer comunicación con un equipo que pertenece a la red privada y poder transmitir datos de forma segura.

3.5. Seguridad en los dispositivos

El otro punto de vista es centrar la seguridad en la comunicación entre los distintos dispositivos que forman parte de la red.

3.5.1. AES

El cifrado AES es un sistema criptográfico simétrico utilizado para encriptar los mensajes entre el emisor y el receptor. Fué elegido por el instituto Nacional de Normas y Tecnología como sucesor de DES. Este algoritmo también es conocido por el nombre de “Rijndael” y utiliza un esquema de cifrado de bloques.

Este algoritmo utiliza una clave, que tiene que ser la misma para el emisor y el receptor, con la cual se cifra el mensaje a transmitir. El único problema de este algoritmo es la distribución de la clave utilizada en el proceso.

Este algoritmo se caracteriza por tener una estructura diseñada para resistir los criptoanálisis lineales y diferenciales. Permite varios tamaños de clave de 128, 192 o 256 bits. Utiliza una matriz de 4 x 4 bites para operar.

En el proceso de encriptación, utiliza las siguientes operaciones:

“AddRoundKey”: el algoritmo comienza con esta operación, en el cual se van a ejecutar las 4 operaciones descritas en este apartado para cada una de las iteraciones en el ciclo.

“SubBytes”: cada byte es reemplazado por un byte de la matriz 4 x 4.

“ShiftRows”: las filas son desplazadas una posición a la izquierda. La fila que se encuentra más arriba no se desplaza, la segunda se desplaza una posición, la tercera dos posiciones y la cuarta tres posiciones.

“Mix colums”: aquí los cuatro bytes de cada columna son mezclados multiplicando por polinomio que ha sido calculado previamente para obtener una nueva columna.

AES se caracteriza por ser un algoritmo de cifrado por bloques y dependiendo de cómo gestione esos bloques se llama modo de cifrado. Los modos de cifrado más significativos son:

ECB (Electronic Code Book Mode)

Este modo de cifrado más sencillo de todos. Consiste en partir el mensaje en trozos y cifrarlos por separado.

CBC (Cipher Block Chaining Mode)

Este modo de cifrado ha sido elegido como estándar por el Instituto Nacional de Estándares y Tecnología (NIST). Actúa como el modo ECB, pero añadiendo algunos aspectos de seguridad.

Consiste en trocear el mensaje a cifrar y añadir cada mensaje cifrado a la siguiente parte del mensaje a cifrar mediante la operación XOR. Como en el primer mensaje no se puede añadir ningún mensaje ya cifrado, se utiliza un vector de inicialización. Es necesario utilizar un vector de inicialización que sea aleatorio para evitar ataques de diccionario.

CTR (Counter Mode)

CTR usa un cifrado de bloque para producir un flujo pseudo aleatorio. Posteriormente, este flujo se añade al texto plano mediante la operación XOR para dar lugar al mensaje cifrado.

Para generar la llave se cifra un contador con un numero aleatorio mediante ECB y se va incrementando. Es necesario que ambas partes de la comunicación conozcan este valor.

OFB (Output Feedback Mode)

OFB es un sistema de flujo que también ha sido estandarizado por el NIST. En este caso, genera una llave a partir del cifrado del bloque anterior de la llave. Para la primera iteración se utiliza un vector de inicialización.

CFB (Cipher FeedBack Mode)

CFB también ha sido estandarizado por el NIST y es muy similar a OFB. La principal diferencia es que para generar la llave utiliza el último bloque de cifrado en lugar del último bloque de llave como utiliza OFB.

3.5.2. SSL/TLS

Transport Layer Security (TLS) es el sucesor de Secure Socket Layer(SSL), que actualmente se encuentra obsoleto por IETF (Internet Engineering Task Force), son protocolos criptográficos que nos proporcionan comunicaciones seguras sobre una

red. Lo que nos intentan proporcionar es seguridad e integridad de los datos en las comunicaciones que se llevan a cabo entre los distintos dispositivos que se encuentran en la red.

Este protocolo a su vez, se divide en dos protocolos en capas distintas para poder llevar a cabo la comunicación:

Protocolo TLS Record: lleva a cabo la autentificación para que la transmisión de datos sea privada y fiable

Protocolo TLS Handshake: aquí se negocia el contenido del mensaje de forma segura. En cada mensaje se especifica el “content_type” y se cifra y se empaqueta con un código de autentificación.

Por lo tanto, TLS se encarga de establecer la comunicación, gestionar los certificados para autentificarse y llevar a cabo la transmisión de datos.

(insertar ejemplo de esquema de comunicación)

3.5.3.MQTT

Protocolo de paso de mensaje ligero caracterizado por ser M2M (Machine to machine). Fue diseñado como un sistema de suscripción muy ligero. Se utiliza para conexiones con zonas remotas, como por ejemplo comunicar distintos tipos de sensores y almacenar las distintas informaciones que recogen en una base de datos. Algunos de los principios en los que se basa mqtt son:

- Simplicidad
- Sistema de suscripción/publicación
- Proveer de calidad de servicio
- Tener la menor cantidad de acciónes inesperadas que requieran de supervisión humana.

3.5.4.Radius

Radius (Remote Authentication Dial in User Service) es un protocolo que define una serie de reglas para establecer las comunicaciones entre los distintos dispositivos. Principalmente aporta las siguientes 3 funcionalidades:

- Autentificar usuarios o dispositivos para permitirles acceso a la red.
- Autorizar a esos usuarios o dispositivos para tareas específicas en la red.
- Infomes para realizar un seguimiento a como se utilizan estos servicios
- Los tres servicios anteriormente nombrados, son reconocidos como la triple AAA (Authentication, Authorization and Accounting).

Está soportado en el puerto 1812 y normalmente es usado en servicios DSL (digital subscriber line), VPNs (virtual private networks).

Autentificación: se denomina autentificación al proceso de validación de la identidad.

Autorización: se refiere al proceso de determinar que permisos son garantizados para cada usuario.

Reporte: hace referencia al almacenamiento de todas las fuentes que son utilizadas por cada usuario el tiempo que ellos se encuentran dentro de la red.

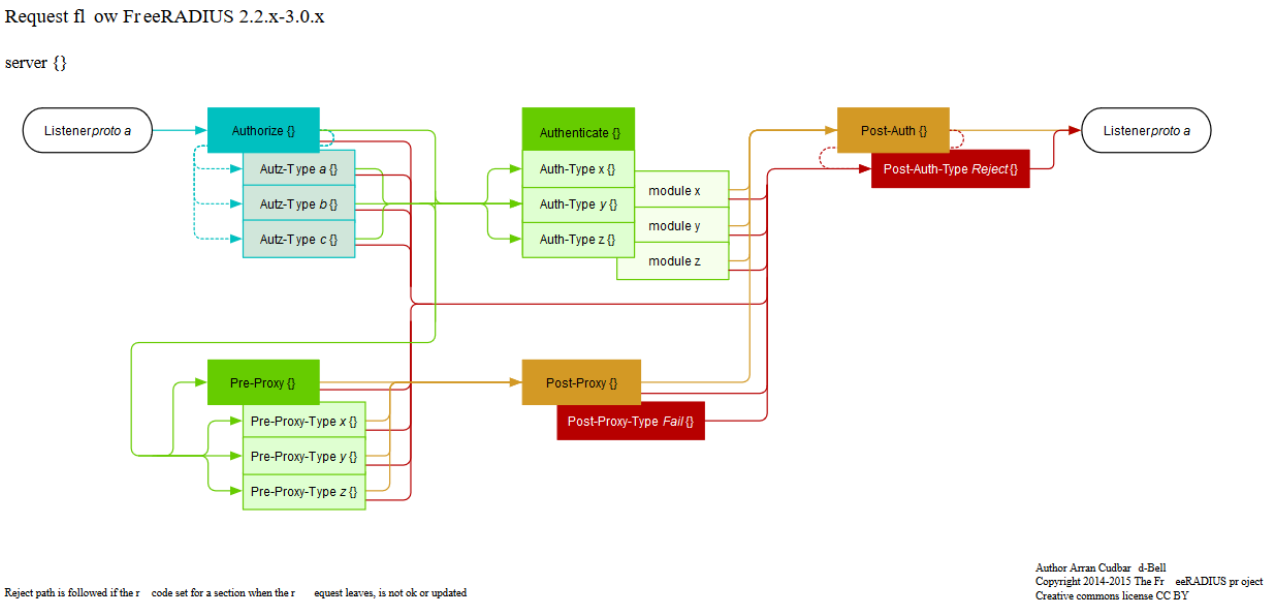
Además, también posee un sistema de auditoría que consiste en el análisis exhaustivo de los logs obtenidos en la parte de Reportes. La información que nos podemos encontrar acumulada en esta parte puede estar relacionada con el tiempo de uso del sistema, la cantidad de información enviada o recibida por cada sesión.

Radius es un protocolo de red que define un sistema de reglas necesarias para la comunicación entre la red de dispositivos. Utiliza una estructura cliente-servidor donde el cliente es el encargado de enviar peticiones al servidor Radius. Mientras que el servidor procesa la petición y envía su respuesta. La lista que se encuentra a continuación muestra todos los elementos que forman parte de Radius y que función poseen:

RADIUS Components		
Component Name	**Functions**	**Examples**
User / Device	Requests access to the network.	Laptop Asymmetric Digital Subscriber Line (ADSL) Modem VOIP Phone
Network Access Server (NAS)	Provides access to the network for the user/device.	Switch Wireless Access Point DSLAM VPN Terminator
Authentication Server	Receives authentication requests from the NAS. Returns authentication results to the NAS. Optionally requests user and configuration information from the database or directory. May return configuration parameters to the NAS. Receives accounting information from the NAS.	FreeRADIUS Radiator IAS NPS ACS
Data Store	Optional database or directory with user authentication and authorisation information. RADIUS server communicates with the data store using DB API or LDAP.	SQL Database Kerberos Service Server LDAP Directory

(freeradius, s.f.)

