Draft

Technical Report TR1: ISIF ASIA Funded Project

Study of Security Attacks against IoT Infrastructures

Prof Vijay Varadharajan, Dr Uday Tupakula and Kallol Karmakar

Advanced Cyber Security Engineering Research Centre (ACSRC)
Faculty of Engineering and Built Environment
The University of Newcastle

May 2018

Acknowledgements:

The authors would like to thank ISIF ASIA for their financial contribution to the Project

Executive Summary

The report presents project summary on the "Software Defined Networks based Security Architecture for IoT Infrastructures" project work done during January 2018 – May 2018 funded by the ISIF ASIA. There are three milestones for the project with specific deliverables for each milestone. In this report we present a summary of the outcome of first milestone "Report on Security Attacks on IoT Systems" which consists of the following tasks: i) Detailed survey of attacks related to IoT, ii) Analysis of previously proposed techniques and countermeasures and iii) An outline of security requirements to deal with attacks in IoT.

Any physical or virtual device having network connectivity to communicate with the outside world and other devices is termed the Internet of Things (IoT). This connectivity provides flexibility to control devices remotely and in some cases ease in data acquisition from the physical sensor devices. There has been a rapid growth in the Internet of Things with an ever-increasing number of physical devices being connected to the Internet at an unprecedented rate, with recent forecasts suggesting the number of IoT devices to reach 50 billion in 2020. The number and diversity of IoT devices has been growing rapidly; these devices offer many new applications for end users, and will offer even more in the future covering a range of vertical application areas including Smart Homes and Cities, Intelligent Public Transport, Smart Grids, Smart Cars, Smart Airports, eHealth and Smart Hospitals. Although IoT has huge potential, such an IoT environment with heterogeneous devices with different operating systems and connectivity ranging from Zigbee to wireless to mobile networks poses significant challenges in security and privacy.

A major security challenge is a dramatic increase in the attack surface with new vulnerabilities and threats, as many IoT devices do not have any security functionality, and even the ones that have are often primitive and hence can be easily attacked. IoT devices can become the entry points into smart homes and organisation networks including critical infrastructures. IoT devices are small, have limited energy source and capabilities, and less powerful with respect to processing of information. Hence, an attacker is able to easily target these things, by sending fake requests, intercepting/ manipulating valuable sensor data in transit or capturing a physical device and transforming it to zombies to launch attacks. Furthermore, often it is not possible to enhance security of these devices using traditional defense mechanisms as they are located in open premises and they incur extra processing load to small IoT devices. As part of this milestone, we have conducted a detail investigation of the attacks against IoT infrastructures.

In this report, we will first present an overview of the IoT Architecture with discussion on the different layers of the architecture and protocols that are used at different layers of the IoT Architecture. Then we present threat model for the IoT Architecture with discussion on the several attacks that are possible at different layers of the IoT Architecture. We also propose the design and development of feature distributed malware attacks for Internet of Things in the context of smart homes. Smart homes consist of various smart home devices, providing convenient services to the tenants. These devices are usually connected to the Internet, and the tenants can integrate individual devices and make smart home workflows using Internet services such as IFTTT. The focus of our work is on how the attackers can generate various cyber-physical and advanced cyber-attacks by exploiting this integration aspect in smart home. In particular, we have designed and developed an advanced feature distributed malware that can be used to perform various malicious activities such as stealing the victim's IFTTT cookies, using smart devices as information sources of the malware and distributing malware functionalities to other devices, performing malicious without being noticed, causing financial damages to the victim, and evasively exfiltrating data. Our experiments show that the proposed attacks enable the attackers to control any smart home device integrated with IFTTT, without requiring compromise of individual smart home devices. We believe our approach of using the integrating Internet service as an attack vector and distributing malware features to the smart home devices will be helpful to the development of more secure smart home environments and different IoT applications. Finally, we have analysed some of the previously proposed techniques to deal with the attacks in IoT applications and identified few security requirements that need to be considered for designing and for developing security architecture for IoT applications.

The authors would like to thank ISIF ASIA for their financial contribution to the Project.

Contents

1. Introduction

There has been tremendous growth in the use of the Internet. At the same time the exponential growth in the number of smart devices (e.g. smart phones, tablets, intelligent circuits, sensors and actuators) makes it more convenient to use Internet-based applications than ever before [1]. This results in a situation where 'everyone' (e.g. people) and 'everything' (e.g. systems, machines, equipment and devices) connect to create an expanded digital environment. This paradigm shift is referred to as the 'Internet of Things' (IoT) [2].

Many IoT solutions are becoming available or are being developed for deployment in the near future, including sensors to better understand patterns of daily life and monitor health, smart controls for home functions, from locks to heating and water systems, devices and appliances that anticipate a consumer's needs and can take action to address them (such as devices that monitor inventory and automatically reorder products for a consumer). In addition, when coupled with data analysis and machine learning, IoT devices may be able to take more proactive actions, expose interesting data patterns, identify anomalies or make suggestions to end users that may improve their health, environment, finances, and other aspects of their lives.

The IoT is not the result of a single novel technology, instead, several complementary technical developments provide capabilities, that added together, help to bridge the gap between the virtual and physical world. The IoT can be seen as a transformation rather than an evolutionary technological advancement for both traditional and non-traditional application domains [3].

Typically, each IoT device consists of sensors, actuators, communication infrastructure and a processing unit with some software (firmware). A common role of an IoT device is to sense data and send it to a remote location or to receive data and perform some limited actions on it. These IoT devices can be small and resource constrained as well as embedded in other real world objects. According to an industry estimate, the number of IoT devices is expected to reach 50 billion by 2020 [4]. In IoT applications, the behaviour of the devices are controlled by the applications. In recent times, we have witnessed the emergence of new and innovative IoT applications ranging from precision agriculture, environment monitoring, health monitoring, to traffic management. A large number of smart applications are being built using IoT devices. We take an example of IoT enabled solar farm to explain further.

In case of an IoT enabled Solar farm one can control the tilt for a group of big solar panels situated in a remote location based on the sun's position at a particular time. IoT provides an infrastructure for such variety of applications to perform acquisition of the physical sensor data; for instance, temperature, humidity, time, angle, and position logs from the solar panel. These sensed data are transmitted and stored in a cloud server for future analysis. Such stored data can be made available to other users or applications. For example, a power distributing company can analyse the solar panel logs and find the optimum panel angle respective to sun's position throughout the day. The panels can be remotely programmed to rotate in a particular manner to provide maximum efficiency [5].

Although IoT has huge potential, it comes with a number of key security challenges [6-9]. As has been the case with the Internet, security has become an after-thought, and many IoT devices and applications are already in operation without any security mechanism. The heterogeneity of the IoT devices, underlying communication infrastructure and the different types of protocols used by these devices increase the complexity of the IoT infrastructure and make it vulnerable to security attacks. Due to the resource constrained nature of IoT devices, an attacker is able to target them easily by

sending forged requests, intercepting and illegally manipulating valuable sensor data in transit or capturing a physical device and transforming it to a zombie to launch attacks on other systems. Denial of service (DoS) and energy depletion attacks are the most common IoT attacks [8, 9]. Often it is not possible to implement security on these devices using traditional defence mechanisms as they are located in open environments and they would incur extra computational load on small IoT devices.

As part of this milestone, we have conducted a detail investigation on the different types of attacks against the IoT infrastructures. The report is organised as follows. Chapter 2 provides a brief overview of the IoT architecture and a threat model for the IoT architecture. We will present different types of attacks that are possible at different layers of the IoT architecture. In Chapter 3, we propose the design and development of feature distributed malware attacks for Internet of Things in the context of smart homes. In Chapter 4, we present analysis of previously proposed techniques to deal with the attacks in IoT applications. In Chapter 5, we discuss some of the security requirements that need to be considered for designing and developing security architecture for IoT Applications. Chapter 6 presents some concluding remarks.

2. IoT Architecture and Threat Model

Although IoT has huge potential, such an IoT environment with heterogeneous devices with different operating systems and connectivity ranging from Zigbee to wireless to mobile networks poses significant challenges in security and privacy. There is a dramatic increase in the attack surface with new vulnerabilities and threats, as many IoT devices do not have any security functionality, and even the ones that have are often primitive and hence can be easily attacked. IoT devices can become the entry points into smart homes and organisation networks including critical infrastructures. IoT devices are small, have limited energy source and capabilities, and less powerful with respect to processing of information. Hence, an attacker is able to easily target these things, by sending fake requests, intercepting/ manipulating valuable sensor data in transit or capturing a physical device and transforming it to zombies to launch attacks.

In this chapter, we present an overview of generic IoT Architecture with a discussion on different layers and protocols that are used in different layers of the architecture. Then we present the threat model describing attacks that are possible at different layers of the IoT Architecture.

2.1 IoT Architecture Overview

IoT architecture consists of four basic layers: Perception Layer, Network Layer, Middleware Layer, and Application Layer [10] . Table 1 shows the components of each layer and their tasks in details.

Table1: IoT layer details

Layer	Components	Tasks
Application Layer	Third Party Application, Websites, Consoles, and Touch panel.	Machine Learning, Business Models, Graphs, and Flow charts.
Service Layer	Vendor Specific Third Party Application	Machine Learning, Processing, Pre-processing, and Real time action.
Network Layer	Nodes, Gateways, Firmwares	Device Management, Processing and Secure Routing
Perception Layer	Sensors (Temperature, and Humidity) and Actuators (Motor, and Relays)	Identify, Monitor, Acquisition, and Action.

2.1.1 Perception Layer

This layer consists of devices like actuators and sensors. These devices help to control and monitor any physical environment. For instance, sensors are used to sense temperature, motion, vibration, acceleration, humidity, air quality, and physical location. On the other hand, actuators are used to control the activity of physical devices, like controlling the acceleration of a car, air cooler, fan, and controlling the power grid activity. Thus, this layer is also known as "Device Layer." One of the major responsibilities of Device Layer is the identification of the sensors and actuators. These sensors generate an enormous amount of data. For further processing and secure routing, these device data are transferred to the Network Layer. IoT infrastructure must be capable of handling such huge amount of data.

2.1.2 Network Layer

Network Layer processes and securely routes or transmits the data throughout the IoT infrastructure. It is also known as "Transmit Layer." Device data can be transmitted in wired or wireless manner. Depending on the vendor requirements and IoT infrastructure limitations, different protocols like Zigbee, Bluetooth, IR, and 6LowPan are used for data transmission. For further processing and action, Network Layer depends on the Middleware Layer.

2.1.3 Middleware Layer

This layer manages the vendor specific services for different IoT device data. It acts as a bridge between the Network Layer and Application Layer. This bridge helps to process, pre-process, and store IoT device data based on the vendor and device requirement. For instance, a real-time response from the actuators based on the sensor data; if the radioactive sensor reading crosses certain threshold, services deployed by the sensor vendor in the Middleware Layer automatically triggers an alarm and closes certain doors (activates motor function). This process involves storage of a massive amount of sensor data as well as real-time monitoring of sensor data. This layer prepares the data for further processing in Application Layer. One of the popular IoT middleware is OpenIoT [11].

2.1.4 Application Layer

In the IoT infrastructure, the top most layer is the Application Layer. This layer operates on the processed data provided by the Middleware Layer. These operations are IoT device vendor specific. A collection of the operations or processes is known as an application. The smart behaviours of IoT devices are powered by the smartness of these apps. Apps are designed smartly to handle individual IoT device action. For instance, a health monitor app running on a smartphone, which constantly monitors the pacemaker implemented in one of the patients. The pacemaker is synced to the smartphone. Whenever the pacemaker heart rate fluctuates for more than a minute, it triggers a notification to the GP as well as to the person. This notification setting can be changed using the pacemaker manufacturer app. Other examples of application level automation are smart city, smart home, and smart car.

