IoT Security: What Engineering Leaders Need to Know

Learn how to build IoT solutions with security best practices in mind.

Very

Introduction

Security implementation tends to lag behind technology adoption, and few technologies have seen growth as massive as the Internet of Things (IoT).

Despite the rapid expansion of the market for connected devices, security has all too often been an afterthought in the push to be first to market, creating an unprecedented opportunity for hackers worldwide. Bad actors have infiltrated everything from internet-connected gas pumps to smart thermometers and medical devices. Some of these, like the medical devices, may store or communicate sensitive data that holds personally identifiable information.

These unique threats require a new approach to security based on best practices for securing each part of an IoT product: the device or “thing” itself, the device firmware, the web applications that interact with the device, and the cloud infrastructure that supports the product. However, security initiatives always come with trade-offs, and it can be difficult for organizations to decide where and when those trade-offs need to take place.

In this white paper, we’ll explore these aspects of IoT security, dig into the most cost-effective ways to protect your products, and establish some ground rules to guide your next IoT project.

© Very 2020

CHAPTER 1

IoT Device Security

Device Authentication & Authorization

Device authentication is a key aspect of IoT security that can easily lead to a breach if approached incorrectly. If a malicious person can convince your web backend that they are a valid device, they can start to do anything that the impersonated device has been authorized to do. That could mean access to private data like security camera footage, or they could overload your web infrastructure from a seemingly valid device, causing disruptions for other customers.

By using proper device authentication, you can ensure that a trusted set of devices are used in your IoT solution. There are various security mechanisms that can be used to establish that trust, but when security is paramount, we recommend using client-side certificate authentication.

Server-Side vs Client-Side Certificate Authentication

Before we dive into the benefits of client-side certificate authentication, it's important to have a basic understanding of what certificates are used for. Certificates are commonly associated with SSL (Secure Sockets Layer), which is an industry-standard security protocol for establishing an encrypted connection between entities. It has since been superseded by TLS (Transport Layer Security), but their names are often used interchangeably. They both use certificates to establish an encrypted connection that guarantees that all data passed between the two machines or devices stays private and secure.

Millions of websites protect their customers' private information by using TLS to secure communication between a web client and a web server.

© Very 2020

These websites serve traffic over HTTPS instead of HTTP, where the 'S' stands for 'secure', indicating HTTP over SSL/TLS. This is most commonly an example of server-side SSL, where only the client wants to verify the identity of a web server before proceeding.

In client-side SSL, sometimes referred to as mutual or two-way authentication, both the client and server authenticate each other. Passing around X.509 certificates may seem impractical for individual people, but in the case of IoT development, the client is often an automatically provisioned device.

Server-side and client-side SSL both use public key infrastructure (PKI) for authentication. However, there is one significant difference between the two. Unlike server certificates, client certificates aren't used for encrypting data in transit – they're installed for validation purposes only.

Benefits of Client-Side SSL for IoT Device Authentication

The use of client-side certificates isn't new, but isn't exactly widespread, especially in comparison to server-side certificates used for websites. However, it's quickly emerging as the key identity and authentication mechanism for IoT scenarios and machine-to-machine (M2M) communications. Client certificates are more secure than other authentication mechanisms available because they're based on public and private keys, where the private keys aren't ever shared.

Client-side certificates are superior to other authentication methods because the device owns the secret instead of the server. That means the only way a person could impersonate the device would be to get the private key data from it. To do that would require a security flaw exposing the filesystem, or extracting the key from the physical device, both of which should be quite difficult. And since device certificates can be individually revoked, client-side certificates also make it easier to contain a breach if it happens.

Using client-side certificates is the preferred way to authenticate with AWS IoT, Microsoft Azure, and Google Cloud. It's also the only way to authenticate with NervesHub, an extensible web service that allows you to manage over-the-air (OTA) firmware updates of devices in the field. So, why don't all IoT projects use it for device authentication? The answer is pretty simple – it's more difficult to do, and it can be expensive.