El esquema de comunicación de radius es:



[1]En el modelo anterior, podemos observar cómo se llevan a cabo las comunicaciones y con las diferentes casuísticas que se pueden presentar, tanto de aceptación del mensaje como de su rechazo.

[1] Imágenes obtenidas de la documentación oficial de freeradius

4. Seguridad en internet de las cosas (IoT)

4.1. Introducción al Proyecto

La idea de este proyecto surgió a través de diversas asignaturas relacionadas con el ámbito del internet de las cosas, un mundo que se encuentra en actual expansión y se están realizando diversos proyectos como: las ciudades inteligentes las cuales apagan las luces dependiendo del tránsito de los peatones durante la noche, los diversos sistemas que poseen los coches ya sean de estacionamiento autónomo. Pero también se han llevado a cabo diversos ataques maliciosos como: la red mirai que estaba constituida por unas cámaras ip de las cuales se aprovechaban de la configuración por defecto para poder mantenerlas bajo su poder y hacer diversos ataques, la evolución de ataques de DoS a DDoS debido a la creación de las botnets, ataques de phising y de click fraudulento.

Todo esto me llevó a elaborar este proyecto, estudiar la topología de las redes que se conforman, analizar que técnicas que están utilizando para evitar estos ataques y obtener conclusiones poniéndolas en marcha.

4.2. Explicación de la metodología utilizada

La metodología que he utilizado para este proyecto ha sido una basada en prototipado. Esta metodología consiste en realizar pequeñas pruebas y simulaciones con las distintas técnicas a estudiar, ponerlas en funcionamiento y analizar cómo se comporta e intentando que las pruebas sean lo más sencillas posible (aunque todo el tema de la seguridad sea muy complejo por definición y haya grupos de desarrollo de cientos de personas dedicados solamente al estudio de todas estas técnicas y a su puesta en funcionamiento para diversas empresas).

El modelo de prototipado se caracteriza por tener unas estapas muy claras y facilmente distingibles:

- Recolección de requisitos: en esta estapa hay que identificar que queremos conseguir e identificar las partes principales de nuestro proyecto.
- Modelado: hacer un esbozo sobre nuestra prueba, la tecnología a usar y como se estructuraría la topología de la red
- Construcción del prototipo: puesta a punto del modelo. Crear una primera impresión del modelo.
- Desarrollo del modelo: realizar modificaciones hasta que funcione correctamente, incluso introduciendo nuevas tecnologías
- Refinamiento del prototipo: mejorar el prototipo hasta la consecución de los objetivos

Y como todos los sistemas, posee una serie de ventajas y desventajas. Las ventajas son:

- Reduce el riesgo de construir sistemas que luego no tengan la funcionalidad esperada
- Reduce el costo en tiempo y aumenta la probabilidad de exito
- Exige disponer de una herramientas que se adecuen a la finalidad establecida
- Ofrece un mejor enfoque del problema al intentar reducirlo a su minima expresión y complejidad

Y la principal desventaja que presenta esta metodología es:

- Caer en la tentación de ampliar el prototipo sin conseguir previamente todas los objetivos que se habían propuesto

4.3. Dispositivos utilizados para el desarrollo

4.3.1. Router

El router utilizado para todas las pruebas ha sido el cisco rv 120w. Este router se caracteriza por tener las siguientes características:

- Alta conectividad, conectividad inalámbrica basada en las normas 802.11n
- Switch de 10/100 de 4 puertos integrado
- Posibilidad de montar redes privadas virtuales
- Posibilidad de utilizar esas VPN con IPsec
- Firewall con inspección de paquetes SPI (stateful packet inspection)
- Interfaz web intuitiva para poder configurar el dispositivo
- Compatibilidad con la utilidad Cisco FindT Network Discovery Utility

Añadir fuente descripción cisco

[2]

4.3.2.Switch

Referenciado en el modelo de router anteriormente explicado.

4.3.3.Arduino Mega

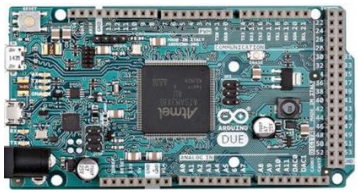
En este proyecto, se ha utilizado un arduino Mega, concretamente el mega 2560 el cual cuenta con 54 pines digitales I/O, 16 analógicos, 4 UARTs (Puerto serie hardware), un power Jack, un ICSP header y un botón de reset. Este dispostivo, es una versión mejorada del arudino mega y aporta mucha más versatilidad que alguno de sus predecesores como el arduino Uno.

[2] Imagen obtenida de las especificaciones dadas por el fabricante

[3]

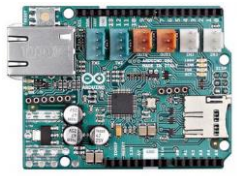
4.3.4.Arduino Due

Otro dispositivo que se ha utilizado en este proyecto ha sido el arduino Due. Es la primera placa arduino que posee un nucleo ARM de 32 bits. El procesador es Atmel SAM3X8E ARM Cortex-m3 CPU. Posee 54 I/O digitales, de las cuales 12 pueden ser utilizadas con PWM (Pulse width modulator), 12 analógicas, 4 UARTs (Puertos series hardware), un reloj de 84 MHz, un usb otg, posee dos DAC (digital to analog), 2 TWI, un power Jack, un SPI header y otra JTAG, un botón de reset y un botón para eliminar datos que contenga.



4.3.5.Ethernet Shield II

Esta placa fue utilizada para dar la posibilidad de conexión a internet, tanto al arduino mega como al due, a través de una red. Solamente hay que acoplar este módulo al arduino que queremos conectar y utilizar un cable con clavija RJ45.



Se caracteriza por trabajar a 5V (la cual es soportada por la placa arduino a la que se conecta), el controlador ethernet es un W5500 con un búfer interno de 32k,

[3] Imágenes proporcionadas por arduino

tiene una velocidad de conexión de 10/100 mb y se conecta a la placa arduino a través del puerto SPI.

4.3.6. Wifi Shield R3

Esta placa realiza una función bastante similar a la anteriormente descrita, con la diferencia de que esta se utiliza para conectarse a una red mediante Wi-Fi. Las características que presenta son:

Requiere una placa Arduino

- Funciona a 5V (soportada por la placa Arduino)
- Es compatible con Arduino due
- Se conecta mediante 802.11b/g
- Utiliza encriptación WEP y WPA2 personal
- Se conecta con la placa Arduino mediante el Puerto SPI
- Posee un slot para una tarjeta SD
- Tiene cabeceras ICSP
- Utiliza conexión FTDI para la depuración de la wifi shield
- Posee un mini USB para actualizar el firmware de la Wifi Shield

[4]

4.3.7. Wemos D1 Mini

Esta placa se caracteriza por poseer como procesador un ESP8266 el cual nos proporciona la posibilidad de conectarse mediante Wi-Fi. Se programa exactamente igual que un Arduino y posee 11 I/O digitales y todos los pines poseen interrupciones,

[4] Imagen proporcionada por arduino

pwm, I2C, one- wire except el D0, una entrada analógica a 3,2V, un Puerto micro USB para programarlo y presenta compatibilidad con Arduino, nodeMCU y MicroPython.

[5]

4.3.8.Esp8266 12-E

Es una placa NodeMCU muy similar a la anterior ya que comparten procesador ESP8266 además de aportar la misma versatilidad para la conexión a redes Wi-Fi. Se programa exactamente igual que un Arduino.

Sus especificaciones son las siguientes:

- Voltaje: 3.3V
- Wifi-direct(P2P)
- Consumo: 170 mA
- Memoria flash disponible: 16MB max
- Integra pila para los protocolos TCP/IP
- Procesador: ESP8266
- Ram: 32k + 80k
- GPIO: 17(multiplexado con otras funciones)
- Analógico a digital: 1 salida
- Soporte para 802.11b/g/n
- Número maximo de conexiones concurrentes TCP: 5

Cabe destacar que D0 solamente puede ser usado como GPIO de lectura/escritura, no soporta interrupciones, ni pwm, ni i2C ni one-wire.

[5] Imagen proporcionada por su fabricante

4.3.9. Raspberry

Raspberry pi es un ordenador en una placa reducida, y tiene capacidad de computo suficiente para la labor que tiene que desempeñar en este proyecto. Actúa como servidor centrar donde reside toda la información que le llega de los distintos arduinos y la almacena. Una de sus grandes virtudes es su escaso consumo.

