CLIPS: Customized Levels of IoT Privacy and Security

Rohith Yanambaka Venkata, Krishna Kavi

Computer Systems Research Lab
Dept. of Computer Science and Engineering
University of North Texas
Denton, Texas 76207, USA
Email: RohithYanambakaVenkata@my.unt.edu, Krishna.Kavi@unt.edu

***Abstract*—Internet of Things (IoT) refers to systems that can be attached to the Internet and thus can be accessed and controlled remotely. Such devices are essential for creating "smart things" like smart homes, smart grids, etc. IoT has achieved unprecedented success. It offers an interconnected network where devices (in the consumer space) can all communicate with each other. However, many IoT devices only add security features as an afterthought. This has been a contributing factor in many of the recently reported attacks and warnings of potential attacks such as those aimed at gaining control of autonomous cars. Many IoT devices are compact and feature limited computing resources, which often limits their ability to perform complex operations such as encryption or other security and privacy checks. With capabilities of devices in IoT varying greatly, a one-size-fits-all approach to security can prove to be inadequate. We firmly believe that safety and privacy should both be easy to use, present little inconvenience for users of non-critical systems, yet be as strong as possible to minimize breaches in critical systems. In this paper, we propose a novel architecture that caters to device-specific security policies in IoT environments with varying levels of functionalities and criticality of services they offer. This would ensure that the best possible security profiles for IoT are enforced. We use a smart home environment to illustrate the architecture.**

Keywords–Internet of Things (IoT); Software Defined Networking (SDN); IoT Security.

I. INTRODUCTION

Internet of Things (IoT) refers to systems that can be attached to the Internet and thus can be accessed and controlled remotely. Such devices are essential for building "smart things" like smart homes, smart grids, etc. The proliferation of IoT has been undeniably progressive and uninhibited.

The total investment in IoT is predicted to touch the $5 trillion mark in the next five years [1]. More specifically, a recent market study shows that the market with the fastest growing adoption rate is smart homes [2] with the market predicted to generate $ 2.5 billion [2]. Consumers are choosing to invest in IoT for convenient and comfortable lives [3]. Devices such as refrigerators, light bulbs and thermostats can be controlled remotely, which can also result in power savings and reduced operational costs [3].

With an industry as diverse and prevalent as smart homes, security is paramount. Users like to be assured that the devices they invested in are safe and cannot be attacked. Providing this assurance, however, is not an easy task.

Securing network infrastructure often entails shutting services down. For example, a widely accepted response to a denial of service attack (DoS) is to shut down network services. Ideally, securing the devices in a network should go a long way toward ensuring the security of the network itself. Frequently, companies enforce a uniform policy model (specific to a department or section). This compromises flexibility and control. With a traditional networking model, control over the network topology is limited. By leveraging Software Defined Networking (SDN), fine-grained control over the network topology is possible. A diverse network comprised of devices with varied capabilities would be constrained by the device with the least computational or networking abilities. For example, if one of the security requirements is end-to-end network encryption, a non-IP addressable device with minimal computational power may be unable to enfforce such a requirement. We firmly believe that security and privacy should both be easy to use, present little inconvenience to users of non-critical systems, yet be as robust as possible to minimize breaches.

A device-specific and user-selectable approach that caters to the need of all devices in IoT is required. Security policies, tailored to the network and computing capabilities of those devices will result in a good IoT security posture. Otherwise, users will either object to the privacy/security requirements or undermine the security with default or weak security configurations.

We propose a device-specific approach to security using SDN that addresses the needs of individual classes/categories of devices that are trying to access the LAN or WAN network in a smart home. Such an approach to security provides flexibility and control over device and network security.

The main contribution of this paper is an SDN administered security framework that caters to device-specific protection needs in IoT environments with varying levels of functionalities and criticality of the services they offer.

This paper is organized as follows. We present our architecture for enforcing device-specific security policies for IoT devices in Section II. In Section III, we describe the topology detailing the various threat protection and avoidance schemes that are implemented in the architecture. We detail the countermeasures in place against some of the most important attacks targeting IoT in Section IV. Section V contains some related work that is aligned with this research. Section VI describes the configuration and set up of a prototype, followed by conclusions.