© Very 2020

Let Your Use Case Be Your Guide

It's important to consider the implications of a breach and let that guide the amount of time and money you invest in authenticating your Internet-connected devices. If you are using simple beacons to do inventory tracking, the consequences of a spoofed device may be small. In situations like this, we recommend using a simpler type of device authentication such as token-based authentication.

However, some connected devices incorporate risk that could endanger lives or jeopardize the wellbeing of your company. In these cases, IoT device authentication is of the ultimate importance and client-side SSL should be used.

Physically Securing IoT Devices

Device authentication is crucial for communication security – but what about securing your actual, physical IoT device? When physically exposed, IoT devices can be stolen and hacked to obtain sensitive information or exploit vulnerabilities.

Consumer IoT devices, including smart home products, often operate within environments that typically have limited security controls in place to protect against attackers. On top of all this, it's difficult for consumers and enterprises to know whether their connected devices are secured against physical attacks because there are no well-known security certification programs for IoT devices.

Wireless security systems are additionally risky because they rely on the security of a network that is accessible from well outside of a building. A wireless network could potentially be exploited by hackers who are nearby, without even needing to enter a building.

Assume That Hackers Will Steal Your IoT Device

What best practices should be in place for IoT device security?
When beginning any security project, it's standard practice to assume that an attacker is going to have full knowledge and access to your code. In IoT, likewise, you should assume that the hacker will have access to your connected device.

© Very 2020

Especially out in the field, it's difficult to control whether or not someone will be able to get their hands on a device, extract the code, decompile it, and figure out what's going on. For this reason, it's important that when developers are actually writing code that they're not taking shortcuts. Developers shouldn't include secrets inside of the code, like secret tokens. If you're storing private information, such as WiFi credentials, it's important that information is encrypted, rather than just stored in plain text.

Install Tamper-Proof Hardware

In addition to securing your code, you'll want to install physical hardware to protect your device and sensitive data.

Cryptoprocessors are microprocessors that you can include in your IoT products to outsource your encryption for you – handling your tokens, certificates, etc. The main processor can communicate with the cryptoprocessor to store and retrieve secrets and perform hardware-accelerated cryptographic operations. Typically, cryptoprocessors are designed to be tamper-proof. If a hostile party tries to open the device, the cryptoprocessor may be designed to stop working or self-destruct.

Think About What's Being Stored on the Device

What about especially vulnerable devices that store a lot of sensitive data? Before you develop an expensive security protocol for a device, ask yourself: why am I leaving something containing such critical information in a place where people can easily access it? Is this absolutely necessary for my product to achieve its core function?

For example, think about a personal smartphone or computer. These objects probably contain a lot of personal data, so most people wouldn't leave them out on their porch if they lived right next to a busy pedestrian street. Similarly, if it can be avoided, you shouldn't make a habit of leaving other connected devices in a location easily accessible to actors with unknown intentions.

The best method for keeping sensitive data secure is often to keep it far, far away from any potential risks. If your device must store or interact with sensitive data, make sure it's in a place where it's always monitored closely. If the device communicates via network connections, take the necessary steps to secure the network before releasing the device into the wild.

CHAPTER 2

Why Does Firmware Matter for IoT Security?

Another critical aspect of IoT security is protecting your firmware – the code running on your hardware that is critical to the hardware's operation. Hackers have targeted platform firmware as a place to embed malware and hide other malicious code that can ultimately compromise a system. As such, it's hard to overemphasize the importance of considering security when developing firmware for Internet-connected devices.
We recently completed work on an Internet-connected project where we applied security best practices to the development of a smart fish tank.

The device has a pretty limited feature set:

- It will broadcast an ad hoc WiFi network.
- Once a user has connected to the device's ad hoc WiFi network, it will accept WiFi credentials from a mobile app and connect to an existing, Internet-enabled network.
- It will register itself with a remote server.
- It can send state updates to the remote server.
- It can receive payloads from the remote server.

