Learning Internet of Things Security "Hands-on"

Constantinos Kolias, Angelos Stavrou, Jeffrey Voas, Irena Bojanova, Richard Kuhn

Abstract

The Internet of Things (IoT) encompasses a wide range of processes: sensing, computation, communication, time, context, and data, to name only a few. How do all of these function as a system when using commercially available components that can be purchased from anywhere and at a low cost, and with little or no component pedigree available? To provide some practical answers to the these questions, we purchased components and created a set of small use cases to see how it all interoperated. We additionally wanted to ask what this means in terms of security, given many reports warning that IoT often lacks security features. Said another way, what could we glean from buying cheap parts, creating our own use cases with them, and learning how to exploit the vulnerabilities of those off-the shelf parts? The answer is a lot.

Introduction

Although the seeds of what we consider now as the Internet of Things (IoT) were planted in 1999, IoT technologies have become widely available only recently, as a result of advancements in nanotechnology, telecommunications, and capacitor technology. The primary design tenet 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 any reliance on human intervention.

Applications of IoT have expanded from strict industrial and closed-loop systems, to commercially available products that address common user needs. Gartner estimated that today there are 5 billion devices connected to the Internet, while by 2020 this number will increase to 25 billion (Rivera, 2014). At the same time, many major IT players have gotten involved with the IoT, either with the development of operating systems (e.g., Google's Brillo, Microsoft's Windows 10 IoT), hardware (e.g., Samsung's Artik, Intel's Galileo), protocol stacks (e.g., Google's Weave, Apple's HomeKit framework), or cloud services (IBM's Bluemix, Amazon's AWS IoT). In the near future, IoT devices are poised to become more and more mainstream, shaping technology innovation to application areas ranging from healthcare (through health monitoring wearables), retail (with flyable crafts delivering orders placed online), to transportation (via self-driving vehicles able to safely deliver users to their destination). IoT technologies are transitioning from monolithic boards of sensors/actuators towards modular appliances focused on applications that satisfy real-life needs. Currently, IoT has evolved into an ecosystem comprised of specialized hardware, network connectivity, and cloud counterparts all designed to facilitate data collection and processing. As we show in this article, the fast productization of IoT technologies may sometimes leave users vulnerable,

unaware, and in many cases unable to defend against security and privacy risks that stem from the use of IoT products and frameworks.

Security implications of IoT and its applications have already turned into hurdles for its wider adoption (Teng, Wendt, & Potkonjak, 2014), (Pescatore, 2014). On the one hand, as the size of the IoT market grows, so does its attack surface because new interconnected devices are added to the chain; each of which can become the new weakest link for an adversary to exploit. Moreover, the increased demand and adoption may 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 (Shelby, Hartke, & Bormann, 2014), (Kim, Kaspar, & Vasseur, 2012) but they may not have been thoroughly tested to prove their trustworthiness. Lastly, IoT has become an umbrella term for many different 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 of the use cases can be a daunting task, and standards and best practices committees are still determining how to address this challenge.

We learned first-hand about the potential pitfalls of IoT applications and components as applied to the exemplary use cases. Our primary goal is to raise awareness regarding deficiencies in current practices and lack of standards pertaining to IoT security and privacy and their possible implications to the public and widespread adoption. To that end, we present a set of exemplary use cases that leverage commercial off-the-self products and services. Both the IoT application type and the implementation methodology were kept simple on purpose, mimicking closely the design decisions the average user could have made to achieve a desired functionality using commercial off-the-self components. We did not attempt to provide wide coverage but rather to highlight some of the most severe, yet easy to abuse, security and privacy threats that exist in simple IoT use cases, namely, (a) the emanation of information relative to user location, (b) the leakage of sensitive information, and (c) the remote exploitation of device functionality by unauthorized users. We refrain from exposing the commercial products used in our example scenarios by name because the goal of this work is to evaluate IoT risks, not to compare products.

Leakage of Personal Identifiable Information (PII)

One of the most desirable yet sometimes controversial features of IoT applications is their ability to perform activities that employ user and location awareness. IoT applications can have certain actions being triggered by the occurrence of specified events. For instance, push notifications in the mobile device when its owner enters or exits an area, or launching an app when two objects come within close proximity. Initially, the geo-fencing feature–materialized mainly through the GPS technology–had precision of a few meters. Today, through the utilization of techniques that take advantage of the strength of the signals of various wireless networks, including WiFi and Bluetooth, the location precision has improved to a few centimeters. This highly accurate pinpointing of location is extremely appealing for IoT applications in the retail sector including targeted advertising and asset tracking or for other use cases such as check-in in airports (ABIresearch, 2013).

Of course, promising as this capability may sound, one cannot overlook its potential privacy implications, most notably the risk for user tracking. Indeed, application vendors may 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 over time? Another concern revolves around unauthorized parties that may capitalize on the leakage of information during transit (by wiretapping communication channels) or storage (by hacking application components). The underlying risk becomes more severe due to the amount of uniquely identifiable information that the average user is broadcasting on a daily basis. This phenomenon occurred initially with the introduction of smartphones and lately with the proliferation of wearable devices. For example, researchers have demonstrated that it is straightforward to track wearable devices such as fitness trackers by exploiting the transmitted Bluetooth Low Energy signals (BLE) (Lester, 2015).

