IoT Security Best Practices

CONTENTS

WRITTEN BY GEOFFREY VAUGHN
SR. SECURITY ENGINEER AT SECURITY INNOVATION

Introduction

The Internet of Things is a concept involving many different areas of technology. At its core, IoT refers to connecting traditional devices and machinery to the global Internet. This practice is not new, in the sense that device manufacturers have been bringing their systems online since the beginnings of the Internet.

What makes this phase unique is that the micro-controllers and chipsets capable of network communication have never been smaller, cheaper, or easier to integrate. This transformation in embedded systems has drastically altered the market, making it much easier and less cost-prohibitive to connect even more devices, even those with incredible complexity.

In addition, this practice, alongside reliable wireless network infrastructure, has created entirely new categories of IoT devices that were not previously feasible, such as smart locks, home appliances, wearables, and connected vehicles.

As with all technology, new use cases and features bring new security considerations. It is important that these systems, whether new or legacy, be considered from the perspective of a new connected attack surface; one to which any actor on the Internet may be a threat. Security researchers within the IoT space have already begun identifying anti-patterns in IoT security architecture and development, reminiscent of forgotten security lessons from previous decades.

In this Refcard, we look to define the scope of what systems are encapsulated within the broader category of IoT. We further look to define a risk profile for organizations looking to create security policies around a connected device architecture.

The IoT Ecosystem

The components of an IoT ecosystem can vary depending on the specific technologies in place, though many follow a specific pattern.

DEVICE

This is your physical machine that will be Internet-connected. Typically, this involves sensor inputs for reading external data and output channels for executing an action. Example devices can include items common in the home (thermostats, refrigerators, door locks), machinery used in factories and worksites (industrial control systems, forklifts), next-gen automotive vehicles (cars, scooters, truck fleets), and much more.

SECURITY INNOVATION

30 billion

connected devices are estimated to be active by 2020
- Mozilla

61%

of device manufacturers have experienced an IoT security incident
- Trustwave

10%

felt fully confident their devices had adequate security precautions in place
- Kaspersky Labs

If you are involved in IoT you need to know how to secure it!

Security Innovation believes in helping software developers build secure connected products that consumers can trust. We aim to accomplish this through training, security-oriented design, and rigorous testing.

Check out our online IoT toolkit to access valuable content, including blog posts, tip sheets and on-demand videos to help your team mitigate IoT risk at your organization: **http://bit.ly/IoTtoolkit.**

MICROCONTROLLER

This is the integrated chipset responsible for network communications. Microcontrollers such as the STM32 are small in size and can be purchased and deployed at a relatively low cost.

These controllers are typically flashed with a minimal operating system such as RTOS or other embedded Linux variants. The firmware of these chips is responsible for coordinating and processing all interactions with the I/O of the device and can communicate over various wireless protocols. Depending on the use cases, some protocols that can be supported include Bluetooth, LTE, Wi-Fi, NFC, and others.

CLOUD COMPONENT

Once the device is capable of networked communications, it will reach out to an externally facing API to send and receive data. This is typically hosted in a cloud environment but can also be hosted on-premise in certain cases.

The choice of whether to host an API for IoT communication using on-premise hardware vs. cloud provider infrastructure comes with some tradeoffs. If the data is considered highly sensitive and falls under strict regulation, it may be advantageous to host the backend components on-premise to ensure that all necessary compliance guidelines can be upheld and policies can be tightly controlled. If the data being processed does not fall under strict compliance requirements and the need for flexible scalability and vertical elasticity is present, it will likely be the most beneficial and cost-effective to utilize cloud technology.

ADMIN/MANAGEMENT FRONTEND

These applications are typically written in a frontend framework as a web application or mobile app. It can be responsible for all configuration and settings of the device. Depending on whether the IoT device is equipped with display panels, these applications may also serve as a source of information and diagnostics.

Risk Profile

In this section we aim to enumerate typical areas of focus when assessing the risk present in an IoT system. These individual components listed previously are seldom new. Many have been well understood from a security perspective for decades. However, by combining them in this way, new emergent threats become present.

EXPANDED ATTACK SURFACE

The biggest threat to IoT systems comes from the introduction of legacy systems or devices that were never designed with Internet connectivity in mind.

Because many of these devices have been active in a non-connected form for years, manufacturers may determine that as a minor redesign, adding network connectivity does not warrant a new security architecture review. This often leads to implicit security assumptions that are no longer valid, causing potentially serious security risks.