During the development of this project, we discovered a few tips that you can use when developing firmware for Internet-connected devices.

Reduce Your Attack Surface

The smaller the attack surface, the easier you can sleep at night. Even though our device needed to broadcast a WiFi network and run an HTTP server (for accepting credentials to the Internet-enabled network), we did not need to do those things after the user had configured the device. By carefully and intentionally disabling the ad hoc WiFi network and HTTP server, we ensured that neither of those features could be used for nefarious purposes during the regular usage of the device.

In addition, after the initial configuration step was complete, the mobile app never communicated directly with the device again. All communications

© Very 2020

were limited to a single WebSocket channel that was negotiated between the device and the remote server. The mobile app sent all commands and payloads through the remote server, and the server forwarded that communication to the device. This way, we didn't have to support multiple communication paths on the device, and we could focus all of our concerns on the single existing channel.

We also reduced the total amount of information stored on the device to ensure that if the device fell victim to an attack, the attacker would be very limited in the amount of information they could extract from the device. We architected the application so that no user data was stored locally. The only data that we did store locally were the WiFi credentials so that the device could automatically reconnect to WiFi after being rebooted, and a token that would allow the device to send data to the remote server. And of course, all of these were encrypted.

No Plain Text

This point cannot be overstated. Do not store sensitive information in plain text. Anywhere. Ever. Even though we store three potentially sensitive data points from the user on our devices, we make sure that those data points are encrypted before they go into the EEPROM.

As mentioned previously, we did not store any user information on the device. However, we did allow the device to receive user information from a mobile app during device configuration. The device then forwarded that data on to the remote server and did not retain a local copy. We did this to ensure that the device, when it was registered with the remote server, was coming from a valid user. However, this data was not sent in plain text either – it was an encrypted JSON Web Token (JWT) that was encrypted before it was sent to the device and only decrypted after it had been passed to the remote server. This way we could ensure that sensitive user data was never exposed on the device itself.

This also applies to communications between devices. When possible, always communicate over secure channels: WSS or HTTPS for web traffic and MQTT over TLS rather than simple TCP communication. Depending on hardware design restrictions, it can be difficult to implement secure communication protocols. In this case, good secure design is just as important as secure implementation. Insecure channels are not inherently

© Very 2020

bad (and can be a plus from a cost perspective) as long as the data being sent is not sensitive, and you are prepared for the implications of potential bad actors consuming that data.

Test for Potential Vulnerabilities

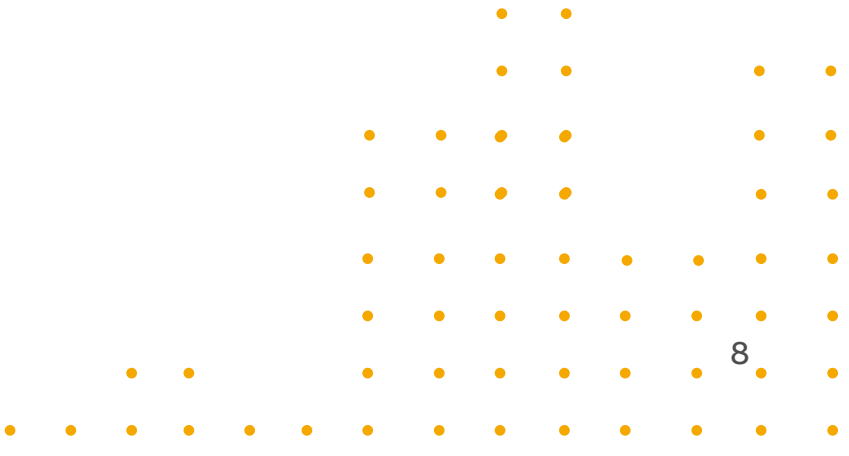
Another way to avoid vulnerabilities is to test for them. When an area of code may be problematic, such as accepting user input, it is possible to write tests that can check for obvious vulnerabilities and ensure that your code will not regress to a point where those vulnerabilities can be exploited in that code.