The ease by which tracking of user activities can be peformed mandated the development of opt-out mechanisms such as Do Not Track (DNT). In the IoT realm, where intelligent applications are associated with individuals rather than online personae, and the browsing history reflects physical locations rather than virtual domains, the severity of the outcomes of user tracking increase multifold. Identity and location information for a person over a duration of time might be exploited in various ways: from (a) simple user annoyance in the form of aggressive advertising (e.g., personalized spam at point of sales locations), to the more serious (b) advanced surveillance (e.g., tracking of routes followed by users and constructing profiles of user habits), or even the grievous outcomes of (c) intelligent terrorism (e.g., triggering of criminal activities based on the the presence of high profile individuals in an area).

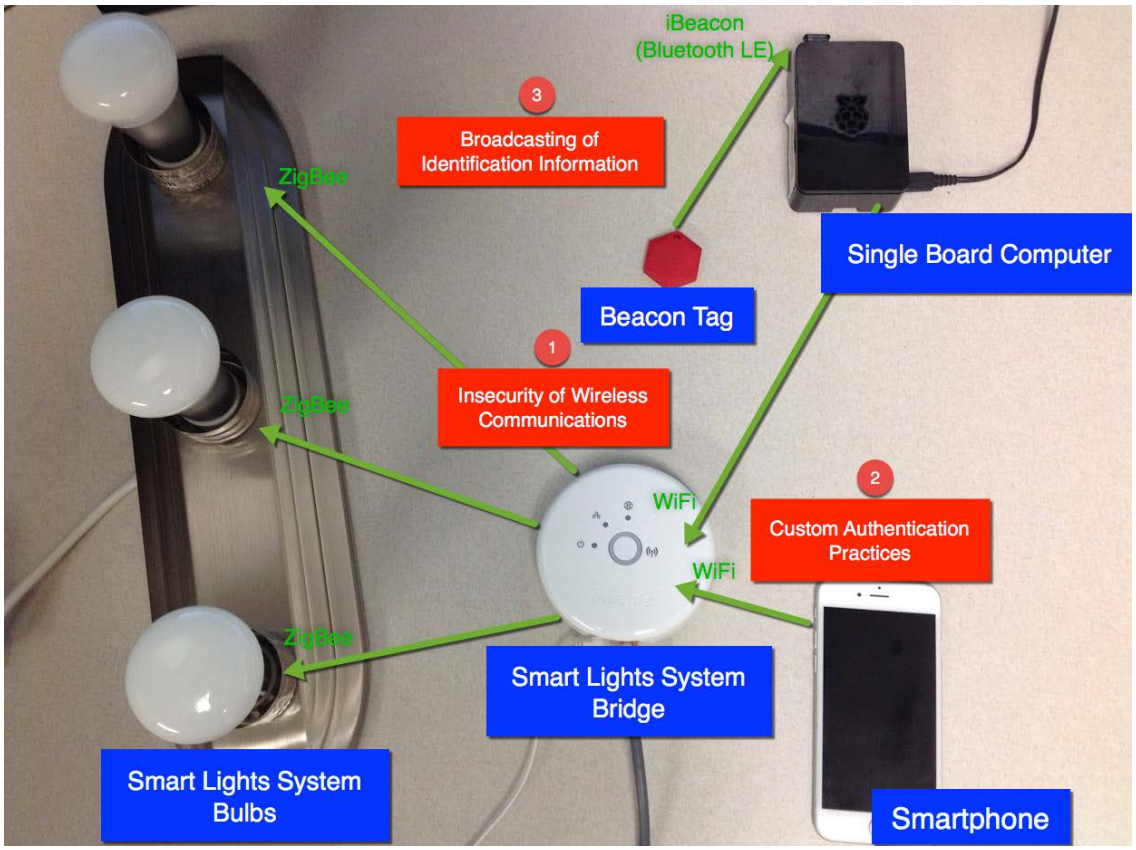

Figure 1: Use Case for Personalized Light Switch System composed of a Beacon tag & a Smart Lights

Personalized Light Switch System

To highlight the simplicity and low cost of the privacy risks, we put forward a possible user-improvised system conceived as an IoT version of the motion-sensing light switch. Commercial systems of light control can be found in modern office environments and industrial settings as they can contribute to the reduction of energy consumption. Their limited functionality might be adequate for many office uses but it lacks personalization characteristics that home automation users often seek. In such situations, different users might have a different color and/or intensity preferences or even diurnal preferences.

At minimum, such functionality requires a component that signifies the presence of a specific user in an area, a component that is able to sense the user's presence, and a component to produce the action i.e., have the lights turn on/off. For this scenario, we relied on a) commercially available proximity tags, b) a computer connected to the network that has Bluetooth capabilities and c) an off-the-shelf smart lighting system, for each of these tasks respectively.

Proximity tags are simple coin-sized microcontrollers that are unable to perform any meaningful functionality other than constantly emitting "beacons". These are BLE messages of specific format that contain a unique identifier. Application logic is located in the corresponding

cloud/mobile applications, which in turn are programmed to respond to the presence of beacons in a specific way. Typically, these devices are attached to objects of interest. If they get misplaced, the corresponding application locates them with high accuracy even in indoor environments. In our scenario we assume that the tags are bound with a personal item each user possesses e.g., their keys. In that way, the unique identifier transmitted indirectly identifies the user.

