Learning Internet-of-Things Security "Hands-on"

Constantinos Kolias and Angelos Stavrou | George Mason University
Jeffrey Voas, Irena Bojanova, and Richard Kuhn | National Institute of Standards and Technology

What can you glean from using inexpensive, off-the-shelf parts to create Internet of Things (IoT) use cases? As it turns out, a lot. The fast productization of IoT technologies is leaving users vulnerable to security and privacy risks.

Although the Internet of Things' (IoT's) seeds were planted in 1999, IoT technologies have only recently become widely available as a result of nanotechnology, telecommunications, and capacitor technology advancements. The primary design tenet has remained the same: infuse common electronic devices with the impression of intelligence by allowing them to integrate seamlessly with their environment and automatically interact with other devices, thus minimizing reliance on human intervention.

IoT applications have expanded from strict industrial and closed-loop systems to commercially available products that address common user needs. An estimated 5 billion devices are connected to the Internet today, and this number is expected to increase to 25 billion by 2020.[1] At the same time, major IT players have gotten involved with IoT by developing OSs (for example, Google's Brillo: https://developers.google.com/brillo, https://dev.windows.com/en-us/iot; and Microsoft's Windows 10 IoT Series: https://www.microsoft.com/en-us/WindowsForBusiness/windows-iot), hardware (for example, Samsung's Artik: www.artik.io; and Intel's Edison: http://www.intel.com/content/www/us/en/do-it-yourself/edison.html), protocol stacks (for example, Google's Weave: https://developers.google.com/weave; and Apple's HomeKit framework: https://developer.apple.com/homekit), and cloud services (for example, IBM's Bluemix: https://console.ng.bluemix.net; and Amazon's Amazon Web Services IoT: https://aws.amazon.com/iot). In the near future, IoT devices are poised to become increasingly mainstream, shaping technology innovation to application areas ranging from healthcare (through health-monitoring wearables) to retail (with flyable crafts delivering online orders) to transportation (via self-driving vehicles). IoT technologies are transitioning from monolithic sensor and actuator boards to modular appliances focused on applications that satisfy real-life needs. Currently, IoT is an ecosystem composed of specialized hardware, network connectivity, and cloud counterparts all designed to facilitate data collection and processing. As we'll discuss, the fast productization of IoT technologies might leave users unable to defend themselves against security and privacy risks stemming from IoT products and frameworks.

Security Implications of IoT

IoT's security implications are creating hurdles for its wider adoption.[2,3] As the IoT market grows so does its

 Copublished by the IEEE Computer and Reliability Societies 1540-7993/16/$33.00 © 2016 IEEE

attack surface because new interconnected devices are added to the chain, each of which can become the weakest link for an adversary to exploit. Moreover, increased demand and adoption might make it hard for the industry to assess critical aspects of IoT security and privacy. For instance, new IoT-specific protocols are being designed constantly[4,5] but might not be thoroughly tested for trustworthiness. Lastly, IoT has become an umbrella term for many applications and industry use cases, each having its own security requirements but relying on the same fundamental IoT technologies. Designing security that encompasses and applies to all use cases is a daunting task, one that standards and best-practices committees are still struggling to determine how to address.

We learned firsthand about the potential pitfalls of IoT applications and components as applied to exemplary use cases. Our goal was to raise awareness of deficiencies in current practices and the lack of IoT security and privacy standards as well as their possible implications for the public and widespread IoT adoption. To that end, we present a set of exemplary use cases that leverage commercial off-the-shelf products and services. We purposely kept the IoT application type and implementation methodology simple, to mimic the design decisions an average user might take to achieve the desired functionality using similar components. We didn't attempt to provide wide coverage but rather to highlight some of the most severe, yet easy to abuse, security and privacy threats to simple IoT use cases, namely:

- leakage of personal identifiable information (PII),
- leakage of sensitive user information, and
- unauthorized execution of functions.

We refrain from naming the commercial products used because our goal is to evaluate IoT risks, not to compare products.

Our goal was to raise awareness of deficiencies in current practices and the lack of IoT security and privacy standards.

Leakage of PII

A desirable but often controversial feature of IoT applications is user and location awareness. IoT applications can trigger certain actions when specific events occur. For instance, they might provide mobile push notifications when a device's owner enters or exits an area, or launch an app when two objects come within close proximity. Initially, geofencing—mainly through GPS technology—was precise to only a few meters. Today, techniques that use the strength of various wireless networks' signals, including Wi-Fi and Bluetooth, have improved location precision to a few centimeters. This highly accurate location pinpointing is extremely appealing for retail IoT applications, such as targeted advertising and asset tracking, or for use cases such as airport check-ins.[6]

As promising as this capability might sound, we can't overlook its potential privacy implications, most notably the risk of user tracking. Indeed, application vendors might collect and store user location information over time. If they collect location data on a massive scale and for extended periods, how valuable does this information become? Another concern is unauthorized parties capitalizing on information leakage during transit (by wiretapping communication channels) or storage (by hacking application components). The underlying risk becomes more severe owing to the amount of PII that the average user broadcasts daily. This phenomenon first appeared with the introduction of smartphones and escalated with the proliferation of wearable devices. For example, researchers demonstrated how straightforward it is to trace wearable devices such as fitness trackers by exploiting the transmitted Bluetooth low energy (BLE) signals.[7]