For instance, while we didn't care about sanitizing input that would never make its way to a user interface, we did care about how that data was stored. If we suspect that a particular part of our String handling may be vulnerable to a buffer overflow, we can write a test that will allow us to check how that code works with specific String length or content.

However, we obviously cannot reasonably expect to be able to foresee all potential vulnerabilities, which is why it is important to...

Allow for Updates

No matter how well-thought-out a security practice may seem, there can be flaws. No system is perfect. It is important to be able to mitigate attacks before they can spread. Over-the-air updates are a great solution for addressing bugs or vulnerabilities that are discovered after a product has shipped. Arduino supports these OTA updates natively and they can be configured to allow the user to approve updates or to be applied automatically.



CHAPTER 3

Protecting the Application Layer for IoT Security

IoT security depends upon protecting data at multiple levels, from the IoT device itself to the low-level network protocols, to the application layer where humans are interfacing with the system. Zeroing in on the application layer, what are some best practices for protecting valuable data?

What Is Application Layer Security?

To understand what the "application layer" is when we're talking about IoT security, the first thing we need to learn is the Open Systems Interconnection (OSI) model. The OSI model is a conceptual framework that describes all the communication that's going on in a network system. To illustrate this, the model uses seven layers, which are built on top of one another, starting from the physical layer.

As shown in the graphic on the next page, the application layer is the topmost layer, built on top of everything from real, physical wires and fiber optic cables in layer one, to data transport protocols in layer four, to encryption and decryption of data just below at layer six.
The application layer is the part of this model that humans actually interact with, facilitating communication between a person and a computer over a network. Application layer security simply means protecting this layer and the messages being sent there.

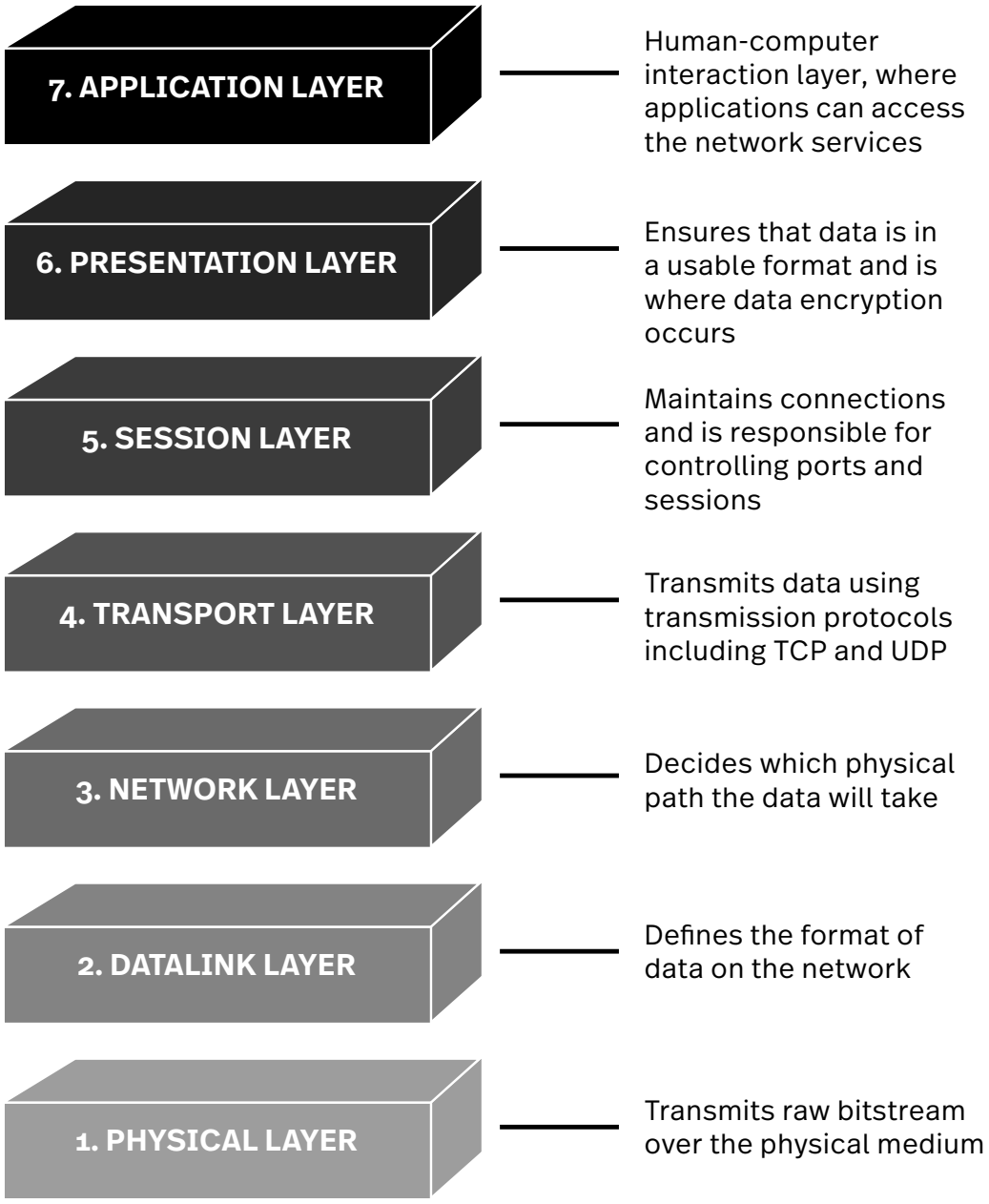
Why Do We Need Application Layer Security?

