**IoT Security Overview**

The Internet of Things is the wave of the future, offering businesses immediate and real-world opportunities to reduce costs, increase revenue, and transform their business. Many businesses, however, are hesitant to deploy IoT in their organizations due to concerns about security, privacy, and compliance. A major point of concern comes from the uniqueness of the IoT infrastructure, which merges the cyber and physical worlds together, compounding individual risks inherent in these two worlds. Security of IoT pertains to ensuring the integrity of code running on devices, providing device and user authentication, defining clear ownership of devices (as well as data generated by those devices), and being resilient to cyber and physical attacks.

Then, there’s the issue of privacy. Companies want transparency concerning data collection, as in what’s being collected and why, who can see it, who controls access, and so on. Finally, there are general safety issues of the equipment along with the people operating them, and issues of maintaining industry standards of compliance.

Given the security, privacy, transparency, and compliance concerns, choosing the right IoT solution provider remains a challenge. Stitching together individual pieces of IoT software and services provided by a variety of vendors introduces gaps in security, privacy, transparency, and compliance, which may be hard to detect, let alone fix. The choice of the right IoT software and service provider is based on finding providers that have extensive experience running services, which span across verticals and geographies, but are also able to scale in a secure and transparent fashion. Similarly, it helps for the selected provider to have decades of experience with developing secure software running on billions of machines worldwide, and have the ability to appreciate the threat landscape posed by this new world of the Internet of Things.

Protecting IoT solutions requires that businesses ensure each of the following:

* secure provisioning of devices
* secure connectivity between these devices and the cloud
* secure data protection in the cloud during processing and storage

Working against such functionality, however, are resource-constrained devices, geographic distribution of deployments, and a large number of devices within a solution.

https://docs.microsoft.com/en-us/azure/iot-fundamentals/iot-security-ground-up

https://www.microsoft.com/en-us/internet-of-things/security

https://docs.microsoft.com/en-us/azure/iot-fundamentals/iot-security-architecture

<https://docs.microsoft.com/en-us/azure/iot-fundamentals/iot-security-best-practices>

**IoT Threat Modeling**

In this topic, you'll learn:

* The definition of a threat model
* How threat modeling applies to IoT security
* How to develop your own threat model

**IoT Security**

As we've seen in previous lessons, planning for security in an IoT solution is not only essential but it can get complicated very quickly. IoT solutions involve data-collecting devices, cloud services (which includes storage and analytics), and can involve personal or sensitive data (a lot of it). Each of these can represent “soft targets” for hackers or others with malicious intent so understanding how a solution can be vulnerable should be an integral part of any IoT architecture.

Intel, in their IoT Platform Reference Architecture document [what they refer to](https://www.intel.com/content/www/us/en/internet-of-things/white-papers/iot-platform-reference-architecture-paper.html) as the “security layer” in their architecture. They describe it this way:

Robust hardware- and software-level protection are essential for ensuring world-class security, which is a foundational IoT tenet. Security is more like a process than a product because it depends upon evaluating the threat model for specific use cases and addressing each possible threat. A layered security approach is highly recommended since it establishes multiple defense mechanisms against hackers.

Let's look more specifically at how this layered approach to security works in an IoT architecture.

**Threat Modeling**

When designing a system, it is important to understand the potential threats to that system, and add appropriate defenses accordingly, as the system is designed and architected. The objective of threat modeling is to understand how an attacker might be able to compromise a system and then make sure appropriate responses and repairs are in place.

The movie *The Big Short* includes a quote (falsely attributed to Mark Twain) states, “It ain’t what you don’t know that gets you into trouble. It’s what you know for sure that just ain’t so.” In the context of IoT security, this means that we can get in the most trouble when we confidently assert that something won't (or will) happen. Part of threat modelling is planning for the things you can't plan for or anticipate. This might be as radical as taking a server offline or sending technicians to collect devices from the field if there's a breach. But it could include a lot of much less dramatic measures to ensure that data is safe and hackers are thwarted.