The ease with which user activities can be tracked led to the development of opt-out mechanisms such as Do Not Track. In the IoT realm, where intelligent applications are associated with individuals rather than online personae and browsing histories reflect physical locations rather than virtual domains, the severity of user-tracking outcomes increases multifold. A person's identity and location information collected over time might be exploited in various ways: from simple user annoyance in the form of aggressive advertising (such as personalized spam at point-of-sale locations) to more serious advanced surveillance (such as tracking user routes and constructing user habit profiles) and even grievous intelligent terrorism (such as triggering criminal activities based on high profile individuals' presence in an area).

Personalized Light-Switch System

To highlight the simplicity and low cost of exploiting this privacy risk, we proposed a possible user-improvised system: an IoT version of a motion-sensing light switch. Modern office environments and industrial settings often use commercial light control systems to help reduce energy consumption. These limited systems are adequate for office use but lack the personalization

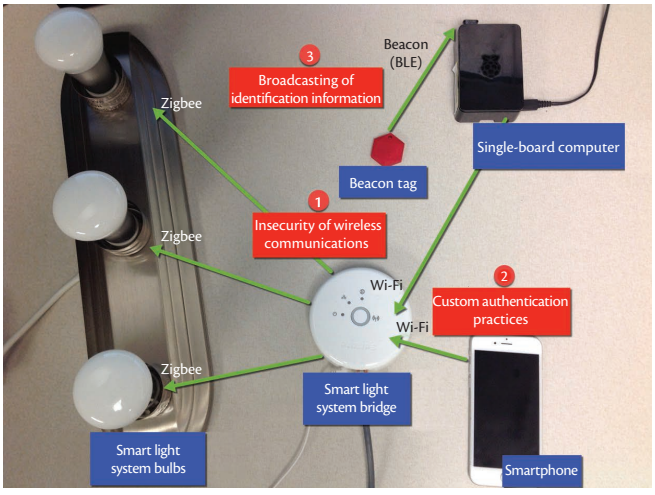

Figure 1. Internet of Things (IoT) use case of a personalized light-switch system employing inexpensive, commercially available components. This simple system has many vulnerabilities—including (1) insecure wireless communications, (2) custom authentication practices, and (3) broadcasting of identification information—that could lead to leakage of users' personal identifiable information. BLE is Bluetooth low energy.

that home automation users often seek. For instance, users might have different color and intensity preferences or even diurnal preferences.

At a minimum, such functionality requires a component that signifies a specific user's presence in an area, a component that senses the user's presence, and a component that turns the lights on or off. For each of these components, respectively, we used commercially available proximity tags, a Bluetooth-enabled computer connected to the network, and an off-the-shelf smart lighting system (see Figure 1).

Proximity tags are simple coin-sized microcontrollers whose only meaningful function is to constantly emit *beacons*—BLE messages of a specific format that contain a unique identifier. Application logic is located in the corresponding cloud or mobile applications, which in turn are programmed to respond in a specific way to the beacons. Typically, these devices are attached to the objects of interest. If the devices get misplaced, the corresponding application locates them with high accuracy, even in indoor environments. In our scenario, we assumed that the tags would be bound to a personal item that the user possesses, such as keys. In this way, the unique identifier transmitted would indirectly identify the user.

The remote-control lighting system included a set of smart bulbs, and an Ethernet/Wi-Fi enabled bridge that could receive and forward remote commands regarding the lights' status. Commands were transmitted via Wi-Fi as simple HTTP requests from the smartphone to the lighting system's bridge component. In turn, the bridge used the ZigBee protocol (www.zigbee.org/non-menu-pages/zigbee-pro-download) to forward commands to the light bulbs. Typically, users must first pair the bridge with a smartphone installed with the corresponding app. Users can then control the bulbs' output through the app; however, it's possible to authorize another device on the network to issue analogous Web requests.

Systems like the one described must maintain a database of pairs of unique identifiers that correspond to users and their preferences (for example, color). At the same time, they must constantly monitor for known identifiers and issue the corresponding HTTP requests as a response. Any computer connected to the network with Bluetooth capabilities is adequate; the only constraint is that its location must be static to ensure consistent readings. We used an inexpensive and well-known credit card–sized computer. Figure 1 displays the sample implementation's main components.

Identified Risks

In this use case, attackers might use alternative points of vulnerability to inflict harm or steal private information. More specifically, the system presents the following security concerns.