In the OSI model, if one layer of the system is compromised, all of the layers "above" it are also vulnerable, because they're built on top of one another. If the transport layer is attacked, therefore, the application layer may also be affected. In contrast, however, an attack on the application layer won't affect the layers beneath it.

But this doesn't mean that you should leave the application layer wide open

© Very 2020

Open Systems Interconnection Model



to threats – in fact, quite the opposite. The application layer has a wider attack surface because it's on top. It's responsible for communicating messages to real people, and the data in those messages could very well be sensitive and need protection.

Any time you're sending data over any distance, you'll need to adhere to the CIA triad. While this might sound like something you'd find in a secret government agent's arsenal, in computer science, it's just an acronym that stands for confidentiality, integrity, and availability. You should have confidence that the data is secret, has not been tampered with in any way, and that it's available to the people who are supposed to have access to it.

Different kinds of security threats can affect every layer of the OSI model, but the application layer is often subject to distributed denial-of-service attacks (DDoS) attacks, HTTP floods, SQL injections, cross-site scripting, parameter tampering, and Slowloris attacks.

Security Solutions for the Application Layer

How you go about securing your application layer all depends on your specific application. Applications have all sorts of different purposes and uses, so your security needs to be tailored to your unique situation. Additionally, you'll need to be aware of the trade-offs that may accompany tighter security measures. More security can mean more costs in data and computing power.

For example – some security solutions, while effective in preventing attacks, may slow your application down slightly. If you're dealing with highly sensitive data like financial or medical information, this small consequence may pale in comparison to the idea of a security breach.

It's kind of like putting seven locks on the door to your home instead of the standard single deadbolt. With seven locks, it will take you a little longer to get in and out of your house, but if you live in an area where there have been a lot of break-ins, that trade-off might be worth it to you.

That being said, what are some options you have for securing your application layer?

© Very 2020

Communication Protocols

Application layer security begins with the communication protocol you choose, whether it's HTTP, MQTT (a popular choice for IoT projects), or one of many others. Each individual protocol will have its own methods for performing user authentication – some more secure than others – so it's important to be familiar with the patterns present in each so that you know about any security adjustments you'll need to make.