II. ARCHITECTURE OVERVIEW

For the purpose of this paper, we use a smart home as an example environment and illustrate how our architecture can

implement device-specific security and privacy policies. We will describe individual components in this Section.

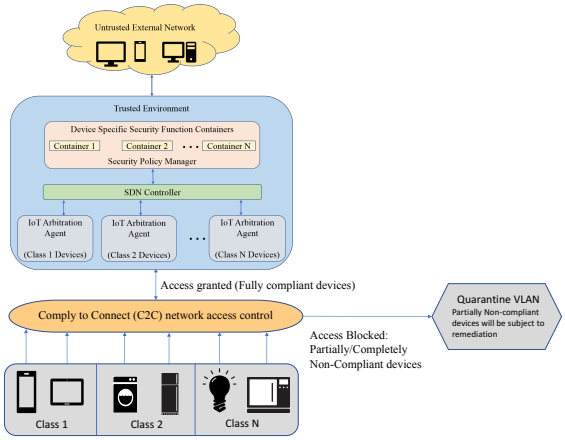

Figure 1. CLIPS architecture.

Figure 1 shows a high-level view of the proposed architecture which consists of a single standalone device that replaces a wireless router. The device hosts a secure trusted environment to which the IoT devices connect to communicate with each other, i.e., Machine to Machine (M2M) or to access the Internet. The functions offered by our device are two-fold:

- Provide networking functionality using SDN.
- Provide a secure environment for device communication (M2M and access to the Internet).

A. *Untrusted external networks*

An untrusted network is one which provides no information regarding data safety, the authenticity/identity of the communicating device, or the communication link itself. This is the most common source for attacks on devices. We need to have mechanisms in place in the smart home network to detect and handle any attacks. The only sensible approach is to monitor and filter traffic at the device itself. This is generally achieved by configuring firewalls and intrusion detection systems to monitor network traffic. Additional protection schemes are discussed in Section III.

B. *Trusted environment*

A trusted environment is one where there is a good degree of confidence in the integrity and confidentiality of the network components (see Figure 2). The first level of defense in the trusted environment is a set of Intrusion Detection Systems (IDS) such as Snort and a firewall service. The final objective is to enforce device (device class) specific security policies. Kerberos provides all the authentication needs for network communication. Additional policies are discussed later in this Section.

1) Security policy manager: Device-specific security policies are stored in Linux containers which are assigned to each category of IoT devices that may be present in a smart home network. Once the device has been identified and grouped, the

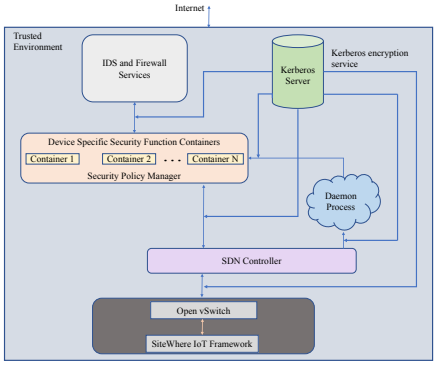

Figure 2. A closer look at the trusted environment.

corresponding container is invoked and the SDN controller initiates a daemon process that runs in the background. This daemon process is then tasked with invoking one of the many device-class specific security function containers that are either suspended or powered down. Powering down containers minimizes the chances of corrupting the code of the container. The container, once invoked, retrieves the required security policies for the identified device. These rules are downloaded onto the SDN controller via the northbound interfaces and the security function container goes back to the suspended state or powers down. The communication link between the container and the SDN controller is encrypted.

The reason for the security policies residing in containers and not configured into the SDN controller is to minimize exposure of the policies to adversaries. When these policies are configured onto the controller, the attackers may find an exploit (in the source code) and gain access to the rules. If, however, they were to reside in a sand-boxed environment, it would make things much harder for them to be accessed, since the policies are exposed to attacks only when the container enforcing these polices is active.

2) IoT arbitrator: An IoT arbitrator resides in the trusted environment and its responsibilities are to host the virtual switch that the SDN controller communicates with (an SDN controller can only communicate with switches and not the devices themselves) and to host an IoT framework for low-powered devices (constrained devices that are not IP accessible) to communicate with each other.