Esta desarrollada por la fundación Raspberry PI y posee las siguientes características:

- Procesador: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LDDR2 SDRAM
- 2.4GHz and 5GHz IEEE 802.11 b/g/n/ac Wireless LAN, bluetooth 4.2, BLE
- Gigabit ethernet sobre Usb 2.0
- 40 pines GPIO
- HDMI
- 4 puertos USB 2.0
- CSI camera para conectar a una raspberry pi camera
- DSI display para conectar a un raspberry pi touchscreen display
- 4 salidas estéreo y un puerto de composición de video
- MICRO SD para cargar el sistema operativo
- 5V/2.5 DC alimentación
- Soporta POE (Power-over-Ethernet)

[6]

4.4. Herramientas utilizadas para el desarrollo

[6] Imagen proporcionada por raspberrypi.org

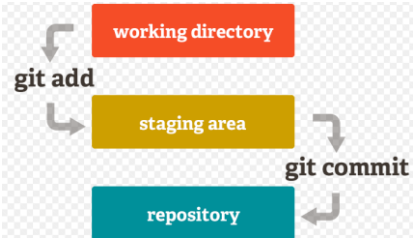
En este apartado del proyecto, procedo a mencionar y explicar cada una de las distintas herramientas que se han usado tanto a la hora de desarrollar el código de las distintas pruebas realizadas en los prototipos como las diferentes herramientas de control que he usado para contabilizar cada uno de los aspectos que faltaban por desarrollar y apuntar las partes que ya se encontraban desarrolladas.

4.4.1.Herramientas de control

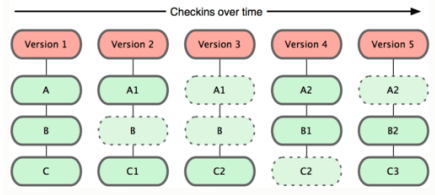
Estas herramientas han sido utilizadas para gestionar y clarificar cuales eran los distintos objetivos que nos encontramos en el proyecto, contabilizar que cosas habían sido desarrolladas y cuáles eran los objetivos que aún nos faltaban por conseguir.

4.4.1.1. Qué es git

Git es una herramienta VCS (version control system) destinada a llevar un control exhaustivo del código que se desarrolla. Esta herramienta es muy utilizada tanto en desarrollos individuales como en desarrollos grupales ya que nos permite obtener códigos que se encuentran en etapas anteriores a la que nosotros podamos poseer en nuestro dispositivo.



Una de las principales diferencias que presenta git frente al resto de VCS es que almacena almacena sus datos. Se caracteriza por almacenar “instantáneas” del sistema de archivos como refleja la siguiente imagen:



En la imagen se puede apreciar el funcionamiento de git. Podemos observar que, a cada instante, en cada una de las versiones se generan 3 ficheros (versión 1) y

[7] Imágenes obtenidas de git-scm

vemos el funcionamiento de esto en versiones posteriores. Y esta característica provoca que Git tenga casi todos los aspectos de control en consideración al ser un mini sistema de archivos más que un VCS al uso.

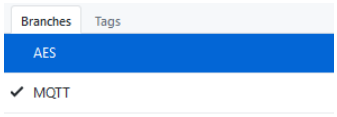
4.4.1.2. Github

Es una plataforma en la cual se nos permite alojar cualquier desarrollo que hagamos utilizando el VCS (sistema de control de versiones) de Git. Esta plataforma se caracteriza por poder almacenarlos de forma pública y privada (los estudiantes tienen una versión que tienen que pedir, para poder tener de forma ilimitada repositorios privados, si no, la versión gratuita solamente te permite 3). Se desarrolló en ruby on rails.

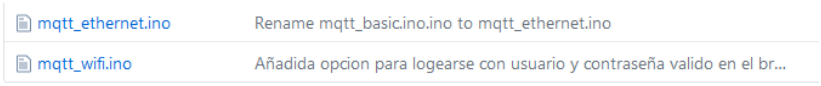
Cabe destacar que la gran alternativa de código abierto es Gitlab.

4.4.1.3. Estructura del repositorio

La estructura llevada a cabo para el desarrollo ha sido de estructurar todas las pruebas realizadas en sus respectivas ramas y no se ha subido nada de código relacionado con la estructura o configuración del servidor (que en nuestro caso es las raspberry).



Y dentro de cada rama, encontramos todo lo relacionado con el código que hemos desarrollado para cada uno de los arduinos empleados en cada prueba.



Cabe destacar, que tampoco se encuentra nada de información relacionada con la seguridad empleada en cada prueba, ya sea aclaración de los algoritmos usados o creación de certificados.

4.4.2.Lenguaje utilizado: C++

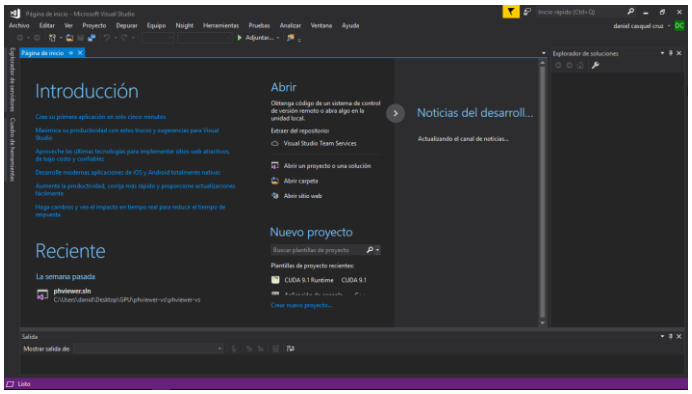
El lenguaje empleado para el desarrollo de todos los prototipados ha sido C++. Este lenguaje se ha utilizado para la creación de un “servidor” en el cual se encargaba de cifrar y descifrar mensajes mediante AES con CBC.
Otro de los motivos por el cual fue elegido este lenguaje es por su compatibilidad con arduino. Para poder programar los distintos arduinos se utiliza un lenguaje C además de la velocidad, eficacia y eficiencia que nos proporciona, teniendo todos los elementos que intervienen bajo nuestro control.

4.4.3.Visual Studio

Este ide es propiedad de Microsoft y cabe destacar que es la mejor herramienta para el desarrollo de un lenguaje como el que hemos elegido anteriormente debido al magnífico compilador que nos proporciona.

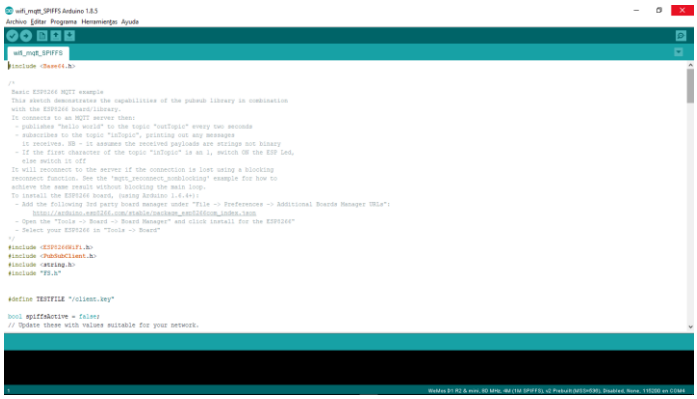
Hoy en día, cuenta con 3 versiones distintas que son: community, Enterprise y professional. Cabe destacar que se encuentra tanto para el sistema operativo Windows como para MacOs. Lamentablemente, para Linux no se encuentra y tendríamos que usar visual studio code para dar formato a nuestro código, generar manualmente nosotros el makefile de nuestro proyecto y utilizar gcc (hay otros muchos compiladores, solamente he indicado uno de ellos) como compilador para nuestra solución.

Hay que mencionar que no solamente proporciona desarrollo para C++, si no también C#, JScript, Python, NodeJs, desarrollo de ASP.Net y web y desarrollo de videojuegos integrando unity con C++.

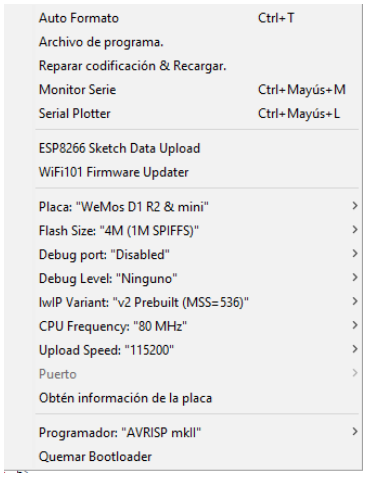


4.4.4.Arduino IDE

Este IDE es el proporcionado de forma oficial por Arduino. Tiene soporte tanto para Windows, como para MacOs y para Linux. Cabe mencionar que automatiza todo el proceso de crear proyecto, manejando el por si solo las distintas rutas que debe de establecer con los ficheros.



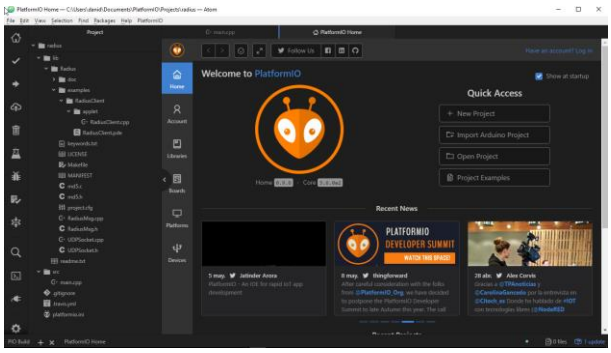
Además, tienes que configurar que tipo de placa vas a usar, puerto en el cual la has conectado.



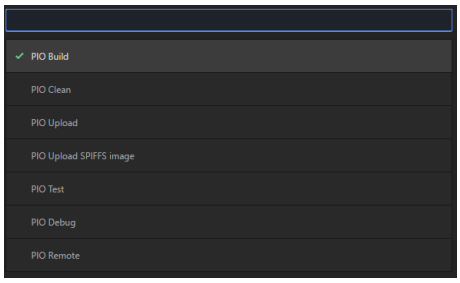
4.4.5. Platform IO

Este es un ide que ha sido creado por la comunidad para poder desarrollar código para Arduino de una forma mucho más sencilla.

