Consideraciones de Seguridad en Soluciones de Internet de las Cosas con Microsoft Azure

Luis Garreta

Índice

Introducción

La plataforma Microsoft Azure IoT (o de forma resumida, Azure IoT) maneja la seguridad en una solución de Internet de las Cosas (IoT) desde diferentes perspectivas relacionadas directamente en la forma como se organizan los distintos componentes de la solución, es decir su arquitectura. Así, la seguridad se organiza teniendo en cuenta la seguridad de los dispositivos, la seguridad en la transmisión de datos, y la seguridad en la nube. De acuerdo a esto, Azure IoT ofrece tanto guías para el desarrollo seguro de soluciones IoT como mecanismos específicos que se implementan en los distintos componentes de la solución.

En este documento vamos a presentar una descripción de estas guías y mecanismos organizados desde tres perspectivas: la seguridad desde la arquitectura, la seguridad desde la implementación, y las recomendaciones de seguridad o mejores prácticas para los diferentes actores de la solución. Especialmente, vamos a centrarnos con más detalle en la parte de seguridad desde la implementación ya que se refiere a los mecanismos concretos que deberían usarse para implementar y ejecutar una solución IoT de forma más segura. Al final, vamos a mostrar en detalle los pasos para obtener un token de autorización que se puede usar para manejar recursos en Azure desde un entorno de programación, en nuestro caso vamos a mostrar ejemplos en PowerShell y Visual Studio C#.

Disponibilidad del Código:

Todo el código que se presenta en este documento lo puede descargar desde:
`https://github.com/lgarreta/AzureIoT-Security`

1. Seguridad desde la arquitectura

La seguridad desde la arquitectura [5] se refiere principalmente al diseño seguro del sistema desde su concepción y se enfoca en identificar y entender las posibles amenazas que pueden presentarse y las

posibles defensas que deben seguirse en los diferentes componentes que abarca un sistema IoT. Para esto Microsoft recomienda crear un modelo de riesgos que básicamente consta de cuatro pasos [4] :

- Modelar la aplicación
- Enumerar las amenazas
- Mitigar las amenazas
- Validar las mitigaciones

En el caso de IoT, lo que se busca es segmentar la solución IoT y enfocarse en las diferentes amenazas que pueden presentarse por zonas de acuerdo a la arquitectura del sistema. En el caso de la arquitectura IoT de Microsoft [6], las principales zonas con potenciales amenazas son cuatro (Figura 1):

- Zona 1 de dispositivos y orígenes de datos.
- Zona 2 de transporte de datos
- Zona 3 de dispositivo y procesamiento de eventos, y
- Zona 4 de presentación

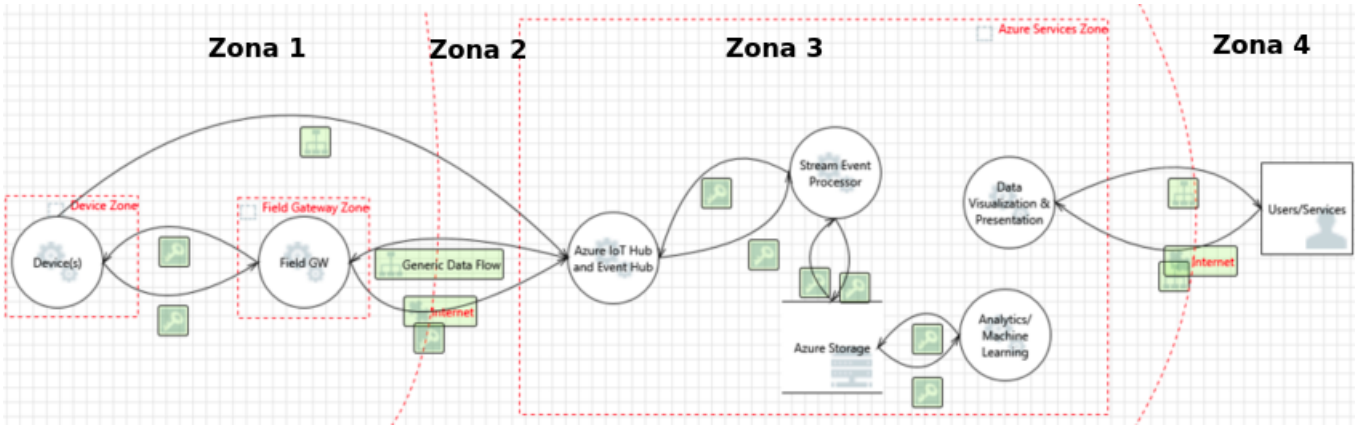

Figura 1: Zonas potenciales a ataques en la arquitectura IoT de Microsoft (Tomado de [5]).