For example, a thermostat without Internet connectivity has a very small attack surface, since it is only accessible from within a locked home. By adding remote connectivity, a whole new threat model emerges. Some of the new considerations include whether an attacker is able turn the heat on to run up the victim's electricity bill, or an attacker monitoring patterns in the network traffic from the thermostat to the cloud API in order to determine when the victim is present in their home and when they are away.

This is similar in nature to a real-life scenario of an IoT-related vulnerability leading to exploitation of a previously protected network. As reported by thehackernews.com, attackers were able to exploit vulnerabilities in the newly installed Internet-connected thermometers of a fish tank in order to pivot their attack towards the internal network of a casino. In this instance, by expanding the networks attack surface, malicious parties were able to cause real monetary damage to the institution.

> *"According to what Eagan claimed, the hackers exploited a vulnerability in the thermostat to get a foothold in the network. Once there, they managed to access the high-roller database of gamblers and "then pulled it back across the network, out the thermostat, and up to the cloud."*
>
> — thehackernews.com/2018/04/iot-hacking-thermometer.html

To counter these new risks, any changes to a system that increase the attack surface, such as enabling Internet connectivity, must require a full threat model and architecture review to ensure that the security assumptions of previous designs still apply or require appropriate adjustments.

OWASP TOP 10

The OWASP Top 10 has been an invaluable resource for security professionals in understanding threats to web applications and services.

Through the OWASP IoT Mapping Project, open source contributors are working to bring this same resource to this emerging technological arena. This aggregation outlines the following top 10 risks to IoT systems:

1. Weak, guessable, or hardcoded passwords
2. Insecure network services
3. Insecure ecosystem interfaces
4. Lack of secure update mechanism
5. Use of insecure or outdated components
6. Insufficient privacy protection
7. Insecure data transfer and storage
8. Lack of device management
9. Insecure default settings
10. Lack of physical hardening

scriptingxss.gitbook.io/owasp-iot-top-10-mapping-project/

From inspecting the draft list, we can recognize many of the classes of vulnerabilities most common to IoT. Many have been prevalent in other areas of technology, including web, desktop, and mobile applications.

In the following sections, we will dive deeper into some of the vulnerabilities inherited from other technological sectors, as well as some of the more unique classes of vulnerabilities found in emergent IoT systems.

INHERITED VULNERABILITY CLASSES

In a sense, it is fair to generalize the early TCP/IP services and early web servers of the previous decades as the original Internet of Things. While the majority of these original applications did not meet the security standards of today's web technologies, best practices emerged to defend these systems as attacks were executed and threats identified.

Many of the common vulnerabilities identified in the top 10 list above share a striking resemblance to the well understood flaws from server technology of the previous decade. Issues regarding hard-coded credentials, default settings, outdated components, and data protection are showing a reemergence in IoT systems, ignoring security best practices on these topics in other fields.

Default passwords in particular have been a pervasive issue in much of the IoT landscape. Developers who are not familiar with the massive attack vectors of an Internet-exposed interface make false assumptions that the only users attempting to log into a service or admin panel are well-intentioned.

Other areas of the web have learned hard-earned lessons about user authentication. These include requiring new users to set a password during a signup process as opposed to starting users with a default. In addition, best practices have emerged regarding password complexity, revocation, recovery, and multi-factor authentication. In order to ensure IoT technologies are deployed with the same level of security, it is necessary that these best practices be applied to any roles authenticating to IoT devices or cloud services.

Similarly, configurations and settings must also be configured through an onboarding process as opposed to using defaults. While this might require a small amount of effort from the user, it can drastically reduce the likelihood of an attack and can be handled in a way that prioritizes a clean and simple user experience.

Insecure network services and use of insecure or outdated components are another set of threats that are directly inherited from early server security. To limit the attack surface of a system, it is always important to minimize the exposed functionality to the absolute minimum necessary for function. If an IoT device internally uses a database, it is almost never appropriate to expose that database directly via a TCP listening service. In that same vein, using third-party services increases the security exposer not only to vulnerabilities in the custom application but also to exploitation of security flaws in these components.

Traditional network security requires firm standards on what services can be exposed and how often they must be updated or patched. In order to prevent IoT users from experiencing similar attacks, these standards must be adopted in this space, as well.

Lastly, insecure handling of sensitive data has reemerged as a common threat to IoT devices and users. While the previous decade has seen great improvements in storing and transmitting sensitive data, through technologies such as TLS, PKI, hashing, and advanced cryptography, the necessity of these protections has not always translated over to IoT.