Es un plugin que se encuentra tanto para Atom como para VSCode. Se caracteriza por tener una interfaz mucho más intuitiva que el ide de Arduino. Automatiza todos los procesos de selección de placa, solamente tenemos que elegir que placa vamos a programar al principio para crear el proyecto. Además añade una serie de tutoriales de cómo se añaden librerías externas que no se encuentren en su propio repositorio de librerias.



También nos aporta distintos metodos como el SPIFFS, del cual hablaremos en apartados sucesivos.



4.4.6.Librerías de Arduino

En este apartado, vamos a explicar y referenciar todas las librerías, tanto las proporcionadas por arduino de forma oficial como aquellas que han sido desarrolladas por la comunidad que da soporte al mismo.

4.4.6.1. Ethernet

Esta librería se diseñó para trabajar de forma cableada (cable de categoría 5e o superior junto a una clavija rj45) con el arduino shield ethernet y es la que nos permite conectar nuestra placa a internet.

Existen dos versiones de la librería ethernet, la 1 que es la que da soporte al chip W5100 y la 2 que da soporte al chip W5500. Entre la versión 1 y 2 solamente difieren del chip que soportan, ya que ambas contienen los mismos métodos. Los métodos que presentan son:

- Begin: Inicializa la librería y sirve para establecer los parámetros de la red, pese a que soporta DHCP
- LocalIP: Método para obtener la dirección ip que posee el arduino
- Maintain: Permite la renovación de todos los parámetros que han sido negociados con DHCP

4.4.6.2. ESP8266

Esta librería ha sido diseñada por la comunidad por la necesidad de tener un soporte para el chip ESP8266. Permite subir sketches de forma similar a como se programa un arduino (funciones y librerías), sin necesidad de la intervención de terceros.

4.4.6.3. WifiSecureClient

Es otra librería desarrollada por la comunidad, esta nos proporciona la conectividad mediante wifi a nuestro arduino pero de una forma segura. Los métodos más importantes que posee son los siguientes:

Connect: para establecer conexión con el destino

SetCAcert, SetCAFile, SetPrivateKey: para añadir tanto el certificado de autoridad, como el certificado del cliente o la clave privada del cliente. Estos métodos respaldan el uso de ssl/tls.

4.4.6.4. AES-Library

Esta es otra librería desarrollada por la comunidad, la cual se utiliza para el algoritmo AES. Consta de los siguientes métodos:

- setKey: para almacenar la llave
- clean: borrar la llave después de usarla
- encrypt: método usado para cifrar
- decrypt: método usado para descifrar

Cabe destacar que posee los métodos tanto de cifrado como de descifrado con cbc, en el cual también hay que pasarle el vector de inicialización

4.4.6.5. AES-Master

Otra librería creada por la comunidad, la cual es una versión modificada de la original que es AVR-Crypto lib. Esta solamente soporta llaves de 128 bits y condiciona a que el vector de inicialización sea de 16 bits.

4.4.6.6. CryptoPP

CryptoPP es una librería de C++ la cual aporta una gran cantidad de algoritmos, entre los cuales se encuentra AES. Es una de las más utilizadas por la comunidad, pero necesita de una mayor capacidad de cómputo para poder utilizar esta librería, por lo que esta ha sido usada en el server en c que se programó con AES y por este mismo motivo, no se pudo utilizar en los arduinos respectivamente.

4.4.6.7. Base64

Es otra librería que ha sido creada por la comunidad, la cual nos proporciona los métodos necesarios para poder pasar la información a base64 y que sea legible para el humano cuando la sacamos por consola. El trasiego de información entre el servidor

y el arduino se hace en una codificación la cual el ser humano es incapaz de entender y para ello, interviene esta librería.

4.4.6.8. PubSubClient

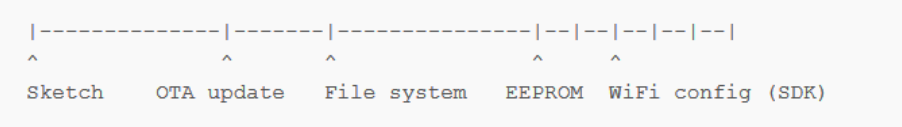
Es una librería que nos proporciona un cliente en arduino para MQTT, permitiendo mensajes de suscripción y de publicación. Posee funciones como:

- Connect: para conectarse al servidor
- Disconnect: para desconectarse del servidor
- Publish: para enviar un mensaje al bróker indicando el topic
- Suscribe: para suscribirse a un topic
- Unsuscribe: para finalizar la suscripción a ese topic
- SetServer: para introducir los detalles del server
- SetCallBack: para introducir los detalles del callback

4.4.7.SPIFFS

SPIFFS es un (File system object) es un sistema encargado de almacenar en el mismo chip del Arduino, todo aquello que nos sea necesario durante el funcionamiento del mismo (imágenes para una web, certificados, etc).

Esta gráfica, muestra cómo se configura el entorno del Arduino:



[8]

Y dependiendo de cada uno de los procesadores que se utilicen, este entorno se confeccionará de manera distinta y tendremos un tamaño de sistema de fichero distinto.

Los métodos que posee son los siguientes:

[8] Imagen obtenida de esp8266.github.io

Begin: se utiliza para inicializar el sistema de ficheros y devuelve un booleano dependiendo de si se ha realizado correctamente o no.

Format: formatea el sistema de ficheros. Devuelve un booleano dependiendo de si se realizó de forma satisfactoria o no.

Open: abre una ruta para poder cargar un fichero. Este método viene acompañado de su respectivo modo de acceso, que puede ser: "r", "w", "a", "r+", "w+", "a+".

Exists: devuelve verdadero o falso si existe el fichero para una ruta dada.

OpenDir: abre un directorio dada una ruta.

Remove: elimina un fichero a partir de la ruta dada.

Rename: cambia el nombre de un fichero a partir de la ruta dada.

Info: devuelve información sobre el sistema de ficheros que se está utilizando.

Cabe destacar, que todas las rutas que se deben de dar de forma absoluta y no de forma relativa.

4.4.8.IDE para raspberry: Geany

El ide elegido para desarrollar todo el software en la raspberry y con compilador en c++ ha sido geany. Se caracteriza por ser muy ligero, una interfaz super intuitiva, por tener resaltado de la sintaxis, autocompletado (entre otras muchas más características).

5. Pruebas realizadas

5.1. AES Arduino con raspberry

5.1.1.Descripción

Hemos desarrollado un prototipo de comunicación entre un arduino due y una raspberry con el objetivo de cifrar la comunicación entre ellos utilizando el algoritmo AES.

5.1.2.Conexión

Ambos dispositivos se han conectado a través del puerto serie del arduino y del usb de la raspberry.

Cabe destacar que este esquema de comunicación intenta reflejar que tanto el arduino como la raspberry (que contiene el server en c que hemos desarrollado) se conectan directamente por el puerto serie y asi es como se lleva a cabo la comunicación.

Como se muestra en el esquema de arriba, la conexión se hace entre el puerto de programación que posee el arduino due y el usb de la raspberry.

5.1.3.Librerías

Las librerías elegidas para llevar a cabo esta comunicación son: cryptoPP por parte de la raspberry y AES y Base64 por parte del arduino, las cuales están definidas en el apartado de herramientas del desarrollo de este mismo documento.

En el proceso de selección de estas librerías, a pesar de no existir estándares para la comunicación utilizando este algoritmo de cifrado, hemos elegido las que más se utilizaban en la comunidad y con las cuales se obtenían mejores resultados.

5.1.4.Pseudocodigo

Esta parte se va a dividir en dos, para reflejar el código utilizado en la raspberry y el código utilizado en el arduino.

En la raspberry el proceso para cifrar es el siguiente:

Librerías CryptoPP y SerialPort añadidas

Definimos parámetros para la comunicación (puerto de comunicación, vector de inicialización, llave)

Introducimos la llave y el vector de inicialización en las variables que posee la librería

Creamos una variable AESEncrypt utilizando la llave y el vector que definimos anteriormente

Se crea una variable cbcEncryption donde se refleja el modo de tratar los datos

Se cifra

Y el proceso para descifrar es el siguiente:

Librerías CryptoPP y SerialPort añadidas

Definimos parámetros para la comunicación (puerto de comunicación, vector de inicialización, llave)

Introducimos la llave y el vector de inicialización en las variables que posee la librería

Creamos una variable AESDecrypt utilizando la llave y el vector que definimos anteriormente

Se crea una variable cbcDecryption donde se refleja el modo de tratar los datos

Se descifra

En el arduino para cifrar con la librería AES.h:

Definimos el vector de inicialización y la llave a utilizar

Introducimos los elementos en las variables de la librería

Utilizamos la función para cifrar con cbc

En el arduino para cifrar con la librería AES.h:

Definimos el vector de inicialización y la llave a utilizar

Introducimos los elementos en las variables de la librería

Utilizamos la función para descifrar con cbc

Y para comunicar los datos con la raspberry, se pasa a base64 para mostrarlo por la pantalla de forma legible y para tener una idea de la secuencia de los datos.

5.1.5.Resultados

El resultado obtenido del estudio de esta tecnología ha sido el siguiente: hemos podido llevar a cabo la comunicación entre los arduinos y la raspberry pero hemos encontrado una serie de problemas, debido a que la aplicación de estos algoritmos no es nada sencillo. Nada más empezar la investigación, nos damos cuenta de que no existe un estándar establecido para arduino por lo que he tenido que utilizar dos librerías distintas para llevar a cabo la comunicación y además no ha sido tarea fácil completar satisfactoriamente la comunicación.