Cada zona está demarcada por un límite de confianza (líneas de puntos roja) que representa una transición de datos o información de un punto origen a otro destino, con la posibilidad que durante las transiciones se presenten distintas amenazas a esos datos o información tales como: suplantación de identidad, manipulación, rechazo, revelación de información, denegación de servicio y elevación de privilegios.

2. Seguridad desde la Implementación

Una vez establecidas las zonas de riesgo dentro de un sistema IoT y sus potenciales ataques, se busca ahora mitigar o disminuir ese riesgo mediante mecanismos específicos que se establecen o implementan sobre los componentes del sistema. Así, Microsoft brinda mecanismos para las tres primeras zonas que son las más vulnerables y que componen las áreas de una solución completa IoT (Figura 2).

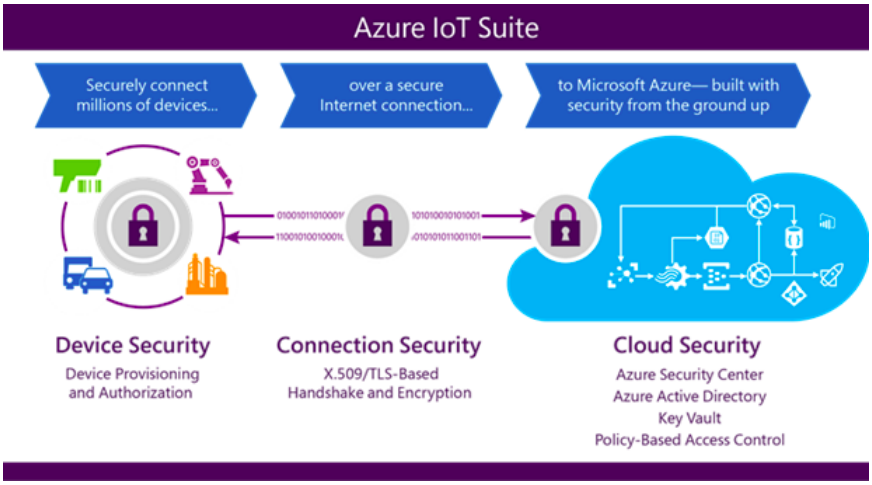

Figura 2: Áreas y mecanismos de seguridad dentro de una solución completa de IoT (Tomado de [3]).

Donde la zona 1 se conoce como seguridad de dispositivo; la zona 2 como seguridad de conexión; y la zona 3 como seguridad de Nube.

2.1. Seguridad de dispositivo

El enfoque aquí es la seguridad en el aprovisionamiento y la autenticación de dispositivos. Para esto Azure IoT ofrece dos medios:

- **Tokens de seguridad:** los dispositivos se conectan a Azure IoT a través de claves o tokens de seguridad utilizando un mecanismo de claves simétricas que evita que se envíen claves durante la conexión. Para que Azure IoT genere estos tokens, primero el usuario asigna a cada dispositivo un identificador que Azure IoT usa como base para generar una clave de identidad única con la cual se comunica con el dispositivo mientras este está operando. Para mayor seguridad, estos tokens se pueden limitar en cuanto al ámbito y el período de validez.
- **Certificado X.509 y una clave privada:** la comunicación entre el mundo externo (dispositivos o puertas de enlace) hacia Azure IoT usa un certificado X.509 (Protocolo de Seguridad de capa de transporte - TLS) y puede usar una clave privada para autenticarse de forma más segura con el IoT Hub de Azure.

Las conexiones entre los dispositivos y puertas de enlace hacia el IoT Hub se pueden realizar con cadenas de conexión simples, pero para más seguridad se debería utilizar tokens de seguridad. La conexión que se establece está basada en TLS y por lo tanto está encriptada y mejora la confiabilidad. El servidor es autenticado cuando este envía su certificado X.509 al dispositivo durante el *handshaking* del TLS. Además, hay que tener en cuenta que todas las conexiones las inician los dispositivos y no el IoT Hub, lo que quiere decir que estos solo se conectarán con quien quieran y al momento que ellos quieran y por lo tanto se evita todas las conexiones entrantes no solicitadas.