An IoT environment may be an amalgam of devices that communicate using a variety of network protocols, each using different MTUs (Maximum Transmission Unit). Hence, a broker needs to mediate the communication between devices. An example of an IoT framework is SiteWhere [4]. It provides a framework to obtain, store, process and migrate data among IoT devices, which happens through the MQTT (Message Queue Telemetry Transport) protocol.

3) Open vSwitch: Once the policies for a specific class of devices are downloaded onto the SDN controller, they have to be conveyed to the respective switch. The SDN controller maintains flow tables and controls the topology of its assigned network. The controller contacts the OVS switch (Open vSwitch) to which the devices in question are connected.

The appropriate flow table entries are made in the switch and additional security policies are downloaded. Once this is complete, the security policies that were downloaded onto the SDN controller are wiped clean (to prevent attackers from gaining access). The switch uses this newfound information to forward the packets accordingly. Each class of devices will have a designated IoT Arbitrator. In essence, the container that hosts an IoT agent is only invoked when a device first connects to the CLIPS security administrator and goes back to the suspended state or powers off when no device of the designated class is connected.

III. Topology

Securing the Internet of Things is a complex process. A one-size-fits-all approach is impractical given that IoT encompasses a wide variety of devices with varying capabilities. Hence, a device (or device class) specific approach is needed.

A. Comply to Connect (C2C)

Comply to Connect (C2C) is described as a standards-based approach to securing devices that connect to a network [5]. C2C consists of a pre-defined set of standards and security profiles that a device must meet before its is permitted to access the network. For example, an Android device trying to access the network may be required to run OS version 6.0 with the latest security patches before it can connect to the network.

Our implementation of C2C performs a series of checks on the device trying to access the network. Some of the parameters collected include the version of the OS, version of the kernel currently running, security patches installed, ports currently open, etc. Additional checks such as deep packet inspection are performed to ensure that the device is secure and can access the network. Each 'class' of devices has a set of security requirements which can be enforced by C2C. For example, all mobile devices (such as tablets, smart phones, smart watches) are expected to support network encryption and two-factor authentication. Devices are first identified by the certificates issued by an internal certification agent controlled by C2C.

If one or more checks fail, the devices are provided limited access until security definitions are updated. If the devices are deemed unfit to access the network until a major upgrade is performed, they are quarantined in a VLAN.

Most of the vulnerabilities listed by OWASP for IoT devices [6] can be prevented or addressed by C2C. At the top of the list is Insecure Web Interface [6] which includes lack of encryption. Use of plain text for storing passwords could lead to issues such as cross-site scripting to inject malicious code [6].

IoT devices are often targeted for DDoS attacks. A famous example is the attack on Dyn's DNS systems that brought down popular services like Netflix and Facebook [7]. A majority of the vulnerabilities can be prevented by ensuring that the recommended software/firmware and security patches are installed and by ensuring that the encryption and authentication schemes used are sufficiently strong. We intend to achieve this through C2C. A brief description of the protection offered against some other IoT attacks is discussed in Section IV.

B. Leveraging SDN to secure the IoT

Software defined networking is one of the most important advances in networking in recent history. SDN has accelerated the rate of connected devices [7]. We intend to leverage SDN to ensure network security. Every class of devices has a pre-defined set of security rules that are stored in Linux containers assigned to them. When a device is first granted access to the network by C2C, the appropriate container is started, corresponding security rules present in the container are downloaded onto the SDN controller via the Northbound API and the container is placed in a suspended state to restrict visibility to the external network.

The secondary objective of the containers is to perform arbitration functions between the devices and the SDN controller, which includes hosting an Open vSwitch with which the SDN controller communicates. Not all security policies need to be exposed to the devices themselves. For example, consider a security policy requiring webcams to be placed under a NAT for a secure (encrypted) network segment created specifically for webcams. All the device needs to know is that the network connection must be encrypted. Since the NAT functionality is handled by the controller and does not rely on native support from the device, it need not be advertised. The devices may remain ignorant of the network topology.