Debido a utilizar dos librerías distintas, cada una trata el bloque de cifrado de forma completamente distinta y eso es una parte importantísima del algoritmo, ya que una mínima variación en la forma de tratar internamente cada uno de los bloques,

puede provocar resultados extraños o el resultado correcto con otros valores redundantes (esto último ha sido lo que nos hemos encontrado al realizar la comunicación y analizar los resultados).

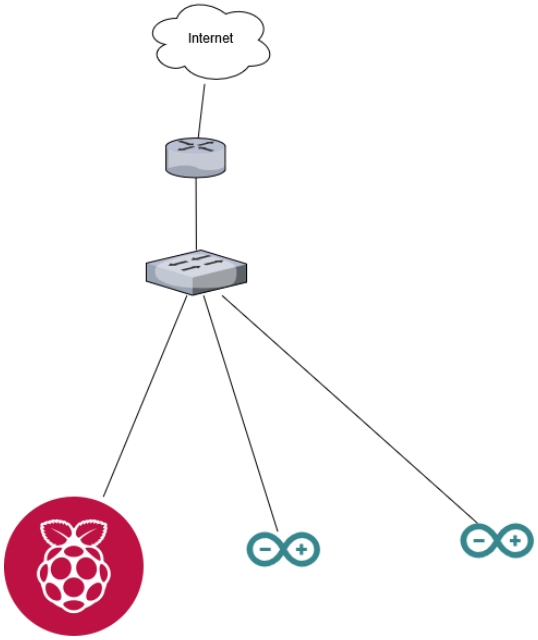
5.2. MQTT Arduino con raspberry utilizando WPA2

5.2.1. Descripción

He elaborado un prototipo de comunicación utilizando el protocolo MQTT. Para ello he utilizado una raspberry pi 3, un arduino mega y un arduino due. Al ser MQTT un protocolo de suscripción, he tenido que utilizar un bróker llamado mosquitto para poder gestionar los mensajes y las suscripciones de los dispositivos. La seguridad que hemos visto en este apartado, ha sido utilizar las tecnologías que nos aporta los routers: WPA2

5.2.2. Conexión

Los dispositivos se han interconectado a través de internet. Para habilitar este tipo de conexión en los arduinos, he tenido que incorporar unas shields ethernet, que se encuentran definidas en este mismo documento en el apartado de dispositivos utilizados para el desarrollo.



Cabe destacar que, en el esquema anterior, las conexiones establecidas a través de los puertos ethernet de los dispositivos irían conectados a un switch para poder comunicarlos en la misma red LAN, y todos estan conectados al mismo nivel aunque posteriormente, todos los arduinos vayan a communicarse con el broker situado en la raspberry.

5.2.3.Librerías

La librería utilizada para el desarrollo de este prototipo en los arduinos es PubSubClient, la cual está las cuales están definidas en el apartado de herramientas del desarrollo de este mismo documento.

Para la raspberry pi3, he utilizado el bróker mosquitto para gestionar todas las suscripciones de los dispositivos a la raspberry. Todos estos elementos han sido elegidos según las opiniones de la comunidad y los resultados obtenidos utilizando estas tecnologías.

Cabe destacar, que el bróker utilizado, es capaz de manejar QoS entre los distintos mensajes, lleva a cabo un protocolo a seguir en caso de que el dispositivo se haya caído y no le haya llegado la información del topic al cual se haya subscrito.

5.2.4. Pseudocodigo

Podemos distinguir tres partes fundamentales, en las cuales nos encontramos el setup, la función loop y la comunicación con la raspberry para poder enviar los mensajes al bróker que se encuentra situado en la raspberry.

En el setup, introducimos todas las credenciales de la red a la cual nos vamos a conectar vía ethernet para tener una IP estática. Una característica que posee arduino es que, si le especificas la mac por software, es capaz de emularla.

```
Serial.begin(57600);

client.setClient(ethClient);
client.setServer(server, 1883);
client.setCallback(callback);

Ethernet.begin(mac, ip);
// Allow the hardware to sort itself out

boolean answer =client.subscribe("outTopic");
//Serial.println(answer);
//client.publish("outTopic","hello world");

delay(1500);
```

Una vez especificado tanto la ip como la mac del dispositivo y la ip del servidor al con el cual nos vamos a comunicar mediante el protocolo MQTT pasamos a la función loop.

En la función loop únicamente comprobamos si el el cliente ha perdido la conexión con el bróker.

```
if (!client.connected()) {
  reconnect();
  delay(5000);
}
client.loop();
```

Y, por último, pero no menos importante nos encontramos la función en la cual llevamos a cabo la comunicación con el bróker. En ella, establecemos el topic del

mensaje para que pueda ser clasificado en el destino para tener un mayor control sobre los mensajes que le llegan.

```
while (!client.connected()) {
  Serial.print("Attempting MQTT connection...");
  // Attempt to connect
  if (client.connect("arduinoClient")) {
    Serial.println("connected");
    client.subscribe("outTopic",1);
    // Once connected, publish an announcement...
      client.publish("outTopic","mensaje arduino due");
      Serial.println("Mensaje enviado al broker");
  } else {
    Serial.print("failed, rc=");
    Serial.print(client.state());
    Serial.println(" try again in 5 seconds");
    // Wait 5 seconds before retrying
    delay(5000);
  }
}
```

5.2.5. Resultados

Los resultados de la puesta en marcha de este prototipo han sido los siguientes: se ha conseguido comunicar los dispositivos con el bróker. Introduciendo la SSID del Wi-Fi junto a la respectiva contraseña para conectarse, definiendo una ip para cada uno de los arduinos, se ha podido comunicar con el bróker, subscribirse y dar de baja la suscripción de cada uno de los temas que se crean para clasificar la información que llega de cada uno de los arduinos y observar cómo se almacenan los datos en la base de datos propia que posee el bróker y como maneja toda la información.

5.3. MQTT Arduino SSL con raspberry

5.3.1. Descripción

Basándonos en la prueba que he documentado anteriormente, decidí añadir un nivel más de seguridad, y a raíz de esta decisión me decanté por SSL. Elaboré unos certificados tanto para cliente como para servidor y un CA, que es una autoridad de certificación encargada de emitir y revocar cada uno de los certificados. Para la creación de todo esto, se ha utilizado openssl x509, además de trabajar con distintos formatos para los certificados como son PEM, DER, PCKS12

5.3.2. Conexión

La conexión de los dispositivos se ha llevado a cabo mediante internet. Para ello he utilizado una raspberry como servidor donde alojar el bróker que necesita MQTT para funcionar y dispositivos esp8266 con wifi, para poder comunicarme con ellos. Además de esto, he tenido que configurar todo el sistema de certificados y el servidor, ya que por defecto no viene establecido como opción predeterminada el uso de SSL/TLS en MQTT. El esquema es muy similar al anterior pero utilizando arduinos con WifiShield o bien con Wemos D1 que son los que nos aporta la funcionalidad de la conectividad Wi-Fi.

5.3.3. Librerías

Las librerías utilizadas son muy similares al ejemplo anterior de MQTT. Solamente cabe añadir como nueva, el uso de SPIFFS que es una herramienta que se ha utilizado para cargar los certificados del cliente al esp8266 y posteriormente se ha usado para poder verificar la identidad con el bróker de MQTT.

5.3.4. Pseudocodigo

Primero tenemos que establecer las credenciales de la red WIFI a la cual queremos conectarnos definiendo el ssid y la contraseña y el *fingerprint* perteneciente al server.

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <string.h>
#include <B64.h>
#include "FS.h"

#define TESTFILE "/client.key.der"

bool spiffsActive = false;
// Update these with values suitable for your network.

const char* ssid = "********";
const char* password = "**********";
const char* mqtt_server = "1*********";

//const char* mqtt_user = "esp8266";
//const char* mqtt_passwd = "wifi8266";

const char* fingerprint = "DC:78:9C:E9:04:07:E4:DC:AB:A3:63:97:6F:CF:E1:3D:76:72:D0:8D";
```

Después de definir esos parámetros, procedemos a la carga de certificados a través de SPIFFS para poder subirlo al wimos D1. Cabe destacar que es recomendable subir los archivos después de haber flasheado el código que queremos ejecutar en el dispositivo, ya que hay veces que se borran los archivos.

```
void load_tls(){
  //We have to use SPIFFS to integrate TLS with MQTT to read CA
  if(SPIFFS.begin()){
    Serial.println("SPIFFS Active");
    Serial.println();
    spiffsActive = true;
  }else{
    Serial.println("Unable to activate SPIFFS");
  }

  if(spiffsActive){
    if (SPIFFS.exists(TESTFILE)){
      Serial.println("Exist!!!!");
      ca = SPIFFS.open("/client.crt.der", "r");

        File key = SPIFFS.open(TESTFILE, "r");
      if(!key) {
        Serial.println("Couldn't open key");
        return;
      }else{
        Serial.println("Open key");
      }
```

Los métodos siguientes, cargan cada uno de los elementos que queremos subir a nuestro sistema, además de comprobar si se ha subido de forma correcta.

```
if(wifiClient.loadPrivateKey(key) == true) {
  Serial.println("Loaded Key");
} else {
  Serial.println("Didn't load key");
}

ca = SPIFFS.open("/client.crt.der", "r");
if(ca == true) {
  Serial.println("open cert");
}else{
  Serial.println("Couldn't open cert");
}

if(wifiClient.loadCertificate(ca) == true) {
  Serial.println("Loaded Cert");
} else {
  Serial.println("Didn't load cert");

}
//ca.close();

if(wifiClient.loadPrivateKey(key) == true) {
  Serial.println("Loaded Key");
} else {
  Serial.println("Didn't load Key");
}
```

Teniendo en cuenta lo mencionado anteriormente, realizamos la misma funcionalidad en el loop donde realizamos la comprobación de la conexión entre el cliente (wemos D1) y el bróker de MQTT situado en la raspberry.