Insecure wireless communications. The wireless medium is open by nature, so actions such as jamming, eavesdropping, or message injection are more practical and can go unnoticed. In most cases, it's possible to manipulate the execution of the wireless protocol via the transmission of forged media access control (MAC) layer messages. More precisely, the 802.11 (Wi-Fi) protocol has been shown to be susceptible to denial-of-service (DoS) and man-in-the-middle (MiM) attacks as well as to cracking of the secret key.[8] The ZigBee protocol has documented weaknesses in key distribution because it relies on a single master key that's transmitted over the air, and to replay attacks, because it uses only the frame counter field to achieve message freshness.[9] In this use case, attackers with the appropriate equipment could easily launch successful DoS attacks against the Wi-Fi network—making the system unresponsive to any validly issued command—or to replay commands in the ZigBee network—causing anomalous behavior and annoyance to users.

Custom authentication practices. The limited-capability hardware utilized in most IoT commercial products (especially those in the home automation sector) necessitates lightweight security practices. Nonetheless, because of a lack of corresponding standards, many vendors rely on custom security mechanisms that are usually kept secret to achieve security through obscurity. The particular smart lighting product we used

implements a custom authentication mechanism based on tokens generated by simple hashing of the device's MAC address. Nitesh Dhanjani demonstrated that attackers in close proximity to the lighting system can easily forge control commands by spoofing whitelisted authentication tokens, thus capitalizing on this vulnerability to annoy users.[10]

Broadcasting user identification information. The described system identifies users and senses proximity through a device that constantly broadcasts a unique identifier within a short range. This is a typical example of theoretically harmless and highly desirable extra functionalities—identity and location awareness—having significant consequences for user privacy. The most important inefficiencies stem from the fact that identifiers are broadcasted in plaintext and that tags are attached to users' personal items, thus creating a correlation. We'll explain the mechanics that allow such behavior in further detail.

Beaconing of Unique Identifiers

Beacons estimate proximity based on the received signal strength indicator field for BLE signals. One of the killer apps of beacons is smart advertising. For example, a beacon transmitter can be placed in a relatively static location, such as inside a store in a shopping mall, and users that come in close proximity will receive promotional messages on their smartphones. Another popular application is the accurate tracking of items: users attach portable tags to valuable items, such as keys, that an app will help locate should they be misplaced.

Today, beacon devices are available in different shapes and sizes (for example, tags, USB sticks, or larger static appliances) but are inexpensive to construct and are considered expendables. In most cases, unlike conventional BLE devices, beacon transmitters can't pair with other devices and exchange data; thus, they transmit a single message throughout their entire lifetime. The message is usually a rather large identifier (so it's virtually unique) along with other information following one of these formats: iBeacon (Apple; https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf), altBeacon (Radius Networks; http://altbeacon.org), or Eddystone (Google; https://github.com/google/eddystone). Then, higher-layer applications (for example, installed in smartphones) perform actions when a message with an ID from a predefined set of universally unique identifiers (UUIDs) is sensed in close proximity.

The iBeacon specification, for example, assumes that beacon messages are transmitted in plaintext and that the UUID isn't secret. One can immediately see that spoofing such messages would be trivial. Although such an attack might annoy users (for example, by sending a notification at the wrong time), it's of little practical value. Possibly more serious for beacon technology is the risk of a user becoming associated with a constantly broadcasted number. It costs approximately US$80 to create a small board-based device with a motion sensor, a camera, and a BLE dongle that captures and stores high-quality images of unaware persons carrying beacon-transmitting devices. To avoid such conditions, several vendors offer customized hardware Beacon implementations that include mechanisms for changing the UUID. Our experience shows that many of these mechanisms simply rotate UUIDs and aren't based on cryptographic functions. These devices' limited computational power makes it difficult to provide strong security.

Hiding Identity

Because of these risks, users might not want to use beacon-broadcasting devices to tag highly personal objects that they carry daily. If they do, however, security-related amendments to the protocols might be required. A quick fix is to allow message broadcasting to be manually or automatically enabled or disabled based on user location. However, this solution introduces the additional challenge of securely enabling beacon devices remotely, requiring support for a lightweight authentication mechanism. More reliable solutions might require standards optimization to support changing UUIDs. In this case, this field must be altered unpredictably, probably relying on cryptographic operations such as hashing or encryption. This raises concerns regarding faster battery exhaustion and increased hardware costs.

Leakage of Sensitive User Information

From health data to payment details to arbitrary sensor information that potentially reveals user habits and preferences, many IoT applications deal with sensitive user data. Hence, one of the most far-reaching security threats is leakage of sensitive information, which the Open Web Application Security Project identifies as one of the most common vulnerabilities in the IoT ecosystem.[11]

Applications commonly collect redundant data or data not directly relevant to their purpose. There are several possible reasons for this. First, application developers might overestimate the requirements of future, improved application versions. In this case, information might leak to the application vendor and anyone with access to its back end (including malicious users). Second, applications often don't communicate properly with their users regarding the type of sensitive information being collected or don't provide opt-out options.

This might happen because the vendor wants to resell the data and so is gathering as much information as possible. In this case, the leaked information isn't limited to the application vendors but is available to any party that might acquire it. Finally, the flow of sensitive information is sometimes the outcome of bad protection practices during data transmission (for example, not using Transport Layer Security). In this case, anyone capable of eavesdropping on the communication might access the information exchanged.