With device manufacturers unwilling to offer security configuration options for devices, ensuring security by enforcing protection schemes at the devices themselves can be challenging. In fact, this is listed as number 8 in OWASP's Top 10 Vulnerabilities in the IoT report [6]. We aim to address this issue by employing an arbitration agent that mediates operations between the controller and a device.

C. Integrity measurement using RADIUM

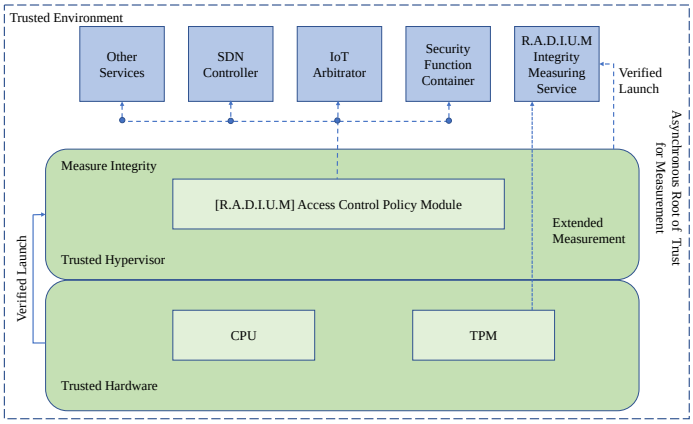

Figure 3. Integrity measurement using RADIUM.

RADIUM (Race-free On-demand Integrity Measurement Architecture) is, as the name suggests, an architecture that measures trustworthiness by providing on demand measurement (of integrity and trustworthiness) of software components. Figure 3 demonstrates this process. To ensure that the measured components are trustworthy, they have to be compared against some the integrity measurement in known "good" state (such as when the component was first developed or a trusted patch was made). This architecture was created using research conducted at the University of North Texas [8].

RADIUM establishes a chain of trust between software and the underlying hardware. This is achieved as follows. The hardware is tasked with measuring the firmware, the firmware is tasked with measuring the system software and the system software measures the application [8]. One way to establish the chain of trust is through Dynamic Root of Trust for Measurement (DRTM). The idea is to create a secure, isolated and measured environment for software to be executed (called measured launch environment MLE) [8].

To initiate a virtualized environment, a hypervisor is the primary component to be invoked. It is done so during the system boot using DRTM. The bootloader is responsible for invoking a DRTM MLE using a set of special instructions to prepare an isolated execution environment [8]. The hypervisor is then measured and compared to a previously known "good" value which is stored in the trusted platform module (TPM). If everything checks out, the hypervisor is deemed trustworthy and executed. The hypervisor then takes control of the platform [8].

For the hypervisor to be mindful of the existence of a measuring service, registration needs to take place. During registration process, it is the responsibility of the measuring service to provide all the ACPM rules for its own functioning to the hypervisor. Also, the hypervisor measures the measuring service itself and saves its "good" value for future comparisons. The measuring service is encrypted and the key is stored in the TPM.

Any virtual machine, that has to run on the hypervisor needs to be registered with the hypervisor before executing. During this registration, a set of policy rules are provided to the hypervisor, which contain the ID of the measuring service that can access the target virtual machine with the appropriate permissions. The hypervisor registers the ID of the target virtual machine and it will be invoked after it's registration.

The simplest method to measure integrity is to store the hash codes of the entire binary (for each software) in a container. RADIUM is configured to compute hash codes at regular intervals. The computed hash code is then compared against the stored hash code. If everything checks out, RADIUM measuring service goes back to a suspended state (or is powered down) and is only invoked when a measurement is again necessary. If there is a mismatch between the measured and stored hashes, however, the application is run on an Intel SGX enclave, which functions as a sandbox. The integrity of the measuring service itself is ensured by the TPM that is present along with the CPU.

All the containers that are invoked in the trusted environment run on a single hypervisor. This makes it a lot easier to ensure their integrity and trustworthiness.

IV. IoT VULNERABILITIES AND ATTACK VECTORS

In this Section, we attempt to identify attack vectors for IoT and describe our approach to addressing them. Nawir et. al do a good job of tabulating and describing the taxonomy of attacks targeting IoT [9].