The remote control lighting system includes a set of smart-bulbs and an Ethernet/Wi-Fi enabled bridge that is able to receive and forward remote commands regarding the status of the lights. In the adopted system, any command is transmitted from the smartphone to the bridge component of the lighting system via Wi-Fi in the form of a simple HTTP request. In turn, the bridge communicates with the light bulbs themselves via the ZigBee protocol (ZigBee, 2012) to forward the command. Typically, in an initialization step, the bridge must be paired with a smartphone device that has the corresponding app installed. The user may control the output of the bulbs through the app however, it is possible to authorize another device on the network to issue analogous web requests.

A database of pairs of unique identifiers that correspond to users, along with their preferences (color), must be maintained. At the same time, constant monitoring for known identifiers must take place and the corresponding HTTP requests must be issued as a response. Any computer connected to the network that has Bluetooth capabilities is adequate and the only constraint is that its location must be static so that its readings are consistent. In our implementation, we relied on a well-known credit card-sized computer due to its low cost. Figure 1 displays the main components of this sample implementation.

Risks with the Personalized Lights

In the use-case described above one may take advantage of alternative points of vulnerability to inflict harm or steal private information. More specifically the system includes security concerns such as:

1. ***Insecurity of wireless communications*** – the wireless medium is open by nature and as such, actions like jamming, eavesdropping or message injection are more practical and can go unnoticed. In most cases it is possible to manipulate the execution of the wireless protocol via the transmission of forged MAC layer messages. More precisely, the 802.11 (WiFi) protocol has been proved susceptible to Denial of Service (DoS) attacks, Man-in-the-Middle attacks as well as cracking of the secret key (Kolias, Kambourakis, Stavrou, & Gritzalis, 2015), while the ZigBee protocol has documented weaknesses regarding the key distribution as it relies on a single Master Key that is transmitted over the air, and replay attacks as it uses just the frame counter field to achieve message freshness (Yuksel, Nielson, & Nielson, 2008). In this use case, it is considered trivial for an attacker with the appropriate equipment to launch successful DoS attacks against the Wi-Fi network thus making the system unresponsive to any validly issued command or to replay commands in the ZigBee network, causing anomalous behavior and annoyance to the end-user.
2. ***Custom authentication practices*** – the limited-capabilities hardware utilized in most IoT commercial products (especially those used in the home automation sector) generates

the need for lightweight security practices. Nonetheless, due to the lack of corresponding standards many vendors have to rely on custom-tailored security mechanisms which are usually kept secret as an attempt to achieve security through obscurity. The particular smart-lights product used in this experiment, implements a custom authentication mechanism that is based on tokens generated by simply hashing the MAC address of the device. Dhanjani demonstrated at Black Hat Asia (Dhanjani, 2015) that attackers in close proximity can easily forge control-commands by spoofing the whitelisted authentication tokens, and then capitalize on this vulnerability to create annoyance for the victim user.

3. ***Broadcasting of user identification information*** – in the described system, the identification of users and proximity sensing is achieved through a device that constantly broadcasts a unique identifier within a short range. This is a typical case where a theoretically harmless and highly desirable extra functionality i.e., identity and location awareness can have significant consequences for user privacy. The most important inefficiencies stem from the fact that the broadcasting of identifiers is done in plaintext and that such tags are attached to personal things a person may carry, thus creating a correlation. The mechanics that allow such behavior will be explained in further detail.

Beaconing of Unique Identifiers

Beacons are Bluetooth LE signals emitted by devices that are part of indoor proximity systems. The proximity estimation is done based on the Received Signal Strength Indicator (RSSI) field for these signals. One of the killer apps of beacons is smart-advertising. Such scenarios assume that the beacon transmitter would be placed in a relatively static location e.g., inside a store in a mall where users that come in close proximity would receive promotional messages on their smartphones. Another popular application is the accurate tracking of items. The user would attach portable tags on valuable items e.g., keychain and an app would assist in locating these items in case they ever got lost.

Today, beacon devices are available in different shapes and sizes (e.g., 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 are unable to pair with other devices and exchange data, thus they are bound to transmit a single message throughout their entire lifetime. The message is usually a rather large identifier (so that is virtually unique) along with other information following one of the available formats: iBeacon (Apple), altBeacon (Radius Networks), or Eddystone (Google). Then, higher layer applications (e.g., installed in smartphones) perform actions when any from a predefined set of UUID's is sensed in close proximity.

The iBeacon specification, for example, assumes that beacon messages are transmitted in plaintext and the Universally Unique Identifier (UUID) information is not considered secret. One can immediately realize that spoofing such messages is considered trivial. Actually such an attack may easily create annoyance to the end-user (e.g., receive a notification in a wrong time) even so it is considered of relatively low practical value. Possibly more serious for beacon

technology is the risk for a user to be associated with a number that is being broadcasted constantly. Today, it costs approximately $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 have implemented custom versions that include mechanisms for changing UUIDs. Our experience has shown that many of these approaches adopt a simple rotating scheme of UUIDs and are not based on cryptographic functions. Limitations in computational power of such devices can make it difficult for this approach to provide strong security.