Research into web communication and open-source tools such as Firesheep helped many leaders in web technology understand the risks associated with the plaintext HTTP protocol. Since then, browsers have continually adopted UX improvements that assist users in understanding the security of their data. Some examples of this include warnings when a network connection or any third-party scripts have been included without TLS, as well as using the URL bar to indicated trust-levels of the certificates being used (for example, displaying EV information).

Many of the interfaces used by IoT technologies do not have the same UX components as modern web browsers, so it is much more challenging for users to verify the security of their data. In order to ensure the IoT space moves security in the correct direction, users must be empowered to verify the integrity of their services. TLS must always be required, as anything less puts users at risk of attack. Additionally, all stored data with sensitive content must be encrypted or hashed as convention demands.

Most importantly, manufacturers of IoT systems must take note of security lessons from other fields, including common vulnerability types, known solutions, and best practice recommendations, and integrate that knowledge when designing security into their systems.

UNIQUE SECURITY CONSIDERATIONS AND VULNERABILITY TYPES FOR IOT

While many of the threats to IoT are inherited from existing technologies, there are some security properties that present specific challenges to IoT systems that are worth examining from a new perspective.

AVAILABILITY EXPECTATIONS

Many systems may be designed under the assumption that a network connection will always be available. Just like how code must be written so that negative cases are handled and dealt with appropriately, these unique IoT failure cases must be given the same thought and preparation.

A false assumption that devices will always remain connected can lead to these specific failure cases and can have costly effects. For example, if a remote vehicle, scooter, or other transportation device can only be started from an Internet API call but finds itself in a cellular dead-zone, this device might become inoperable. Depending on how well thought-out this failure case is, this may lead to a costly denial of service. It is important that all failure cases of IoT systems be formally modeled with broken connectivity included and accounted for.

SYSTEM UPDATES AND PATCHING

Since many of these systems are integrating modern software for the first time, best practices regarding regular security updates and patches, including having a process for frequent updates to critical software based on security advisories, have not been considered. In the event that a high-severity vulnerability is discovered in an IoT firmware, manufacturers may not be prepared to issue a full new deployment.

Since it is never possible to guarantee perfect security, procedures must be in place for handling security incidents in IoT systems. An effective strategy for deploying updates is a crucial ingredient to securing a userbase.

This requirement may be challenging in that users are not typically accustomed to performing firmware updates on a regular basis. For these reasons, it is recommended that whenever possible, firmware and software updates be performed behind the scenes automatically. Ensure that firmware updates are signed such that downgrades are not possible. Additionally, in order to minimize the risk of lockout, consider using an M-of-N key system on the bootloader. This way, in the event that a manufacturer loses a key or is compromised, other public keys on the device's bootloader can be used to ensure valid firmware images can always be signed and deployed.

BOTNETS

The concept of a botnet (a large network consisting of compromised machines) has existed for many years. Sophisticated software has emerged to control these networks of slave machines in order accomplish various levels of malfeasance, such as targeting companies with distributed denial of service (DDOS) attacks or to mine cryptocurrencies. With the explosion of new devices connecting to the public Internet, the possible pool of bots proportionally increases.

The following snippet of code is sourced from the leaked codebase of the *Mirai* botnet. This code is executed by the coordinator to construct a list of hosts that a network of compromised devices (including routers, security cameras, and DVRs) will use as a target for a DDOS attack.

