A Host-based Intrusion Detection and Mitigation Framework for Smart Home IoT using OpenFlow

Mehdi Nobakht
Data61 | CSIRO, Australia and
University of New South Wales (UNSW)
Syndey, Australia
Email: mehdi.nobakht@data61.csiro.au

Vijay Sivaraman
University of New South Wales (UNSW)
Syndey, Australia
Email: vijay@unsw.edu.au

Roksana Boreli
National ICT Australia (NICTA)
Syndey, Australia
Email: roksana.boreli@nicta.com.au

***Abstract*—Smart devices are gaining popularity in our homes with the promise to make our lives easier and more comfortable. However, the increased deployment of such smart devices brings an increase in potential security risks. In this work, we propose an intrusion detection and mitigation framework, called IoT-IDM, to provide a network-level protection for smart devices deployed in home environments. IoT-IDM monitors the network activities of intended smart devices within the home and investigates whether there is any suspicious or malicious activity. Once an intrusion is detected, it is also capable of blocking the intruder in accessing the victim device on the fly. The modular design of IoT-IDM gives its users the flexibility to employ customized machine learning techniques for detection based on learned signature patterns of known attacks. Software-defined networking technology and its enabling communication protocol, OpenFlow, are used to realise this framework. Finally, a prototype of IoT-IDM is developed and the applicability and efficiency of proposed framework demonstrated through a real IoT device: a smart light bulb.**

Keywords*—*Internet of Things (IoT); Smart-home; SDN; OpenFlow; Machine learning; Anomaly detection; Attack mitigation

I. Introduction

The phenomenon of the Internet of Things (IoT) is about connecting devices over the Internet. There are many applications of IoT in different fields ranging from industry automation and education to smart homes and smart cities. In its home application, consumers are able to control and manage smart appliances within and outside the home. As such these smart devices bring intelligence to our traditional appliances by incorporating computational and network capabilities and employing sensors and actuators. In a more advanced smart home, smart appliances can be programmed to operate autonomously, e.g., a smart fridge can monitor its contents and automatically order foods online when stocks are running low. IoT devices are becoming more pervasive and being flooded into the market at a large scale. According to a prediction by Cisco's Internet of Things Group, there will be over 50 billion connected devices by 2020 [1], outnumbering the entire world population by 6.5.

While these smart devices promise to make our lives easier, they also raise a set of security and privacy concerns. Technically savvy adversaries can gain access to smart home appliances and utilities and create emergent threats to people's possessions. A report by the BBC, for example, warned of how thousands of baby monitors are accessible to anyone from the Internet [2]. These security exposures may arise due to a number of different factors such as insufficient authentication and authorization, lack of transport encryption and insecure firmware. Thanks to cheap hardware and open source software, making IoT devices is not hard today. Indeed, there are hundreds of IoT-related companies manufacturing these devices. However, as revealed in an HP report [3], security is not taken seriously in the development of many of these devices. The report was part of Open Web Application Security Project (OWASP) that reviewed 10 popular IoT devices ranging from TVs and remote power outlets to door locks and scales. The results of this study showed the high average number of vulnerabilities per device. Regardless of these vulnerabilities, even a more secure IoT device might be compromised when it is managed poorly due to the lack of its user's expertise.

As the lack of security and privacy properties raise concerns in deploying such smart appliances throughout our homes, it is crucial to monitor the events occurring in today's home networks and analysing them for signs of potential security risks associated with these devices. Once a security attack is identified, it is also necessary to consider a proper defence mechanism to stop the adversary from having a harmful effect. One possible approach for securing the myriad smart devices within the home is to redesign and embed security agent inside them. However, such an approach would not scale, nor would be affordable. This motivates us to propose a framework that will be capable of detecting and mitigating security threats within smart home environments at network-level, whereby it does not require any amendment to their design. To this end, we propose utilizing the capabilities of Software-defined Networking (SDN) architecture and in particular OpenFlow protocol to realise this framework. The main contributions of this study are summarized below.

- We propose a host-based Intrusion Detection and Mitigation framework called IoT-IDM, which provides computer security services for risks associated with smart IoT devices within the home. IoT-IDM harnesses (***i***) the advent of SDN technology, which offers network visibility and provides flexibility to configure, manage and secure the network remotely and (***ii***) the maturity of machine learning techniques in detection of network anomaly patterns.
- We implement IoT-IDM in Java as module in Floodlight; a well-known SDN controller.
- Finally, we demonstrate the applicability of IoT-IDM

by conducting a case study through a popular smart lighting system. The results show that our framework works well and incurs limited computation and communication overhead.

The rest of this paper is organized as follows. Relevant prior work is summarized in § II. § III provides a background to IDS and OpenFlow. A high-level overview of IoT-IDM is given in § IV followed by a description of the principal components of it. We describe our prototype implementation in § V. In § VI, we describe a real smart home IoT device to demonstrate the applicability of our approach. Then, we evaluate the feasibility of IoT-IDM in § VII and finally the paper concludes in § VIII.

II. Related work

Smart home environments are envisaged as a key part of IoT applications. The widespread adoption of smart devices throughout today's homes led researchers focus on security and privacy properties of home technologies and evaluate the level of such important properties as security breaches within home technologies will result in variety of harmful impacts on end users. In this section, we first survey the risk associated with home-based IoT, then look at early research on securing IoT in general, home-based IoT, and using SDN to provide security.

A. Risk Analysis

There are several research endeavours in the community focusing on risk analysis of IoT and its home application to gauge the impact of security attacks against them. Denning et al. [4] survey potential security attacks against home-based IoT and provide a structure for reasoning about the different security needs. They consider various attack scenarios against such smart devices including traditional attacks as well as novel attacks that are not viable with traditional computing platforms. Furthermore, they propose an informative framework to evaluate a risk posed by in-home IoT based on three components: the feasibility of an attack on the system, the attractiveness of the system as a compromised platform, and the damage caused by executing a successful attack. The human assets and security goals are two elements used in their work to reason the impact of potential attacks. The advantage of this paper is the evaluation of the risks associated with smart home devices. However, they do not elaborate the proposed framework to evaluate traditional attacks.