Some researchers have shown an extension named "Business Layer" to "Application Layer". This layer deals with the representation of the IoT data. Business models, flowcharts, and graphs are used to represent the received application data. To benefit the service provider, business models can be implemented to the data for optimized output [12]. For instance, Smart Solar panels can provide temperature data at different angles. Based on these angles, where the temperature is high at which time of the day, the app can automatically rotate the panel. This smart adjustment of the panel will incur maximum sunlight and thus maximize the power generation.

2.2 Different IoT Protocols

In this section, we present protocols used in different layers of the IoT Architecture.

Layer Name	**Protocol Used**
Application Layer	**HTTP, CoAP, DDS, AMQP, MQTT, MQTT-SN, XMPP, HTTP REST**
Network Layer	**MDNS, DNS-SD, RPL, 6LoWPAN, IPv4/IPv6**
Perception Layer	**LTE-A, EPCGlobal, IEEE 802.15.4, Z-Wave.**

Perception Layer: The primary driving protocol for this layer is Low-rate wireless personal area networks (IEEE 802.15.4). IEEE 802.15.4 focuses on providing low-cost, low-speed ubiquitous communication between IoT devices. The basic framework can contribute to a transfer rate of 250kbit/s over a communication range of 10 meters.

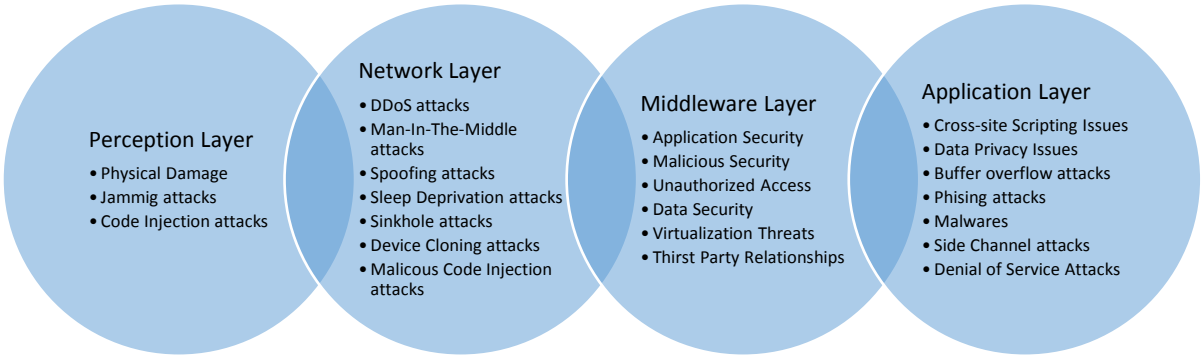
Network Layer: The two most important protocols in Network Layer is 6LowPAN and RPL. 6LowPAN uses IPv6 over low power Wireless Personal Area Networks. It requires both destination and intermediate addresses for mesh routing. On the other hand, RPL (Routing over Low Power) uses very low power for routing. It is designed for sensor data collection and unlike 6LowPAN, RPLs major focus is routing.

Application Layer: Message Queue Telemetry Transport (MQTT) is a lightweight protocol developed by IBM and standardised by OASIS for M2M devices in 2013. This protocol is the most dominating protocol for IoT devices in the application layer. It uses TCP which causes a high protocol overhead. On the other hand, MQTT_SN (MQTT_ Sensor Networks) uses UDP, hence it provides less protocol overhead.

AMQP (Advanced Message Queuing Protocol) uses TCP/IP. It is point to point and uses publish-subscribe model. COAP (Constrained Application Protocol) is designed for UDP. It uses request-response model similar to HTTP.

2.3 IoT Threat Model

In this section we present discussion on the attacks that are possible at different layers of the IoT Architecture.



2.3.1 Attacks in Perception Layer:

A. **Physical Damage:** Most of the cases IoT devices are placed in open premises. Thus enabling an adversary to physically damage the device. Sometimes an adversary can use this device to tamper firmware code and make it a Zombie or it can be used by the adversary to log in into the IoT gateway. After getting access to the IoT gateway it can modify and capture both device traffic and instructions from the users.

B. **Jamming attacks:** IoT devices use wireless communication to program (the devices), get instructions (from the user), or to transmit data to the cloud storage. An adversary can use similar communication frequency of the IoT device to Jam the communication between IoT devices and gateways. This is a very common attack and was previously used in mobile frequency jamming and RF frequency jamming for WSN nodes.

C. Code Injection attacks: An adversary sitting close or having physical access to the IoT devices can tamper the firmware by injecting malicious codes into the devices. With this approach, the adversary can target a specific device only or can target to compromise the whole IoT communication network.

2.3.2 Network Layer:

A. **DDoS attacks:** In the IoT infrastructure, things are vulnerable. An adversary captures the things credentials and gains access to the node/gateways. Typically, gateways preserve thing (IoT) information in its local database. An adversary can use this thing intelligence to exploit the other devices. Finally, it can launch a DoS attack by sending dummy packets and disrupting IoT network communication.

B. **Man-In-The-Middle attacks:** With the current IoT architecture, it is possible for an adversary to launch a Man-In-The-Middle attack. Other conventional attacks like eavesdropping, damaging the confidentiality and integrity of the data is also possible.

C. **Spoofing attacks:** Current IoT infrastructure is not secure. Things can be connected to the network either directly or via a node or gateway. In both cases, physical security of the things, as well as the node, is necessary. An adversary can physically compromise things and node/ gateways. It can then be replaced/reprogrammed with a malicious thing or node. The adversary can also capture specific credentials from the stolen things and start spoofing attacks from some other devices. Similarly, if an adversary learns about some of the details of nodes/gateways, he can launch spoofing attacks. To avoid this, things and nodes/gateways must be authenticated.

D. **Sleep Deprivation attacks:** Each IoT device/ Wireless Sensor nodes are packed with batteries. In general, their battery size is limited. To conserve power resources, IoT devices drive into sleep mode while they are not functioning. An adversary can compromise an IoT device and keep itself busy to drain the batteries. This type of attack is known as Sleep Deprivation attack.

E. **Sinkhole attacks:** In this attack, an adversary creates sinkholes to attract traffic from all the IoT devices. This attack is harmful to privacy and confidentiality of IoT devices. The device traffic can later be diverted to any other devices than the destination devices.

F. **Device Cloning attacks:** An adversary having physical or wireless access to the IoT devices can create an exact clone of the devices. Later, they can compromise the IoT network infrastructure using these devices.

G. **Malicious Code Injection attacks:** In this attack, adversary uses compromised IoT devices to inject malicious codes into the IoT network infrastructure. Once they are compromised, they can control the whole IoT network domain.

2.3.3 Middleware Layer:

A. **Application Security:** IoT devices act as a data acquisition medium. After the acquisition, they transfer the data to the cloud for further processing and storage. An adversary can easily

intercept, steal or manipulate the data either in the cloud or the processing IoT gateways. This is a software issue linked with application services running in the cloud or the IoT gateways. A lot of application security information issues are pointed out by Open Web Application Security Project (OWASP) in Software as a Service (SAAS).

B. **Malicious Insider:** An adversary can be a malicious insider hiding inside the IoT cloud service provider. In this case, an adversary might have easy access to IoT device data and can manipulate it for his benefit. This type of attacks is tough to detect.

C. **Unauthorized Access:** IoT networks are relatively weak when it comes to enforcement of security. An adversary can easily use open ports and back doors to get access to the IoT network infrastructure. Once they are inside the network, they can launch different type of attacks.

D. **Data Security:** IoT device stores the data in the cloud. As mentioned previously, cloud data security is mostly dependant on SaaS. Also in most of the cases third-parties backup the data from IoT clouds. There is a huge possibility that data can be compromised during the backup phases.

E. **Virtualization Threats:** IoT infrastructure can be built on top of a virtualised network, and the cloud servers are motly deployed as virtual servers. Both virtualised network/ the cloud servers pose different kinds of threats. Thus, making the IoT infrastructure vulnerable.

F. **Third Party Relationships:** Different types of third-party components/platforms are used to build the IoT infrastructure. If these third-party platforms are not secure, they can be easily compromised by an adversary.

2.3.4 Application Layer:

A. **Cross-site Scripting attack:** Every IoT infrastructure comes with a central control dashboard or a mobile application based control/ display interface. These interfaces help the user to control the IoT devices. If these interfaces or mobile applications are not properly sterilized, an adversary can launch cross-site scripting attacks.

B. **Data Privacy issues:** IoT sensors generate a massive amount of data. These data, as well as the policies (Access control and Intrusion Prevention) for IoT devices, are stored in the same database. It is possible for an adversary to compromise the privacy of the database.

C. **Buffer Overflow Attacks:** Application subsystems are always vulnerable to buffer overflow attack. An adversary can use this to trigger malicious codes. Finally, the whole IoT infrastructure can be compromised.

D. **Phishing attacks:** This attack targets on stealing the user login credentials for any IoT interfaces. Here an adversary advertises fake or spoofing links to attract legitimate users. Once they provide the login credentials to the fake interface, the adversary steals them. Finally, the adversary uses this credentials to log into the personal IoT interface.

E. **Side-Channel Attack:** In this attack, the malicious adversary searches for the encryption keys. In this type of attacks, adversaries use electromagnetic analysis and device power consumption analysis.

F. **Denial of Service Attacks:** An adversary can redirect or Jam the IoT web/mobile application interface by code injection resulting in service unavailability for the users.

3. Design of Feature Distributed Malware Attacks

In this chapter, we propose the design of feature distributed malware attacks for Internet of Things in the context of smart homes. Smart homes consist of various smart home devices, providing convenient services to the tenants. These devices are usually connected to the Internet, and the tenants can integrate individual devices and make smart home workflows using Internet services such as IFTTT. The focus of our work is on how the attackers can generate various cyber-physical and advanced cyber-attacks by exploiting this integration aspect in smart home. In particular, we have designed and developed an advanced feature distributed malware that can be used to perform various malicious activities such as stealing the victim's IFTTT cookies, using smart devices as information sources of the malware and distributing malware functionalities to other devices, performing malicious without being noticed, causing financial damages to the victim, and evasively exfiltrating data. Our experiments show that the proposed attacks enable the attackers to control any smart home device integrated with IFTTT, without requiring compromise of individual smart home devices. We believe our approach of using the integrating Internet service as an attack vector and distributing malware features to the smart home devices will be helpful to the development of more secure smart home environments.

3.1 Introduction

Smart homes are equipped with various technologies that provide the tenants with a wide range of information about the state of their home, and allow them to control all the connected smart devices/appliances. Especially when the smart home technologies are combined with the Internet, the tenants can monitor and control their smart devices remotely, from where personal area network (PAN) or home area network (HAN) cannot reach. On top of this remote control capability, a smart home is usually able to detect changes to its connected devices, to notify the tenants of such changes, and to take appropriate actions. Smart home devices include smart switches/power sockets, smart home security suites, smart Wi-Fi cameras, smart motion monitors and so on. Home routers and network attached storages (NAS) are also indispensable components of the smart homes, since the former integrate PAN/HAN with the Internet, thus enabling remote monitoring and control, and the latter, together with other devices such as smart TVs and tablets, play a central role in the converged media of the smart homes [16].

Remote monitoring and controlling a smart home used to be initiated only by human tenants. However, the inter-connectivity between devices has made it possible to automate various smart home workflows such as detecting status changes, notifying the tenants of the changes, and reacting to them. All these actions from monitoring to responding are performed by the smart devices in the smart home. This home automation has increased over the last few years partly because technologies for smart home devices and services have been improved and developed to the level where deployment and market entry are practical. In addition, smart home devices have become more affordable, and there are many available smart home devices from several vendors. Considering the continuously improving inter-connectivity between devices, the smart home area is expected to be an attractive field for development and deployment.