```
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Create a random client ID
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);
    // Attempt to connect
    if (client.connect(clientId.c_str(),mqtt_user,mqtt_passwd)) {
      if(!wifiClient.verify("**********************************","**********")){
        Serial.println("connected");
        // Once connected, publish an announcement...
        client.publish("outTopic", "hello world");
        // ... and resubscribe
        client.subscribe("inTopic");
      } else {
        Serial.println("Not verified");
      }
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      // Wait 5 seconds before retrying
      delay(5000);
    }
  }
}
```

5.3.5. Resultados

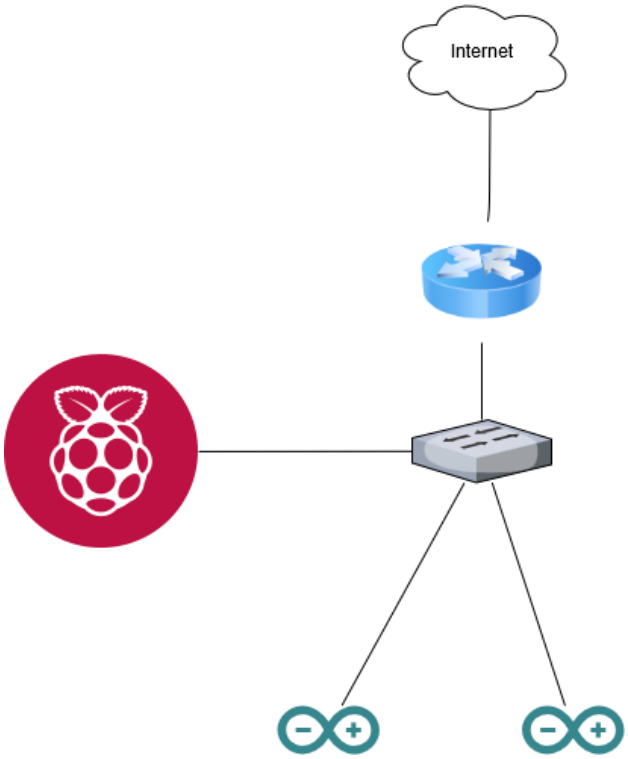
Los resultados obtenidos han sido los siguientes: no ha habido ningún problema con la creación de los certificados, ya que hay que hacerlo de una forma metódica y con un orden para que no den fallo. La puesta a punto del servidor y el correcto uso de los certificados ha sido la parte más problemática ya que no ha sido posible la integración completa del sistema, debido a que todos los fallos relacionados con SSL/TLS reportan el mismo tipo de fallo dando la menor información posible, para evitar informar al atacante de un sistema, además de la complejidad que presenta la integración de esta tecnología en la topología de una red.

5.4. Radius en Arduino con WPA2

5.4.1. Descripción

Se ha elaborado un prototipo para analizar el uso de radius en el internet de las cosas. Para ello se ha preparado en la raspberry un pequeño servidor que es capaz de reconocer y autorizar a los distintos dispositivos que se conectan a nuestra red. La seguridad que se ha añadido a esta prueba ha sido WPA2, que es la que permite que todas las comunicaciones vayan cifradas.

5.4.2. Conexión



En la imagen, se muestra el esquema de conexión de la topología de la red y identifica que la raspberry es el dispositivo encargado de alojar el sistema radius y los dispositivos arduinos seran los encargados de ponerse en contacto con el switch previamente antes de tener conexión a internet.

Se ha situado en un nivel horizontal la raspberry con el switch para esclarecer que cualquier dispositivo que quiera tener conexión con internet, debe primero autentificarse con el sistema radius que posee la raspberry. Una vez establecida esa autentificación, se procede a la conexión a internet y al acceso a los distintos servicios que posee la raspberry.

5.4.3.Librerias

Para esta prueba, hemos utilizado una librería desarrollada por la comunidad llamada Radius, la cual se ha definido anteriormente en el apartado de librerías utilizadas en este Proyecto.

Nos ha aportado todos los elementos necesarios para poder elaborar cada uno de los mensajes que se tienen que destinar al server radius para poder comunicarse con el.

5.4.4.Pseudocodigo

Para realizar esta prueba, primero hay que proceder a instalar freeradius en la raspberry. Después de la puesta a punto, añadiendo los usuarios y ejecutando freeradius en modo verbose para mostrar todos los pasos que se lleven a cabo en la comunicación con el servidor radius, se procede a elaborar un script, el cual se carga en el Arduino para que se pueda verificar en el servidor Radius y poder llevar a cabo su funcionalidad de forma completa, ya que de otro modo, le sería imposible entrar a formar parte de la red.

En el script, hacienda uso de la librería externa de Radius, utilizando las credenciales de la red a la que nos vamos a conectar (SSID y su respectiva contraseña), se elabora el mensaje de radius que irá directamente al servidor Radius, y se esperará una respuesta de ese servidor, para comprobar si finalmente tenemos acceso a el o no.

```
Serial.println("Creamos el mensaje");
RadiusMsg  msg(RadiusCodeAccessRequest);
msg.addAttr(RadiusAttrUserName, 0, "testing");
msg.addAttr(RadiusAttrUserPassword, 0, "password");
msg.addAttr(RadiusAttrNASPort, 0, 0x01020304);
msg.sign((uint8_t*)"testing123", 10);
```

Una vez elaborado este mensaje, se procede al envio al servidor y espera a la respuesta del mismo. Cabe mencionar que tiene un TTL (time to live), el cual es establecido por la librería y como su propio nombre indica, continua con la ejecución en caso de no recibir ninguna respuesta por parte del servidor.