Andreas et al. [5] review the state of the art in the context of smart home IoT security and privacy, and apply a risk analysis to evaluate smart home automation system vulnerabilities and threats and their potential impact. They used Information Security Risk Analysis (ISRA) method by Pelrier [6]. In the ISRA method, the system's risk exposure is systematically reviewed based on the three basic requirements: confidentiality, integrity, and availability. They view a smart home automation system consisting of six elements: connected sensors/devices, a home gateway, a cloud server, a mobile device, mobile device apps, and an API as an interface between mobile device apps and smart home devices. In such architecture, a total of 32 risks were identified and associated risk values calculated in a range of 1-25. The strong merit with this paper is the systematic evaluation of risks associated with home-based IoT. However, as smart home automation systems often have heterogeneous architectures, the proposed risk analysis would not be applicable to different technology designs. Furthermore, their approach is more beneficial in the design and development phases rather than attack identification and prevention during the operational phase.

B. Securing IoT

The research into IoT security is in its early stage and many work focus on identifying potential threats [7], [8], [9]. Some solutions for specific problems are also proposed. In the work by Reza et al. [10], an intrusion detection system is proposed for 6LoWPAN (a compressed version of the IPv6) networks of IoT. They primarily targeted network layer and routing attacks such as sinkhole and selective-forwarding attack. For evaluation, an implementation of the proposed IDS has been implemented in the Contiki OS for IoT. However, their approach is not applicable to home-based IoT as 6LoWPAN and Contiki OS are mainly used in IP-connected wireless sensor networks and not common in smart home automation systems. In another attempt, Prabhakaran et al. in [11] and [12] propose an IDS framework for IoT empowered by 6LoWPAN. They aimed Denial of Service (DoS) attacks such as jamming, cloning, eavesdropping and routing attack. The developed IDS framework implemented within the EU FP7 project ebbits, which is not a common environment for smart home systems. On the contrary, the scope of attacks in our work is not limited to routing attacks; but instead, we focus on considering other attacks as well; such as unauthorized access to home-based smart devices, which are among OWASP top ten IoT vulnerabilities.

In smart home context, Lee et al. [13] pointed out the increasing popularity of smart home application of the emerging IoT and the need to provide adequate level of protection for potential cyber-attacks against these resource-constrained smart devices. They review the smart home technologies (applications, devices, operating systems, and communication protocols) and discuss the main security challenges and threats against them. Das et al. [14] designed a security system providing essential security for home automation system. They use the information from motion detectors and video capturing devices to manage operations on smart home appliances like TV, lighting systems, and microwave. Their security system has two main components: an iOS application and the server-side script running from a cloud server. While this approach seems to perform well in case enough contexts would be available from capturing devices, it is obvious that it poses a major limitation when no video capturing or motion detector devices are being used within the network.

C. Network security using SDN

There is a growing body for prior work on using SDN to manage network security for various scenarios including campus, business or home. Several research works have investigated SDN ability to protect the network with different intentions [15], [16]. SDN applications for network security ideally must be able to detect and isolate network devices that have been compromised. Feamster in [17] proposes to outsource the management and operation of home and small enterprise networks to a professional third party who has

security expertise and a general view of network activities. In another attempt, authors in [18] use flow statistics in SDN architectures to reveal anomalies by malicious events such as DDoS, worm propagation and port scan. The merit of this paper is using flow statistics to detect anomalies, which reduces the overhead on central controller. However, this approach is unable to detect other types of attacks such as insufficient authentication/authorization.

III. Background

Before explaining our proposed framework, we briefly review the enabling technologies used by IoT-IDM.

A. IDS

Intrusion Detection is the process of monitoring the events happening in a computer network system and inspecting them for signs of possible threats [19]. An Intrusion Detection Systems (IDS) is a device or software application aiming for such detection. There are two main types of such systems: *network-based* and *host-based* IDS. A network-based IDS monitors network traffic for a particular network segment and analyses different network layers to identify possible threats and suspicious activity, e.g., unusual traffic flows like Denial of Service (DoS) attack.

A host-based IDS monitors the activities and characteristics of a single-host in a network for suspicious activities [19]. Host-based IDSs often deploy a detection unit known as agents installed on the target hosts. Some host-based IDS products may use dedicated agents on remote devices instead of placing them on individual hosts. In this case, each remote agent is positioned to monitor the network traffic going to and from a particular host. Technically, these agents could be considered network-based IDSs, as they are deployed inline to monitor network traffic. However, they usually monitor activity for only one specific host.

In typical IDSs for computer networks, sensor elements monitor the network activities of target systems. The information events acquired by these elements later are being analysed for signs of possible incidents by other components. Sensors elements are typically deployed as *inline* or *passive*. In inline sensors, the network traffic pass through them and then being forward to the target hosts, whereas passive sensors monitor a copy of the actual network traffic. In legacy network, switches equipped with *spanning port* enable their users to monitor the network traffic in a passive manner.

IDSs typically use three main methodologies to investigate incidents: (***i***) *signature-based*, (***ii***) *anomaly-based*, and (***iii***) *stateful protocol analysis* [19]. When a threat is known, the pattern corresponds to that threat or signature can be identified. Signature-based detection (aka misuse detection) is a process in which signatures are compared against observed network traffic to identify possible malicious activity.

Anomaly-based detection can be used when there is no previously known threat. In this process, the normal activity is compared against the observed network activity in an attempt to identify any deviation between them. Thus, a profile representing the normal behaviour needs to be developed. The normal behaviour profile is required to be updated as the system and network change over time.

Stateful protocol analysis (aka deep packet inspection) is relying on deep inspection of protocol activity and tracks the state of application protocols used by network or hosts. In this method in fact a normal activity of a protocol is compared with observed events to identify deviations. For example, inspecting the status code in an authentication process can reveal the suspicious activity to access an IoT device using stateful protocol analysis.