The effectiveness of remote activities is maximised when the smart home devices and the tenants' smart devices such as smart phones are inter-connected via the Internet so that the tenants are notified of changes, and monitor or control their smart home whenever and wherever they want to. Moreover, Internet services such as IFTTT are providing an integrated interface between smart home devices and users. Although some smart home devices (e.g. temperature sensors) may not have the Internet connection, they are still connected to a form of device manager that aggregates information

and adapts to changes, and this manager device is connected to the Internet so that it can send notifications to the tenants and communicate with other smart home devices deployed in the smart home. Standard protocols like Z-Wave, Bluetooth, and ZigBee are broadly used for this local connectivity. This entire smart home environment is an example of the Internet of things (IoT) where non-human devices communicate with each other, and automate various tasks.

However, similar to any other emerging technology, currently available smart home devices are known to be vulnerable to relatively uncomplicated attack vectors; examples are smart TV [19, 32], smart thermostat [31], smart lights [25] and baby monitor [26]. These studies have shown that smart home devices can be easily compromised via hardware alterations, firmware modifications, uncomplicated wireless network attacks such as traffic replay and malicious packet injection (although relatively new technologies such as ZigBee and Z-Wave are used), and software exploitation (zero-day or even not patched old exploits). Most academic studies are also addressing security issues of hardware, firmware and network protocols [16, 35, 43, 44, 48, 49].

In this chapter, we take a different perspective. Rather than attacking individual devices or network protocols, we utilise the IoT aspect of the smart home in order to achieve several types of attacks. While the current efforts in the smart home and IoT security are mainly focused on attacking network protocols and individual smart home devices, our approach considers the smart home as a single and integrated system. We use a malware to obtain the access to an Internet service that integrates target smart home devices, and perform malicious activities without compromising individual smart home devices. This is a more devastating attack than the previously suggested ones in the sense that it does not require the compromise of multiple devices (some devices require physical access or a certain proximity to them for attack), and hence the attackers do not need to have any shape of known or zero-day exploits against the target smart home devices. Furthermore, the attackers can use the Internet service from anywhere on Earth to monitor and control all the directly or indirectly connected target smart home devices, which enables smart attacks such as automatic unlocking of the smart lock after the tenants leave the home and notifying the event to the attackers.

We also adopt a concept of an advanced malware, called feature-distributed malware (FDM) [34] , to steal the target's specific Internet service account, and to achieve cyberattacks that are impossible or very hard to perform on highly secured PC's due to the security tools installed on them. Feature-distributed malware on a machine can distribute certain malware functionalities to smart home devices that are not equipped with serious security tools, and bypass the PC security tools' policies enforced on the machine; most of small and low-cost smart home devices that we have surveyed do not have mature security functionalities, because they are designed primarily for the ease of set-up, use and interconnection, and their computing power is relatively low [16]. In addition, smart home devices such as motion sensors and Wi-Fi cameras can extend the input capability of malware. We have implemented a feature-distributed malware that steals an Internet service account of the victim, utilises the service for various attacks against the victim's smart home, and performs malicious activities from unlocking the smart lock to evasive data exfiltration. The attack scenarios and experimental results show that such attacks are practical, which implies that such attacks have to be considered from the design and implementation stages of smart home devices and services.

To our knowledge, this is the first work that analyses the IoT aspect of the smart home from a security point of view, and uses it as an attack vector against the smart home integrated with an Internet service, even though the possibility of the threat has been stated [16]. Since a compromised smart home can lead to physical damages (e.g. penetration into the house), any security issue of smart home devices and technologies have to be addressed, and we believe this work will benefit the future development of secure smart home devices and services.

The remainder of this Chapter is structured as follows. Our proposed attacks are explained in Section 3.2 together with essential attack components such as a smart home integrating Internet service (IFTTT) and feature-distributed malware. Section 3.3 presents our practical implementation of a feature-distributed malware and the proposed attacks, and shows our experimental results. The generality of our attacks and the currently available mitigations are discussed in Section 3.4.

3.2 Proposed Attacks against Smart Home

3.2.1 Integrated Attack Vector: IFTTT

IFTTT (https://ifttt.com) started as an Internet service that connects two different Internet services and automates certain tasks. It allows its users to create a series of conditional statements (`If This' in **IFT**TT) that are triggered when such conditions are satisfied (e.g. a new mail in the user's Gmail account). The users also specify an action statement that will be performed on each trigger (`Then That' in IFT**TT**). This set of statements called ``recipes'' in IFTTT, and the users can create their own recipes using more than 180 channels (i.e. Internet services like Gmail, Facebook, Twitter, Instapaper, and RSS feeds) supported by IFTTT.

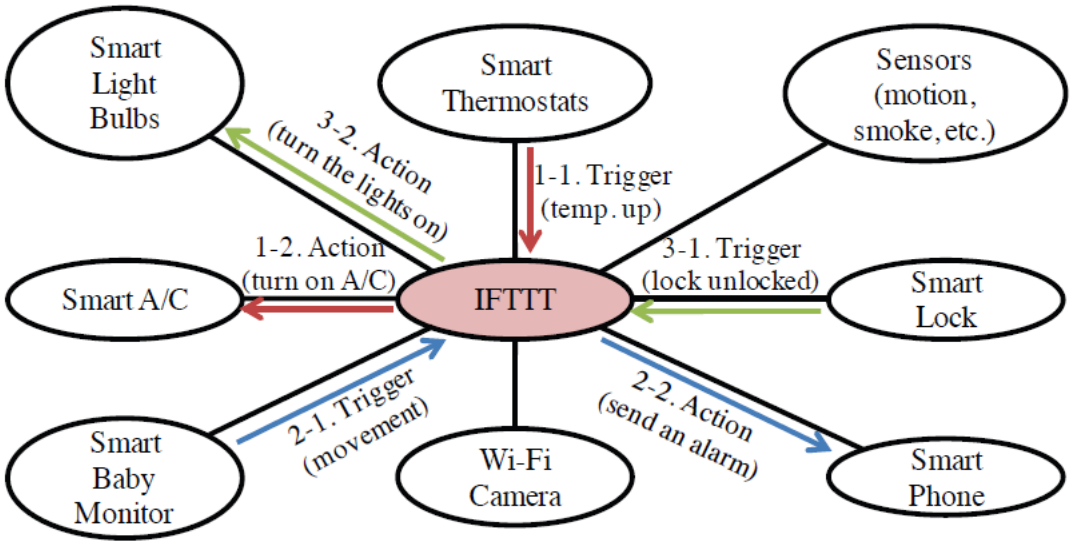


Figure 3.1: IFTTT recipe examples for smart home

Since the emergence of smart home devices, IFTTT started supporting such devices so that its users can automate a wide range of smart home tasks such as maintaining a certain temperature using smart thermostats and an air conditioner as illustrated in Figure 3.1. Since IFTTT supports multiple vendors, the users can integrate smart home devices from different vendors. In other words, the users do not have to buy a single smart home suite from a specific vendor, which is one reason for IFTTT's popularity. There are more than 20 channels and 2,000 public (shared) recipes that are related to the smart home, which are given in Table 1. One extreme example of the shared recipes is to use IFTTT to physically open doors using triggers and actions shown in Table 2.

Once a user activates channels on IFTTT with his/her own smart home devices, recipes can be built based on the triggers and actions that the activated channels support. For instance, the user can make a recipe that is triggered when the temperature goes too high (measured by an activated smart thermostat channel), and then turns on an activated smart air-conditioner. This is a typical example

of the Internet of things (IoT) where two smart home devices communicate to each other and co-operate through the Internet (more precisely, an Internet service called IFTTT. In this way, a number of smart home related tasks can be readily automated, such as maintaining economic power consumption, automatic lighting on the tenant's arrival, and getting an alarm as a push message sent to a mobile device in case of emergency. Major triggers and actions of smart home related channels on IFTTT are summarised in Table 2.

Table 1: IFTTT channels related to smart home

Category	Channels
Lighting & shading	Hue, LIFX, Lutron
Thermostat, sensing & A/C	nest, ecobee, wallyHome, Wink Aros
Switch/power socket	WeMo, Scout, Wireless Sensor Tags, SmartThings, Smappee
Camera	Manything
Integrated systems	SmartThings, WeMo, Scout, Wireless Sensor Tags, HomeSeer, Honeywell
Others	Harmony (Logitech), bttn, Iro

Table 2: Smart home related channel categories and their respective triggers & actions

Category	Triggers	Actions
Lighting & shading	Not available	Turn on/off lights, Change colour of lights, Blink lights, Set light level, Move shades up/down
Thermostat, sensing, A/C	Temp. too high/low, Fire detected, Set to home/away, Low battery	Set temp., Turn on/off A/C Turn on/off fan for XX minutes, set vacation mode
Switch/power socket	Switched on/off, Standby mode entered, Daily minutes/cost reached	Turn on/off switch/socket, Toggle on/off, Turn on/off then off/on switch/socket
Camera	Motion detected, No movement for XX minutes, Power dis/connected, Low battery	Start/stop recording, Start/stop taking photos, Set time interval, Use front/rear camera, Turn flashlight on/off, Un/mute audio
Integrated systems	Temp. high/low, Sensor out of range, Door/window open/closed, Switched on/off, Sensor timed-out, Motion detected	Lock/unlock doors, De/activate alarm, Beep a tag/an alarm, En/disable temp. monitoring, De/activate smart devices, e.g. a Wi-Fi camera, Dis/arm motion sensor, open/close door
Others	Button pressed	Start a specific activity, e.g. turn on TV

Although Internet services like IFTTT provide tremendous benefits and convenience to their users, they are also a very lucrative target for the attackers. Since different types of devices and services are set up and integrated with in IFTTT, the attackers do not have to compromise any target device as soon as they get an access to the target's IFTTT account. This means that (1) no hardware, firmware or more high-level software exploitation is not required, (2) no networking attacks against network protocols such as ZigBee and Z-Wave are not needed, and therefore (3) a certain level of proximity to the target devices for PAN/HAN network attacks is also not required. It is a huge gain for the attackers because they do not have to attack individual devices, even if the current state of security of smart home devices are unsatisfactory and such devices are relatively easy to compromise as discussed in Section 3.1. Hence, using an integrated Internet service as an attack vector is one of the core ideas of this work. When this approach is combined with an advanced malware, several attacks that are discussed in Section 3.2.4 have a huge impact on the victim.

Depending on the security implementation of IFTTT, obtaining an access to the victim's IFTTT account can be performed in a few ways such as keylogging for IFTTT credentials by malware, and session hijacking via eavesdropping, session fixation attack or data exfiltration (in this case stealing the cookie data) by malware. Unfortunately, it turned out IFTTT's session management and general security measures are not robust enough to prevent the stolen cookies from being abused by the attackers as discussed in Section 3.3.4. In this work, we have experimented both session hijacking and keylogging; the malware deployed on the victim's device steals the relevant cookies or credentials, and then either directly uses them to manipulate target's IFTTT recipes or sends them back to the attackers so that recipes can be created, modified, and disabled with the stolen cookies or credentials.