The HTTP Basic authentication method, for example, is usually frowned upon because it's not inherently a very secure method. If you need more security, you might choose another method or protocol.

Alternatively, you could add in a more secure form of multi-factor authentication, like HTTP token-based authentication. With token-based authentication, the application validates a user's credentials on their first login, then provides the client with a signed token. The client stores the token and then must provide it with every login request, helping to prevent against CSRF attacks (where unauthorized commands are transmitted from a user that the application trusts).

Also of note – most protocols will have both a standard version and a secure version. While the standard version will be more lightweight and less secure, the secure version will be more complex and offer more protection.

Follow Best Practices for Encryption

One common misstep in protecting the application layer actually has to do with other layers, like the transport layer.

As mentioned earlier, due to the structure of all the layers, an attack at a given layer can affect all of the layers above it, though not the ones below. And while it's critical to secure the transport layer and those other deeper layers on their own, you shouldn't always rely on those layers to handle all your encryption. If for some reason an attacker exploits a vulnerability in the transport layer, like the infamous Heartbleed bug, data not encrypted at the application layer could be suddenly available.

For this reason, it's always a good idea to follow best practices for encrypting your data at the application layer to avoid unplanned exposure.

Firewalls

You can also use application firewalls to guard your application layer, plus other layers as well. Keep in mind that most firewalls are built with specific applications in mind, though many firewalls can be configured for multiple applications.

The firewall can control all network traffic to make sure that weird connections aren't happening in places you don't expect, and that all communications are following the desired protocols.

Making it a Reality

Whichever route you choose for application security, based on your own unique needs, make sure that your security isn't an afterthought. Put the safety of your users – and by extension, your business – first by following these best practices.

CHAPTER 4

Educating Consumers on IoT Security Measures

Often, the user is your most valuable asset for maintaining IoT security. User or customer education is always important in technology security. When the web first started out, for example, people were creating multiple personal accounts and using weak passwords that put their accounts at risk of being compromised. We had to explain to consumers that "123456" and "password" were too easy to guess. Similarly, with IoT, we need to educate consumers about best practices, like:

- **Guarding their smart devices.** Consumers need to understand that if their IoT devices have sensitive information, it's important that they don't leave it in an exposed area. If the device connects to the internet, for example, it may contain the customer's WiFi credentials, which could be manipulated by malicious parties.

- **Inspecting the IoT products they buy.** Keeping in mind that there are no well-known certification bodies for IoT security, customers need to be thoughtful about their purchases. What kind of data does this device store? How does it interact with other smart devices and IoT networks? If you're the vendor selling the device, it's your responsibility to notify your consumers of security information relevant to your product.

- **Proper disposal of IoT devices.** Consumers need to know the proper procedures for selling, giving away, or disposing of IoT devices. Many of these products, like smart light bulbs, may still store WiFi credentials that they wouldn't want to share with others – especially not people digging through trash cans looking for opportunities.

By educating those users, carefully crafting your IoT solutions, and implementing secure processes, you can protect yourself and your customers.

© Very 2020

Taking it with You

There are currently no industry-wide standards for IoT device security, leaving both individual consumers and businesses open to too much risk. By following these best practices to secure your device, firmware, web applications, and cloud infrastructure, however, you can begin to ensure the safety of your customers' data as you launch your IoT project.

Additionally, you'll be better prepared to make the right trade-offs when securing your product, without compromising overall user experience, timelines, and budget requirements.

Bring Your Product to Life

SEE WHAT WE DO

If you need additional resources or insights to make your IoT product vision a reality, Very has an experienced team of engineers, designers, and data scientists that care deeply about security. We deliver high-quality IoT applications by combining best-in-class web and embedded technologies with battle-tested security protocols, over-the-air firmware updates, and extensible code.

As a full-service IoT application development agency, we build every product with privacy, security, and scalability in mind. Our team becomes an extension of your team, and we'll use our hands-on experience to be true partners on your product development journey.

© Very 2020