Remote Watering System

To demonstrate inadvertent transmission of sensitive user data, we assembled a smart watering system (see Figure 2). Conventional watering systems rely on clock settings to automate the watering process for outdoor gardens or flower pots. Such systems require manual reconfiguration when changes in the environment occur, including fluctuations in temperature and moisture (such as rain). Using open source hardware IoT components, we easily built an advanced watering system that provided live feeds of environmental readings, including ground moisture, temperature, and luminosity, while simultaneously allowing remote control, or some automation, of the watering system.

This use case required a component that provided environmental readings, a module that implemented user decisions, and a unit that connected the user to the rest of the system. We relied on a single-board computer to execute all the sensing and actuating functionality and a Web application to provide the business logic and user interface.

The single-board computer we used is a popular open source hardware kit equipped with an 8-bit Atmel processor capable of handling multiple external circuits through its digital and analog I/O pins. It can be programmed using C/C++ language and a set of open source software libraries that aid common operations with these circuits. In our example, the board was the main system component. It was connected to moisture, temperature, and luminosity sensors that constantly provided readings to the Web application. At the same time, the board was connected to a relay linked to an external power source and a water valve. Finally, it was attached to a Wi-Fi shield through which it connected to a home wireless network. All communication occurred through Wi-Fi. We programmed the board to execute HTTP Post to the Web application in prespecified intervals to insert new readings. To stay current on the water valve's status (that is, on or off), the board polled the Web application. The Web application received the readings from an authorized device and stored them in a back-end database. Authenticated users could receive live feeds on conditions and alter the water valve's status. The system's main components are depicted in Figure 2.

Points of Failure

In this simple example, we identified multiple points at which data leakage, or other undesirable results, could occur. We describe the most important points of failure here.

Insecure Web application counterparts. Web applications interact with their users via dedicated user interfaces (UIs). In many cases, invalidated user input inserted into a UI entry field contains special sequences that form malicious code in the application layer. This can lead to XSS and SQL injection attacks. Such attacks can annoy users (which leads to reduced revenues) and compromise their privacy. Generally, failure to enforce good security practices, such as adopting strong credentials during the application development cycle, leads to compromised accounts and unauthorized access by malicious users.

Insecure wireless communications. As we discussed, wireless protocols have an array of vulnerabilities. In addition to trivial DoS attacks, which, in this case, might be significant—consider what would happen if the system was attacked while watering—MiM attacks are also relatively easy to accomplish. For example, an attacker can approach the valid network with a software-enabled access point (SoftAP) bearing the same service set identifier (SSID) as that network but no protection. Then, it can temporarily exile all clients (including IoT devices) from the valid network by broadcasting deauthentication packets (unprotected messages that are defined by the 802.11 specification and, thus, are easy to spoof). At that point, all devices will attempt to reconnect to the access point that advertises their known SSID and has the strongest signal: the attacker's SoftAP. Advanced OSs might evade the trap, but the less feature-rich OSs of many IoT devices won't understand the difference and will connect. So, attackers can eavesdrop on the unencrypted traffic of all devices connected to the SoftAP, compromising privacy if security isn't also enforced at a higher layer.

Unprotected communications. While communication protection is a de facto choice in desktop environments, it's not always practical with IoT, primarily because of the increased hardware costs versus the application type. Adoption of additional message protection mechanisms on higher layers might be necessary. Here, we elaborate on the reasons for this limitation.