Hiding the Identity

Because of the risks identified above, many users may not want to use beacon broadcasting devices to tag highly personal objects that the users tend to carry with them on a daily basis. If that behavior is desirable, revised protocols may be required. A quick fix is to allow the broadcasting of messages to be enabled/disabled manually or automatically based on the user location. This solution however, introduces the challenge for the beacon devices to be securely enabled remotely, thus requiring support for some sort of lightweight authentication mechanism. More reliable solutions may require optimization of standards to support changing UUIDs. In this case, this field must be altered in an unpredictable fashion, probably relying on cryptographic operations (e.g., hash or encryption). The latter case raises concerns regarding the increase in the battery exhaustion rate as well as the overall cost of the hardware.

Leakage of Sensitive User Information

From health data, to payment details, or arbitrary sensor information that can potentially reveal user habits and preferences, many IoT applications deal with sensitive user data. Hence, one of the most far-reaching security threats is the leakage of sensitive information, a vulnerability which is identified as one of the most common in the IoT ecosystem by OWASP (OWASP, 2014).

In this context, applications that collect redundant data, or data not directly relevant to the application's purpose are common. There might be several reasons behind this phenomenon: A) One possibility is that application developers are overestimating the requirements of future, improved versions of the application. In this case, information may leak towards the application vendor and anyone with access to its backend (possibly including malicious users). B) In other situations, applications do not properly communicate with their users what type of sensitive information is collected and frequently they do not provide the possibility to opt-out. This may happen because the vendor is interested in reselling the data and is seeking to gather as much information as possible. In this case, the leaked information is not limited to the application vendors, but is also available to any party that may acquire it. C) Finally, there are cases where the flow of sensitive information occurs as an outcome of bad protection practices during the transmission (e.g., no use of TLS) of data. In this case, anyone who has the capability of eavesdropping on the communication may gain access to information exchanged.

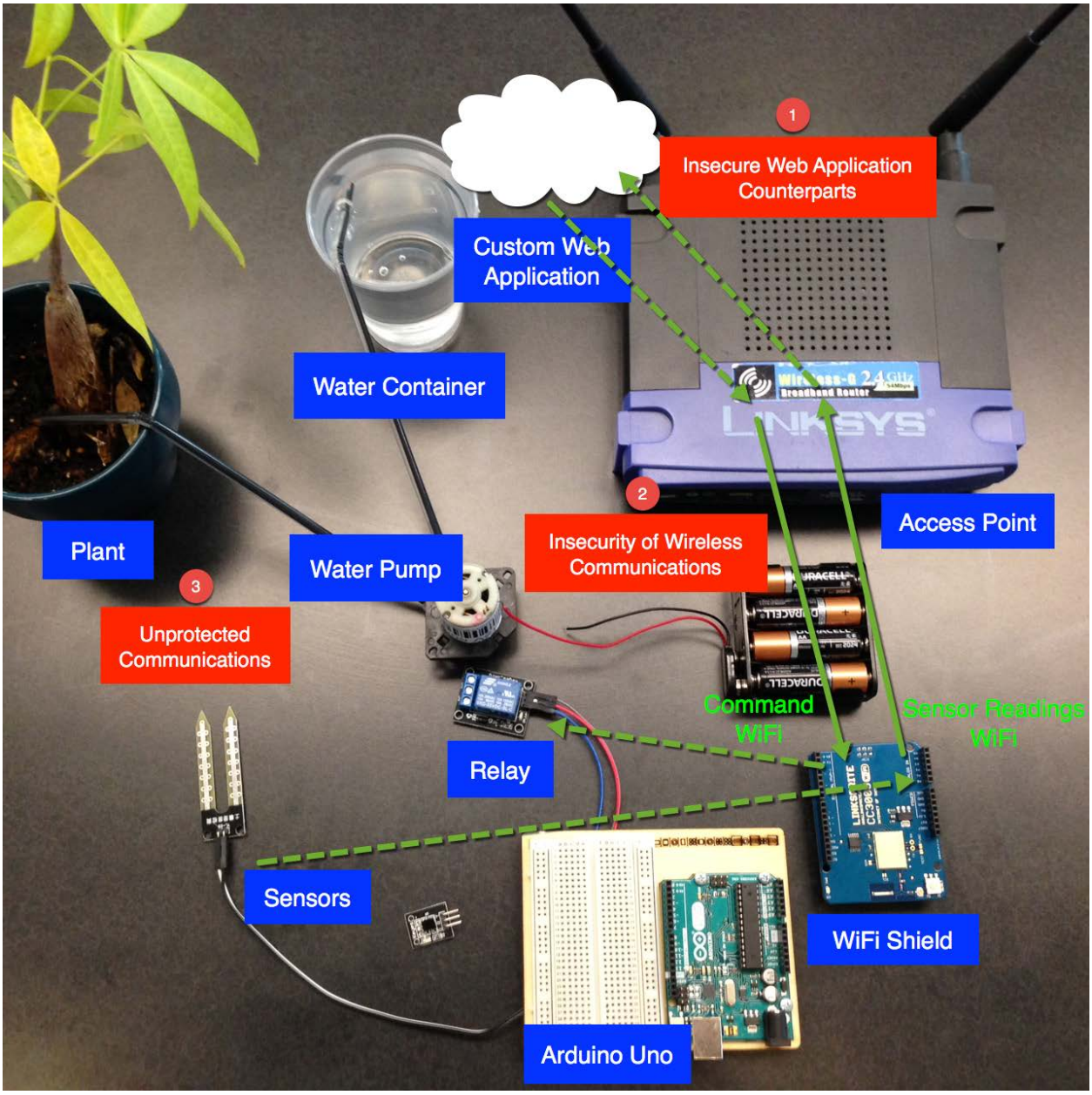

Figure 2: User Case of Remote Watering System

Remote Watering System

To demonstrate the inadvertent transmission of sensitive IoT data, we put together a smart watering system. Conventional watering systems rely on clock settings to automate the watering process in gardens or flower pots that lay indoors. Such systems require manual reconfiguration when changes in the environment occur including temperature and moisture (rain). Using open-source hardware IoT components, we can easily build an advanced watering system that can provide live feeds of environmental readings including moisture of the ground,

temperature, and luminosity and at the same time allow remote control of the watering system overall or some level of automation.