B. OpenFlow

Switches/routers in Computer networks have two main functionalities. The first is *control plane*, which is responsible to make decisions about where traffic is sent, and the second is *data plane* which forwards traffic to selected destination. In legacy networks, switches/routers run proprietary software where these two functions are coupled. Software-Defined Networking (SDN) is a new promising architecture of computer networking in which the control plane is decoupled from the data plane [20]. SDN deploys control plane remotely from a software-based entity called *controller*. The controller is implemented in software in a logically centralized manner, and the data plane is implemented in commodity hardware. The controller communicates with the data plane using compliant protocols. In this way, SDN enhances network visibility, scalability and virtualization. The SDN architecture with its rich functionalities in traffic monitoring and management can also be utilized to perform traffic anomaly detection algorithms.

OpenFlow [20], which is an open standard managed by Open Networking Foundation, is leading and dominant protocol suggested for communication interface between the controller and underlying switches. It enables remote controllers to modify the behaviour of networking devices through a well-defined forwarding instruction set; in this way, the controller determines the path of network packets through the network of underlying switches. In OpenFlow protocol, each flow entry consists of *match fields*, *counters*, and a set of *instructions* to apply to matching packets. Matching starts at the first flow table and may continue to additional flow tables. If a matching entry is found, the instructions associated with the specific flow entry are executed. If no match is found in a flow table, the outcome depends on configuration of the table-miss flow entry. For example, the packet may be forwarded to the controller over the OpenFlow channel, dropped, or may continue to the next flow table.

IV. IoT-IDM: Host-based Intrusion Detection and Mitigation

The most relevant contribution of this paper is proposing a framework to detect computer security attacks against smart objects within home networks and to trigger the proper defence actions to countermeasure the identified attacks. We consider the following principles in the design of this framework. First of all, the design should be able to provide security services remotely and the end users of home-based IoT should not be burdened with such heavy tasks. Typical home users of these smart devices often lack enough expertise and vigilance to secure a network of in-home smart devices, which often pose heterogeneous architectures and varying degrees of security properties. To address this challenge, we propose employing SDN technology as it provides the possibility for remote

management. Thus, a third entity, which has security expertise, could take responsibility for security management on behalf of end users and provide Security as a Service (SaaS).

The second design objective is the efficiency of the framework: the scheme should incur low communication and computation overheads on the home router. For this requirement, we consider host-based IDS to design our solution. This limits the volume of the network traffic that needed to be analysed, as host-based IDSs monitor the activities of a single host of interest for suspicious and malicious activities. Furthermore, the scheme should be able to filter out even the network traffic of the specified host to reduce the overhead introduced to the home router. For instance, if the detection method requires investigating the traffic of a particular protocol (like HTTP) or a specific source or destination IP address, appropriate filters must be considered to reduce the volume of overall traffic.

Last but not the least, the framework should consider scalability as new smart home IoT are emerging in our lives at a fast pace from various manufacturers. Novel technologies are incorporated within these emerging devices to provide a wide variety of capabilities. This trend leads to a great degrees of heterogeneity in their architectures and in turn could create different types of security exposures. To cope with this challenge, the framework needs to support the new coming devices and technologies at a large scale.

To substantiate our claim, we propose a host-based Intrusion Detection and Mitigation framework for home-based IoT, named IoT-IDM. It aims to provide security services using SDN technology along with machine learning techniques. The IoT-IDM framework, which we place it on the top of SDN controller, consists of five key modules: *Device Manager*, *Sensor Element*, *Feature Extractor*, *Detection*, and *Mitigation*. Fig. 1 shows a high-level view of IoT-IDM including its key modules and the communication between IoT-IDM with a typical network of smart home. In this section, we describe the IoT-IDM architecture and explain its main components in details. We also mention the advantage of our framework over to the already existing solutions.

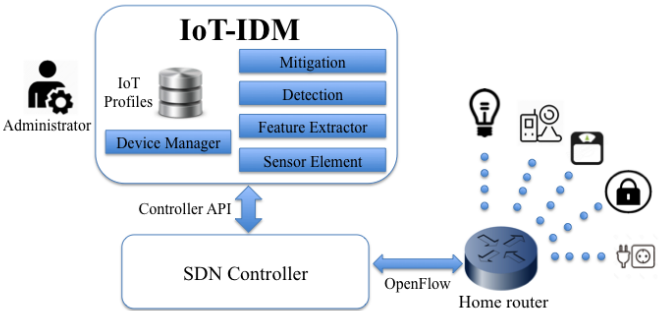


Fig. 1. IoT-IDM Architecture and a typical smart home network.

A. Device Manager

The IoT-IDM architecture incorporates a database to simplify the storage and management of a list of all known in-home IoT devices along with the potential security risks associated with them, the detection method of known attacks, and if it is applicable, appropriate defense mechanisms. The SaaS provider for home networks maintains this database and updates the repository as new devices appear in the market. This database, known as *IoT Profile*, can be located inside the device manager module or on a remote site that is accessible by this module. Remote database helps easy scalability of IoT-IDM. The profile in general contains: **(*i*)** an ID associated with the IoT device, **(*ii*)** attributes to create customized rules to redirect the traffic toward IoT-IDM, **(*iii*)** traffic features, to be used for classification, and **(*iv*)** a classification model, to be used for detection process. A detailed explanation will be provided later when this information is being used by IoT-IDM.

At present, the presence of smart devices is not discovered automatically and needs to be done manually. We plan to employ discovery techniques such as SSDP protocol for the registration purpose in the future work. In the first step, the administrator of IoT-IDM is required to register in-home IoT devices of interest, which are being used in the home. The required information for registration is **(*i*)** *device ID* and **(*ii*)** *device location*. The device ID is a unique identifier representing the IoT device type and corresponds to the device ID in the database of profiles. The device location represents the network location where the IoT device is installed and can be either the IP address of the device or the port number of OpenFlow-enabled switch connected to the device.