Lack of Encrypted Communications

IoT application data might be transmitted in plaintext for

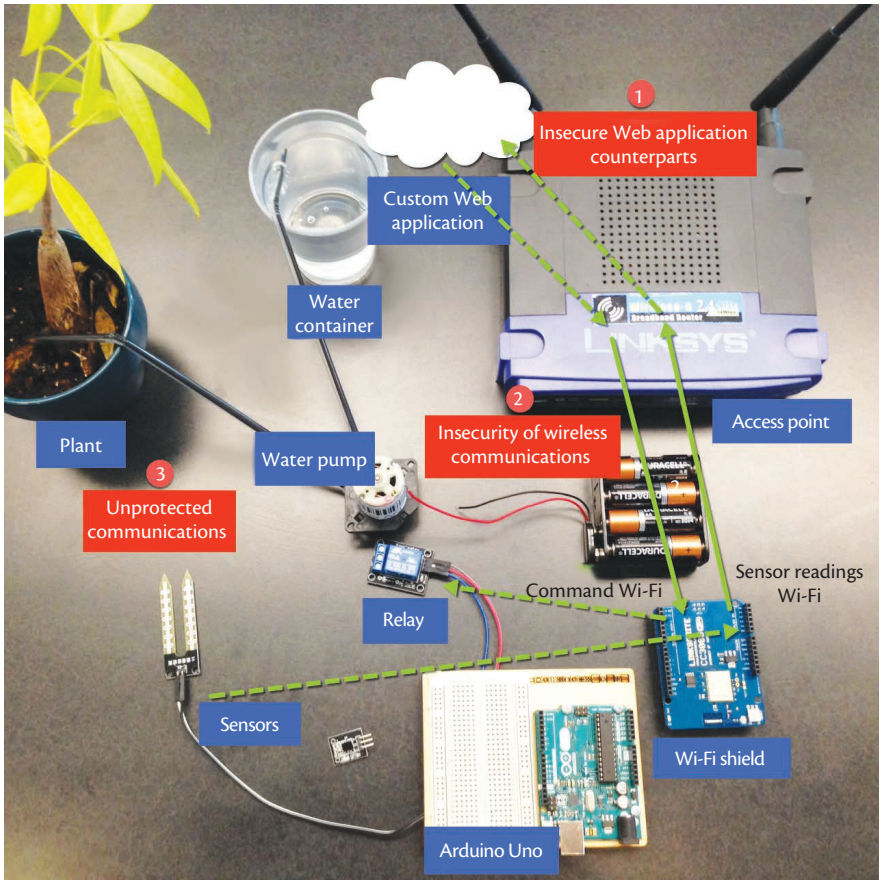

Figure 2. IoT use case of a remote watering system. The system's (1) insecure Web application counterparts, (2) insecure wireless communications, and (3) unprotected communications might inadvertently transmit sensitive user information.

many reasons. One common reason is the poor design decision to treat only the most obviously private user information as sensitive. In a home automation system, sensor data like temperature readings might not be considered sensitive. However, an eavesdropper monitoring such readings temporarily might be able to infer whether a user is at home by tracking sudden temperature changes or significant deviations from outside conditions (for instance, if the user starts the air conditioner).

Another reason for transmitting unprotected data is the choice of hardware. Many IoT products are inexpensive components with limited memory and computational resources. Such devices might be unable to support the computationally intense cryptographic functions of public-key cryptography. Hence, they might be incapable of supporting the SSL/TLS protocol, which is the industry-standard transport protection mechanism. In our use case, even if system designers considered the privacy implications of unencrypted data, they would have limited encryption options because of the hardware platform.

As a result, system designers have two choices: create their own lightweight security protocols or implement modified, stripped-down versions of well-known security protocols. The first choice runs the risk that the new mechanism will be vulnerable in practice and incur significant development costs. However, the second choice carries a great likelihood of a security vulnerability. Thus, custom security schemes or hardware-adapted protocol implementations might result in data being transmitted without meaningful protection. For example, a system designer might implement a custom TLS protocol, intentionally leaving out the computationally heavy certificate verification step. Evidence suggests that such a modified protocol would run efficiently even on small single-board computers;[12] however, it would create an opportunity for MiM attacks. Actually, this last case is the most deceptive of the three, because it gives users the illusion of industry-standard protection without really providing it.