You should threat model the solution as a whole and also focus in the following areas:

* The security and privacy features
* The features whose failures are security relevant
* The features that touch a trust boundary

Three rules of thumb to keep in mind when building a threat model:

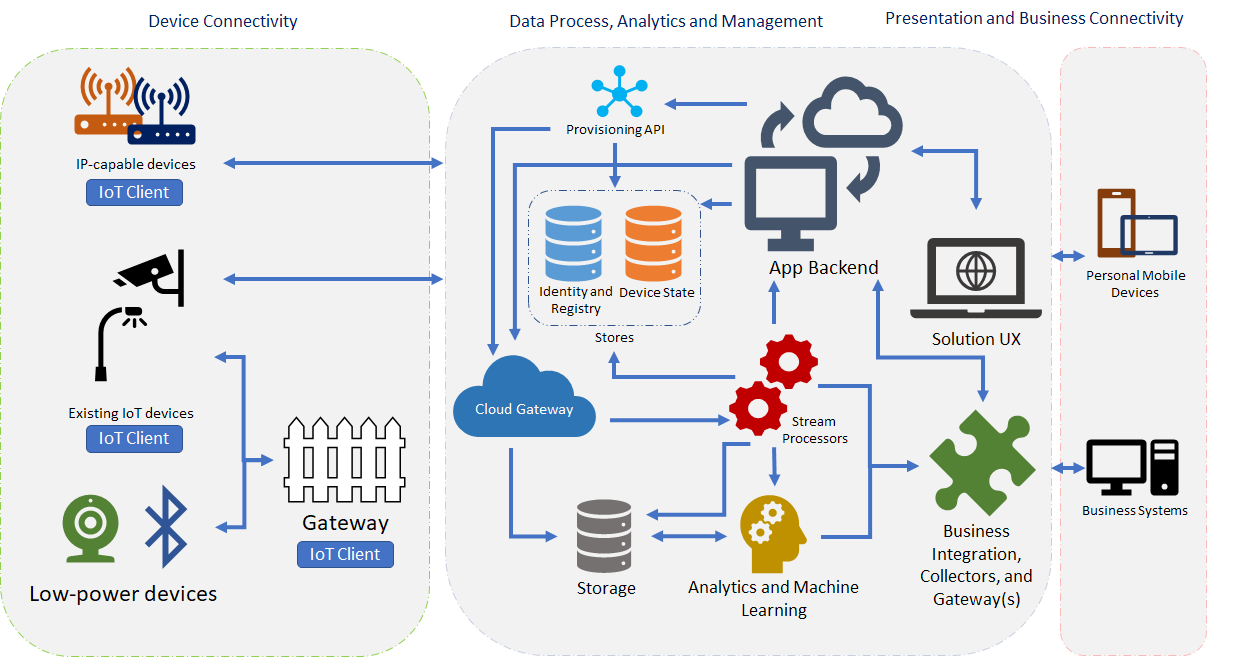
* Create a diagram out of reference architecture.
* Start breadth-first. Get an overview, and understand the system as a whole, before deep-diving. This approach helps ensure that you deep-dive in the right places.
* Drive the process, don’t let the process drive you. If you find an issue in the modeling phase and want to explore it, go for it! Don’t feel you need to follow these steps slavishly.

**Threat modeling and IoT architecture**

Microsoft's [guidance on threat modeling](https://docs.microsoft.com/en-us/azure/iot-fundamentals/iot-security-architecture) includes four main areas of focus. Each of these will have specific needs and involve particular “threat vectors” (ways the area can be attacked). They include:

* Devices and Data Sources
* Data Transport
* Device and Event Processing
* Presentation

The diagram below illustrates an IoT architecture with each of these areas designated. The blue arrows indicate paths the data can take through the system. While this looks complex, it's important in a threat model to understand where your data is coming from and *every possible place it can go*. Missing just one path can create a serious vulnerability.



**Attack Vectors to Consider**

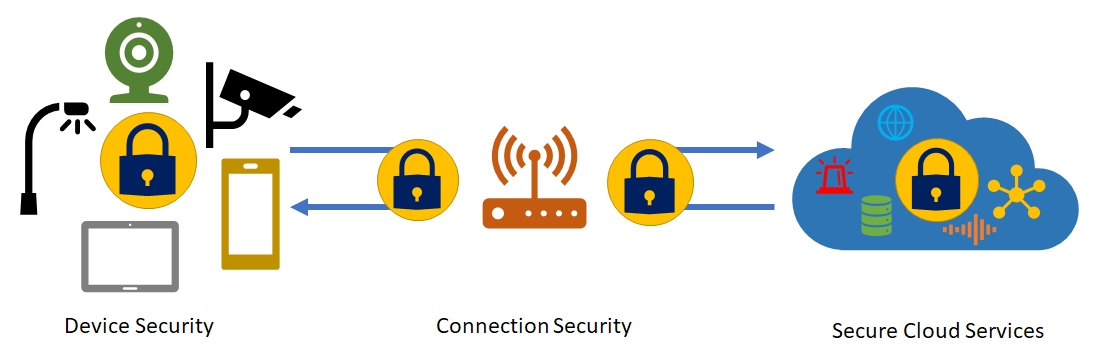
Again, an attack vector is simply a particular way that a hacker or person with malicious intent could compromise your IoT system. Microsoft documentation [lists many possible options](https://docs.microsoft.com/en-us/azure/iot-fundamentals/iot-security-architecture) and we'll summarize a few of the most important here.