This use case requires a component that provides environmental readings, a module that implements user decisions and a unit that connects the user with the rest of the system. We relied on a single board computer to execute all the sensing and actuating functionality while the business logic and user interface were provided through a web application.

The single board computer 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 the C/C++ language and a set of open source software libraries that aid common operations with these circuits. In this example, the board is the main component of the system. It is connected to moisture, temperature, and luminosity sensors that constantly provide readings to the web application. At the same time, it is connected to a relay that connects to an external power source and a water valve. Finally, it is attached to a WiFi shield through which it connects to a home wireless network. All communication takes place through WiFi. The board is programmed to execute HTTP Post to the web application in pre-specified intervals to insert new readings. To stay up-to-date regarding the the status of the water valve (i.e., on/off) it polls the web application. The web application receives the readings from an authorized device and stores them in a backend database. It also allows authenticated users to receive live feeds regarding the conditions, as well as to alter the status of the water valve. The main components of this example are depicted in Figure 2.

Points of Failure of the Watering System

In the simple example previously described, one can identify multiple points that may result in leakage of data or other undesirable results. The most important are:

1. ***Insecure web application counterparts*** – Web applications interact with their users through dedicated UIs. In many cases, invalidated user input inserted into an entry field of the UI may contain special sequences that form malicious code in the application layer. This can lead to XSS and SQL injection attacks. Such attacks whether successful, can result into annoyance for the end-user (which leads into reduced revenues) and possibly compromise the privacy of its clients. Generally, failure to enforce a set of good security practices during the application development cycle e.g., the adoption of strong credentials, may lead to compromise of accounts and unauthorized access by malicious users.
2. ***Insecurity of wireless communications*** – From the previous use case it was made clear that many wireless protocols have been identified with an array of vulnerabilities. Besides the trivial DoS attacks which in this case may be significant (consider the case where the system gets attacked while it is watering), MiM attacks are relatively easy to accomplish. For example, an attacker can come into the vicinity of the valid network, with a soft-AP bearing the same 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 (these are unprotected messages, defined by the

802.11 specification and as such they are easy to spoof). At that point, all devices will attempt to reconnect to the AP that advertises their known SSID and has the strongest signal (i.e., the attacker's AP). Advanced OSs may be able to evade this trap but the less feature-rich OSs of many IoT devices will not understand the difference and will connect anyway. In that way, the attacker will be able to eavesdrop on all unencrypted traffic of any device connected to it, compromising any privacy if security is not enforced in a higher layer also.

3. ***Unprotected communications*** – While protection of the communications is a de-facto choice in the desktop environments, it is not always practical with IoT. The primary reasons revolve around the increased cost of hardware utilized versus the application type. The issues discussed previously suggest that the adoption of additional message protection mechanisms on higher layers is a necessity. Below, we elaborate on the reasons behind this limitation.

Lack of encrypted communications

There are many reasons why IoT application data may be transmitted in plaintext. One common cause is poor design decisions that treat only the most obviously private user information as sensitive. In a home automation system, at first glance sensor data like temperature readings might not be considered sensitive. However, an eavesdropper that is able to monitor such readings temporarily might be able to infer whether a user is at home by tracking sudden changes of temperature or significant deviation from the outside conditions (i.e. the user might have started the air conditioner).

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

As a result, system designers are left with two choices: (a) create their own lightweight security protocols from scratch or (b) implement modified, stripped-down versions of well-known security protocols. The first choice includes the danger that the newly created mechanism is proven to be vulnerable in practice and the associated development costs are significant. The second choice, on the other hand, carries the likelihood of a security vulnerability. Thus, custom security schemes or hardware-adapted implementations of protocols may result in data being transmitted without meaningful protection. For example, in this use case someone might decide to implement a custom version of the TLS protocol where the computationally heavy certificate verification step would be intentionally left out. Evidence suggests (Ardiri, 2014) that this modified protocol would run efficiently even on small single board computers, however this creates an opportunity for trivial MiM attacks. Actually, this last case is the most deceptive of the three, as it gives the illusion of an industry standard grade protection to its users, without really providing it.

Plugging the Leaks

A dedicated computer forwarding all traffic generated from sensors to the corresponding web application and vice-versa, may be able to address the security requirements of the previously described architecture when it comes to securing the communications. This “proxy” device has the advantage of being able to communicate with different types of sensors through shorter range wireless protocols (e.g, 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 channels of communication and then rely on the proxy device to protect the traffic through SSL/TLS when it is transmitted to the Internet. Another advantage of this architecture is that it may scale without a significant increase in cost, and can support multiple communication protocols from different types of sensors without extensive re-configuration of the system.