Note that many of the hosts that were compromised by Mirai were exploited via unchanged default administrative credentials.

```
func NewAttack(str string, admin int) (*Attack, error) {
    atk := &Attack{0, 0, make(map[uint32]uint8),
    make(map[uint8]string)}
    args, _ := shellwords.Parse(str)
    var atkInfo AttackInfo
    // Parse attack name
    if len(args) == 0 {
        return nil, errors.New("Must specify an attack
        name")
    } else {
        if args[0] == "?" {
            validCmdList := "\033[37;1mAvailable attack
            list\r\n\033[36;1m"
            for cmdName, atkInfo := range attackInfo
            Lookup {
                validCmdList += cmdName + ": " + atkInfo.
                attackDescription + "\r\n"
            }
            return nil, errors.New(validCmdList)
        }
        var exists bool
        atkInfo, exists = attackInfoLookup[args[0]]
        if !exists {
            return nil, errors.New(fmt Sprintf("\033[33;
            1m%s \033[31mis not a valid attack!",
            args[0]))
        }
        atk.Type = atkInfo.attackID
        args = args[1:]
    }
    // Parse targets
    if len(args) == 0 {
        return nil, errors.New("Must specify prefix/
        netmask as targets")
    } else {
        if args[0] == "?" {
            return nil, errors.New("\033[37;1mComma
            delimited list of target prefixes\r\nEx:
            192.168.0.1\r\nEx: 10.0.0.0/8\r\nEx:
            8.8.8.8,127.0.0.0/29")
        }
        cidrArgs := strings.Split(args[0], ",")
        if len(cidrArgs) > 255 {
            return nil, errors.New("Cannot specify more
            than 255 targets in a single attack!")
        }
        for _,cidr := range cidrArgs {
            prefix := ""
            netmask := uint8(32)
            cidrInfo := strings.Split(cidr, "/")
            if len(cidrInfo) == 0 {
                return nil, errors.New("Blank target
                specified!")
            }
            prefix = cidrInfo[0]
            if len(cidrInfo) == 2 {
                netmaskTmp, err := strconv.Atoi(cidr
                Info[1])
                if err != nil || netmask > 32 || netmask
                < 0 {
                    return nil, errors.New(fmt Sprintf
                    ("Invalid netmask was supplied, near
                    %s", cidr))
                }
                netmask = uint8(netmaskTmp)
            } else if len(cidrInfo) > 2 {
                return nil, errors.New(fmt.Sprintf("Too
                many /'s in prefix, near %s", cidr))
            }
            ip := net.ParseIP(prefix)
            if ip == nil {
                return nil, errors.New(fmt.Sprintf
                ("Failed to parse IP address, near %s",
                cidr))
            }
            atk.Targets[binary.BigEndianUint32(ip[12:])]
            = netmask
        }
        args = args[1:]
    }
    …
```

Source: github.com/jgamblin/Mirai-Source-Code/blob/master/mirai/cnc/attack.go

IoT manufacturers must take into special consideration the risk of devices becoming part of a malicious botnet. Adequate monitoring, at the API level or through diagnostic interfaces on the individual device, is an important step into customer awareness so that detection of bot software can occur early and countermeasures be taken.

PRIVACY

With more and more devices reporting metadata to a central server for aggregation, privacy has become an even larger concern. This can prove to be a new method of revenue for some organizations, though through its nature, this can create a much larger impact for any security failure.

Privacy concerns have especially become topical in the last few years as continual breaches have shown just how much data is there to be stolen. Data aggregation and collection will continuously grow as storage becomes cheaper and processing becomes faster, so it is important that companies that are collecting this data understand what it is and realize the importance of its security.

It is important that all data collected through IoT devices be analyzed through a privacy preserving framework in order to understand what customer PII (personally identifiable information) might be involved. Data such as names, credentials, online habits, locations, and other personal traits can all be incredibly private to a user and must be respected as such. Even metrics such as timing between requests could be revealed to a malicious party so that they can gather details on a target through side-channel analysis.

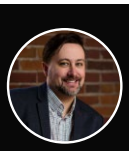
Manufacturers must take care to ensure that the data collected on their customers is not only sufficiently protected but is also gathered in a consent-preserving manner that does not violate the expectations of the user.

Conclusion

The Internet of Things is not a new concept. It is a collection of technologies that have existed in some form for several years now. With recent technology advancements, it has become easier than ever to integrate connectivity into machinery and devices, and this trend will not be slowing down. As more and more devices that we interact with come online, our digital and physical attack surfaces will continue to grow in parallel.

In order for device manufactures to protect their users' appliances and data, lessons from other sectors of technology must be applied. Well-known vulnerabilities in the components of IoT must be considered in this new arena to ensure best practice mitigations are included. Security must be a high priority at every point in the development lifecycle. Companies must actively participate in threat modeling, design review, penetration testing, and security maintenance.

By carrying on these lessons and participating in security-first policies, manufacturers can actively tide the growing threats of our increasingly Internet-connected world.



Written by **Geoffrey Vaughn,** *Sr. Security Engineer at Security Innovation*

Geoff Vaughn is a Sr. Security Engineer and the IoT Center of Excellence lead at Security Innovation. He specializes in finding exploitable vulnerabilities in software applications as well as reverse engineering binaries to locate vulnerable code. Geoff spends the majority of his time hacking and securing web applications, mobile apps, robots, 3D printers, infrastructure, embedded devices, and anything with a Biometric. He is passionate about security and helping others build secure products.

Devada, Inc.
600 Park Offices Drive
Suite 150
Research Triangle Park, NC

888.678.0399 919.678.0300