3.2.2 Feature-Distributed Malware (FDM)

In the sophisticated attack scenarios that are described in this section, an advanced malware is used to perform various malicious activities. In this section, we introduce the core concept of this malware, called feature-distributed malware (FDM). Feature-distributed malware was first suggested in the work by Min and Varadharajan [34], and is an advanced malware which dynamically distributes its features to multiple software components on a victim system. One of the key ideas of the feature-distributed malware is to piggyback malware features on various applications' legitimate features. For example, the networking feature of the malware is migrated into an email client, while the local data collection feature that requires administrator or even SYSTEM privilege is migrated into a security tool's service running under SYSTEM account. These distributed components cooperate as a single virtual malware. Such distribution of the features enables the malware to bypass various security policies and mechanisms such as application whitelisting, application-based permission model, and egress filtering. In addition, the feature-distributed malware can also convert a malware instance that is detected by major anti-virus solutions' behavioural engines (as well as file detection engines) into an undetectable one [34].

We have applied this malware technique to our proposed attacks against the smart home for the following three reasons. First, the feature-distributed malware makes several malicious activities including data exfiltration possible, which would be impossible or extremely hard to perform otherwise, due to end-point security tools installed on the target machine. As mentioned in Section 1, smart home devices, home routers and media centres are not yet armed with serious security tools. Therefore, distributing malware features to these devices will enable the malware to bypass security tools and policies such as egress filtering enforced on the target machine. Second, a smart home is a kind of distributed system that consists of multiple smart home devices. Although these devices have different purposes from measuring temperature to detecting motions, they work together as an integrated smart home system to achieve various smart home tasks. The individual purposes can be mapped to or transformed into malware features, which makes the concept of distributed malware to nicely ``fit'' the smart home domain. In particular, smart home devices can be utilised as an extended input/output for malware; a Wi-Fi camera can work as an additional data source, a smart lock can function as another credential stealer that is normally performed by a key stroke grabber in traditional malware, and a home router can perform data exfiltration feature. It is worth noting that malware features are piggybacked on individual devices' original features such as recording videos, locking/unlocking a door, and communicating data with the Internet. This makes the feature-distributed malware more stealthy with extended capabilities. Lastly, the feature-distributed malware can adapt to the dynamics of the smart home where new devices can be added and/or existing devices are removed or replaced, because it can dynamically distribute malware features to newly installed devices (Section 3.2.3).

3.2.3 First Stage Attack: Preparation

Our proposed attacks begin like usual malware operations; the victim system is infected with a malware. This malware performs traditional malware activities including keylogging, screenshot grabbing, password hash dump, and network scanning for further propagation. At the same time, it tries to detect target's activities on IFTTT, and then starts the following procedure if any activity is identified in order to execute second stage attacks described in Section 3.2.4:

I. Steal the target's IFTTT account credentials and/or valid session data, i.e. cookies.

II. Search for activated smart home related channels from the IFTTT account.

III. Search for home routers and NAS that are essential to a smart home, but not managed by IFTTT. These devices are compromised, if possible, and used for malware feature distribution.

IV. If there exist one or more activated smart home channels on the target's IFTTT account, collect all the information on these channels and recipes created by the victim. This step is crucial in the proposed attacks because the malware and the attackers get to know the target's smart home configuration at this stage. Another advantage of compromising an integrated Internet service like IFTTT is that the target will create new recipes whenever he/she buys new smart home devices. The attackers automatically get the new configuration via IFTTT, and the malware automatically becomes able to use the newly deployed devices, which can be thought as an automatic malware feature extension in the sense that it gets the access to the new devices as soon as they are added by the victim.

V. Begin the second stage attacks based on the gathered information.

The attackers can intervene in this procedure at any moment using the malware's command and control (C&C) channel, and manually manipulate target's IFTTT recipes.

3.2.4 Second Stage Attacks: Actual Payloads

In this section, we first briefly describe the characteristics and requirements of the proposed attacks, and then discuss several approaches to using feature-distributed malware and IFTTT from an offensive point of view.

Any attack targeting physical devices of the smart home has to remain undetected and unnoticed by security tools and the victim, at least until its goal is achieved. This is usually harder to fulfil than remaining stealthy in cyber-only attacks, because the impact of such an attack is physical, and hence easy to spot. This kind of attack against cyber-physical systems (CPS) is called ``deceptive'' attack [15, 33] in the sense that the attackers need to ensure that the victim does not notice that his/her smart home is under attack. Deceptive attack is possible since most CPS including smart home are remotely monitored and controlled through software and/or hardware systems. The notorious Stuxnet showed that the deceptive attack is clearly feasible for the attackers, and Min and Varadharajan [33] have derived the following three core requirements that are essential to the deceptive attack:

RQ1: Modifying the behaviour of target physical devices (for physical impact).

RQ2: Hiding physical impact via software or data manipulation (for deception).

RQ3: Preventing automatic and manual remote recovery attempt.

These requirements are fulfilled with the proposed attacks with physical outcome, e.g. turning on/off a smart home device, as follows:

RQ1: Create/modify recipes of IFTTT to modify the behaviour of smart home devices. Modification is recommended since a newly created recipe can be noticed by the victim.

RQ2: Disable/modify/delete existing recipes of IFTTT, which are made for monitoring, to hide the attack and its physical impact. Again, modification is most recommended, and deletion is least preferred way of recipe manipulation.

The last requirement can be satisfied by intercepting any access to IFTTT by the victim, and displaying a forged screen that shows the victim's action on IFTTT was successful. This is easier to achieve in SCADA systems because only a few control software installations on a limited number of computer systems that are connected to the closed control network are required to be compromised [33]. However, in the case of Internet services, even after the attackers compromised all the devices from

the PC's to the smart phones and the tablets that belong to the victim in order to intercept and modify any access to IFTTT on these devices, he/she can access IFTTT on an arbitrary computer such as a public computer available at the hotel the victim is staying for a vacation. On the other hand, SCADA and other critical infrastructure systems are always monitored by human beings, whereas smart homes are not. Therefore, instead of directly addressing the last requirement, we have considered this difference between SCADA and smart home, and designed the attacks in a way that they can hardly be detected by the victim, e.g. by starting the attack only when the smart home becomes empty for a certain amount of time. In addition, IFTTT recipes are designed with a philosophy of ``setup once and forget'', the details of recipes such as the specific action that will be performed are not displayed in my recipe pages; only channels involved in the recipe are displayed. As a result, even after the malware changed the details of the existing recipes, it is hard for the victim to notice the changes. Furthermore, there is no reason for the victim to frequently check the details of the recipes unless he/she realises that the account has been compromised.

3.2.4.1 Extending malware's capability

The attackers can use the target smart home devices to extend existing malware features. Traditional malware has various functionalities such as propagation, data collection, backdoor installation, and networking. Among these, data collection depends on the number of input types of the target machine, including keystrokes, screenshots, microphone recording, files, connected device list, and saved passwords. Networking (usually data exfiltration and C\&C) relies on the number of output types such as Ethernet, Wi-Fi, Bluetooth, and even speakers. A feature-distributed malware operating in the smart home can extend input and output types based on the smart home devices deployed in the target smart home. For instance, any device with a camera and Wi-Fi connectivity (e.g. smart TV [32] can be used as an input for an extension to screenshot grabbing or even target surveillance. Credential stealing activity is extended from keylogging and password hash dump to controlling smart locks via IFTTT. Recording sources may now include VoIP phones as well as PC microphone. A new type of input method is added; presence service usually implemented with sensors will enable the attackers to detect the target's leave and arrival. When it comes to output, traditional output methods can be extended to other devices with network connectivity. The extended input and output sources can be used in the exactly same way as in the traditional PC malware, i.e. data collection and networking (cyber espionage). This type of attack is categorised into the passive and autonomous one in Table 3, which monitors the target smart home and notifies the attackers of any desired event such as tenant's leave.

Table 3: Types of CPS Attacks against smart home using **IFTTT**

	Passive	**Active**
Autonomous	Monitoring & notifications	Actions performed or prevented in recipes
Manual	-	Actions preformed or prevented by attacker

As discussed in the following sections, the extended input and output methods can also be utilised in several CPS attacks that have physical impact and cyber attacks that surmount the security of PC with less secure smart home devices.

3.2.4.2 Manipulating smart home devices (CPS attacks)

This attack is a kind of CPS attack that uses the extended input and output sources in a very active manner. From all the new input and output methods listed in Table 2, one can imagine virtually unlimited number of malicious combinations of them. In this Section we discuss three particular attack scenarios among many possibilities. These attacks are implemented as IFTTT recipes and the malware automatically turns on and off them as shown in Section 3.3.

Unlocking a smart lock

I. The smart home detects target's leave using the motion/presence sensors and the smart lock.
II. The victim may change the smart home system's mode to the vacation mode, which is detected by IFTTT.
III. IFTTT notifies the attackers of the above status through the recipes modified or created by the malware/attackers (RQ1).
IV. IFTTT automatically opens the smart lock after a certain period of time so that the attackers (or co-working thieves) can enter the victim's premises (RQ1).
V. The malware temporarily turns off all the presence/motion detection related recipes so that the thieves' entry is not notified to the victim (RQ2). In particular, recipes that are triggered when the smart lock is unlocked or when motion/presence sensor detects someone (widely used to turn the lights on when the tenant comes in) have to be disabled.
VI. The malware temporarily turns off all the power/running status-related recipes for security-related smart home devices, such as Wi-Fi camera and sensors (RQ2). The thieves may want to turn off or stop the operation of Wi-Fi camera, but this can be notified to the victim if a relevant recipe is set up. For instance, the victim may have set up a recipe that notifies him/her when the power adapter of the Wi-Fi camera is disconnected. This category of recipes has to be disabled by the malware.
VII. After the thieves completes their job, the malware restores all the recipes it has disabled so that the victim cannot notice any change in the recipe status list (RQ2).

This attack belongs to an active and autonomous attack in Table 3. The attackers set up relevant recipes in the victim's IFTTT account, and wait for the recipes triggered and the actions involved in the recipes are executed.

As discussed earlier, the recipe list screen of IFTTT shows only the channels involved in each recipe and the recipes' status (i.e. turned on or off); detailed actions or conditions of the recipes are only displayed when the user clicks a certain recipe. Therefore, in all the attacks explored in this section, recipes can be modified instead of being turned off for maximum stealthy property.

Causing financial damages to the victim

I. Same 1-3 steps as the previous attack.
II. IFTTT turns on all the appliances from air conditioner to lights and to any devices connected to the smart switches through the recipes modified or created by the malware/attackers (RQ1).
III. The malware temporarily turns off all the temperature/humidity monitoring/controlling recipes so that all the appliances remain turned on and no notification is sent to the victim. All the notification-style recipes connected with the turned-on devices also have to be temporarily disabled (RQ2).

IV. After a certain amount of time, the malware turns off the above attack recipes, and restores all the disabled recipes (RQ2).
V. Regularly repeat the above steps (RQ1). This will increase the power consumption of the target smart home, which will bring financial damage to the target.

This attack is also classified as an active and autonomous attack in Table 3. This attack is most effective if the time duration in step 4 is short enough to make the victim think that the electricity bill is reasonable, but the attack period (step 5) is long enough to raise the victim's electricity use in the long term.

Leaving critical events uninformed

I. The malware/attackers turns off the notification-style baby monitor recipe (RQ1). It may modify the recipe to send notifications to the attackers, not the victim, in order to minimise the risk of detection. In addition, any other actions such as beeping a buzzer tag are modified or disabled.
II. As a result, movements of the baby will not be reported, and damages to the baby such as suffocation can happen without being alerted.
III. After a certain amount of time, the malware re-enables the turned-off recipe (RQ2).

Since the attackers are using a compromised Internet service account, they do not need to compromise the baby monitor, which makes the attack more feasible; on the contrary, previous work [26] requires direct attacks on the baby monitor for the same offensive purpose. This attack is an example of active and manual attack in Table 3 in the sense that the attack is not automatically triggered by IFTTT recipes, but manually instructed by the attackers.