1. **Spoofing**. Device spoofing takes place when an analogous device or virtual device takes the place of an intended device without the system knowing a switch was made. Spoofing can happen with services, APIs and other parts of an IoT system. Certificates can help reduce spoofing but an IoT architecture should have mechanisms in place to ensure that the devices and services deployed to the solution are the ones you intend to be there.
2. **Denial of Service**. Denial of Service (DOS) attacks are something we hear about in the news every so often because they can negatively affect our ability to reach a web site or service provider. One type of DOS attack involves overwhelming a service or device with garbage data or requests so the service or device can't operate normally. General a DOS attack is any hack where the device or service designed to perform a particular function is rendered useless. These types of attacks (besides being annoying) can prevent critical data from reaching a destination or enable hackers to attack other parts of an IoT installation.
3. **Elevation of Privilege**. This type of attack causes a device or service that has a set of capabilities that are limited by permissions or function (e.g. an automobile accelerator that has a mechanism that prevents the car from exceeding a certain speed) to function beyond their imposed limitations. You can imagine an API or device that has permission to collect or store impersonal data being tricked into collecting or storing personal data that could cause harm if it got into the wrong hands (e.g. credit card information).

As mentioned above, there are many other attack vectors to consider and a threat model should include mitigations for as many as possible.

**A Secure Ecosystem**

As you think about the security of an IoT solution, it can help to break down each aspect of the problem into functional categories. In the threat model we considered above, we saw four areas of focus. We can abstract these even more to help us build a threat model. In an [article for *Network World*](https://www.networkworld.com/article/3266375/internet-of-things/best-practices-for-iot-security.html), author Dean Hamilton echos the guidance that we'll be looking at in this module. He recommends that IoT architects focus on securing devices, the network, and data. We'll talk about security in three primary areas: **devices**, **connection and communication**, and **cloud services**. We'll call this our secure IoT Ecosystem.



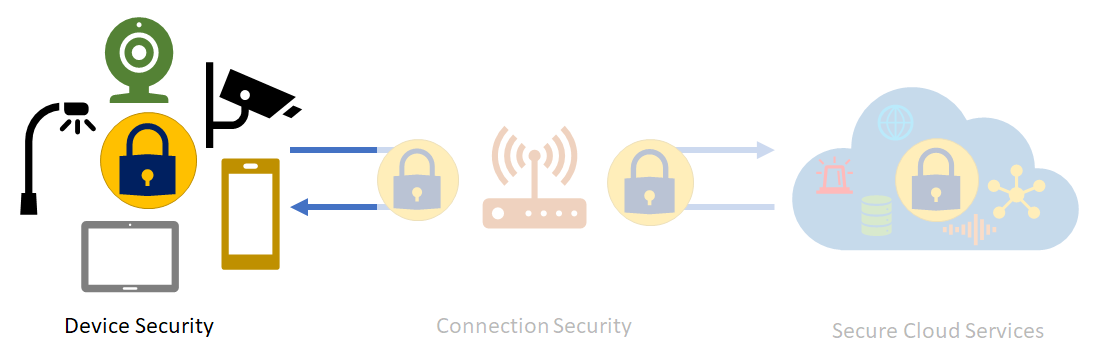
In the next three lessons, we'll look at each of these categories in turn.

**Device Security**

In this topic, you'll learn:

* How a secure device model fits into a secure IoT ecosystem
* An introduction on how to secure IoT devices
* How to develop your own device security model

Device security includes primarily device provisioning and authentication. But a threat model for devices should also include how data is stored and transmitted as well as how to protect devices from spoofing and DOS attacks.



**Basic Device Security**

Intel, in their IoT Platform Reference Architecture document [describes device security](https://www.intel.com/content/www/us/en/internet-of-things/white-papers/iot-platform-reference-architecture-paper.html) by claiming that a device is secure when an IoT solution can:

Protect device and user identities, ensure device integrity, and protect operational and personal data on every device. Each device should guarantee authentication without jeopardizing individual privacy and have the ability to automatically self-assess and resolve any situation.