1) Distributed Denial of Service (DDoS): On 21 October 2016, a large scale DDoS attack on Dyn's DNS systems resulted in popular services such as Netflix, Facebook and Google becoming inaccessible [10]. The attack was perpetrated using Mirai botnet that infected thousands of devices around the world. Some 100,000 devices bombarded Dyn's systems with network traffic at 1.2 Tbps which is a new record [11]. The attackers used a simple brute force attack to infect various DVR players, smart televisions, refrigerators and CCTV cameras using the default passwords that the products were shipped with. The objective in this case was evident; to utilize the popularity of IoT, the lack of sophistication or technical literacy on the part of end users to infiltrate IoT devices to sabotage networks.

We intend to address this issue by protecting devices against brute force attacks by screening them through the C2C architecture discussed in Section III-A. Assuring that they are running the latest software with the appropriate security patches installed will safeguard against a majority of the vulnerabilities.

Ultimately, we intend to create a hierarchical and distributed SDN network segment that shares the network load. This should add an additional layer of security against DDoS attacks.

2) Ransomware attacks: The summer of 2017 saw the emergence of the dreaded WannaCry ransomware which targeted windows machines. The attack used an exploit in the SMB transactions. When it gains access to a machine, the worm encrypts the file system. The attackers then offer to unlock the system in exchange for a ransom. Some 400,000 devices were affected around the world [12]. What is interesting to note is that almost 98% of the devices affected by this attack were running an outdated version of Windows 7.

An attack such as this would be catastrophic for IoT, especially for business-critical systems or devices in the health care domain, as evident from the ransomware attack on a children's hospital in Boston [13] where some personally identifying patient records were deleted.

Our architecture has several safeguards in place to prevent such attacks:

- C2C ensures that devices are running the latest software and have the latest security patches installed. Most ransomware attacks use an open port to inject the encryption program. Oftentimes, the vulnerability is already patched by the manufacturer of the product. Hence, safeguarding against such attacks is a simple matter of updating the software and security patches on the devices.
- The RADIUM architecture ensures integrity of the application.
- Anomaly detection ensures that suspicious activity on the network will either invoke shutdown or containment protocols.

3) Man-in-the-Middle Attacks: Man-in-the-Middle attacks involve an adversary impersonating a legitimate communication node in a network. A successful impersonation will result in sensitive/confidential data being shared with the adversary which leads to a breach or gaining access to a device. Li et. al describe an approach using fog nodes that mediate between servers and clients [14]. One of the countermeasures they proposed was to modify package types in OpenFlow [14].

We intend to use RADIUM to establish a chain of trust between software applications and the underlying hardware. This ensures that a device is properly identified, authenticated

and monitored at all times in a network. Applications are run in secure containers called Enclaves, which an illegitimate user would be unable to access because each process would have its own trusted environment with integrity being measured by RADIUM.

4) Spoofing Attacks: An adversary may be able to masquerade as a legitimate user/entity in a network by spoofing IP addresses, ARP entries or MAC addresses. Preventing such attacks is easier than detecting them. Xiao et. al proposed a method of identifying spoofing using reinforcement learning in wireless networks [15]. The spoofing detection algorithm aims to identify spoofing attacks using Q learning [15].

We intend to prevent spoofing attacks by employing a few protection schemes:

- We aim to prevent IP spoofing by employing packet filtering that is usually achieved by configuring a firewall and IDPS (Intrusion Detection and Prevention System). We use the signature matching algorithm proposed by Meng et. al [16], but we replace the pre-filtering of packets with access control through the C2C architecture. The challenge is to ensure that these systems are properly configured, so we follow Cisco's guide to best firewall configuration practices [17].
- We employ encryption during device authentication and communication to overcome vulnerabilities arising as a result of the design of the TCP suite.
- We are exploring the advantages offered by IPv6 in securing network communication.
- Preventing MAC spoofing may be somewhat challenging in a Smart home because most embedded devices with limited capabilities are not IP addressable and do not possess a MAC address. Consequently, the arbitration agent has to spoof MAC addresses for such devices. Our approach to preventing malicious spoofing does not rely on authenticating by MAC addresses. Rather, device certificates that are validated by an internal certification authority are used by the C2C architecture to control and moderate access to the network.