3.2.4.3 Overcoming stricter security on PC

This is not a CPS attack, but a cyberattack that utilises the extended output in order to overcome the stricter security configurations of the compromised PC.

As cyberattacks have become increasingly sophisticated, the security research community has been improving end-point and network security techniques. Almost every commodity PC from major vendors such as HP and Dell has an anti-virus product pre-installed, and anti-virus solutions have been incorporating behavioural detection engines (as well as traditional signature-based detection and sandboxed execution/emulation-based file scanning engines) to detect malicious behaviours such as Dynamic Link Library (DLL) injection on the fly. In addition, major PC operating systems including Microsoft Windows, Linux and Mac OS X are armed with various software exploit mitigation techniques such as Data Execution Prevention (DEP) and Address Space Layout Randomization (ASLR), and are protected by secure booting procedures such as Secure Boot [36] and Early Launch Anti-Malware [37] of Microsoft Windows. In a more secured environment like corporate systems and networks, even more strict security policies and tools are deployed. For instance, a firewall with egress filtering rules, an end-point intrusion detection or prevention system (IDS or IPS), an application whitelisting tool, and a file integrity monitoring tool may be running on a highly secured corporate laptop. As a result, it is becoming increasingly challenging for the attackers to bypass the detection of security tools to achieve their goals.

However, as discussed in Section 3.1, smart home devices are still premature from a security point of view, and they are vulnerable to relatively out-dated offensive techniques. More importantly, these devices tend not to be equipped with any security tool partly due to their limited computation power and resources. Considering local area networking is more allowed than wide area networking (i.e.

Internet) [20, 24, 25], bypassing strict security policies and techniques that are enforced on a PC can be accomplished using the smart home devices that reside in the same local area network. Since it has been shown that the feature-distributed malware can migrate the data exfiltration feature to a process within the same system that is allowed in the egress filtering rules and bypass such rules, migrating the external networking feature of the malware to an Internet-connected smart home device should be also possible. Home router is the best target device for this purpose because its original functionality is networking (the idea of the feature-distributed malware is piggybacking malware features on benign applications' legitimate features). We have implemented this feature distribution functionality on the malware side and the data exfiltration module on the home router side as described in Section 3.3, and our evaluation results has shown that migrating the data exfiltration module from the malware on the victim machine to the home router enabled the malware to evade the egress filtering rules that used to successfully prevent information stealing on the machine.

3.3 Implementation

In this section, we fist explain the details on how the proposed attacks are implemented in a real smart home setup, present our smart home testbed implementation, and then evaluate the attacks carried out against the testbed.

3.3.1 Feature-Distributed Malware for Smart Home

As discussed in Section 3.2.3, we assume the target PC has been infected with a malware, which is assumed by most related work in the malware research area [13, 14, 18, 28, 34, 39-42]. Specific initial attack vectors for PC, home router, and NAS are not the scope of this paper. Our focus is to show *what* and *how* malware and a compromised Internet service account can bring to the smart home and IoT security.

On the victim's PC, we have followed usual operations of the original feature-distributed malware [34]. On top of this, we have added new features for smart home so that the malware can compromise a certain models of home router and NAS, and distribute malware features to these devices. First, the notorious shellshock [1] exploit has been loaded in the feature-distributed malware. This exploit was used to infect both the home router and the NAS during our experiments. Next, netcat (nc) compiled for busybox Linux distribution has also been embedded in the malware in case that netcat is not built in the target home router and the NAS. Netcat is a small but versatile network utility that is optimal for our purposes, i.e. getting a shell and transferring data. Using netcat, the malware gets a reverse shell from the home router and the NAS (-e switch), and sends stolen data to the home router and then to the attackers (traditional netcat relay). Our smart home configuration follows the current state of the art, but security features currently provided by the home router and the NAS only include simple traffic inspection for known malware, denial of service mitigation, and traffic control using firewall (packet filtering rules). Corporate-level threat detection at the network level (e.g. monitoring and analysing traffic online and offline for anomaly or unknown threat detection) is not the norm for the smart home yet. Therefore, it is viable to use netcat as the communication tool between the PC module and the home router and the NAS modules of the feature-distributed malware. However, if the attackers decide that netcat may be detected or prevented at the network level of the target smart home, encapsulation or tunnelling with a well-known protocol like HTTP(S) is always an option as it is widely used in PC malware. Lastly, we have implemented a Python module for NAS to control the

[1] There are many ways of exploiting this vulnerability and, at least six CVE's (CVE-2014-6271, CVE-2014-6277, CVE-2014-6278, CVE-2014-7169, CVE-2014-7186, CVE-2014-7187) have been registered for it

target's IFTTT recipes, which is detailed in the following section. NAS is an optimal target for IFTTT control because it is almost always turned on and connected to the Internet.

3.3.2 IFTTT Manipulation by Malware

The core of the malware and the proposed attacks is the use of IFTTT that automates various smart home workflows. Our aim is to automate IFTTT user interactions as much as possible so that the malware automatically manipulates the target recipes (i.e. automated workflows) as the attackers want to.

During the preparation stage (Section 3.2.3), the malware monitors cookie data on the victim PC and performs keylogging. In addition to this typical malware activities, it sends IFTTT credentials or cookies to the attackers when such information is found. At the same time, the feature-distributed malware distributes the IFTTT manipulation module to NAS together with the stolen credentials or cookies. From this point in time, the malware automatically manipulates the target's IFTTT recipes according to the commands from the attackers; the attackers can manually manipulate the recipes as well.

```
{"commands": [
  {
    "recipe" : "0",
    "mode" : "disable",
    "contents" : ""
  },
  {
    "recipe" : "3",
    "mode" : "create",
    "contents" : "trigger and action
                  for new recipe"
  }
]}
```

The automatic manipulation is implemented in Python, since Python is officially supported by our target NAS (Synology products) and it is easy for the malware and the attackers to update the module. In the current implementation, the attackers upload a command to their server in the form of JSON that specifies how to manipulate (to enable/disable/create/delete) which recipes (designated by the recipe number) with what contents (in the case of recipe creation). Modification is not yet fully implemented in the current version of the malware, so the attackers need to manually do this. A command form like the following is retrieved by the IFTTT module of the malware, and then parsed and stored using json.loads() function:

In order to implement IFTTT-manipulating features from logging in to navigating and to manipulating recipes, we analysed the HTML elements and JavaScript code employed on IFTTT, because there is no quick and simple way to find UI elements that consist of HTML and JavaScript; Cascading Style Sheets (CSS) is mainly used for styling so analysing it was not needed. In essence, we analysed Document Object Model (DOM) structures and JavaScript code (mostly Ajax and asynchronous code), and inspected each HTML element that we needed to interact with and exact timing of element interactions using the developer tools provided by Google Chrome, following the guidelines from Black Hat Python[46]. Then we have used Splinter (https://splinter.readthedocs.org/) to implement the required interactions with IFTTT. After these analyses, we have implemented the Python code that manipulates the target IFTTT recipes as stated by the command. The code accesses to the ``My Recipes'' page on IFTTT, finds the recipe specified in the command (the number of top most recipe is 0), turns on/off according to the command, or clicks ``Create a Recipe'' button to create a new recipe based on the description of the ``contents'' of the JSON command form. In the case of using the

credentials, not the cookies, the code first looks for the email and password fields, sets them with the stolen data, and submit the form to log in to IFTTT as the victim.

An attack command usually creates a recipe for restoration after the attack. For instance, the physical penetration attack command adds a recipe that sends a push message to the attackers when the smart lock is locked again or there is no movement in ten minutes. Upon receiving the push message, the attackers' machine uploads a recipe restoration command on the attack server. This command is a JSON file that orders the malware to delete all the modified or newly created recipes and to create the original ones.

3.3.3 Target Smart Home Configuration

The configuration of our target smart home consists of a smart lock, a baby monitor, a Wi-Fi camera (iPod Touch), smart sensors (temperature, motion, and/or humidity), beeping tags, smart switches, sensor & tag mangers, a home router and a NAS, as given in Table 4 and illustrated in Figure 3.2. SmartThings Hub and connected devices communicate using the Z-Wave standard, which enabled us to connect the SmartThings Hub and the smart lock from a different vendor. On the contrary, Wireless Sensor Tags Manager and tags seem to use a proprietary protocol, communicating at the radio frequency of 433.186 MHz.

Table 4: Smart home devices used in the target smart home

Product	Purpose
Schlage FE599NX	Smart lock for front door
WeMo Motion	Baby monitor
iPod Touch	CCTV with Manything app
SmartThings products	Integrated smart home system
Wireless Sensor Tags products	Integrated smart home system
WeMo Switch	Light bulbs and Hi-Fi control
Buffalo N450	Home router
Synology DS215j	Media hub (NAS)

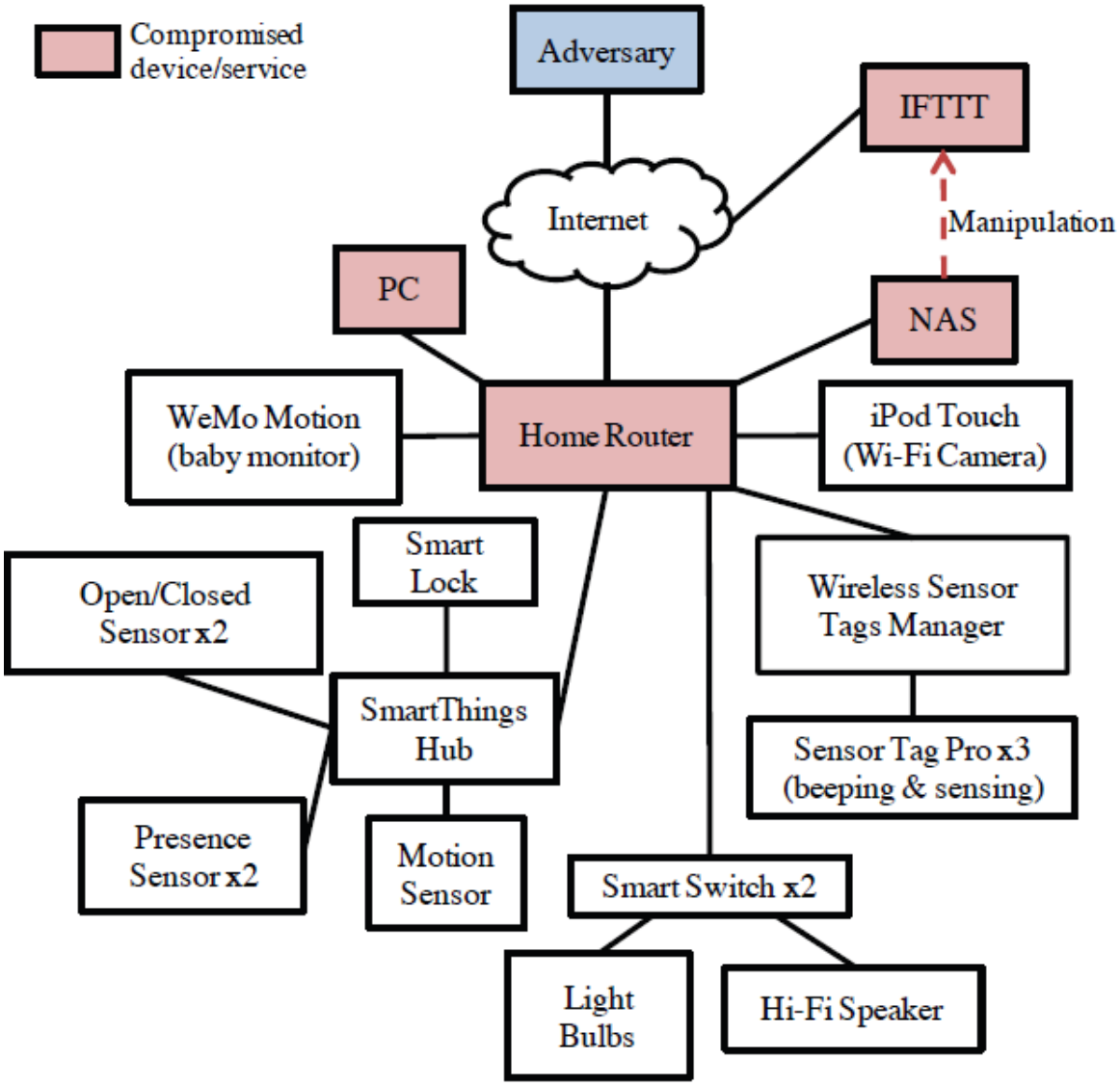


Figure 3.2: Target smart home configuration

After finishing the hardware deployment, we signed up an IFTTT account that will be used by the victim, and enabled all the relevant channels. Then we created several smart home recipes based on the popular and featured recipes provided by IFTTT in order to make our smart home configuration as realistic as possible. For instance, one recipe beeps a tag when the baby monitor detects a certain level of movement, and another one sends a security alarm to the user's smart phone if the open/closed sensor installed at a window detects the window is open. Lighting a room when presence is detected and automatically locking the smart lock at a specific time are also the recipes set up for the test-bed.