That's a tall order but this level of device security should be considered essential for a secure IoT ecosystem. Similarly Microsoft recommends IoT architects focus on the physical security of devices and sensors.

**Physical tamper proofing and safety**

Sensors and devices can and must often be placed in public areas, where anyone may potentially have physical access to them. Also, tampering with the device is not just the act of manipulating the device hardware or software. A digitally trustworthy sensor may be tricked into reporting misleading data by dismounting and relocating it. Or an attacker could impact the environment around the device, creating misleading physical conditions in the immediate proximity of the device, pushing the overall system into an erroneous reaction. A lit lighter held near a smoke or temperature sensor might, for instance, trick a digital building control system into flooding a hotel hallway with the sprinkler system.

As the IoT space blurs digital and physical concerns, it also blurs security with safety. Suddenly, security threats become safety threats. If something “goes wrong” with automated or remote controllable devices—from physical defects to control logic defects to willful unauthorized intrusion and manipulation—production lots may be destroyed, buildings may be looted or burned down, and people may be injured or die. That is a different class of damage than someone maxing out a stolen credit card limit. The security bar for commands that make things move, and also for sensor data that eventually results in commands that cause things to move, must be higher than in any e-commerce or banking scenario.

Some exemplary measures that can be taken to improve the security of the physical device are:

* Choosing microcontrollers/microprocessors or auxiliary hardware that provide secure storage and use of cryptographic key material, such as trusted platform module (TPM)63 integration.
* Secure boot loader and secure software loading, anchored in the TPM.
* Using sensors to detect intrusion attempts and attempts to manipulate the device environment with alerting and potentially “digital self-destruction” of the device.[^1]

[^1]: For more information on how the Azure IoT framework can help keep devices secure, see the [Azure IoT Reference Architecture](https://aka.ms/iotrefarchitecture).

**Device Security and Azure IoT Hub**

You can secure devices out in the field by providing a unique identity key for each device, which can be used by the IoT infrastructure to communicate with the device while it is in operation. With Azure's IoT framework, the process is quick and easy to set up. The generated key with a user-selected device ID forms the basis of a token used in all communication between the device and the Azure IoT Hub.

Device IDs can be associated with a device during manufacturing (that is, flashed in a hardware trust module) or can use an existing fixed identity as a proxy (for example CPU serial numbers). Since changing this identifying information in the device is not simple, it is important to introduce logical device IDs in case the underlying device hardware changes but the logical device remains the same. In some cases, the association of a device identity can happen at device deployment time (for example, an authenticated field engineer physically configures a new device while communicating with the solution backend). The Azure IoT Hub identity registry provides secure storage of device identities and security keys for a solution. Individual or groups of device identities can be added to an allow list, or a block list, enabling complete control over device access.

Azure IoT Hub access control policies in the cloud enable activation and disabling any device identity, providing a way to disassociate a device from an IoT deployment when required. This association and disassociation of devices is based on each device identity.

To support these features, the Azure IoT framework supports the following features through the device identity store:

* **Device identity authority**. The device identity store is the authority for all device identity information. It also stores and allows for validation of cryptographic secrets for the purposes of device client authentication. The identity store typically does not provide any indexing or search facility beyond direct lookup by the device identifier; that functional role is taken on by another store that keeps the application specific domain model. These stores are primarily separated for security reasons; lookups on devices should not allow disclosing cryptographic material.
* **Provisioning**. Device provisioning uses the identity store to create identities for new devices in the scope of the system or to remove devices from the system. Devices can also be enabled or disabled. When they are disabled, they cannot connect to the system, but all access rules, keys, and metadata stay in place. A solution’s provisioning workflow takes care of processing individual and bulk requests for registering new devices and updating or removing existing devices. It will also handle the activation, and potentially the temporary access suspension and eventual access resumption.The provisioning workflow ensures, that the device is registered with all backend systems that need to know about its identity and additional metadata attributes as needed.[^2]

[^2]: See the Azure reference architecture document for more details on this.

These features help ensure that device identity is securely managed and that onboarding or suspending a device from the solution can be managed centrally which can help ensure that only secure devices are included in the ecosystem when a threat has been detected.

Device security is a logical first step as you consider how to secure your IoT solution. Using a centrally managed framework like the Azure IoT hub can help make the complicated task of device deployment and management simpler.

**Connection Security**

In this topic, you'll learn:

* A security philosophy for communication as it applies to IoT architectures
* How to use tools to properly secure communication networks
* How Azure's IoT framework can assist in developing secure communication

Durability of messaging is an important feature of any IoT solution. The need to durably deliver commands and/or receive data from devices is underlined by the fact that IoT devices are connected over the Internet, or other similar networks that can be unreliable.