B. Sensor Element

IoT-IDM framework adopts the inline sensor to monitor network activities on a target IoT device. Once an IoT device has been registered to monitor for suspicious activities, IoT-IDM builds a virtual inline sensor as an application on top of the SDN controller. It then creates and pushes out OpenFlow rules in underlying switches in an attempt to redirect the traffic between the IoT device and the rest of the network toward the sensor element. In this way, only the network traffic of a registered IoT device will pass through the sensor element, thereby less traffic needs to be processed and the load on SDN controller will be decreased. Sensor elements in IoT-IDM also perform logging of network activities of associated IoT devices. This data can be used in later steps to investigate incidents.

The IoT-IDM framework is capable of detecting wide range of attacks. However, since the sensor element in IoT-IDM is positioned on top of SDN controller and due to the high volume of network traffic in general, it is not technically feasible to use IoT-IDM for intrusion detection and mitigation process of all devices in a home network. Thus, this approach is only applicable on selected smart home IoT devices in the network that do not add a lot of overhead on SDN controller.

C. Feature Extractor

The network traffic of a registered IoT device captured by the sensor element is used to extract features out of it. This framework gives its user (SaaS provider) the flexibility to decide which fields need to be extracted, as the entire traffic is available. Therefore, the users of IoT-IDM have the ability to select features depends on different problem instances.

Features basically represent the most critical aspects of the network traffic, so their definitions require extensive domain

knowledge of the hosts and the possible attacks as well. Features can be *packet-based* or *flow-based*. Packet-based features are a set of IP packet attributes including *source* or *destination IP address*, *Layer 3 protocol type*, e.g., TCP or UDP, *source* or *destination port* and finally *class of service*, e.g., HTTP, telnet, etc. A network *flow* is a unidirectional sequence of packets that all shares the IP packet attributes [21]. Thus, it is required to aggregate packets in each flow into a single flow object to extract flow-based statistics out of them. Table I shows a few examples of flow-based features.

TABLE I. EXAMPLES OF FLOW-BASED FEATURES

feature name	*description*
src IP	Source IP Address
dst IP	Destination IP Address
duration	length of the flow
protocol type	type of the protocol, e.g. TCP, UDP
service	network service on the destination like HTTP
src bytes	number of data bytes from source to destination
dst bytes	number of data bytes from destination to source
mean fit	mean amount of time between two packets sent in forward direction
mean bit	mean amount of time between two packets sent in backward direction

D. Detection Unit

IoT-IDM framework examines the network traffic captured by sensor elements or the statistics of network traffic extracted from the previous step to identify suspicious activities. The programmability aspect of SDN technology enables IoT-IDM framework to utilize any machine learning algorithms to build predictive models for detecting malicious traffic. The detection process through this framework can deploy signature-based, anomaly-based or stateful protocol analysis methodologies either separately or integrated to provide efficient and accurate detection. Due to the modular design of IoT-IDM, the SaaS providers using this framework have the ability to design the detection module, which fits best to the characteristics of attack vectors.

Once a suspicious activity is detected, the detection module in IoT-IDM framework identifies attacker and the victim of the attack. This module then either raises an alert when a suspicious activity is found but the attack mitigation is not possible, or exposes all related information to the mitigation module.

E. Mitigation

The mitigation module aims to take appropriate countermeasures to prevent identified attacks from having harmful effect. IoT-IDM uses OpenFlow rules in order to either block or redirect the malicious traffic in accessing the target IoT device. As a host within the network might be compromised to launch the attack, IoT-IDM takes advantage of OpenFlow, which gives the flexibility to quarantine the infected host or just limit its access to the IoT device.

V. IMPLEMENTATION

We implemented a prototype of IoT-IDM using Floodlight [22], a well-known open SDN controller for OpenFlow networks. Floodlight can be freely used almost for any purposes under Apache-license. The core of Floodlight controller comprises Java-based modules implementing basic network services to inquire and control an OpenFlow network. Apart from core modules, Floodlight comes with a collection of application modules for different purposes, such as a module for installing a specific flow entry into a specific switch or the load balancer module. The application modules have been located on top of the core modules and Java APIs interface them together. In addition, Floodlight modules may expose REST APIs via specified REST port. In this way, REST applications would be able to retrieve information and invoke services using HTTP REST commands.

Corresponding to components of IoT-IDM framework, the five key modules of the prototype are realised as Floodlight application modules, and have been implemented in approximately 1,100 lines of Java. Fig. 2 illustrates these sub-modules and their instantiation hierarchy. The base class and the starting point of IoT-IDM module is *IoT-IDM* class. It implements the necessary Floodlight listeners. Once initiated it creates an instance of *Device Manager* to start the process. At first, *Device Manager* creates a data structure to hold the profiles of IoT devices. In current version, the administrator must register the intended devices along with related information including device ID and location in the module directly. An interface application would place between the administrator and this module in the future version to facilitate registration and updating the related information.

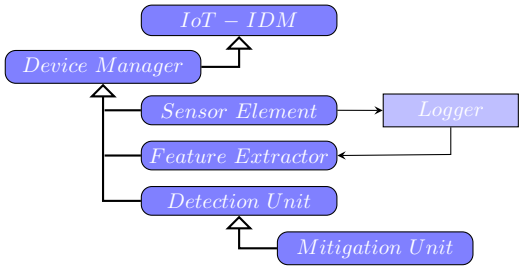


Fig. 2. Components of IoT-IDM and their relationship.

Once an IoT device is registered in *Device Manager* and its corresponding profile is created, it then creates instances of three other classes: *Sensor Element*, *Feature Extractor* and *Detection Unit*. *Device Manager* is also responsible for cleaning up the objects of disconnected IoT devices.