2.2. Seguridad de Conexión

La seguridad en la conexión entre dispositivos y nube se puede medir mediante tres parámetros: durabilidad de los mensajes, relacionada con la entrega confiable de mensajes y comandos; eficiencia de la comunicación, tanto en recursos como en entrega de mensajes; y escalabilidad, relacionada con la interoperación de forma segura con una gran variedad de dispositivos.

Para garantizar la durabilidad, Azure IoT tiene en cuenta la poca confiabilidad de la Internet y ofrece mecanismos para mejorarla a través de un sistema de comunicación mediante reconocimientos (*acknowledgments)* combinado con un sistema de almacenamiento o *cache* de los mensajes de hasta siete días para los mensajes y dos días para los comandos [3]. La eficiencia la garantiza Azure IoT a través del soporte de protocolos de comunicación seguros cómo HTTPS (eficiencia de recursos) y el par de protocolos MQTT y AMQPS (eficiencia en la entrega de mensajes). Y para la escalabilidad,

el Hub de Azure IoT garantiza la conexión segura tanto para dispositivos habilitados para IP como los no habilitados. Para los primeros, la conexión es directa con el HUB de IoT y por lo tanto es segura. Mientras que para los segundos, donde los dispositivos pueden estar restringidos de recursos y conectarse con protocolos de corta distancia (e.g. Zwave, ZigBee, o Bluetooth) se usa una puerta de enlace de campo *(field gateway)* que los agrega y hace la conversión del protocolo.

Adicionalmente, la conexión con el Hub de IoT está asegurada por el uso del estándar TLS y la autenticación mediante certificados X.509. También, los dispositivos y no el Hub de IoT son los que inician las conexiones y por lo tanto están protegidos de conexiones no solicitadas. Finalmente, el Hub de IoT tiene en cuenta el hecho que la comunicación con los dispositivos puede ser intermitente y esporádica y por tanto espera su conexión guardando los mensajes hasta dos días.

2.3. Seguridad en la Nube

La seguridad del IoT Hub se centra en el uso de permisos para limitar el acceso a sus diferentes puntos de conexión basado en la funcionalidad [1]. El administrador del IoT Hub puede conceder permisos ya sea a través de directivas de acceso compartido o a través de credenciales de seguridad de cada dispositivo.

En el primer caso, **políticas de acceso compartido a nivel de IoT Hub**, este define un conjunto de permisos para otorgar acceso a todos los puntos de conexión del IoT Hub. Estas políticas pueden conceder cualquier combinación de permisos y se pueden definir ya sea desde el portal de Azure IoT o mediante programación [**?**]. Estos permisos están relacionados con:

- **Política iothubowner:** todos los permisos
- **Política registryRead**: permiso RegistryRead, solo lectura del registro de identidad
- **Política registryReadWrite**: permisos RegistryRead y RegistryWrite, lectura/escritura del registro de identidad. Por ejemplo, esta política es. usada por el componente de administración de dispositivos.
- **Política device**: permiso DeviceConnect, usado por los dispositivos. Concede acceso de conexión del dispositivo hacia los puntos de conexión para comunicación de dispositivo hacia la nube. Se puede usar para otorgar permisos para enviar mensajes de dispositivo hacia la nube (D2C) y recibir mensajes de la nube hacia el dispositivo (C2D).
- **Política service**: permiso ServiceConnect, conexión de servicio hacia los puntos de conexión para comunicación desde la nube hacia los dispositivos. Se puede usar para otorgar permiso a los servicios que residen en la nube de Azure para recibir mensajes de dispositivo, enviar mensajes desde la nube hacia el dispositivo, y recuperar confirmaciones de entrega de los mensajes. Por ejemplo, esta política es usada por el componente de procesador de eventos.

En el segundo caso, **credenciales de seguridad de cada dispositivo,** cada IoT Hub contiene un registro de identidad y a cada uno de sus dispositivos se le puede configurar credenciales de seguridad que otorguen permisos DeviceConnect relacionados al dispositivo específico. También se pueden otorgar políticas de acceso compartido a nivel de hub (que ya están definidas por defecto).