Referring again to the [Intel IoT Reference Architecture](https://www.intel.com/content/www/us/en/internet-of-things/white-papers/iot-platform-reference-architecture-paper.html), the document describes connection security (their term is “network level” security), as an IoT solution's ability to:

Ensure secure application, traffic, and data security in transit through every type of wired and wireless network connection.

Microsoft similarly recommends IoT solutions focus on secure communication as a top architectural priority. As a foundational principle, all cloud communication with devices or [field gateways](https://stackoverflow.com/questions/27749149/azure-event-hub-what-is-a-field-gateway) must occur through secure channels when the devices talk directly to endpoints.

The Microsoft reference architecture adopts the following principles of Clement Vasters's [Service Assisted Communication](https://blogs.msdn.microsoft.com/clemensv/2014/02/09/service-assisted-communication-for-connected-devices/) model:

* Devices do not accept unsolicited network connections. All connections and routes are established in an outbound-only fashion.
* Devices generally only connect to or establish routes to well-known service gateways that they are peered with. In case they need to feed information to or receive commands from a multitude of services, devices are peered with a gateway that takes care of routing information downstream and ensures that commands are only accepted from authorized parties before routing them to the device.
* The communication path between device and service or device and gateway is secured at the transport and application protocol layers, mutually authenticating the device to the service or gateway and vice versa. Device applications do not trust the link-layer network.
* System-level authorization and authentication should be based on per-device identities, and access credentials and permissions should be near-instantly revocable in case of device abuse.
* Bidirectional communication for devices that are connected sporadically due to power or connectivity concerns may be facilitated through holding commands and notifications to the devices until they connect to pick those up.
* Application payload data may be separately secured for protected transit through gateways to a particular service.

**Trustworthy and secure communication**

Information received from and sent to a device must be trustworthy, if anything depends on that information. Trustworthy communication means that information is of verifiable origin, correct, unaltered, timely, and cannot be abused by unauthorized parties in any fashion.

Even telemetry from a simple sensor that reports a room’s temperature every five minutes should not be left unsecured. If any control system reacts to this input, or draws any other conclusions from it, the device and the communication paths from and to it must be trustworthy.

Unless a device can support the following key cryptographic capabilities, its use should be constrained to local networks and all inter-network communication should be facilitated through a field gateway:

* Data encryption with a provably secure, publicly analyzed, and broadly implemented symmetric-key encryption algorithm, such as AES with at least 128-bit key length.
* Digital signature with a provably secure, publicly analyzed, and broadly implemented symmetric-key signature algorithm, such as SHA-2 with at least 128-bit key length.
* Support for either TLS 1.2 for TCP or other stream-based communication paths or DTLS 1.2 for datagram-based communication paths.
* Updateable key-store and per-device keys. Each device must have unique key material or tokens that identify it toward the system. The devices should be able to store the key securely on the device (for example, using a secure key-store). The device should be able to update the keys or tokens periodically, or reactively in emergency situations in case of system breach. Key update might occur over the air or through some other means, but updateability is required.
* The firmware and application software on the device must allow for updates to enable the repair of discovered security vulnerabilities.

**Legacy Devices**. If (legacy) devices must use insecure or nonstandard and proprietary communication paths into the cloud system, they should be connected through a separately hosted custom protocol gateway or a local field gateway.[^1]

[^1]: For more information on how the Azure IoT framework can enable secure communication, see the [Azure IoT Reference Architecture](https://aka.ms/iotrefarchitecture).

**Connection Security and the Azure Iot Hub**

We saw in Lesson 1 of this module that understand how data is flowing through your solution is an important first step in creating a robust threat model and ensuring data flow is secure.

Azure IoT Hub offers durability of messaging between cloud and devices through a system of acknowledgments in response to messages. Additional durability for messaging is achieved by caching messages in the IoT Hub for up to seven days for telemetry and two days for commands.

Efficiency is important to ensure conservation of resources and operation in a resource-constrained environment. HTTPS (HTTP Secure), the industry-standard secure version of the popular http protocol, is supported by Azure IoT Hub, enabling efficient communication. Advanced Message Queuing Protocol (AMQP) and Message Queuing Telemetry Transport (MQTT), supported by Azure IoT Hub, are designed not only for efficiency in terms of resource use but also reliable message delivery.