Plugging the Leaks

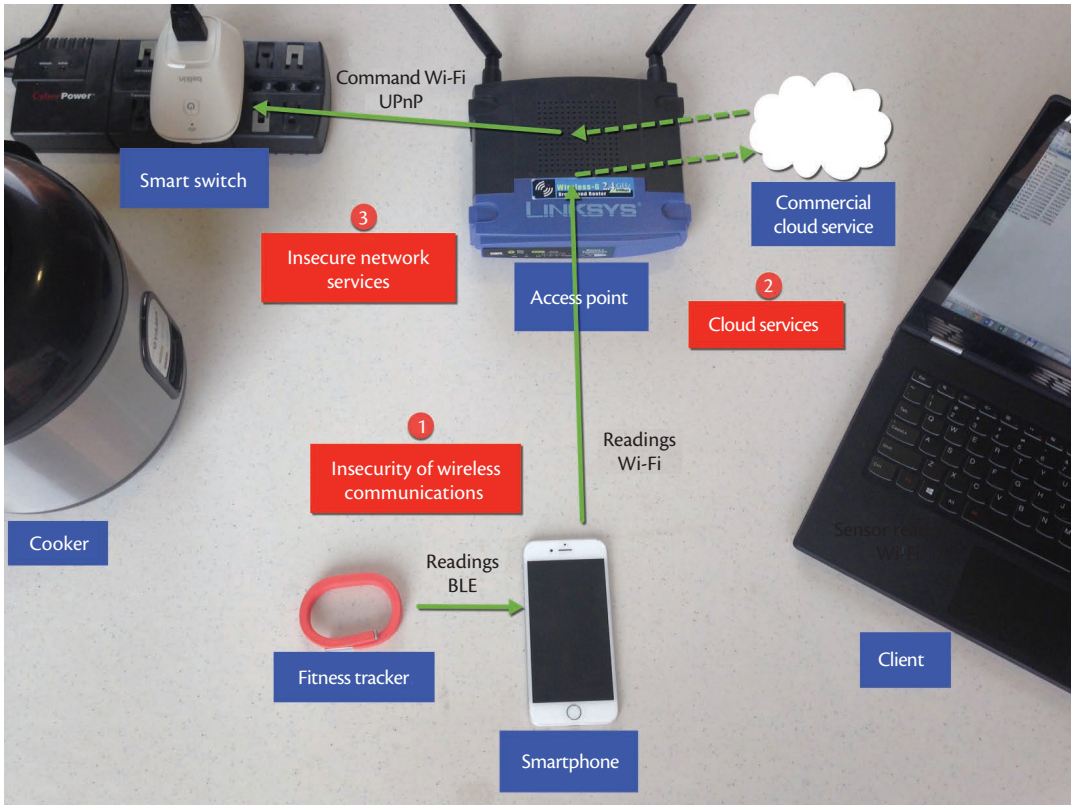

Figure 3. IoT use case of a system that automatically powers off devices (such as a cooker). By relying on (1) insecure wireless communications, (2) cloud services, and (3) insecure network services, such a system could allow attackers to control the device. UPnP is Universal Plug and Play.

A dedicated computer that forwards all sensor traffic to the corresponding Web application and vice versa might address the previously described architecture's security requirements by securing communications. This "proxy" device can communicate with different sensor types through shorter-range wireless protocols (for example, BLE), forcing attackers to be closer to their targets. Depending on the application, it might be possible to employ computationally efficient symmetric cryptography to protect these communication channels and then rely on the proxy device to protect the traffic through SSL/TLS when it's transmitted to the Internet. Another advantage is that this deployment strategy can scale without significant cost increases and support multiple communication protocols from different sensor types without extensive system reconfiguration.

Unauthorized Execution of Functions

Usability is a defining factor for any application's success, yet it's often viewed as in opposition to security. Frequently, security compromises or strong assumptions (for example, that the application is functioning within a secure network) must be made for a product to meet the market's user requirements.

Today, attackers have many opportunities to infiltrate a network: malware might evade antivirus checks, mobile apps with back doors might allow remote code execution, vulnerable services running on clients might allow buffer overflow attacks and unnecessarily open ports might welcome unauthenticated malicious entities. Given these risks and modern networks' increasing complexity, it's unrealistic to assume that the exposed clients are reliable and trustworthy. In fact, a compromised client inside the network is often used as a stepping stone for an unauthorized entity outside the network to issue commands that affect the status of local devices.

The threat is significant for the IoT ecosystem because IoT devices interact with the physical world and users. Surprisingly, our use-case analysis indicates that some IoT products adopt insecure mechanisms by default to provide a more user-friendly plug-and-play product.

Automatic Control of Devices

We assembled an IoT system that could automatically power off potentially dangerous appliances (for example, a cooker) on detecting that the user had fallen asleep (see Figure 3). This scenario is similar to that of powering on a coffeemaker in the morning. This use example

required a device that monitored the user's sleep status, a switch that could turn conventional appliances on and off, and an application that brought these two devices together. We relied on a well-known cloud service, a fitness tracker, and a smart switch.

The commercial cloud service we utilized permits users to create applications that trigger specific actions when specific events occur by using Web API calls.

The fitness tracker was a wearable device that monitors wearers' calories burned, sleep habits, and fitness by measuring their steps taken and heartbeat. Typically, it must be paired with a smartphone installed with the corresponding app. The two devices communicate in intervals, transmitting data from the fitness tracker to the smartphone. The smartphone then acts as a proxy by forwarding the data to the corresponding Web service via a Wi-Fi or 3G/LTE connection. In addition, it presents the statistics in a comprehensive UI. The triggering part of the cloud application was the fitness tracker's detection of sleep.

The smart switch is essentially a relay that can be connected to a wall socket and any conventional appliance. Initially, the switch must join the wireless network and pair with a smartphone installed with the corresponding app. The user can then control the device remotely from the app's UI. When users are within the home network, the communication occurs directly through Universal Plug and Play (UPnP) commands transmitted through Wi-Fi. When users are outside the home network's range, the device issues an HTTPS request to the corresponding Web service through 3G/LTE; that service then activates or deactivates the switch. The action part of the cloud application is the change in smart switch status. Figure 3 presents this use case's components.

Architectural Vulnerabilities

This use case presents opportunities for attackers to control the device, with potentially life-threatening results. The following are the most important vulnerabilities.

Insecure wireless communications. Once more, assailants can exploit the Wi-Fi protocol's weaknesses to constantly deauthenticate the device. In our sample case, if the status of the device is "off" during the moment of the attack, the user outcome will most likely be simple annoyance. On the other hand, if the status is "on," the attack might prevent the device from turning off.

Cloud services. Some off-the-shelf IoT products use cloud services for storage and command and control. Typically, when users purchase a product, they implicitly trust their cloud counterparts as well. In this particular use case, however, the lack of interoperability protocols to permit direct communication between the two applications (the fitness tracker and the smart switch) meant that an external cloud service had to serve as the middleman. Hence, trust had to be placed in yet another entity. As this circle of trust grows, so does the risk that one of these services follows less strict security practices. A breach of one service might result in loss of sensitive user information (such as health data and sleep habits) and remote control of home appliances.

Insecure network services. Our use-case analysis indicated that some commercial IoT products still adopt insecure network services and protocols—UPnP is one of the most popular—to facilitate their seamless setup. This introduces the risk that unauthorized insiders can control the switch as we describe next.