Finalmente, la autenticación la brinda el IoT Hub que verifica primero el token enviado por el dispositivo contra las políticas de acceso compartido a nivel de Hub y después contra las credenciales de seguridad del dispositivo.

3. Mejores Prácticas de Seguridad

Una estrategia de seguridad completa en una solución de IoT debe abarcar toda la infraestructura y ser ejecutada por todos los actores que intervienen en la misma, desde los fabricantes de hardware hasta los operadores de la solución. Teniendo en cuenta esto, Microsoft recomienda seguir un conjunto de mejores prácticas por actor [2], las cuales se resumen a continuación.

Fabricantes o integradores de hardware:

- Incluir sólo las características necesarias para que el hardware pueda operar, es decir evitar características adicionales que puedan usarse para atacar al dispositivo (e.g. puertos adicionales por los cuales se podría realizar un ataque).
- Crear el hardware a prueba de saboteos de tal manera que el hardware registre o avise sobre eventos no usuales (e.g. abrir la caja donde está el dispositivo).
- Construir hardware integrado mecanismos de seguridad (e.g. almacenamiento encriptado o arranque seguro a través de protocolos estándar de seguridad).
- Realizar actualizaciones seguras (e.g. actualización del firmware del dispositivo usando medios encriptados).

Desarrolladores de soluciones:

- Seguir metodologías de desarrollo seguro que guíen tanto la selección de plataformas, lenguajes y herramientas, como el desarrollo seguro desde la concepción hasta el despliegue (e.g. la metodología de Microsoft *The Microsoft Security Development Lifecycle*).
- Escoger con cuidado el software de código abierto buscando que este software esté soportado por una comunidad visible y vigente donde aspectos críticos de seguridad estén detectados, contrario a software del que no se conoce bien de sus potenciales fallas de seguridad.
- Integrar software con cuidado buscando crear interfaces que contengan sólo los elementos necesarios para su operación y así ocultar funcionalidades provistas por diferentes componentes (e.g. APIs) que no son necesarias para la operación (e.g. funciones adicionales de escritura/lectura del dispositivo).

Encargados del despliegue de soluciones:

- Desplegar el hardware de forma segura: teniendo en cuenta el sitio físico donde se va a desplegar (e.g. espació público o sitio vigilado). Sea cual sea el sitio, se debe tratar de hacer el despliegue lo más seguro de alteraciones externas, por ejemplo, cerrando el sitio o encerrando el hardware en una caja donde solo queden expuestos las entradas y salidas necesarias.
- Mantener las claves de autenticación físicamente seguras solo accesibles a los directamente responsables de configurar los dispositivos.

Operadores de las soluciones:

- Mantener el sistema actualizado (e.g. versiones, drivers, parches de seguridad).
- Proteger contra actividades maliciosas (e.g. antivirus o antimalware, si el sistema lo permite).
- Auditar continuamente el sistema.
- Proteger físicamente la infraestructura IoT.
- Proteger las credenciales de conexión a la nube.

4. Autenticación en aplicaciones

Para el caso de las aplicaciones, que permiten automatizar la creación y uso de recursos de Azure de forma programada (e.g. PowerShell, C#, o node.js), cada una de estas debería tener sus propias credenciales de autenticación, las cuales se pueden crear ya sea a través de certificados o claves *(passwords)* [**?**]. Aquí vamos a mostrar la forma obtener un token de autorización a través de contraseñas en vez de certificados, lo cual es más apropiado cuando se va a iniciar sesión en Azure desde una aplicación. Después, usaremos el token para obtener las credenciales para crear de forma segura recursos en una aplicación en # bajo Visual Studio.

4.1. Creación de los IDs para Autenticación

El proceso de autenticación se realiza a través de la creación de dos objetos: el directorio activo o *Azure Active Directory (AAD)* y la entidad de aplicación o *service principal*. Después se asigna el role *(owner)* al *service principal*. Al final los valores que nos interesa guardar son los siguientes cuatro:

ApplicationId: Este es el ID para sus aplicación en su directorio activo.

Password: Este es la clave para su entidad de aplicación

TenantId: Este es el ID de su directorio activo.

SubscriptionId: El ID de su suscripción.