Scalability requires the ability to securely interoperate with a wide range of devices. Azure IoT hub enables secure connection to both IP-enabled and non-IP-enabled devices. IP-enabled devices are able to directly connect and communicate with the IoT Hub over a secure connection. Non-IP-enabled devices are resource-constrained and connect only over short distance communication protocols, such as Zwave, ZigBee, and Bluetooth. A field gateway is used to aggregate these devices and performs protocol translation to enable secure bi-directional communication with the cloud.

Additional connection security features include:

* The communication path between devices and Azure IoT Hub, or between gateways and Azure IoT Hub, is secured using industry-standard Transport Layer Security (TLS) with Azure IoT Hub authenticated using X.509 protocol.
* In order to protect devices from unsolicited inbound connections, Azure IoT Hub does not open any connection to the device. The device initiates all connections.
* Azure IoT Hub durably stores messages for devices and waits for the device to connect. These commands are stored for two days, enabling devices connecting sporadically, due to power or connectivity concerns, to receive these commands. Azure IoT Hub maintains a per-device queue for each device.

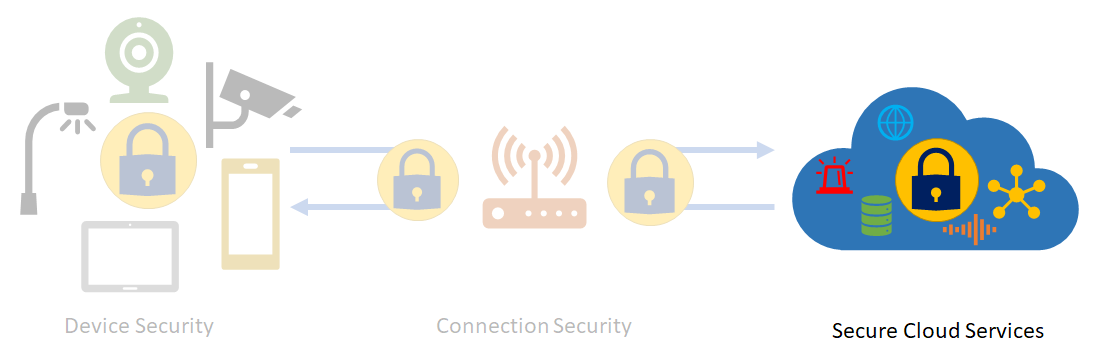
https://docs.microsoft.com/en-us/azure/iot-fundamentals/iot-security-ground-up

**Cloud Security**

In this topic, you'll learn:

* How to think about cloud security in an IoT solution
* About Azure IoT cloud-based components that can be used in developing an IoT solution

Cloud security involves secure storage and processing of information as well as device management.



Once again, Intel in their [IoT Reference Architecture](https://www.intel.com/content/www/us/en/internet-of-things/white-papers/iot-platform-reference-architecture-paper.html) includes this component in their model. The “cloud level” of an IoT is secure according to Intel when it can:

Deliver the necessary trust for data centers and multi-tenant public cloud environments to unleash powerful IoT services and analytics while protecting data and ensuring privacy.

The previous two topics address some of the cloud services that can be used to secure an IoT system. In what follows, we'll look at other cloud services that can be used to secure an IoT architecture.

**Azure and IoT Cloud Security**

**Azure Active Directory**

Using Azure Active Directory (AAD) for user authentication and authorization provides a policy-based authorization model for data in the cloud, enabling easy access management that can be audited and reviewed. This model also enables near-instant revocation of access to data in the cloud, and of devices connected to the Azure IoT solution accelerators.

Once data is in the cloud, it can be processed and stored in any user-defined workflow. Access to each part of the data is controlled with Azure Active Directory, depending on the storage service used.

All keys used by the IoT infrastructure are stored in the cloud in secure storage, with the ability to roll over in case keys need to be re-provisioned. Data can be stored in Azure Cosmos DB or in SQL databases, enabling definition of the level of security desired. Additionally, Azure provides a way to monitor and audit all access to your data to alert you of any intrusion or unauthorized access.

**Virtual Private Network (VPN) technology**

Virtual private network (VPN) technology allows for integrating and isolating a network, creating a single address space functionally equivalent to a local network, while in reality spanning multiple underlying networks. It provides mechanisms to securely join and participate in an isolated network but does not secure the traffic inside the network. Without further components like per-endpoint firewalls, it intentionally does not limit how the participants of the virtual network can communicate with each other. In scenarios where devices participating in a VPN are in physical control of users or potentially unknown intruders, the virtual network environment must be considered as hostile as the Internet environment.