Unauthorized Execution of Functions

Usability is a defining factor for the success of any application, yet it is often viewed as an opposing force to security. Frequently certain security compromises or strong assumptions (e.g., the application is functioning within a secure network) must be made for a product to meet the user requirements in the market.

Today, there are many opportunities for an attacker to infiltrate a network: malware may evade anti-virus checks; mobile apps with backdoors may allow remote execution of code; vulnerable services running on clients may allow buffer overflow attacks; unnecessary open ports may welcome unauthenticated malicious entities. Given these risks and the increasing complexity of modern networks, it is unrealistic to assume reliability and trustworthiness of the exposed clients. Actually, a compromised client inside the network is often used as a stepping stone for an unauthorized entity from outside the network to issue commands that affect the status of local devices.

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

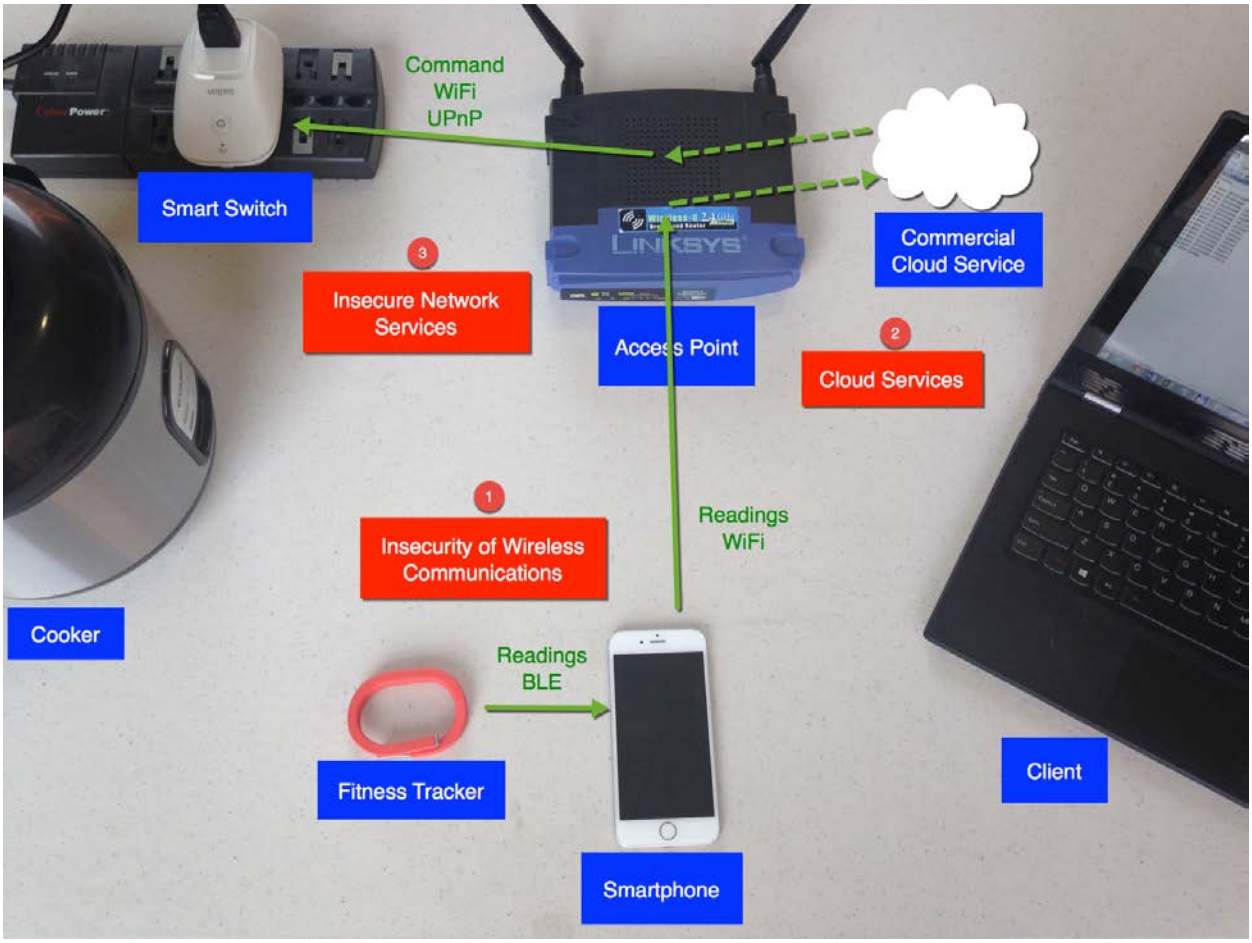

Figure 3: Use case of Powering Devices

Automatic Control of Devices

For this use case, we put together an IoT system in which potentially dangerous appliances (e.g., cooker) can be powered off automatically upon detection that a user has fallen asleep. This scenario is similar to the scenario of starting via power on, of a coffee maker in the morning. This example requires a device that monitors when a user has fallen asleep, a switch that can turn on/off conventional appliances and an application which will bring together these two devices. In our implementation, we relied on a) a fitness tracker, b) a smart switch and c) well known cloud service.

The commercial cloud service utilized permits the creation of applications that trigger specific actions upon the occurrence of specific events, using Web Services API calls.