Insecure Protocols Running on the Network

UPnP is a set of network protocols that Microsoft standardized in 1999. It provides a convenient way to introduce new devices into the network, facilitate their discovery by other devices, and permit their control. UPnP requires routers to have the corresponding feature enabled. It's widely used in Voice over Internet Protocol (for example, Skype), peer-to-peer (for example, uTorrent), and gaming applications. One of UPnP's most serious security issues is that it trusts network clients. It doesn't encrypt data, nor does it require users to authenticate before triggering the execution of functions, such as searching for UPnP-enabled devices, querying for their supported capabilities, and activating them.

In this use scenario, any client on the local network, and not just the authorized smartphone device, could query the remote-control switch for its supported functions and issue a simple, unencrypted HTTP request with the right SOAP body to manipulate the device at will.

Future Prospects

Although IoT testing has received relatively little attention, assuring security and privacy is a central concern as systems proliferate and connect to safety or security-critical applications. This is evident even from our small set of examples.

In many ways, the challenges of IoT assurance amplify those associated with testing more familiar software and hardware platforms.[13] But for testing, the most significant characteristic of IoT systems is the sheer variety of devices and means of communication. To complicate matters, variations in processor speed, memory, protocols, and application types are much bigger with IoT than with traditional desktops, laptops, or smartphones. This inherent heterogeneity requires testing a

wide range of IoT application platforms and associated tools. Unfortunately, many organizations don't have sufficient resources to perform the required testing or to keep their testing current. However, practical assurance approaches addressing this resource problem have been developed: distributed frameworks allowing geographically separated parties to cooperate on testing are becoming available, providing a full complement of shared testing resources to reduce cost and testing time.[13,14] Test frameworks allow components owned by multiple cooperating organizations to communicate as if they were adjacent, making it possible to test complex interactions and interoperability among various IoT devices.

> **Standards bodies and industry experts must begin formulating suitable guidance and identifying the right security and privacy primitives.**

IoT testing is further complicated by the increased number of interacting devices. Most IoT devices exist to send and receive data, and the number of potential communicating pairs increases with the square of the number of devices. Furthermore, many interactions will involve more than just two IoT devices exchanging data, so interoperability testing must consider huge numbers of combinations. Fortunately, combinatorial test methods from the experimental design field allow compression of huge numbers of configuration settings or input variable values into a few tests. These methods could be highly effective for IoT, where component interactions are especially critical.[15,16] Another approach is to use models based on interoperability patterns.[17] Such model-based testing takes advantage of similarities in architecture and behavior for systems in different IoT application domains. Despite these advances, finding ways to provide appropriate assurance levels for the incredibly diverse IoT domain remains a significant research challenge.

Our foray into assembling IoT systems using inexpensive and readily available modules to create appealing, practical use cases suggests that IoT security and privacy aren't always well defined or understood by consumers and manufacturers. Specifically, we demonstrated that some IoT implementations can lead to inadvertent tracking of user identity and behavior if data isn't classified as sensitive and if devices aren't built with privacy as a design tenet. We also showed that data encryption isn't always enabled, and even when it is, the cryptographic libraries might exhibit security flaws that expose the data. Finally, we determined that some IoT systems suffer from the isolation syndrome of embedded devices: weak protocols and practices are sometimes used because some IoT technologies were designed for closed, non-Internet use with proprietary code and no thorough software testing.

Usability and interoperability are important design drivers for IoT manufacturers. It seems prudent to avoid past mistakes and elevate security and privacy as design tenets. There are relatively few standards or best practices for IoT security design and testing, although some related guidance is being developed by the Cyber-Physical Systems Public Working Group[18] and in documents such as the National Institute of Standards and Technology's *Guidelines for Smart Grid Cyber Security*.[19] We believe that standards bodies and industry experts must begin formulating suitable guidance and identifying the right security and privacy primitives. Beginning to identify IoT security and privacy requirements will open many research challenges—challenges that will continue to grow as IoT applications become part of our daily lives. Indeed, IoT was introduced more than 15 years ago, yet no thorough research has identified IoT-specific vulnerabilities and compared them with non-IoT vulnerabilities. Although individual IoT technologies have been studied recently, much work remains to fully describe different IoT systems' behavior when under attack. ■

Disclaimer

Certain products are identified in this document to describe the work conducted, but identification doesn't imply that the National Institute of Standards and Technology recommends or endorses these products or that the products are necessarily the best available for the purpose.