**The Cloud Gateway**

The Cloud Gateway (e.g. Azure IoT Hub service) provides a cloud hub for secure connectivity, telemetry and event ingestion and device management (including command and control) capabilities. The IoT Hub offers built-in secure connectivity, telemetry and event ingestion, and bi-directional communication with devices including device management with command and control capabilities. In addition, the IoT Hub offers an entity store that can be used to store device metadata.

Cloud gateway is a system that enables remote communication from and to devices or field gateways from several different sites across public network space, typically towards a cloud-based control and data analysis system, a federation of such systems. In some cases, a cloud gateway may immediately facilitate access to special-purpose devices from terminals such as tablets or phones. In the context discussed here, “cloud” is meant to refer to a dedicated data processing system that is not bound to the same site as the attached devices or field gateways. Also in a Cloud Zone, operational measures prevent targeted physical access and are not necessarily exposed to a “public cloud” infrastructure.

A cloud gateway may potentially be mapped into a network virtualization overlay to insulate the cloud gateway and all of its attached devices or field gateways from any other network traffic. The cloud gateway itself is not a device control system or a processing or storage facility for device data; those facilities interface with the cloud gateway. The cloud gateway zone includes the cloud gateway itself along with all field gateways and devices directly or indirectly attached to it. The edge of the zone is a distinct surface area where all external parties communicate through.

**Services**

In modern IoT solutions, most services used for an IoT solution are run from the cloud though some services can be run from a field gateway[^1] or even on devices itself.

A “service” is defined for this context as any software component or module that is interfacing with devices through a for data collection and analysis, as well as for command and control. Services are mediators. They act under their identity towards gateways and other subsystems, store and analyze data, autonomously issue commands to devices based on data insights or schedules and expose information and control capabilities to authorized end users.

As we've been discussing in this module, any service used in an IoT solution should be fully secure and risks should be considered in any threat model. Modern cloud services like Azure are built with security in mind but architects still need to evaluate how data flows through any service. Data is the responsibility of the data owner and not of the service itself.

[^1]: Field gateway is a device/appliance or some general-purpose server computer software that acts as communication enabler and, potentially, as a device control system and device data processing hub.

**Security Best Practices**

Securing an Internet of Things (IoT) infrastructure requires a rigorous security-in-depth strategy. This strategy requires you to secure data in the cloud, protect data integrity while in transit over the public internet, and securely provision devices. Each layer builds greater security assurance in the overall infrastructure.

This security-in-depth strategy can be developed and executed with active participation of various players involved with the manufacturing, development, and deployment of IoT devices and infrastructure. Following is a high-level description of these players.

* **IoT hardware manufacturer/integrator**: Typically, these players are the manufacturers of IoT hardware being deployed, integrators assembling hardware from various manufacturers, or suppliers providing hardware for an IoT deployment manufactured or integrated by other suppliers.
* **IoT solution developer**: The development of an IoT solution is typically done by a solution developer. This developer may be part of an in-house team or a system integrator (SI) specializing in this activity. The IoT solution developer can develop various components of the IoT solution from scratch, integrate various off-the-shelf or open-source components, or adopt solution accelerators with minor adaptation.
* **IoT solution deployer**: After an IoT solution is developed, it needs to be deployed in the field. This process involves deployment of hardware, interconnection of devices, and deployment of solutions in hardware devices or the cloud.
* **IoT solution operator**: After the IoT solution is deployed, it requires long-term operations, monitoring, upgrades, and maintenance. These tasks can be done by an in-house team that comprises information technology specialists, hardware operations and maintenance teams, and domain specialists who monitor the correct behavior of overall IoT infrastructure.

See Microsoft's [document on IoT best practices](https://docs.microsoft.com/en-us/azure/iot-fundamentals/iot-security-best-practices) for more detail on each of these.

**Areas of Focus**

In this module, we've categorized security considerations into three main areas: devices, communication, and cloud services. The IEEE focuses their “[best practices](https://internetinitiative.ieee.org/images/files/resources/white_papers/internet_of_things_feb2017.pdf)” guidance on two areas: securing devices and securing networks. As a standards body, this makes sense since they wouldn't focus on specific technologies but on high-level categories. Much of the guidance here amplifies what we've already seen in this module but we'll briefly cover some of the salient points here for easy reference and to provide a wrap up summary for the module.