Sensor Element is instantiated to monitor and log the IoT device activities and basically responsible for two tasks. First, it pushes out OpenFlow rules to underlying OpenFlow switches to redirect all traffic going from and to the IoT device to itself. It uses Floodlight *Static Flow Pusher* REST API to create two rules signifying a bidirectional communication. These two flows are then will be inserted in the flow table of destination OpenFlow switch. Second, it logs the redirected traffic to be used for further extensive investigations.

Feature Extractor is responsible for extracting the necessary information from the logged flows in a structure that the *Detection Unit* can interpret. It first queries the *Device Manager* module for a set of features to be used for classification. Then, it parses the logged flows to derive features. Finally, it uses a simple data structure to store the feature statistics.

Detection Unit module first fetches the stored classification model from *Device Manager* and then applies this model on the statistic of features to predict whether the communication is legitimate or not. In this way, training needs to be done in advance and in off-line manner. It also creates an instance of *Mitigation Unit* to avert the possible attacks using static rules blocking intruders dynamically.

VI. Case Study

In order to demonstrate the applicability of IoT-IDM, we selected one of the popular smart home IoT devices, Philips Hue. It is a smart lighting system that can be controlled wirelessly. Compromised security of lighting systems can have non-trivial consequences, for example, an intruder might gain access of lighting systems in a home or a public venue and remotely shut off lights resulting in accidents. We briefly describe the architecture of Philips Hue and discuss possible attack vectors against it. This provides the context for the security mechanism that will be used by IoT-IDM.

Philips Hue is a smart lighting system that allows its user to wirelessly control a series of light bulbs. The user can switch On/Off the lights and also set their brightness and colour. The Hue lighting system consists of four components: **(*i*)** a series of Hue light bulbs, **(*ii*)** an Android/iOS App, as a way of controlling the lights within the home network, **(*iii*)** a web-based control panel, allowing a user of Hue to control the lights from anywhere in the Internet, and **(*iv*)** an Ethernet enabled bridge for communication between the App/web portal and the light bulbs. A wired connection of the bridge to the home router provides communication to the outside world and wireless *Zigbee* protocol is a means of communication between the bridge and the light bulbs. In the Hue's architecture model, the bridge acts as server that accepts commands from its users via HTTP protocol.

A. Scope of Attack and Attack Model

In this study, we focus on communication between the bridge and the App and do not analyse the Zigbee communication of the bridge and the lights. The Hue was initially hacked in 2013 [23], which the attack against it lies on the plaintext data communication between the App and the bridge. This unencrypted communication provides an attack vector for malicious attackers to infer operations that a legitimate user performs on the bulb. Furthermore, the Hue maintains a list of registered users and their associated secret keys known as whitelist tokens for authentication purposes. The Hue sends these whitelist tokens in plaintext in response to the access request of any legitimate user, which is indeed very scary. A malware on an infected machine on the smart home internal network may be able to passively eavesdrop on the data/control messages and obtain a list of registered tokens within the bridge. This enables an attacker to masquerade as a legitimate user by using its token and gain control of the lighting system.

We exploited above-mentioned vulnerabilities and developed a Python script to launch an attack against the Hue to take control of light bulbs. The script takes two steps in the attack scenario. First, it snoops passively the Hue network traffic to learn tokens of registered clients and current status of light bulbs. Next, it performs a range of commands on the Hue light bulbs by impersonating as a legitimate user using the learned token. These commands include switch on/off, brightness as well as colour setting, and the bridge setting like transition time. Fig. 3 describes the attack model algorithm, which implemented by the use of Python script.

Require: IP ▷ IP address of the Hue bridge
  repeat
    $snooping$ ▷ passive snooping the network
  until T ▷ a registered token is obtained
  $url = "http://" + IP + "/api" + T$
  $status \leftarrow GET(url)$
    ▷ HTTP method to get current status of the light bulbs
  while $True$ **do**
    ON/OFF in [True , False]
    BRI in [0 , 255]
      ▷ draw a random number to set the brightness
    CLR in [153 , 500]
      ▷ draw a random number for colour
    $data = \{"on" : ON/OFF, "bri" : BRI, "ct" : CLR\}$
      ▷ form a command
    $PUT(url, data)$ ▷ issue the command
    n in [100 , 1000]
    $wait(n)$ ▷ wait *n* msec then send the next command

Fig. 3. Attack Model Logic

B. Data Gathering

We performed two experiments and collected packet traces for the classification purpose. In the first experiment, we installed a Hue system in our lab and used the Hue iOS App version 1.8.2 on an iPhone to control the light bulbs. The app and its host were registered within the Hue bridge. We then issued a range of commands and collected the network traffic. This experiment consists of 366 commands, which performed over a period of approximately 5.5 hours. The commands include switching on/off, brightness setting and changing the light colour of bulbs.

We used the developed attack script mentioned earlier and conducted the second experiment in an attempt to disclose the registered clients within the bridge and to emulate unauthorized access to take control of the light bulbs. In this experiment 297 illegitimate commands were issued from a host within the Hue network over a period of approximately 2 hours. For these two experiments, we employed `tcpdump` tool and captured traffic packet traces at the interface between the App and the bridge of the Hue.

C. Feature Selection

We analysed the captured traffic and exercised several statistics features within two available captured traffic traces to find out the most critical ones that can be employed for detection in IoT-IDM. We selected our set of features heuristically based on the behaviour of the Hue as well as the behaviour of the attack model. To this end, three features were chosen: **(*i*)** *the number of bytes in command packets*, **(*ii*)** *the number of bytes in acknowledgement response packets*, and **(*iii*)** *inter-packet time interval*. Other features that we considered initially but decided not to use were connection duration and number of packet per connection. A more detailed

description of selected features and the reason behind their selection will be provided below.