References

1. J. Rivera, "Gartner Says 4.9 Billion Connected 'Things' Will Be in Use in 2015," *Gartner*, 11 Nov. 2014; www.gartner.com/newsroom/id/2905717.
2. X. Teng, J.B. Wendt, and M. Potkonjak, "Security of IoT Systems: Design Challenges and Opportunities," *Proc. IEEE/ACM Int'l Conf. Computer-Aided Design* (ICCAD 14), 2014, pp. 417–423.
3. J. Pescatore, *Securing the Internet of Things Survey*, white paper, SANS Inst., Jan. 2014; www.sans.org/reading-room/whitepapers/analyst/securing-internet-things-survey-34785.
4. Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," Internet Engineering Task

Force, June 2014; https://tools.ietf.org/html/rfc7252.
5. E. Kim, D. Kaspar, and J. Vasseur, "Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)," Internet Engineering Task Force, Apr. 2012; https://tools.ietf.org/html/rfc6568.
6. "Indoor Location in Retail: Where Is the Money?," ABIresearch, Feb. 2013; www.abiresearch.com/market-research/product/1013925-indoor-location-in-retail-where-is-the-mon.
7. S. Lester, "The Emergence of Bluetooth Low Energy," Context, 21 May 2015; www.contextis.com/resources/blog/emergence-bluetooth-low-energy.
8. C. Kolias et al., "Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset," *IEEE Communications Surveys & Tutorials*, vol. PP, no. 99, 2015.
9. E. Yuksel, H. Nielson, and F. Nielson, "Zigbee-2007 Security Essentials," *Proc. 13th Nordic Workshop on Secure IT Systems* (NordSec 08), 2008, pp. 65–82.
10. N. Dhanjani, *Abusing the Internet of Things: Blackouts, Freakouts, and Stakeouts*, O'Reilly Media, 2015.
11. "Internet of Things Top Ten," Open Web Application Security Project, 2014; www.owasp.org/images/7/71/Internet_of_Things_Top_Ten_2014-OWASP.pdf.
12. A. Ardiri, "Is It Possible to Secure Micro-Controllers Used within IoT?" Evothings, 27 Aug. 2014; https://evothings.com/is-it-possible-to-secure-micro-controllers-used-within-iot.
13. P. Rosenkranz et al., "A Distributed Test System Architecture for Open-Source IoT Software," *Proc. Workshop IoT Challenges in Mobile and Industrial Systems* (IoT-Sys 15), 2015, pp. 43–48.
14. A. Lahmadi, C. Brandin, and O. Festor, "A Testing Framework for Discovering Vulnerabilities in 6LoWPAN Networks," *Proc. IEEE 8th Int'l Conf. Distributed Computing in Sensor Systems* (DCOSS 12), 2012, pp. 335–340.
15. A.H. Patil, N. Goveas, and K. Rangarajan, "Test Suite Design Methodology Using Combinatorial Approach for Internet of Things Operating Systems," *J. Software Eng. Applications*, vol. 8, no. 7, 2015, p. 303.
16. G. Dhadyalla, N. Kumari, and T. Snell, "Combinatorial Testing for an Automotive Hybrid Electric Vehicle Control System: A Case Study," *Proc. IEEE 7th Int'l Conf. Software Testing, Verification and Validation Workshops* (ICSTW 14), 2014, pp. 51–57.
17. P. Grace et al., "Taming the Interoperability Challenges of Complex IoT Systems," *Proc. 1st ACM Workshop Middleware for Context-Aware Applications in the IoT* (M4IOT 14), 2014, pp. 1–6.
18. "Cyber-Physical Systems Public Working Group," National Inst. Standards and Technology, 11 Aug. 2014; www.nist.gov/cps/cps-pwg-workshop.cfm.
19. *Guidelines for Smart Grid Cyber Security*, tech. report NISTIR 7628 Revision 1, National Inst. Standards and Technology, Sept. 2014; http://nvlpubs.nist.gov/nistpubs/ir/2014/NIST.IR.7628r1.pdf.

Constantinos Kolias is a research assistant professor at George Mason University. His research interests include security for 4G/5G communication protocols, wireless intrusion detection, and Internet of Things (IoT) and machine to machine security. Kolias received a PhD in computer science from the University of the Aegean. Contact him at kkolias@gmu.edu.

Angelos Stavrou is an associate professor of computer science and director of the Center for Assurance Research and Engineering at George Mason University. His research interests include large systems security and survivability, intrusion detection systems, privacy and anonymity, and mobile ad hoc network and mobile device security. Stavrou received a PhD in computer science from Columbia University. Contact him at astavrou@gmu.edu.

Jeffrey Voas is a computer scientist at the National Institute of Standards and Technology (NIST). His research interests include IoT and fundamental computer science shortcomings. Voas received a PhD in computer science from the College of William and Mary. He is a Fellow of IEEE and the American Association for the Advancement of Science. Contact him at j.voas@ieee.org.

Irena Bojanova is a computer scientist at NIST. Her research interests include IoT, distributed systems, and formal methods. Bojanova received a PhD in computer science from the Bulgarian Academy of Sciences. She is a Senior Member of IEEE and serves as the Integrity Chair of the IEEE Computer Society Publications Board. Contact her at irena.bojanova@computer.org or irena.bojanova@nist.gov.

Richard Kuhn is a computer scientist at NIST. His research interests include combinatorial methods in software testing and access control models. Kuhn received an MS in computer science from the University of Maryland, College Park. He is a Senior Member of IEEE. Contact him at kuhn@nist.gov.

cn *Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.*