3.3.4 Evaluation

The proposed attack model turns the smart home attacks that normally require compromise of multiple smart home devices with a number of attack vectors into more feasible ones that are achieved via the two matured offensive techniques, session hijacking and malware operation. In this section, we evaluate how much cookie stealing is effective against IFTTT, and discuss how CPS and cyber-only attacks are accomplished.

Security of IFTTT cookies

The current security state of IFTTT cookies is not up to par with its capability. It does not provide any additional security measure that is discussed in Section 3.4.2. Once the attackers and the malware obtained the target's IFTTT cookie data via a type of session hijacking attack, they can impersonate the target at anytime from anywhere until the cookies expire. We have verified that the cookies from the victim who logged in to IFTTT from an Australia IP was valid from IP addresses in the University of Salamanca, Spain (IP addresses of 212.128.0.0/16) and Seoul National University, Korea (IP addresses of 147.46.0.0/16). Even though the victim virtually moved from Australia to Spain within a minute, IFTTT did not invalidate the cookies or notify the user; notification is the most often used mitigation since the user may use different IP's from geographically faraway places using tunnelling mechanisms such as VPN or SSH tunnelling. As a result, the attackers could manually modify the target's recipes and upload a command that was retrieved and executed by the NAS module of the malware. As one can expect, we verified that the malicious behaviours described in Section 3.2.4 were performed by the target smart home devices, thus fulfilling the goal of our proposed CPS attacks; those behaviours are omitted here to avoid unnecessary redundancy.

After the final step of an attack was carried out (e.g. leaving the house after penetrating into the premises and stealing offline assets), the restoration recipe was triggered. Then the attacker's machine automatically uploaded the restoration command that was downloaded by the NAS module, and the malware restored all the original IFTTT recipes by deleting modified or newly created recipes and re-creating the original ones.

Data exfiltration

Data exfiltration is a cyber-only attack among the scenarios explored in Section 3.2.4. We have set up the target PC with the Comodo Firewall (http://personalfirewall.comodo.com) with egress filtering rules. The rules block any Internet access from non-whitelisted software, while allowing LAN communications. This is a typical firewall set-up scenario that is also explained in the Comodo documentation [22]. Because local area networking is allowed, the malware successfully transferred data that is collected by the data stealing modules to the compromised home router, and the router replayed it to the attackers' server. This evaluation is not complex compared to the CPS attacks, but it clearly shows how the feature-distributed malware can bypass a stricter security enforcement of the compromised PC via a smart home device within the same network.

3.4 Discussion

3.4.1 Generality of the Proposed Attacks

Although we have used a particular Internet service, IFTTT, throughout the chapter and implementation, the concept of abusing an Internet service that integrates smart home devices is general, and hence can be applied to other services with the same purpose such as Facebook's Parse (https://parse.com/), KumoApp Engine (http://wirelesstag.net/kumoapp), and Zapier (https://zapier.com). In addition, building one's own smart home system with more general-purpose hardware platforms like Arduino and Raspberry Pi is emerging, and dynamic DNS (DDNS) is often used for remote smart home control in this case [30]. As long as a smart home is connected to the Internet, our attack strategy can be applied. Once the interface to the Internet is compromised (through an attack like session hijacking, man-in-the-middle or keylogging by malware) and the feature-distributed malware is deployed, all the attack types and scenarios analysed in this paper can be performed. Finally, Internet services (even big ones such as Facebook and eBay) used to have (and still have

potential) XSS vulnerabilities that can be used for session hijacking, which will make the proposed attacks even more practical.

Instead of utilising an integrating Internet service like IFTTT, the attackers can embed multiple exploits in the malware so that it can compromise individual smart home devices. A module such as NAS or home router will arrange the compromised devices as the Internet service does. As discussed earlier, in general, currently available smart home devices are more vulnerable than PC and they do not have security tools that detect attacks on themselves[2], which makes this type of attack model plausible. On the other hand, the feature-distributed malware that is deployed to maximise the attack capability (e.g. automatic attack launching and restoration and advanced cyber espionage) may not be used. In this case, the CPS attacks can be manually executed by human attackers after they obtain the target's IFTTT account. The number of compromised components in Figure 3.2 is decreased to one (i.e. IFTTT), in spite of the fact that such attacks can have less practicality and limited impact.

3.4.2 Mitigations

Even though it is impossible to perfectly prevent session hijacking with the currently employed web technology, IFTTT's session management and other security measures can be improved, and should be done so when we consider its position as a centre of integrated smart home. In this section, we briefly describe several techniques that can mitigate the proposed attacks, mainly addressing session hijacking[3]. Also, although there are other session tracking mechanisms like URL rewriting and hidden fields, our discussion on session hijacking is limited to cookie that is the most widely used method.

First of all, HttpOnly and Secure flags [17] can enhance the security of cookie. When HttpOnly flag is used, JavaScript running in a web browser cannot read authentication cookies even in case of cross-site scripting (XSS) exploitation. Secure flag will mitigate eavesdropping issue since cookies are only sent over HTTPS connections when this flag is set, although some HTTPS implementations had vulnerabilities that enabled HTTPS eavesdropping.

Even after the two flags are used, other forms of session hijacking attack (e.g. man-in-the-middle, man-in-the-browser, and pharming) is still possible. Providing against these attacks, cookie can contain user fingerprinting data such as IP address and browser information. Currently, IFTTT does not check any user fingerprinting data so the attackers are able to use the stolen cookies from an arbitrary IP address with an arbitrary web browser. Even if IP address is too strict condition that changes quite frequently, especially on a laptop or a mobile device, it would be better to provide a more secure ``remember me'' feature with other user fingerprinting data such as time zone and screen size so that the attackers cannot use the stolen cookies without reproducing the same user fingerprinting data.

Next, IFTTT and similar services can introduce additional security measures. A watchdog feature that alerts users of any change in their account details and recipes will be a great addition. Modifying account settings or recipes (at least user-selected or CPS-related ones) may require an additional authentication method such as SMS and push message to a trusted device can be introduced for better security. These features are already available in several major Internet services including Google, Facebook and Twitter, and it is highly recommended for smart home related services to adopt them since the impact of a compromised account includes not only the loss of cyber assets but also damages to offline assets via thieving and higher electricity bills.

[2] Home router and NAS sometimes have a simple PC malware inspector and a firewall built in.

[3] Malware issue is more general than session hijacking so we are not covering mitigations for malware, but [34] is suggesting a feature-distributed malware specific defence.

Lastly, relatively new standards such as HTTP Strict Transport Security (HSTS) [27] and new techniques suggested by researchers for session hijacking prevention [47, 29, 45, 23, 21] can be employed. It will take some time for these new techniques to be adopted by the industry, but they will, once deployed, significantly improve the security of session and cookie management.

4. Analysis of previously proposed security techniques for IoT

In this Chapter, we present our analysis on the previously proposed techniques to deal with the attacks in IoT applications and infrastructures. Some of the work has focused on the usage of access control and cryptographic techniques for securing IoT applications from different attacks. As part of milestone 3 of this project, we aim to develop SDN based security architecture for IoT. Hence, we have also considered the techniques that make use of SDN for securing IoT applications.

Das et al. in [50] present a context-sensitive policy framework for IoT devices, to control and protect information sharing between them. Their policies capture the diverse nature of IoT devices and their interaction with network users using attribute-based access control policy. Their work mostly focuses on the privacy of the user data. Beetle [51] is an access control policy framework for operating systems (Linux, Android) to control application interaction with peripheral device resources, and provides transparent access to network devices. Later, Hong used the Beetle framework in home network gateways to control IoT communications [52]. This work does not address authentication of IoT devices, or users, which our architecture does consider. Other work on access control policies for IoT devices can be found in [53-55]. However, these works are mainly concerned with user security rather than IoT security.

Moosavi et al. [76] present a mobility enabled secure healthcare scheme for the IoT. The proposed scheme uses DTLS (Datagram Transport Layer Security) for end-user authentication and authorisation. However, it simply assumes users have valid credentials and policy management is outside the scope of this work. Tarouco et al. [77] explore the interoperability issues between various users and devices in an IoT system through WiFi/Bluetooth gateway supported by the Simple Network Management Protocol (SNMP) or Web services. Their proposal is based on an SNMP agent acting as a proxy between the user device and smart sensor. Access control will be implemented through the agent, but only a high-level description is given. Ren et al. [78] and Lee et al. [79], amongst others, employ ECC for securing healthcare systems. Using ECC, a high level of encryption becomes more practical and feasible as it reduces the key size and computational costs of public key cryptography, however these proposals do not examine wider issues of access control, e.g. policy specification and management.

Liu et al. [80] present an access control model for an IoT system combining ECC with RBAC. ECC is used for key establishment during entity authentication and RBAC is used for managing access control policies. Their approach to using RBAC is highly centralised, with all decisions being made in a central server, and all information being required beforehand. Zhang and Tian [81] present an access control model using RBAC and security-relevant contextual information (e.g. time, location, and environment) for an IoT system. The implication is that the system is highly centralised, which is not an ideal solution for the IoT.

Xu et al. present an IoT-based data accessing design for emergency medical services [82]. While this model shows how to collect, integrate and interoperate IoT data flexibly in order to provide support to emergency medical services it simply abstracts access control into the business activities layer of their model. Zhang and Liu [83] present an ABAC model to provide a fine-grained access control for IoT systems. The proposed model allows permissions to be assigned to a user for accessing resources based on user attributes, resource attributes, environment attributes and current tasks. While their framework includes policy decision and enforcement points, it is not clear where in the system these are implemented. Further, in their design policies also have to be written in advance, based on explicit user identity.

Several proposals discuss capability-based access control for IoT applications. These approaches enable access control to be tailored to the requirements of individual users and avoid fully centralised systems by providing users with access tokens (i.e. capabilities) which smart IoT devices can validate for access. Gusmeroli et al. [84] present a capability-based access control approach for IoT. A similar approach is proposed by Hernandez-Ramos et al. [85] which describes a distributed capability-based access control frame- work for IoT devices. A highly optimised version of Elliptic Curve Digital Signature Algorithm (ECDSA) is implemented inside these devices ensuring end-to-end authentication, integrity and non- repudiation. However, while offering significant detail on capability structure and processing neither of these proposals address capability distribution or questions of how the information defining which users have access to which capabilities is stored. They simply assume that a capability issuer exists. Mahalle et al. [86] and Ondiege et al. [87] present essentially similar capability-based proposal for access control in the IoT.