1) The Number of Data Bytes in Command and Response Packets: As mentioned earlier in this section, users of Hue issue commands to the bridge using the App in an attempt to change the state of the light bulbs. Based on the Hue design, the App communicates with the bridge through provided RESTful interface over HTTP protocol. Once the bridge receives commands, it fulfils the request and in turn will issue an acknowledgement response message to the App informing the latest status of all light bulbs. The Hue uses JSON data exchange format for this response message. We also observed that every time a command is being sent to the bridge, a TCP connection will be established and after receiving the acknowledgement response the connection terminates.

There are many properties that can be controlled with Hue. All of these properties are in the **/state** resource of a light. It is also possible to control all lights at once by addressing the **/groups/0** resource. The bridge can accept ten commands per second to the **/lights** resource and one command per second to the **/group** resource as maximum. We also note that there are different ways of colour setting. The colour can be set either via an array of two values between 0 and 1 or a single value between 153 and 500 representing a warm white to a cold white. Lastly, the Hue App set the brightness or colour separately, though it is possible to set both properties in one command. These variation and combination of properties in a command result in different size of packets.

Depending on how an entity communicates with the Hue bridge, the size of command and response packets may vary. We extracted the number of bytes in command packets and its acknowledgement response messages of the two datasets. Fig. 4 shows the byte count of command and response packets for these two datasets. The intensity of colour in the plot shows the frequency of that example in our observed dataset.

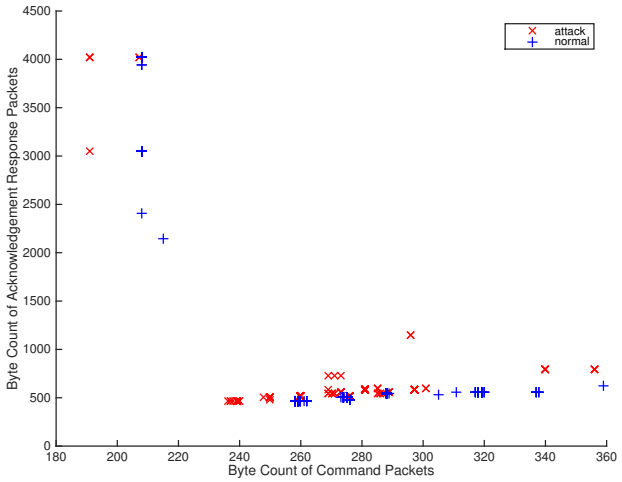
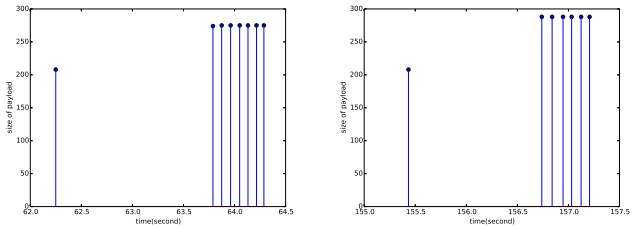


Fig. 4. Number of data bytes in commands and their responses in Hue.

2) Inter-packet time interval: Based on the Hue design the brightness of a light can be set from a range between 0 to 255. In the Hue App, a sliding bar is considered to allow users set the desired light intensity. The App in turn sends the current state of brightness indicator as it senses any change. Thus, several commands are being forwarded to the bridge in a row for any change in brightness. The same principle also applies to the colour setting; users of the Hue can choose the colour from a xy coordinates and the App will send colour change commands, as any change is perceived, thereby several colour setting's commands are sent in a row. Fig. 5 shows issued commands in a typical brightness setting and colour changing.



(a) Brightness Setting (b) Colour Setting

Fig. 5. Consecutive requests to set brightness and colour from the iOS App.

We measured the inter-packet time interval for the normal traffic dataset. The average value of these two observations for normal and attack dataset is given in Table II. From the difference in the time interval, we inferred that the inter-packet time interval could be used as a classification feature in the detection process.

TABLE II. INTER-PACKET TIME INTERVAL

	Legitimate user Traffic	*Attack Traffic*
Brightness setting	83ms	518ms
Colour changes	93ms	521ms

D. Classification

Aiming to detect possible intruders to protect a Hue system from unauthorized access, we sought for a predictive model, i.e., a classifier, capable of distinguishing between legitimate access and attack or intrusion. This requires a set of labelled data as legitimate and attack for learning task. We used the two captured traffic traces, as mentioned in Section VI-B, and generated the *training dataset*. Each instance in the training set contains one class label (legitimate or attack) and three features or attributes including ***(i)*** the size of request command from the App toward the bridge, ***(ii)*** the time passed since the previous command received, and ***(iii)*** the size of response packet to the command.

We first employed *logistic regression* technique with the three features of interest for the Hue. We used the training dataset and implemented gradient descent algorithm on them to find out the optimal parameters of a linear regression model. Alternatively, we used *Support Vector Machines* (SVMs) [24], which are a robust technique for non-linear data classification. SVMs use maximum margin kernel to nonlinearly map samples into a higher dimensional space. We considered Radial Bias Function (RBF) kernel (a.k.a Gaussian) that performs well in practice. In using RBF kernel, there are two parameters (C and γ) that must be determined before the learning process. We used a v-fold cross validation to identify the best parameters and incorporated the LIBSVM library to obtain the optimized classification model.

The accuracy of the obtained linear model using logistic regression on the whole training set was 96.2%, whereas through SVMs the accuracy of the nonlinear model was 100%. We note that the accuracy of classifier depends on the exact learning algorithm so choosing a suitable algorithm is crucial for well performance of IoT-IDM. The obtained model must be stored in the profile of the Hue in the database to be used by detection unit.

VII. Experiments

In this section, we present the empirical experiment to demonstrate the effectiveness and applicability of IoT-IDM framework. To this end, an experimental setup was developed and Philips Hue lighting system is employed as a smart-home IoT device to demonstrate the applicability of our framework.