```
if (msg.sendWaitReply(&client, server, 1812, &reply)
    && reply.checkAuthenticatorsWithOriginal((uint8_t*)"testing123", 10, &msg)
    && reply.code() == RadiusCodeAccessAccept)
{
```

5.4.5.Resultados

Una vez realizadas todas las pruebas con esta tecnología, se procede a analizar y comentar todos los resultados que hemos obtenido.

Las pruebas han sido satisfactorias, se ha podido instalar y montar el servidor radius en la raspberry (dispositivo encargado de alojar este servicio), se ha podido añadir el usuario y por último, se ha conseguido elaborar un script con el cual se ha realizado la comunicación entre el servidor y el Arduino (que es el dispositivo encargado de realizar la función de cliente).

El principal problema se ha encontrado en la comunicación entre los distintos dispositivos a la hora de establecer el trasiego de información entre los participantes, debido a que el servidor no era capaz de reconocer correctamente la petición realizada por el cliente, lo cual es un problema que presenta frecuentemente sistemas de una indole similar a este, que utilizan esta tecnología, ya que su puesta a punto es muy compleja, necesita conocimientos muy específicos de la herramienta con la cual se trabaja y se suele destinar un equipo de desarrollo exclusivamente al funcionamiento de este sistema.

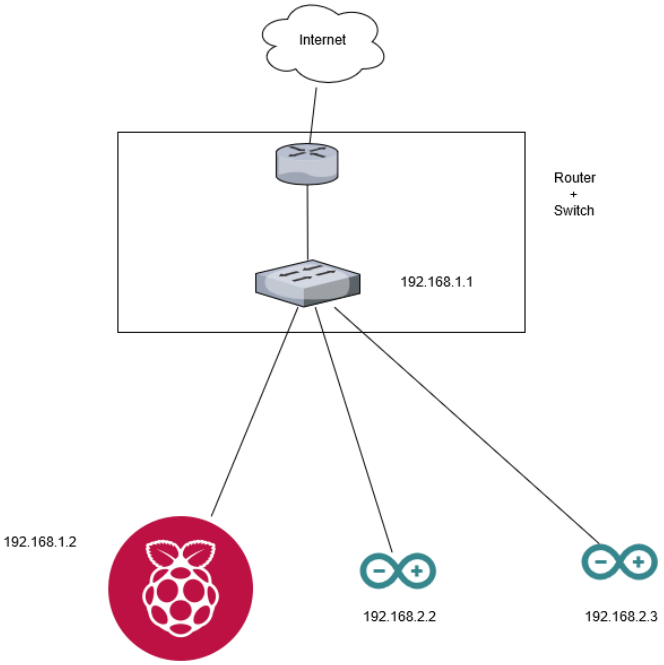
5.5. VLAN

5.5.1. Descripción

Se han estudiado otras técnicas ademas del uso de protocolos o cifrado, más relacionadas con la topologia de la red. Utilizando una red de área local virtual, conseguimos distribuir los distintos arduinos en redes diferentes, separadas lógicamente aunque se encuentren cerca fisicamente.

De esta forma, conseguimos que sea mas dificil comprometer un equipo de los que componen nuestra red y si ello ocurre, sea mas complejo aun compremeter las otras VLANs que forman parte de nuestra red.

5.5.2. Esquema de comunicación



En la anterior imagen, se puede apreciar como esta estructurada la topología de la red, como se distribuyen las VLAN y cuales son los dispostivos que pertenece a cada una de ellas.

5.5.3. Configuración

Para llevar a cabo la configuración de las distintas vlan, tenemos que tener un router que permita la creación de vlan. Cabe destacar, que la vlan puede poseer una etiqueta, la cual nos va a permitir identificar las distintas vlans que vamos a crear.

Primero tenemos que entrar al router, con la dirección ip que tenga, y crear las vlans que queramos. Será necesario ponerle un identificador, una descripción y marcar si queremos comunicación entre las distintas vlans.

Después, procedemos a configurar las vlans, desactivando el modo DHCP ya que supone una mayor cantidad de problemas superiores a los beneficios que nos puede aportar, y seguidamente definir la ip en la cual se va a situar esa vlan.

Una vez definido todo esto, queda conectar los dispositivos y configurar manualmente la ip, ya que hemos desactivado DHCP. Destacar que en arduino, la ip se simula por software, lo que facilita la asignación de la ip para cada dispositivo.

5.5.4.Análisis de resultados

Una vez configurado todo esto, creadas las distintas vlans y configuradas las diferentes ip en los dispositivos, se procede a analizar el trafico que pertenece a las diferentes vlans.

Utilizando una herramienta como wireshark, podemos sacar todas las ip que intervienen en la comunicación y seleccionando los puntos finales, se identifican claramente las 3 vlans que intervienen en la comunicación.

192.168.1.254	4	200	0	0	4	200
192.168.1.255	671	64 k	0	0	671	64 k
192.168.2.1	5.076	2241 k	1.887	2006 k	3.189	235 k
192.168.2.2	3.200	2134 k	1.313	127 k	1.887	2006 k
192.168.2.3	135	9689	135	9689	0	0
192.168.2.5	260	38 k	260	38 k	0	0
192.168.2.30	3.393	198 k	3.393	198 k	0	0
192.168.2.255	282	27 k	0	0	282	27 k
192.168.3.1	2.812	1949 k	1.714	1843 k	1.098	105 k
192.168.3.2	2.912	1957 k	1.198	113 k	1.714	1843 k
192.168.3.255	39	3804	0	0	39	3804

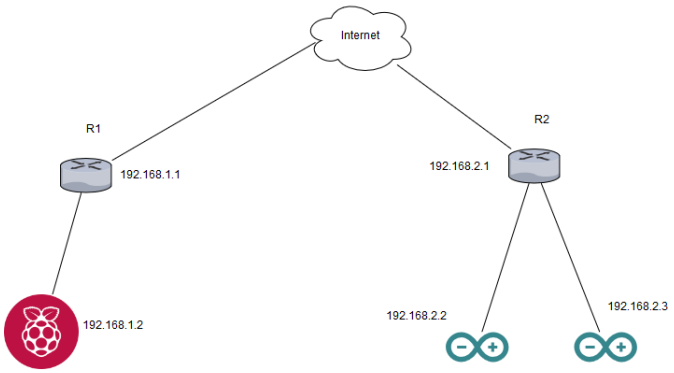
Cabe destacar, que en el análisis de esta prueba, he utilizado distintas ips para verificar que la vlan estaba creada correctamente.

5.6. VPN con IPSec en modo túnel

5.6.1.Descripción

Otra técnica que se ha estudiado para establecer seguridad es VPN (virtual private network). Esta técnica consiste en establecer una comunicación segura aprovechando todas las ventajas que nos proporciona las VPN añadiendo un protocolo seguro como es IPSec. Además, nos proporciona los elementos necesarios para poder complementar esta tecnología con las anteriormente estudiadas, y eso permite crear sistemas muchos mas robustos frente a los ataques a los cuales se pueden ver sometidas las diversas redes.

5.6.2.Esquema de comunicación



El anterior esquema de comunicación refleja como se distribuye la red, el la cual se muestra que dos redes se van a conectar mediante una VPN e utilizando IPSec en modo túnel.

5.6.3.Configuración

Lo primero que hay que configurar, es la creación de la vpn entre los equipos. Una vez decidido sobre que equipos se va a realizar, se procede a realizar la configuración tal y como se relata en el manual que nos aporta el fabricante.

Primero, nos conectamos de nuevo a nuestro equipo, cerciorandose previamente que posee soporte para VPN. Una vez hecho esto, tenemos que configurar la conexión estableciendo el rango de IPs, puerto sobre el cual se va a llevar, procedemos a hacer exactamente el mismo proceso en el otro equipo para poder establecer la comunicación. Cabe destacar que durante este proceso, también hay que configurar IKE polices, lo que nos proporcionará la clave que se tendra que establecer entre los dos hosts.

5.6.4.Análisis de resultados

El resultado obtenido al hacer el estudio de esta tecnología ha sido el siguiente: ha sido una prueba la cual requiere de un gran conocimiento de la topología de la red, saber el funcionamiento exacto de una VPN, el funcionamiento de IPSec en modo tunel (ya que nos interesa que tanto los datos como las cabeceras vayan cifradas para

no aportar ningún tipo de información, ni de cabeceras ni de datos, ante la posibilidad de la intrusión de un atacante en nuestra red) y de las distintas posibilidades que nos proporciona, pero finalmente una vez después de conseguirlo, ha sido satisfactorio. Nos ha permitido añadir una seguridad extra a nuestra red, añadiendo latencia entre los equipos, pero a su vez mucha mas seguridad.

La principal desventaja como se ha comentado anteriormente, es la complejidad que presenta montar este sistema, que se puede complicar aún más si decidimos utilizar certificados SSL/TLS. Otra desventaja que presenta es la ralentización de la comunicación que provoca, ya que este sistema va cifrado y se tiene que descifrar en el destino.

5.7. Problemas encontrados en la comunicación

En este apartado, se van a tratar lo problemas obtenidos desde un punto más generico. Los problemas que han sido encontrados en la puesta a punto de cada uno de los prototipos han sido bastantes tales como: una gran complejidad algoritmica, problemas añadidos por los equipos de comunicación, perdidas de datos provocadas por una mala configuración, entre otras muchas cosas.

Los problemas relacionados con la gran complejidad algoritmica estan directamente vinculados con la necesidad de tener un gran conocimiento algebraico para poder conocer todos los aspectos de cada una de las técnicas que se aplican, saber como se comporta el algoritmo y poder identificar de una forma relativamente sencilla, que parte del algoritmo esta fallando, hallar el punto en cuestión el cual no se esta ejecutando correctamente y proceder a su modificación para solventar ese problema.

Las adversidades relacionadas con los equipos de comunicación son problemas que van intrínsecos con las comunicaciones y los equipos que nos proporcionan las bases suficientes para llevarlas a cabo. Debido a las topologías de red que se utilizan hoy en dia, algunas llegan a tal nivel de complejidad que, su mantenimiento, es algo muy tedioso y complicado.

Pero pese a todo esto, las comunicaciones que han sido llevadas a cabo en este proyecto, han sido con un resultado satisfactorio. Todas ellas han funcionado bajo los márgenes esperados, pese a su complejidad y necesidad de un gran conocimiento matemático (como anteriormente mencioné). Cabe destacar la gran ausencia de estándares en los protocolos utilizados, bien por el deficit de conocimiento que tenemos acerca del internet de las cosas o bien por lo complejo que resulta adaptar todas las técnicas de seguridad que poseemos a unos dispostivos con unas capacidades de computo tan limitadas.

6. Planificación temporal

6.1. Planificación temporal

Se procede a añadir una planificación temporal del TFG.

Toda la información que se refleje aquí sobre la planificación es orientativa, ya que, en primera instancia, se desconocía la complejidad de la técnica a utilizar y su puesta en marcha.

En la siguiente tabla se pueden apreciar la duración de cada una de las pruebas que quedan recogidas en este TFG. Se procede a explicar el significado de cada columna:

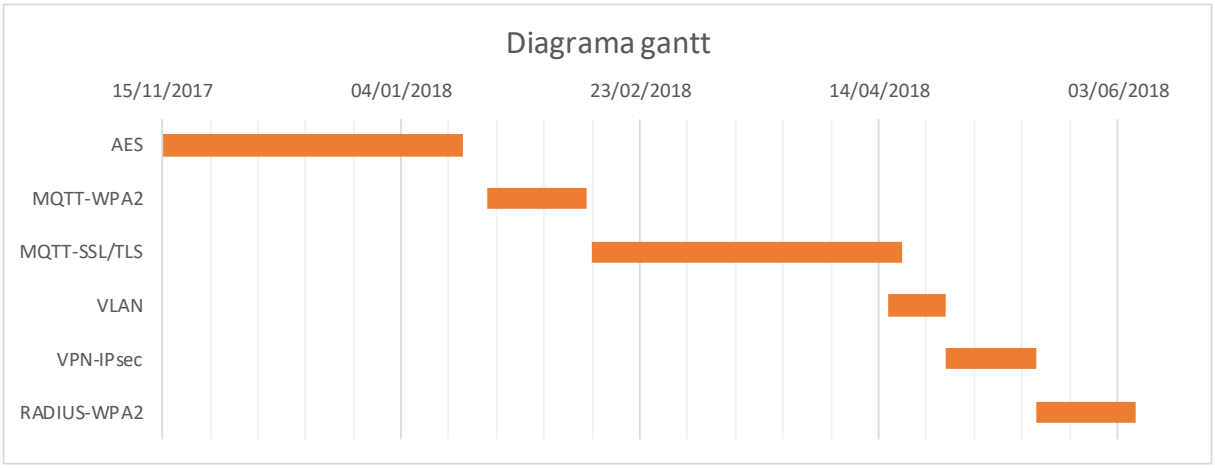
- Nombre: Nombre proporcionado a cada uno de los prototipos elaborados.
- Duración: Número total de dias que comprendió la realización de la tarea
- Comienzo: Fecha de comienzo de la elaboración del prototipo
- Final: Fecha de fin de actividad

Nombre	Duración	Comienzo	FInal

AES	Nueve semanas	Miércoles 15/11/2017	Lunes 22/1/2018
MQTT-WPA2	Tres semanas	Lunes 22/1/2018	Martes 13/2/2018
MQTT-SSL/TLS	Nueve semanas	Martes 13/2/2018	Viernes 16/4/2018