Estos pasos se realizan en Powershell y lo vamos a mostrar de dos maneras: la primera como un script de PowerShell que solo al final imprime los valores de los IDs de autenticación (Listado XXX) y después lo mostramos a manera de commandos, paso a paso, con sus respectivas salidas (los cuadros corresponden a las salidas resultantes):

Algoritmo 1 Script PowerShell Proceso de Autenticación.

```
$login = Login-AzureRmAccount
$tenantId = $login.Context.Tenant.TenantId
$suscriptionId = $login.Context.Tenant.TenantId
$nombreAplicacion = "progiothubapp"
$clave = "iothub2017A"

$app = New-AzureRmADApplication -DisplayName $nombreAplicacion -HomePage "http://$nombreAplicacion/
    home" -IdentifierUris "http://$nombreAplicacion" -Password $clave
$serv = New-AzureRmADServicePrincipal -ApplicationId $app.ApplicationId
$role = New-AzureRmRoleAssignment -RoleDefinitionName Owner -ServicePrincipalName $app.
    ApplicationId

echo "TenantId: $tenantId"
echo "SubscriptionId: $suscriptionId"
echo "ApplicationId: $app.ApplicationId"
echo "Password: $clave"
```

Algoritmo 2 Comandos PowerShell. paso a paso, proceso de autenticación con Azure Active Directoy.

```
// 1. Ingresar a su subscripcion con su usuario y clave y guardar el TenantId y el SubscriptionId:
Login-AzureRmAccount
```

```
Environment           : AzureCloud
Account               : luisgpujiot@outlook.com
TenantId              : 2707f6c8-6127-4a28-858a-1bad498d78b8
SubscriptionId        : 933a5bc9-18cb-4cd3-800a-2a2fdd692fb6
SubscriptionName      : Visual Studio Enterprise: BizSpark
CurrentStorageAccount :
```

```
// 2. Crear un nuevo Azure Active Directory y guardar el valor del ApplicationId
New-AzureRmADApplication -DisplayName iothubapp -HomePage http://iothubapp/home -IdentifierUris
    http://iothubapp -Password iothub2017A
```

```
DisplayName             : iothubapp
ObjectId                : b330059c-d3b3-4d32-bc03-b898922c23c5
IdentifierUris          : {http://iothubapp}
HomePage                : http://iothubapp/home
Type                    : Application
ApplicationId           : 4c1f4d33-e87b-4395-b758-dbc0d4afe25f
AvailableToOtherTenants : False
AppPermissions          :
ReplyUrls               : {}
```

```
// 3. Crear un nuevo service principal para la aplicación usando el ApplicationId
New-AzureRmADServicePrincipal -ApplicationId 4c1f4d33-e87b-4395-b758-dbc0d4afe25f
```

```
DisplayName  Type             ObjectId
-----------  ----             --------
iothubapp    ServicePrincipal 5e33ff06-7905-4925-997b-4cda7a895ac4
```

```
// 4. Asigne un rol al anterior service usando el ApplicationId
New-AzureRmRoleAssignment -RoleDefinitionName Owner -ServicePrincipalName 4c1f4d33-e87b-4395-b758-
    dbc0d4afe25f
```

```
RoleAssignmentId   : /subscriptions/933a5bc9-18cb-4cd3-800a-2a2fdd692fb6/providers/Microsoft.
    Authorization/roleAssignments/91899326-499e-4fd3-b209-b8a055efc892
Scope              : /subscriptions/933a5bc9-18cb-4cd3-800a-2a2fdd692fb6
DisplayName        : iothubapp
SignInName         :
RoleDefinitionName : Owner
RoleDefinitionId   : 8e3af657-a8ff-443c-a75c-2fe8c4bcb635
ObjectId           : 5e33ff06-7905-4925-997b-4cda7a895ac4
ObjectType         : ServicePrincipal
```

4.2. Aplicación C# bajo Visual Studio para crear un grupo de recursos

En el siguiente listado se presenta el método principal con el llamado de los métodos para obtener el token de autorización y después crear el grupo de recursos.