A. Experimental Setup

In order to conduct our experiment, we required an OpenFlow-compliant switch. For this purpose, we used Open vSwtich (OvS) [25] that is a software switch that supports OpenFlow. The software switch was hosted on a normal Ethernet switch to build an OpenFlow-hybrid switch. We used TP-Link TL-WR1043ND as the Ethernet host switch. As the default OS on this switch does not allow to install OvS on it, we installed OpenWrt [26] v12.09 (Attitude Adjustment version) integrated with OvS v2.3.1 on the host switch. OpenWrt is an embedded OS based on the Linux kernel. The defined ports of the host switch, which correspond to hardware interfaces, are mapped to OvS ports to represent OpenFlow physical ports. WiFi connection is also configured within the switch to provide wireless access across the home network using OvS.

We used a realistic setting and assembled a small home wireless router using the OpenFlow-enabled switch. The Floodlight controller was hosted on a 2.5 GHz Intel Core i7 with 16 GB 1600 MHz DDR3 Ram server. One of the physical interfaces on the switch is considered for OpenFlow channel to connect the OpenFlow switch to the controller. Through this interface, IoT-IDM receives events from the switch, and sends packets out the switch. The Hue bridge was connected to one of the wired Ethernet ports of the switch. We also used an Android phone and installed the Hue App on it to control the light bulbs. Therefore, the Hue App communicates with the Hue bridge through the OvS router via WiFi and wired connection. For emulating attack against the Hue, we used a desktop computer connected in the same network to run the attack script. The experimental setup is shown in Fig. 6

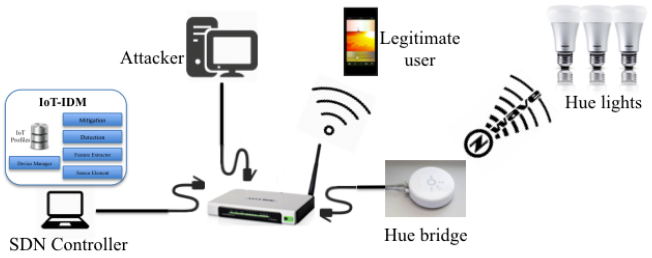


Fig. 6. Experimental Setup.

B. IoT-IDM Setup

After thorough analysis of the Hue given in Section VI, a profile for the Hue created and stored in IoT-IDM database as described in Section IV-A. In the first step, we registered the Hue bridge within device manager module in IoT-IDM. The registration includes the Hue ID and its location specified by IP address of the Hue bridge. Thus, IoT-IDM will be aware that a Hue lighting system is situated in the network with known IP address.

In the next step, IoT-IDM creates a virtual sensor element and then uses the location information (IP address) of the Hue bridge to construct two static OpenFlow rules to redirect the network traffic going to and from the bridge toward the sensor element. For constructing these rules, IoT-IDM uses the Hue profile to redirect the traffic toward the sensor element. In order to avoid flooding the sensor element, the rules may include specific conditions to filter out the redirected traffic. As described in Section VI-C, we are interested in HTTP packets for the Hue. Thus, IoT-IDM added `tcp_dst` field with value of 80 to filter out only HTTP packets. The generated OpenFlow rules are then pushed out in the OvS using HTTP POST command. The generated rules by IoT-IDM for the Hue are given below.

```
{"switch":"*:*", "name":"hue-flow1",
"priority":"32767", "active":"true",
"ipv4_dst":"192.168.2.226",
"tcp_dst":"80", "eth_type":"0x0800",
"ip_proto":"0x06",
"actions":"output=controller"}
```

```
{"switch":"*:*", "name":"hue-flow2",
"priority":"32767", "active":"true",
"ipv4_src":"192.168.2.226",
"tcp_src":"80", "eth_type":"0x0800",
"ip_proto":"0x06",
"actions":"output=controller"}
```

Feature extractor module in IoT-IDM then queries the types of features of interest for the Hue from its stored profile. It then starts extracting of the features from the logged traffic in the sensor element. For each HTTP command receiving to the bridge three features were extracted as described in Section VI-D. The detection module then employs the stored logistic regression prediction model for classification purpose. Finally, once a suspicious activity is identified, the detection module in turn informs the mitigation unit to generate and push out rules to block infected machine in accessing the victim device.

C. Results

It is important to test a system under examination with a different data set as it trained. Thus in our testing scenario, we used an Android App for the normal access to the Hue. We also modified the attack script in a way that it mimics the pattern of commands generated by a legitimate Hue App as much as possible, e.g., brightness and colour setting commands generated separately in different commands and for each property, 5 to 7 commands are generated in a row with 100ms interval.

In our test experiment, we performed commands ranging switch On/Off, brightness and colour setting using the Hue App. In the meantime, we launched the modified test attack from a machine in the same network of the Hue. IoT-IDM was monitoring the Hue activities during this experiment and collected all the traffic of the system under examination. The captured traffic contained 1762 commands over a period of approximately 8 hours. The detection unit in IoT-IDM labelled each access on the Hue based on the developed classification model described in Section VI-D. For accuracy evaluation, we amended IoT-IDM framework such that it also logged the four tuples including source and destination IP addresses and ports along with three features mentioned in Section VI-C. Since we were aware of legitimate App and emulated attack source, we used logged information and labelled commands for evaluation purpose.

The efficiency of IoT-IDM in detecting unauthorized attacks against the Hue was evaluated through *precision* and *recall* metrics. The precision metric indicates that from all accesses to the Hue where IoT-IDM considered them as attack, what fraction actually were illegitimate access and is given by:

$$precision = \frac{TP}{TP + FP} \tag{1}$$

Where TP (True Positive) is an attack access to the Hue classified as attack, and FP (False Positive) is a legitimate access to the Hue classified as an attack access. Similarly, the recall metric reveals that from all attack accesses to the Hue, what fraction IoT-IDM correctly detects as illegitimate access. The recall metric can be seen as follows.

$$recall = \frac{TP}{TP + FN} \tag{2}$$

Where FN (False Negative) is a legitimate access to the Hue classified as an attack access.