VLAN	Dos semana	Viernes 16/4/2018	Viernes 28/4/2018
VPN-IPsec	Tres semanas	Viernes 28/4/2018	Jueves 17/5/2018
RADIUS-WPA2	Cuatro semanas	Jueves 17/5/2018	Jueves 7/6/2018

6.2. Diagrama de gantt

Un diagrama de gantt es una herramienta utilizada para exponer de una forma más visual, la planificación de un proyecto en un largo periodo de tiempo. Este diagrama, permite realizar un control y ver el progreso de las distintas etapas del proyecto, el cual es perfecto para complementar la tabla de duración de cada una de las pruebas realizadas:



7. Conclusiones

Este tfg ha sido desarrollado para dar a conocer la gran importancia que esta adquiriendo el internet de las cosas, dar un nuevo punto de vista, avisar del gran déficit de seguridad que presenta y por último, pero no menos importante, mostrar algunas de las diversas tecnicas que podemos utilizar para minimizar esa vulnerabilidad que presenta.

Todas estas técnicas se caracterizan por la imperiosa necesidad de tener un gran conocimiento sobre seguridad, que a su vez, como ya he reflejado a lo largo de este tfg, también necesita de una gran base matemática y dependediendo de la topología de la red, a veces es necesaria una formación en redes para poder hacer frente a todo aquello que interviene en las redes de IoT.

A nivel empresarial, se están dando pasos enormes para avanzar en este ámbito, el cual también esta siendo impulsado por el desarrollo de nuevos dispostivos con mayor capacidad de computo, esto directamente provoca que se simplifique enormemente la adaptación de nuestros algoritmos a estos pequeños instrumentos.

Partiendo de toda esta base establecida, procedo a aportar una serie de conclusiones a las cuales he llegado después de un análisis exhaustivo y completo de todas y cada una de las técnicas utilizadas:

1. Los riesgos y problemas que provoca esta fuerte exposición surge debido a que en una primera instancia el internet de las cosas no estaba preparada para la gran expansión que ha sufrido

2. Tenemos el conocimiento y la metodologías necesarias para poder intervenir de lleno este problema y reducir su exposición, no erradicarlo ya que nunca se puede llegar a tener un sistema completamente seguro, debido a la definición de internet.

3. Hay que concienciarse de que esta tecnología no es cuestión de futuro, si no que está ya aquí y ahora y ha llegado para quedarse, por eso, cuanto antes atajemos este problema, nos permitirá avanzar más rápido

4. Las empresas tienen la obligación de dedicar grupos de investigación en estos ámbitos si no quieren que la mayor parte de sus datos se vean expuestos y provoquen graves perdidas.

7.1. Opinión sobre cada técnica utilizada

En este apartado, se procede a hablar más concretamente de cada una de las técnicas que han sido evaluadas durante el desarrollo de este TFG, desde un punto de vista más pormenorizado.

En primer lugar, cabe destacar que todas las técnicas que se han utilizado para este proyecto, aportan la suficiente versatilidad como para poder utilizar más de una a la vez para aportar seguridad, por ejemplo, estructurando la topología de la red con VLAN o VPN con IPSec en modo tunel y además utilizando MQTT con SSL/TLS para añadirle otra capa más de seguridad.

AES es un algoritmo de cifrado para el cual no existe un estándar para estos dispositivos, lo cual dificulta muchisimo su implementación. Esto provoca que, al tener que utilizar librerías distintas, se le de un trato diferente a los bloques incluso utilizando el mismo modo de tratar los bloques (en nuestro caso ha sido CBC) pero ello no asegura que internamente cada librería los trate de igual manera. Además, la imperiosa necesidad de una gran capacidad de cálculo para poder hacer uso del algoritmo dificulta mucho más su uso, pese al gran progreso que estan haciendo estos dispositivos en temas relacionados con memoria y procesadores.

MQTT es un protocolo muy ligero, pero tiene un objetivo muy claro, llevar a cabo una comunicación de suscripción. Si necesitas bajas latencias en comunicación, o un protocolo que te asegure comunicación M2M, este no es tu protocolo debido al motivo anteriormente mencionado. Cabe destacar, que añadiendo la variante de SSL/TLS aporta una mayor seguridad, pero a su vez descata por la enorme complejidad que posee su configuración, así como la creación de cada uno de los certificados, los cuales se deben de hacer de una forma muy sistemática para que funcionen correctamente.

Radius es un sistema de autorización muy versátil, aporta una gran seguridad, pero el mayor problema es la alta complejidad que posee su puesta a punto. Se necesita unos niveles de conocimiento de la herramienta muy elevados para que funcione perfectamente. Además, como ocurría con MQTT, también se puede utilizar SSL/TLS para añadir aún más seguridad, pero como mencioné anteriormente, es aún mas complejo, y si se quiere poner a punto para un ámbito empresarial, se tiene que destinar un equipo de desarrollo solamente para su puesta a punto y funcionamiento.

Y por último, pero no menos importante, la topología de red, donde se englobará tanto VPN con IPsec como VLAN. Estas diferentes formas de estructurar la red hacen posible delimitar la parte que puede quedar expuesta en caso de que un atacante se hiciera con el control de una de las redes de las cuales se componen nuestra estructura principal. Pero como ha sido mencionado con anterioridad, para llevar a cabo esto es necesario un conociento de comunicaciones y de red, ser capaz de trabajar con los distintos equipos de comunicación, saber como funcionan además de conocer todas y cada una de las posibilidades que nos ofrecen.

8. Líneas de futuro

Una vez elaborado este proyecto y tras estudiar todas las técnicas y topologías que han sido referenciadas anteriormente, se procede a mencionar cual será el avance de la seguridad en IoT, además de destacar algunos de los trabajos que se realizarán una vez concluido este.

Como ha sido mencionado previamente, por la definición y el objetivo inicial tanto de internet como del IoT, estas no estaban preparadas para la masificación que han sufrido ni tampoco se definieron desde un punto de vista de la seguridad. Ha sido un aspecto que no se ha contemplado en ningún momento.

Sin embargo, poco a poco, esto se está refinando a la vez que se está concienciando del gran problema que presenta tanto para una organización como para un ciudadano de a pie que se encuentra domotizando toda su casa utilizando algún dispositivo con inteligencia artificial (como por ejemplo alexa).

Además, está consiguiéndose avanzar en varios ámbitos los cuales, también muy importantes, como las capacidades de cómputo, diferentes y nuevas metodologías que se llevan a cabo para poder defenderse. También se han puesto en conocimiento las botnets: qué son, cómo identificarlas y defenderse frente a ellas, debido a ser uno de los principales problemas de seguridad que se presenta en IoT.

Como conclusión, cabe destacar que el IoT es el presente y, pese a estos matices comentados, se está preparando para incluirlo en nuestra vida cotidiana.

Como menciona el mayor principio de la informática: “Se debe utilizar toda tecnología posible para facilitar la vida de una persona”.

Bibliografía

Advanced Encryption Standard [sitio web].Acceso: Junio 2018 Disponible en : https://0-link.springer.com.avalos.ujaen.es/content/pdf/10.1007%2Fb137765.pdf

IoT Cisco – Internet of things (cisco) [sitio web]. Acceso: Junio 2018. Disponible en: https://www.cisco.com/c/es_es/solutions/internet-of-things/overview.html

Freeradius – Manual de usuario y conocimiento acerca de radius [sitio web]. Acceso: Junio 2018. Disponible en: https://freeradius.org/documentation/

MQTT-mosquitto- Definición de la libreria y ayuda para la configuración [sitio web]. Acceso: Junio 2018 Disponible en: https://mosquitto.org/man/mosquitto-8.html

MQTT- mosquitto con TLS/SSL: Creación de certificados openssl [sitio web]. Acceso: Junio 2018. Disponible en: https://mosquitto.org/man/mosquitto-tls-7.html

SPIFFS – repositorio de la herramienta que nos permite utilizar esa tecnología [sitio web]. Acceso: Junio 2018. Disponible en: https://github.com/esp8266/arduino-esp8266fs-plugin

WifiClientSecure - repositorio donde se aloja la librería con sus métodos [sitio web]. Acceso: Junio 2018. Disponible en: https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFiClientSecure

AES – repositorio donde se encuentra una implementación de AES para arduino [sitio web]. Acceso: Junio 2018. Disponible en: https://github.com/DavyLandman/AESLib

AES-library – otra librería con otra implementación distinta para AES [sitio web] Acceso: Junio 2018 .Disponible en: http://utter.chaos.org.uk/~markt/AES-library.zip

Arduino – sitio web para obtener tanto las especificaciones técnicas de las placas como para leer toda la documentación relacionada con las librerías que aportan soporte oficial [sitio web]. Acceso: Junio 2018. Disponible en: https://www.arduino.cc/

Foro arduino – sitio web donde habla la comunidad sobre los distintos problemas que le surgen en sus implementaciones [sitio web]. Acceso: Junio 2018. Disponible en: http://forum.arduino.cc/

Foro esp8266 – foro donde preguntar cualquier tipo de duda relacionada con el procesador esp8266 [sitio web]. Acceso: Junio 2018. Disponible en:

https://www.esp8266.com/

Base64 – sitio web que contiene toda la documentación relacionada con la librería [sitio web]. Acceso: Junio 2018 Disponible en:

https://github.com/adamvr/arduino-base64