Algoritmo 3 Llamado de los métodos para manejo de recursos de Azure.

```
using System;
using Microsoft.Azure.Management.ResourceManager;
using Microsoft.Azure.Management.ResourceManager.Models;
using Microsoft.IdentityModel.Clients.ActiveDirectory;
using Microsoft.Rest;
using Microsoft.Azure;

namespace AzureIoTProg {
  class AzureIoTProgClass {
    //---------------------------------------------------------------------
    // Replace de corresponding values taken from the autentication process
    //---------------------------------------------------------------------
    static string tmpApplicationId = "4c1f4d33-e87b-4395-b758-dbc0d4afe25f";
    static string tmpSubscriptionId = "933a5bc9-18cb-4cd3-800a-2a2fdd692fb6";
    static string tmpTenantId = "2707f6c8-6127-4a28-858a-1bad498d78b8";
    static string tmpAppPassword = "iothub2017A";
    //---------------------------------------------------------------------
    // Temporal names for resources
    //---------------------------------------------------------------------
    static string tmpResourceGroupName = "progrg1";
    static string tmpResourceGroupLocation = "East US";
    static string tmpIoTHubName = "progiothub1";
    static string tmpDeployName = "progdeploy1";
    static string tmpStorageAccountName = "progstrgacc1";
    static string tmpStorageAddress = "https://progstrgacc1.blob.core.windows.net/";
    //---------------------------------------------------------------------
    // Main method: retrieve the access token and creates a resource group
    //---------------------------------------------------------------------
    static void Main(string[] args) {
      // Retrieve a token from Azure AD using the application id and password
      var accessToken = GetAuthorizationToken (tmpApplicationId, tmpSubscriptionId, tmpTenantId,
          tmpAppPassword);

      // Create, or obtain a reference to, the resource group you are using
      var rgResponse = CreateUpdateResourceGroup (accessToken, tmpSubscriptionId,
          tmpResourceGroupName, tmpResourceGroupLocation);
    }

    //---------------------------------------------------------------------
    // Method that returns the authorization token
    //---------------------------------------------------------------------
    private static string GetAuthorizationToken (string applicationId, string subscriptionId,
        string tenantId, string appPassword)  {
      Console.WriteLine("Getting authorization token credentials" + "...");
      var credential = new ClientCredential(applicationId, appPassword);
      var authContext = new AuthenticationContext(string.Format("https://login.windows.net/{0}",
          tenantId));
      AuthenticationResult token = authContext.AcquireTokenAsync("https://management.core.windows.
          net/", credential).Result;
      if (token == null) {
        throw new InvalidOperationException("Failed to obtain the token");
      }
      return token.AccessToken;
    }
    //-------------------------------------------------------------
    // Method that creates an Azure Resourse Group using an access token
    //-------------------------------------------------------------
    private static ResourceGroup CreateUpdateResourceGroup(string accessToken, string
        subscriptionId, string rgName, string rgLocalization) {
      Console.WriteLine("Creating/Updating resource group: " + rgName + "...");

      var tokenCredentials = new TokenCredentials(accessToken);
      var rmClient = new ResourceManagementClient(tokenCredentials) {SubscriptionId =
          subscriptionId};

      var rgResponse = rmClient.ResourceGroups.CreateOrUpdate(rgName, new ResourceGroup(
          rgLocalization));
      if (rgResponse.Properties.ProvisioningState != "Succeeded") {
        throw new InvalidOperationException("Failed to creating/updating resource group");
      }
      Console.WriteLine("Resource Group:\n" + rgResponse.ToString());
      return rgResponse;
    }
}
```

Recursos

[1] Microsoft Docs. Azure IoT Hub overview. URL: https://docs.microsoft.com/en-us/azure/

iot-hub/iot-hub-what-is-iot-hub.

[2] Microsoft Docs. Internet of Things security best practices. URL: https://docs.microsoft.com/en-us/azure/iot-suite/iot-security-best-practices.

[3] Microsoft Docs. Internet of Things security from the ground up. URL: https://docs.microsoft.com/en-us/azure/iot-suite/securing-iot-ground-up.

[4] Microsoft Docs. SDL Threat Modeling Tool 3. URL: https://www.microsoft.com/en-us/sdl/adopt/threatmodeling.aspx, 2011.

[5] Microsoft Docs. Internet of Things Security Architecture. URL: https://docs.microsoft.com/en-us/azure/iot-suite/iot-security-architecture, 2016.

[6] Microsoft Docs. Microsoft Azure IoT Reference Architecture Azure. (November):231, 2016.