The results obtained from the test experiment showed that the linear logistic regression classification model in IoT-IDM predicted 1201 commands as illegitimate access where 69 of them are False Positive. It also predicted 561 commands as legitimate access which 199 of them were False Negative. Considering these result, it turned out that IoT-IDM detected unauthorized access to the Hue with precision rate of 94.25% and recall rate of 85.05%. Similarly, in case of using nonlinear classification model, the prediction precision was 98.53% and prediction recall was 95.94%. Although the detection accuracy depends on the employed detection algorithm, it also demonstrates the applicability of IoT-IDM. Once an attack is detected, an infection profile will be created in IoT-IDM and appropriate OpenFlow rule will be created to quarantine the infected host to access to Hue.

VIII. Conclusion

The Internet of Things (IoT) paradigm provides an opportunity to access and control smart devices at any place and at any time through using Internet Protocol (IP). Providing such opportunity, smart home application is making our lives easier by employing IoT technology in nearly anything: wired and wireless security detectors, cameras, thermostats, smart plugs, lights, entertainment systems, locks, and appliances. However, these smart devices are susceptible to attacks due to the lack of sufficient security consideration or poorly management. Thus, appropriate actions must be taken to mitigate security concerns in smart home technologies.

This study offers network-based intrusion detection and mitigation framework, called IoT-IDM, to identify and address potential attacks in smart home environments. To achieve this aim, we employed the SDN architecture in IoT-IDM to take advantage of its network visibility and programmability to provide a more secure and trustworthy environment for smart homes. IoT-IDM uses machine learning techniques to detect compromised hosts, whereby attacks are launched. After identifying the source of attacks, IoT-IDM generates appropriate policies and pushes to underlying routers/switches to avert attacks against victim IoT devices. A prototype of IoT-IDM is developed as a module of Floodlight SDN controller and its applicability demonstrated through a real smart lighting system.

References

[1] "The Internet of Things How the Next Evolution of the Internet Is Changing Everything," White Paper, Cisco, April 2011.

[2] (2015) A Technology report. BBC. [Online]. Available: http://www.bbc.com/news/technology-30121159

[3] "Internet of things research study," White Paper, HP, July 2015.

[4] T. Denning, T. Kohno, and H. M. Levy, "Computer Security and the Modern Home," *Communications of the ACM*, vol. 56, no. 1, pp. 94–103, January 2013.

[5] A. Jacobsson, M. Boldt, and B. Carlsson, "A risk analysis of a smart home automation system," *Future Generation Computer Systems*, vol. 56, pp. 719 – 733, 2016.

[6] T. R. Peltier, *Information Security Risk Analysis*. Auerbach, 2010.

[7] R. Roman, J. Zhou, and J. Lopez, "On the Features and Challenges of Security and Privacy in Distributed Internet of Things," *Computer Networks*, vol. 57, no. 10, pp. 2266–2279, Jul. 2013.

[8] S. Sicari, A. Rizzardi, L. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Computer Networks*, vol. 76, pp. 146 – 164, 2015.

[9] S. L. Keoh, S. Kumar, and H. Tschofenig, "Securing the Internet of Things: A Standardization Perspective," *Internet of Things Journal, IEEE*, vol. 1, no. 3, pp. 265–275, June 2014.

[10] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time Intrusion Detection in the Internet of Things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2661–2674, Nov. 2013.

[11] P. Kasinathan, G. Costamagna, H. Khaleel, C. Pastrone, and M. A. Spirito, "DEMO: An IDS Framework for Internet of Things Empowered by 6LoWPAN," in *ACM SIGSAC Conference on Computer and Communications Security, CCS'13*, November 4-8 2013, pp. 1337–1340.

[12] P. Kasinathan, C. Pastrone, M. A. Spirito, and M. Vinkovits, "Denial-of-Service detection in 6LoWPAN based Internet of Things," in *9th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob '13*, October 7-9 2013, pp. 600–607.

[13] C. Lee, L. Zappaterra, K. Choi, and H.-A. Choi, "Securing Smart Home: Technologies, Security Challenges, and Security Requirements," in *IEEE Conference on Communications and Network Security (CNS) Workshops*, Oct 2014, pp. 67–72.

[14] S. R. Das, S. Chita, N. Peterson, B. A. Shirazi, and M. Bhadkamkar, "Home Automation and Security for Mobile Devices," in *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, March 2011, pp. 141–146.

[15] H. Kim, A. Voellmy, S. Burnett, N. Feamster, and R. Clark, "Lithium: Event-Driven Network Control," Georgia Institute of Technology, Tech. Rep., 2012.

[16] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*, Oct 2010, pp. 408–415.

[17] N. Feamster, "Outsourcing Home Network Security," in *Proceedings of the 2010 ACM SIGCOMM Workshop on Home Networks*, ser. HomeNets '10. New York, NY, USA: ACM, 2010, pp. 37–42.

[18] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Computer Networks*, vol. 62, no. 0, pp. 122 – 136, 2014.

[19] K. A. Scarfone and P. M. Mell, "Guide to Intrusion Detection and Prevention Systems (IDPS)," National Institute of Standards and Technology of USA, Gaithersburg, MD, United States, Tech. Rep. SP 800-94, February 2007.

[20] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," *SIGCOMM Computer Communication Review (CCR)*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[21] "Introduction to Cisco IOS NetFlow - A Technical Overview," White Paper, Cisco, May 2012.

[22] (2015) Floodlight Homepage. [Online]. Available: http://www.projectfloodlight.org/floodlight/

[23] (2013) Hacking Lightbulbs. [Online]. Available: http://www.dhanjani.com/blog/2013/08/hacking-lightbulbs.html

[24] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92. New York, NY, USA: ACM, 1992, pp. 144–152.

[25] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenker, "Extending Networking into the Virtualization Layer," in *In: 8th ACM Workshop on Hot Topics in Networks (HotNets-VIII)*, New York City, October 2009.

[26] (2015) OpenWrt Website. [Online]. Available: https://openwrt.org/