The fitness tracker is a wearable device that monitors the calories burned, sleep habits, and fitness of the wearer by measuring the steps they make and their heartbeat. Typically, it has to be paired with the corresponding app installed on a smartphone. The two devices communicate in intervals to transmit data from the fitness tracker to the smartphone. Then the smartphone acts as a proxy by forwarding the data to the corresponding web service via Wi-Fi

or 3G/LTE connection. Additionally, it presents the statistics in a comprehensive User Interface (UI). In our case the triggering part of the cloud application is the detection of sleep by the fitness tracker.

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 that has the corresponding app installed. Then, the user can control the device remotely from the app's UI. When the user resides inside the home network, the communication is done directly through Universal Plug and Play (UPnP) commands transmitted through WiFi. When outside the range of the home network, the device issues an HTTPS request to the corresponding web service through 3G/LTE and then that service activates/de-activates the switch. The action part of the cloud application is the change of status of the smart switch. Figure 3 presents all the components used in this use case.

Vulnerabilities of the Architecture

In the use case described previously there are opportunities for an attacker to influence the control of the device with potentially life threatening results. The most important are:

1. ***Insecurity of wireless communications*** – Once more, an assailant can take advantage of the weaknesses in the Wi-Fi protocol to constantly de-authenticate the device. In our sample case if the status of the device is "off" during the moment of the attack, then the outcome will most likely be simple annoyance. On the other hand, if it's "on", the attack may prevent the device from turning off.
2. ***Cloud services*** – Some off-the-shelf IoT products use cloud services for storage and command and control. Typically, when the user purchases the product they implicitly put trust in their cloud counterparts too. 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), generated the need for an external cloud service that would assume the role of the middle-man. Hence, trust must also be placed into yet another entity, expanding the circle of trust even further. As this circle of trust grows bigger, so does the risk that one of these services may follow less strict security practices. A breach of one service might result in loss of sensitive user information (health data, sleep habits) and possibly remote control of home appliances.
3. ***Insecure network services*** – Our use case analysis indicated that some commercial IoT products still adopt insecure network services and protocols to facilitate their seamless setup. One of the most popular protocols of this kind is the UPnP. This fact can introduce the risk of control of the switch by unauthorized insiders, as described next.

Insecure Protocols Running on the Network

UPnP is a set of network protocols standardized by Microsoft in 1999. It provides a convenient way to introduce new devices on the network, facilitate their discovery by other devices and permit their control. UPnP requires that routers have the corresponding feature enabled. It has been widely used in VoIP (e.g., Skype), P2P (e.g., uTorrent), and gaming applications. One of

the most serious security issues of UPnP lies in the fact that it places trust in clients of the network. It does not encrypt data, nor does it require users to authenticate themselves before triggering the execution of functions, such as search for UPnP enabled devices, querying for their supported capabilities and activating them. In a sense, any client who is part of the network and has a local IP is considered trusted.

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

Conclusions and Prospects for the Future

While IoT testing has received relatively little attention, security and privacy through assurance is a central concern as systems proliferate and become connected 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 (Rosenkranz, Wahlisch, &Ortmann, 2015), but for testing, the most significant characteristic of IoT systems is the sheer variety of devices and means of communication. To make matters worse, variations in processor speed, memory, protocols, and types of application are much bigger with IoT than with traditional desktops and laptops, or even smart phones. This inherent heterogeneity requires testing on 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 (Rosenkranz, Wahlisch, &Ortmann, 2015), (Lahmadi, Brandin, & Festoe, 2012) are becoming available, providing a full complement of shared testing resources to reduce cost and testing time. 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 a wide variety of IoT devices.

IoT testing is further complicated by the increased number of devices that must be able to interact. 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 among IoT devices will involve more than just two exchanging data, so huge numbers of combinations must be considered for interoperability testing. Fortunately, combinatorial test methods from the statistical field of Design of Experiments make it possible to compress huge numbers of configuration settings or input variable values into a small number of tests. These methods can be highly effective for IoT (Patil, Goveas, & Rangaran, 2015), (Dhadyalla, Kumari, & Snell, 2014) where component interactions are especially critical. Another approach to making assurance more tractable with the extreme heterogeneity of IoT is to use models based on interoperability patterns (Grace, Barbosa, Pickering, & Surridge, 2014). This form of model-based testing can take advantage of similarities in architecture and behavior for systems in different IoT application domains. Despite these advances, a significant research

challenge remains in finding ways to provide appropriate levels of assurance for the incredibly diverse IoT domain.

Our foray into putting together commercial off-the-shelf IoT systems using inexpensive and readily available modules to create appealing, everyday use cases suggests that IoT security and privacy are sometimes not well defined and understood by both consumers and manufacturers. Our use cases demonstrate that (a) some IoT implementations can lead to inadvertent tracking of user identity and behavior because if data are not always classified as sensitive and if devices are not built with privacy as a design tenet. (b) Data encryption may not always be enabled, and even when it is, the cryptographic libraries used may exhibit security flaws that expose the data. Finally, (c) some IoT systems suffer from the isolation syndrome of embedded devices: weak protocols and practices are sometimes used because some of the 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 the mistakes of the past and elevate security and privacy as additional design tenets. There are relatively few standards or best practices to guide the security design and testing of IoT technologies, although some related guidance is being developed by the Cyber Physical Systems Public Working Group (CPSPWG, 2015) and in documents such as the Guidelines for Smart Grid Cyber Security (NIST, 2010). We believe that now is the time for standards bodies and industry experts to begin to formulate suitable guidance and to work towards identifying the right security and privacy primitives. We are only at the beginning of the security and privacy requirements for IoT technologies, with many open research challenges that only grow as IoT applications become part of our everyday lives. Indeed, it has been more than 15 years since the introduction of this new research field and still there are no thorough studies to identify IoT specific vulnerabilities and outline their differences with the non-IoT ones. Although there have been studies of individual IoT technologies over the last few years, there is still a lot to be done to fully describe the behavior of different IoT systems when under attack.