Some relevant related works in the context of network policy control in SDN include [56-60]. However these works did not address security aspects and in particular, did not provide fine grained security policy based network management for IoT services. A network activity based IoT device blacklisting approach has been proposed in [67]. The authors have used SDN features to dynamically block/ quarantine the malicious IoT devices using access control policies. They have shown how a Philips Hue light-bulb can be protected from user impersonation attack. This work only focuses on a particular type of IoT devices. SDN- WISE [80] makes use of the SDN Controller to limit the amount of information exchange between wireless sensor nodes. They introduced an API interface over the ONOS Controller to program the IoT nodes. Hesham et al. in [69] propose a simple Network Access Control (NAC) mechanism for machine-to-machine (M2M) devices using SDN. Their NAC architecture for M2M devices captures minimal network features (such as source and destination IP and bit rates). It is not as fine grained as our policies and does not consider the IoT device specific parameters in access control. Qin proposed MINA (Multinetwork INformation Architecture) [70], which uses SDN to manage IoT network infrastructure. The primary focus of this work is in the management of network services and scheduling flows. The work in [71] is concerned with the quality of service and considers the use of IoT infrastructure for managing large scale data generated from IoT sensors. Xu et al. in [72] used SDN to prevent flow attacks in IoT infrastructure. They utilised dynamic access control mechanism and real-time monitoring of the flow packets to defend the network attacks. Black SDN [73] used SDN features to encrypt the header and payload to prevent some attacks in the IoT infrastructure. Nobakth et al. in [74] proposed a host based intrusion detection system (IDS) for IoT infrastructure. This approach used a Floodlight SDN Controller to enforce the IDS in a smart home network domain and used machine learning algorithms to detect the attacks. Miettinen et al. in [75] developed an IoT device identification system known as IoT Sentinel. IoT Sentinel used SDN to extract IoT device specific attributes. In their model, legitimate IoT devices were identified using predictive modeling.

5. Security Requirements for IoT Applications

In this chapter, we provide some of the security requirements that need to be considered in the design of security architecture for IoT infrastructure and applications.

Monitoring: There is need for secure monitoring techniques to detect malicious devices and devices flows. In particular, there is need for techniques to dynamically isolate the malicious devices or flows from the network.

Core security functionality: There is need to provision core functionalities such as authentication, authorisation, confidentiality, privacy and trust throughout the architecture potentially at different levels of granularity.

Support for Composition: The IoT composes and aggregates services and data from devices to provide diverse applications to users. Appropriate security must be provided at each stage of this composition.

Scalability: The architecture must support the number of things and users that may appear in IoT systems.

Heterogeneity: The architecture must allow for the heterogeneous nature of the IoT in terms of devices, and technologies. The security design and implementation should not be dedicated to a specific technology, rather it should be technology agnostic and support interoperability of components.

Light-Weight Solutions: The architecture should recognise the resource-constrained nature of things and support light-weight solutions. This extends from the cryptographic techniques employed to the load placed on devices by the division of function in security applications.

Identity Management: The architecture should incorporate reliable techniques for registering devices and users. This would provide for seamless integration of various services generating from different devices and users across multiple domains. The architecture should also support flexible identity management and mutual authentication for users, devices and associated services.

Decentralised Management: The architecture must allow for a flexible approach to placement of security provisioning and management to address the issues around centralised versus decentralised architectures. Given the edge-intelligence present in many IoT systems and their potentially large scale, the security provisioning should be placed as close as possible to the point of need, while allowing for resource-constrained devices. This will likely take the form of decentralised management responsible for clustered portions of the system.

End-to-end Security: The architecture must support end-to-end security where the communication path may traverse multiple domains. This will require interoperable security technologies, inter-domain policy management between the endpoints and identities that are verifiable between the endpoints.

Robustness: IoT systems need to be robust due to such factors as mobility, device faults and the increase in the number of attack vectors. The security supporting these systems will need to demonstrate the same characteristics and support systems that are adaptive and self-healing.

Incremental Deployment: The scale and heterogeneity of IoT systems mean that they will often be deployed incrementally, with revisions and adaptations. The security functions must support this style

of deployment, allowing things and users to join and leave the system and system functionality to evolve.

Transiency: Relationships in the IoT may be fleeting, with users, devices and systems that have never encountered each other before interacting, possibly on a onetime-only basis.

There are obvious relationships between the requirements. For example, trade-offs may be involved between decentralisation and light-weight solutions. The requirements to support transiency and incremental deployment will affect the identity management and scalability provisions. We will use these requirements for designing SDN based Security Architecture as part of milestone 3.

6. Conclusion

In this report, we have presented the work done as part of first milestone report for the project "Software Defined Networks based Security Architecture for IoT infrastructures". As part of this milestone, we have conducted a detailed survey of attacks related to IoT, analysed previously proposed techniques and countermeasures to deal with the attacks and developed security requirements to deal with attacks in IoT.

In this report, we have first presented an overview of the IoT Architecture with discussion on the different layers of the architecture and protocols that are used at different layers of the IoT Architecture. Then we presented threat model for the IoT Architecture with discussion on the several attacks that are possible at different layers of the IoT Architecture. We also propose the design and development of feature distributed malware attacks for Internet of Things in the context of smart homes. Smart homes consist of various smart home devices, providing convenient services to the tenants. These devices are usually connected to the Internet, and the tenants can integrate individual devices and make smart home workflows using Internet services such as IFTTT. The focus of our work is on how the attackers can generate various cyber-physical and advanced cyber-attacks by exploiting this integration aspect in smart home. In particular, we have designed and developed an advanced feature distributed malware that can be used to perform various malicious activities such as stealing the victim's IFTTT cookies, using smart devices as information sources of the malware and distributing malware functionalities to other devices, performing malicious without being noticed, causing financial damages to the victim, and evasively exfiltrating data. Our experiments show that the proposed attacks enable the attackers to control any smart home device integrated with IFTTT, without requiring compromise of individual smart home devices. We believe our approach of using the integrating Internet service as an attack vector and distributing malware features to the smart home devices will be helpful to the development of more secure smart home environments and different IoT applications. Finally, we have analysed some of the previously proposed techniques to deal with the attacks in IoT applications and identified few security requirements that need to be considered for designing and for developing security architecture for IoT applications.

The authors would like to thank ISIF ASIA for their financial contribution to the Project.

References

1. S. Srirama, “Mobile web and cloud services enabling Internet of Things,” vol. 5, no. 1, CSI Trans. on ICT, Springer, pp. 109–117, 2017.
2. K. Ashton, “That ‘Internet of Things’ Thing,” RFID, June 2009.
3. H. Suo, J. Wan, C. Zou, and J. Liu, “Security in the Internet of Things: A Review,” in Int. Conf. on Computer Science and Electronics Engineering, vol. 3, IEEE, pp. 648–651, Mar. 2012.
4. R. Bogue, "Towards the trillion sensors market," Sensor Review, vol. 34, no. 2, pp. 137-142, 2014.
5. A. Roy et al., "Energy-efficient Data Centers and smart temperature control system with IoT sensing," in Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2016 IEEE 7th Annual, 2016, pp. 1-4: IEEE.
6. R. H. Weber, "Internet of Things–New security and privacy challenges," Computer law & security review, vol. 26, no. 1, pp. 23-30, 2010.
7. M. Medwed, "Iot security challenges and ways forward," in Proceedings of the 6th International Workshop on Trustworthy Embedded Devices, 2016, pp. 55-55: ACM.
8. L. A. B. Pacheco, J. J. Gondim, P. A. S. Barreto, and E. Alchieri, "Evaluation of Distributed Denial of Service threat in the Internet of Things," in Network Computing and Applications (NCA), 2016 IEEE 15th International Symposium on, 2016, pp. 89-92: IEEE.
9. X. Cao, D. M. Shila, Y. Cheng, Z. Yang, Y. Zhou, and J. Chen, "Ghost-in-ZigBee: Energy Depletion Attack on ZigBee based Wireless Networks," 2012.
10. L. Da Xu, W. He, and S. Li, "Internet of things in industries: A survey," IEEE Transactions on industrial informatics, vol. 10, no. 4, pp. 2233-2243, 2014.
11. J. Soldatos et al., "Openiot: Open source internet-of-things in the cloud," in Interoperability and open-source solutions for the internet of things: Springer, 2015, pp. 13-25.
12. A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "IoT middleware: A survey on issues and enabling technologies," IEEE Internet of Things Journal, vol. 4, no. 1, pp. 1-20, 2017.
13. Alvarez, S. Antivirus (In)Security. In CCC (Chaos Communication Camp) (Finowfurt, Germany, 2007).
14. Alvarez, S., and Zoller, T. The Death of AV Defense in Depth? - revisiting Anti-Virus Software. In CanSecWest (Vancouver, B.C., Canada, 2008).
15. Amin, S., Litrico, X., Sastry, S. S., and Bayen, A. M. Stealthy deception attacks on water SCADA systems. In HSCC '10: Proceedings of the 13th ACM international conference on Hybrid systems: computation and control (Stockholm, Sweden, Apr.2010).
16. Barnard-Wills, D., Marinos, L., and Portesi, S. Threat Landscape and Good Practice Guide for Smart Home and Converged Media. Tech. rep., enisa, Dec.2014.
17. Barth, A. HTTP State Management Mechanism, Apr. 2011. RFC 6265.
18. Bilge, L., and Dumitras, T. Before we knew it: an empirical study of zero-day attacks in the real world. In CCS '12 (Raleigh, NC, USA, Oct. 2012).
19. Bodnar, C. Don't Shop or Bank With a Smart TV. https://blog.kaspersky.co.uk/dont-shop-or-bank-with-a-smart-tv/, Feb. 2014.
20. Byres, E., Kounga, G., and Langill, J. How Stuxnet Spreads: A Study of Infection Paths in Best Practice Systems. Tech. rep., Tofino Security, Abterra Tech, ScadaHacker.com, Feb. 2011.
21. Calzavara, S., Tolomei, G., Bugliesi, M., and Orlando, S. Quite a mess in my cookie jar!:leveraging machine learning to protect web authentication. In Proceedings of the 23rd international conference on World wide web (Seoul, Republic of Korea, Apr. 2014).
22. Comodo. Blocking Internet Access while Allowing Local Area Network (LAN) Access. https://help.comodo.com/topic-72-1-451-4800-.html.