**Securing Devices**

1. **Make hardware tamper resistant**

This guidance focuses on the ensuring the physical device is secure. Since devices exist in the public domain and can be vulnerable to manipulation, protecting the device is critical for ensuring a safe IoT deployment. Specifically the IEEE recommends (as examples):

* + Use tamper-resistant packaging when transporting devices (to ensure the device is not manipulated during transit)
  + Using small plastic containers with a lock to keep ports and apertures from casual manipulation
  + Using port locks (a small device that keeps the physical network port on a device from being manipulated)
  + Setting the device so it automatically is disabled if certain types of tampering occurs
  + Using strong boot-level passwords on the device itself or requiring the device to boot from local storage
  + Close unused TCP/UDP and serial ports
  + Disabling open password prompts

1. **Provide for firmware updates/patches**

As the title implies, IoT architects must ensure that devices are easily patchable and updatable when vulnerabilities are found. This becomes more challenging when you consider that hardware vendors may have little incentive (legal and financial) to keep devices up to date. It is up to the IoT architect to ensure that solid relationships exist with device manufacturers and agreements (preferably written) are in place to keep devices updated with the latest firmware and that vulnerabilities are patched as soon as possible after one is discovered.

1. **Perform** [**dynamic testing**](https://en.wikipedia.org/wiki/Dynamic_testing)
2. **Specify procedures to protect data on device disposal**

This is an item that may be an after-thought to IoT architects but the reality is that devices do fail and most have an end-of-life date when it becomes unfeasible to update, repair, or maintain the device. Ensuring that the device contains no on-board data, passwords, or other information that could compromise security when the device is discarded is an important part of an overall security plan.

**Securing Networks**

1. **Use strong authentication**

While it may seem obvious, ensuring network resources don't use simple passwords, duplicate credentials, or include “[back doors](https://en.wikipedia.org/wiki/Backdoor_(computing))” are fundamentals of a secure system. According to the IEEE,

Each device should have a unique default username/password, perhaps printed on its casing, and preferably resettable by the user. Passwords should be sophisticated enough to resist educated guessing and so-called brute force methods.

Using multi-factor authentication where possible is also highly recommended. Cloud IoT platforms like Azure help IoT engineers manage a lot of this more easily as we've seen in previous topics. For more information on multi-factor authentication and how it works, see the following:

* + [Azure MFA documentation](https://docs.microsoft.com/en-us/azure/active-directory/authentication/concept-mfa-howitworks)
  + [wikipedia artical](https://en.wikipedia.org/wiki/Multi-factor_authentication)

1. **Use strong encryption and secure protocols**

There is no substitute for using encrypted communication over secure protocols when building an IoT solution. As with device patching, the ability of a device or network to use encryption and communicate over a particular protocol may be a function of the device or network itself so architects have to ensure, up front, that the components they're working with will support the type of security they desire. Committing to a particular device family or cloud solution and learning after commitments have been made that the component doesn't meet the required security level can have monetary and scheduling consequences.

1. **Minimize device bandwidth**

IEEE's guidance on this area focuses on reducing the attack surface area for [denial-of-service attacks](https://en.wikipedia.org/wiki/Denial-of-service_attack) (a subject we touched on in the topic on threat modeling). The document includes an intimidating future scenario: if 1.1% of 50 billion IoT devices became compromised in a coordinated attack, they could generate 55 petabytes of data per second overwhelming the current fastest network interface by a factor of 183,333 to 1. To secure devices against these types of attacks, hard vendors should include limiters to throttle network transmissions to the task at hand. They conclude,

Additional kernel-level controls within devices that notice and attenuate large amounts of uploaded traffic or stop other unexpected behavior could further reduce the destructive capabilities of compromised devices without requiring heroic efforts by network defenders. Thus, we recommend serious consideration of the performance requirements of each device and that modest limitations be emplaced that are difficult to circumvent. This will greatly increase the safety of IoT devices and make it possible to safely field many more of them in the future.

1. **Divide networks into segments**

We've seen this guidance repeated often in other material. By using network “zones”, architects can provide more localized firewalls, security gates, and secure interfaces to ensure attacks can be stopped before compromising an entire system. While segmentation can make management more complex, it can reduce the attack surface area by restricting access to other segments of the system if a single segment becomes compromised.The Azure IoT framework provides tools for working with network segments making management easier for engineers.

**Conclusion**

These are just some recommendations for securing an IoT solution. Based on threat modeling, architects should design a security plan that fits the needs of their solution. Security should be a primary consideration when developing an IoT solution. Thankfully, IoT services offer like those offered through Azure are build with security as a “first-class citizen” making building secure systems easier. Even so, security is always the responsibility of the engineers that build the system so making it an integral part of the design from day one can help mitigate problems later in development and deployment