Disclaimer: *Certain products are identified in this document in order to describe work conducted, but identification does not imply recommendation or endorsement by NIST, nor that the products identified are necessarily the best available for the purpose.*

References

ABIresearch. (2013, February). *Indoor Location in Retail: Where Is the Money?* Retrieved October 25, 2015, from ABIresearch: https://www.abiresearch.com/market-research/product/1013925-indoor-location-in-retail-where-is-the-mon/

Apple. (2014, June 2). *Getting Started with iBeacon.* Retrieved October 21, 2015, from iBeacon for Developers: https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf

Ardiri, A. (2014, September 27). *Is it possible to secure micro-controllers used within IoT?* Retrieved October 26, 2015, from evothings.com: https://evothings.com/is-it-possible-to-secure-micro-controllers-used-within-iot/

Dhanjani, N. (2015). *Abusing the Internet of Things: Blackouts, Freakouts, and Stakeouts.* O'Reilly Media, Inc.
Kim, E., Kaspar, D., & Vasseur, J. (2012, April). *Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs).* Retrieved October 25, 2015, from IETF: https://tools.ietf.org/html/rfc6568
Kolias, C., Kambourakis, G., Stavrou, A., & Gritzalis, S. (2015). Intrusion Detection in 802.11 Networks: Empirical Evaluation of Threats and a Public Dataset. *Communications Surveys & Tutorials , PP* (99), 1.
Lester, S. (2015, May 21). *The Emergence of Bluetooth Low Energy.* Retrieved October 23, 2015, from Contenxt: http://www.contextis.com/resources/blog/emergence-bluetooth-low-energy/
OWASP. (2014). *Internet of Things Top Ten.* Retrieved 11 11, 2015, from https://www.owasp.org/images/7/71/Internet_of_Things_Top_Ten_2014-OWASP.pdf
Pescatore, J. (2014). *Securing the Internet of Things Survey.* Bethesda: SANS Institute.
Rivera, J. (2014, November 11). *Gartner Says 4.9 Billion Connected "Things" Will Be in Use in 2015.* Retrieved October 21, 2015, from Gartner: http://www.gartner.com/newsroom/id/2905717
Shelby, Z., Hartke, K., & Bormann, C. (2014, June). *The Constrained Application Protocol (CoAP) Specification.* Retrieved October 25, 2015, from IETF: https://tools.ietf.org/html/rfc7252
Teng, X., Wendt, J., & Potkonjak, M. (2014). Security of IoT systems: Design challenges and opportunities. *2014 IEEE/ACM International Conference on Computer-Aided Design* (pp. 417-423). San Jose: IEEE Press.
Yuksel, E., Nielson, H., & Nielson, F. (2008). Zigbee-2007 security essentials. *Proc. 13th Nordic Workshop on Secure IT-systems*, (pp. 65-82).
ZigBee. (2012, September 7). *ZigBee Pro Standard.* Retrieved October 21, 2015, from ZigBee: http://www.zigbee.org/non-menu-pages/zigbee-pro-download/

NIST. (2010). Guidelines for smart grid cyber security (vol. 1 to 3). *NIST IR-7628*.

Rosenkranz, P., Wählisch, M., Baccelli, E., & Ortmann, L. (2015, May). A Distributed Test System Architecture for Open-source IoT Software. In *Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems* (pp. 43-48). ACM.

Lahmadi, A., Brandin, C., & Festor, O. (2012, May). A testing framework for discovering vulnerabilities in 6LoWPAN networks. In *Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference on* (pp. 335-340). IEEE.

Fernandes, J., Nati, M., Loumis, N. S., Nikoletseas, S., Raptis, T. P., Krco, S., ... & Ziegler, S. (2015, April). IoT Lab: Towards co-design and IoT solution testing using the crowd. In *Recent Advances in Internet of Things (RIoT), 2015 International Conference on* (pp. 1-6). IEEE.

Patil, A. H., Goveas, N., & Rangarajan, K. (2015). Test Suite Design Methodology Using Combinatorial Approach for Internet of Things Operating Systems. *Journal of Software Engineering and Applications*, *8*(07), 303.

Dhadyalla, G., Kumari, N., & Snell, T. (2014, March). Combinatorial testing for an automotive hybrid electric vehicle control system: a case study. In *Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on* (pp. 51-57). IEEE.

Grace, P., Barbosa, J., Pickering, B., & Surridge, M. (2014, December). Taming the interoperability challenges of complex IoT systems. In *Proceedings of the 1st ACM Workshop on Middleware for Context-Aware Applications in the IoT* (pp. 1-6). ACM.

CPSPWG. Cyber Physical Systems Public Working Group. http://www.nist.gov/cps/cps-pwg-workshop.cfm