23. Dietz, M., Czeskis, A., Balfanz, D., and Wallach, D. S. Origin-Bound Certificates : A Fresh Approach to Strong Client Authentication for the Web. In Proceedings of the 21st USENIX Security Symposium (Bellevue, WA, USA, Aug. 2012).
24. Falliere, N., Murchu, L. O., and Chien, E. W32.Stuxnet dossier. Tech. rep., Symantec, 2011.
25. Goodin, D. Crypto weakness in smart LED lightbulbs exposes Wi-Fi passwords. http://arstechnica.com/security/2014/07/crypto-weakness-in-smart-led-lightbulbs-exposes-wi-fi-passwords/, July 2014.
26. [14] Hill, K. Baby Monitor Hacker Still Terrorizing Babies And Their Parents. http://www.forbes.com/sites/kashmirhill/2014/04/29/baby-monitor-hacker-still-terrorizing-babies-and-their-parents/, Apr. 2014.
27. Hodges, J., Jackson, C., and Barth, A. HTTP Strict Transport Security (HSTS), Nov. 2012. RFC 6797.
28. Jana, S., and Shmatikov, V. Abusing File Processing in Malware Detectors for Fun and Profit. In IEEE Symposium on Security and Privacy (S&P) 2012 (San Francisco, CA, USA, 2012), pp. 80{94.
29. Johns, M., Lekies, S., Braun, B., and Flesch, B. BetterAuth: Web Authentication Revisited. In Proceedings of the 28th Annual Computer Security Applications Conference (Orlando, Florida, USA, Dec.2012).
30. Kyas, O. How To Smart Home: A Step by Step Guide to Your Personal Internet of Things, 3rd edition. Key Concept Press, Mar. 2015.
31. Lawler, R. Nest Learning Thermostat has its security cracked open by GTVHacker. http://www.engadget.com/2014/06/23/nest-thermostat-rooted/, June 2014.
32. McAllister, N. You THINK you're watching your LG smart TV - but IT's WATCHING YOU, baby. http://www.theregister.co.uk/2013/11/20/lg_smart_tv_data_collection/, Nov. 2013.
33. Min, B., and Varadharajan, V. Design and Analysis of Security Attacks against Critical Smart Grid Infrastructures. In 19th International Conference on Engineering of Complex Computer Systems (ICECCS) (Tianjin, China, June 2014), pp. 59-68.
34. Min, B., and Varadharajan, V. Feature-Distributed Malware Attack: Risk and Defence. In 19th European Symposium on Research in Computer Security (ESORICS) (Wroclaw, Poland, 2014), pp. 457-474.
35. Miorandi, D., Sicari, S., De Pellegrini, F., and Chlamtac, I. Internet of things: Vision, applications and research challenges. Ad Hoc Networks 10, 7 (Sept.2012), 1497-1516.
36. MSDN. Design, Develop, and Certify Hardware for Windows 8. http://msdn.microsoft.com/en-us/library/windows/hardware/br259114.aspx, 2012.
37. MSDN. Early Launch Anti-Malware.
38. http://msdn.microsoft.com/en-us/library/windows/hardware/br259096, 2012.
39. Oberheide, J., Bailey, M., and Jahanian, F. PolyPack: an automated online packing service for optimal antivirus evasion. In Proceedings of the 3rd USENIX Workshop on offensive technologies (Montreal, Canada, 2009).
40. O'Kane, P., Sezer, S., and McLaughlin, K. Obfuscation: The Hidden Malware. IEEE Security & Privacy 9, 5 (2011), 41-47.
41. Porst, S. How to really obfuscate your PDF malware. In ReCon (Montreal, Canada, July 2010).
42. Rad, B. B., Masrom, M., and Ibrahim, S. Camouflage in Malware: from Encryption to Metamorphism. International Journal of Computer Science and Network Security 12, 8 (Aug. 2012), 74-83.
43. Robles, R. J., Kim, T., Cook, D., and Das, S. A review on security in smart home development. International Journal of Advanced Science and Technology 15, 1 (2010), 13{22.

44. Roman, R., Najera, P., and Lopez, J. Securing the Internet of Things. Computer 44, 9 (2011), 51-58.
45. Ryck, P. D., Nikiforakis, N., Desmet, L., Piessens, F., and Joosen, W. Serene: self-reliant client-side protection against session fixation. In Distributed Applications and Interoperable Systems (Stockholm, Sweden, June 2012).
46. Seitz, J. Black Hat Python: Python Programming for Hackers and Pentesters. No Starch Press, Dec. 2014.
47. Tang, S., Dautenhahn, N., and King, S. T. Fortifying web-based applications automatically. In Proceedings of the 18th ACM Conference on Computer and Communications Security (Seoul, Republic of Korea, Oct. 2011).
48. Weber, R. H. Internet of Things - New security and privacy challenges. Computer Law & Security Review 26, 1 (Jan. 2010), 23030.
49. Yuan, X., and Peng, S. A research on secure smart home based on the Internet of Things. In International Conference on Information Science and Technology (ICIST) (2012), IEEE, pp. 737-740.
50. P. K. Das et al., "Context-sensitive policy based security in internet of things," in Smart Computing (SMARTCOMP), 2016 IEEE International Conference on. IEEE, 2016, pp. 1–6.
51. A. A. Levy, J. Hong, L. Riliskis, P. Levis, and K. Winstein, "Beetle: Flexible communication for bluetooth low energy," in Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, ser. MobiSys '16. New York, NY, USA: ACM, 2016, pp. 111–122.
52. J. Hong et al., "Demo: Building comprehensible access control for the internet of things using beetle," in Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services Companion, ser. MobiSys '16 Companion. New York, NY, USA: ACM, 2016, pp. 102–102. [Online]. Available: http://doi.acm.org/10.1145/2938559.2938578
53. A. L. M. Neto et al., "Aot: Authentication and access control for the entire iot device life-cycle," in Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM. ACM, 2016, pp. 1–15.
54. C.-J. M. Liang et al., "Sift: building an internet of safe things," in Proceedings of the 14th International Conference on Information Processing in Sensor Networks. ACM, 2015, pp. 298–309.
55. A. A. Yavuz, "Eta: efficient and tiny and authentication for heterogeneous wireless systems," in Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks. ACM, 2013, pp. 67–72.
56. N. Foster et al., "Frenetic: A network programming language," in ACM SIGPLAN Notices, vol. 46, no. 9. ACM, 2011, pp. 279–291.
57. T. L. Hinrichs et al., "Practical declarative network management," in Proceedings of the 1st ACM workshop on Research on enterprise networking. ACM, 2009, pp. 1–10.
58. J. Reich et al., "Modular sdn programming with pyretic," Technical Reprot of USENIX, 2013.
59. A. Voellmy et al., "Nettle: Taking the sting out of programming network routers," in Practical Aspects of Declarative Languages. Springer, 2011, pp. 235–249.
60. ——, "Maple: simplifying sdn programming using algorithmic policies," in ACM SIGCOMM Computer Communication Review, vol. 43, no. 4. ACM, 2013, pp. 87–98.
61. K. K. Karmakar et al., "Policy based security architecture for software defined networks," in Proceedings of the 31st Annual ACM Symposium on Applied Computing. ACM, 2016, pp. 658–663.
62. P. Pongle et al., "A survey: Attacks on rpl and 6lowpan in iot," in Pervasive Computing (ICPC), 2015 International Conference on. IEEE, 2015, pp. 1–6.
63. M. Lyu, D. Sherratt, A. Sivanathan, H. H. Gharakheili, A. Radford, and V. Sivara- man, "Quantifying the reflective ddos attack capability of household iot devices," in Proceedings of

the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks. ACM, 2017, pp. 46–51.
64. C. V. Mendoza et al., “Defense for selective attacks in the iot with a distributed trust management scheme,” in Consumer Electronics (ISCE), 2016 IEEE International Symposium on.IEEE, 2016, pp. 53–54.
65. Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, “Iotpot: analysing the rise of iot compromises,” EMU, vol. 9, p. 1, 2015.
66. M. Capellupo, J. Liranzo, M. Z. A. Bhuiyan, T. Hayajneh, and G. Wang, “Security and attack vector analysis of iot devices,” in International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage. Springer, 2017, pp. 593–606.
67. V. Sivaraman, H. H. Gharakheili, A. Vishwanath, R. Boreli, and O. Mehani, “Network-level security and privacy control for smart-home iot devices,” in Wireless and Mobile Computing, Networking and Communications (WiMob), 2015 IEEE 11th International Conference on. IEEE, 2015, pp. 163–167.
68. L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo, “Sdn-wise: Design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks,” in Computer Communications (INFOCOM), 2015 IEEE Conference on. IEEE, 2015, pp. 513–521.
69. A. Hesham et al., “A simplified network access control design and implementation for m2m communication using sdn,” in Wireless Communications and Networking Conference Workshops (WCNCW), 2017 IEEE. IEEE, 2017, pp. 1–5.
70. Z. Qin et al., “A software defined networking architecture for the internet-of-things,” in Network Operations and Management Symposium (NOMS), 2014 IEEE. IEEE, 2014, pp. 1–9.
71. Y. Lu et al., “Sdtcp: Towards datacenter tcp congestion control with sdn for iot applications,” Sensors, vol. 17, no. 1, p. 109, 2017.
72. T. Xu et al., “Defending against new-flow attack in sdn-based internet of things,” IEEE Access, 2017.
73. S. Chakrabarty et al., “Black sdn for the internet of things,” in Mobile Ad Hoc and Sensor Systems (MASS), 2015 IEEE 12th International Conference on. IEEE, 2015, pp. 190–198.
74. M. Nobakht et al., “A host-based intrusion detection and mitigation framework for smart home iot using openflow,” in Availability, Reliability and Security (ARES), 2016 11th International Conference on. IEEE, 2016, pp. 147–156.
75. M. Miettinen et al., “Iot sentinel: Automated device-type identification for security enforcement in iot,” in Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on. IEEE, 2017, pp. 2177–2184.
76. S. Moosavi, T. Gia, E. Nigussie, A. Rahmani, S. Virtanen, H. Tenhunen, and J. Isoaho. 2016. End-to-end security scheme for mobility enabled healthcare Internet of Things. Elsevier Future Generation Computer Systems, 64 (Nov. 2016), 108–124.
77. L. Tarouco, L. Bertholdo, L. Granville, L. Arbiza, F. Carbone, Marcelo Marotta, and Jose J. de Santanna. 2012. Internet of Things in healthcare: Interoperatibility and security issues. In Procedings of the International Conference on Communications (ICC). IEEE, 6121–6125.
78. Y. Ren, R. Werner, N. Pazzi, and A. Boukerche. 2010. Monitoring patients via a secure and mobile healthcare system. IEEE Wireless Communications 17, 1 (Feb.2010), 59–65.
79. Y. Lee, E. Alasaarela, and H. Lee. 2014. Secure key management scheme based on ECC algorithm for patient’s medical information in healthcare system. In Procedings of the International Conference on Information Networking 2014 (ICOIN). IEEE, 453–457.
80. J. Liu, Y. Xiao, and C. Chen. 2012. Authentication and Access Control in the Internet of Things. In Procedings of the 32nd International Conference on Distributed Computing Systems Workshops. IEEE, 588–592.

81. G. Zhang and J. Tian. 2010. An extended role based access control model for the Internet of Things. In The International Conference on Information, Networking and Automation (ICINA), Vol. 1. IEEE, 319–323.
82. B. Xu, L. Xu, H. Cai, C. Xie, J. Hu, and F. Bu. 2014. Ubiquitous Data Accessing Method in IoT-Based Information System for Emergency Medical Services. IEEE Transactions on Industrial Informatics 10, 2 (May 2014), 1578–1586.
83. G. Zhang and J. Liu. 2011. A Model of Workflow-oriented Attributed Based Access Control. International Journal of Computer Network and Information Security 3, 1 (2011), 47–53.
84. S. Gusmeroli, S. Piccione, and D. Rotondi. 2013. A capability-based security approach to manage access control in the Internet of Things. Mathematical and Computer Modelling 58, 5-6 (Sept. 2013), 1189–1205.
85. J. Hernández-Ramos, A. Jara, L. Marín, and A. Skarmeta. 2013. Distributed Capability-based Access Control for the Internet of Things. Journal of Internet Services and Information Security 3, 3/4 (Nov. 2013), 1–16.
86. P. Mahalle, B. Anggorojati, N. Prasad, and R. Prasad. 2013. Identity Authentication and Capability Based Access Control (IACAC) for the Internet of Things. Journal of Cyber Security and Mobility 1, 4 (10 Mar. 2013), 309–348.
87. B. Ondiege, M. Clarke, and G. Mapp. 2017. Exploring a New Security Framework for Remote Patient Monitoring Devices. Computers 6, 1 (2017).