We have focused on some of the most important vulnerabilities in IoT for a Smart home setting to highlight the capabilities of our proposed architecture. We have not considered physical side channel attacks because we have assumed that the users have suitable measures in place to ensure that no unauthorized person may gain physical access to devices in such a setting.

V. Related work

Flaunzac et al. proposed a distributed SDN architecture aimed at preventing DDoS attacks [18]. The objective is to authenticate network devices and ensure that only services that the authenticated user is permitted to access are allowed. A distributed SDN architecture contains border controllers that negotiate security policies with neighboring domains [18]. Our proposed architecture can help counter some of the shortcomings of Flaunzac's approach.

- The definition of security policies is optimized to the network domain in which the devices reside and not to the devices themselves. This may not be the ideal approach to securing a diverse set of devices with varying capabilities. This is not an ideal approach because this architecture can be subject to TOCTOU (Time of Check to Time of Use) attacks. We intend to prevent these using RADIUM.
- Integrity measurement is not implemented in the architecture which exposes it to man-in-the-middle attacks. We have a robust integrity measurement architecture in place to address this.
- It may be unsafe to expose security policies by defining them at the border controller. Instead, we package them into Linux containers that are always in a suspended state and are only started when required.
- There is no pre-screening of devices before they connect to the network. A major portion of the vulnerabilities that exist are a result of outdated software and security patches which, when updated, will mitigate those risks.

Agarwal et. al proposed an architecture using edge computing [19] that is very similar to the one proposed by Flaunzac et al [18]. The idea is to compartmentalize networks into zones with each zone being controlled by a gateway controller. Security is enforced by authenticating a device and collecting TCP (Transmission Control Protocol) dumps to perform Deep Packet Inspection [19]. An analysis of the packets determines if a device is safe to be granted permission to proceed to the next hop in the flow entry [19]. The process continues at every hop. Our proposed architecture employs a C2C architecture that performs pre-screening of devices and recommends fixes which should address a majority of the vulnerabilities. Deep packet inspection and authentication of devices is a subset of all security policies we enforce through our proposed architecture.

Most of the solutions we reviewed focus on enforcing network security by controlling the flow table in SDN. The solution proposed by Bull et al. is also a similar approach [20]. The idea is for a controller to detect malicious activity by constantly monitoring network flow in an SDN network. If a suspicious activity is detected, the data is forwarded to a quarantine zone where further processing of the packet occurs [20]. This is somewhat similar to other anomaly detection schemes employed by Brocade and Rackspace [20].

VI. Prototype

A proof of concept prototype of CLIPS was developed in the Computer Systems Research Lab (CSRL) at the University of North Texas. The test environment contains the following components.

- A Netgear W3800 running OpenWRT functions as a data plane. The control plane (routing brains) of the router has been disabled by creating a virtual bridge using Open vSwitch which renders the device incapable of performing routing operations.
- The Open vSwitch is connected to a Floodlight SDN controller running on a Linux machine. The controller interfaces with the virtual switch using OpenFlow. All networking operations are performed by the controller and communicated to the switch.

The router, functioning as a wireless access point, helps deploy a local network to host the CLIPS architecture. The wireless SDN network is set up using the WiFISDN project [21].

Ideally, from a security standpoint, direct communication between devices (bypassing the SDN controller) should not be permitted, but the only way to achieve this is by disabling machine to machine M2M communication. This can prove to be counter-productive because the objective of this research is to secure the network without placing heavy restrictions on normal use of services.

To work around this issue, we enabled wireless isolation, which prevents devices on the same network from communicating with each other [21]. This forces additional processing of network packets. Additional OpenFlow rules are required to permit communication. Hence, security policies can be designed around wireless isolation. For example, eliminate communication between a refrigerator and a light bulb to keep non-essential communication to a minimum. This reduces the attack vector for DDoS attacks and enables simpler network analysis and anomaly detection.

Before a device connects to the CLIPS network, the user is expected to register it and install the required certificates. In this prototype, we have designed security policies for a microwave oven and a refrigerator using SiteWhere [4]. For a refrigerator to first connect to a network, the user is expected to authenticate it using a password. The bandwidth of the communication channel for a refrigerator is limited using the QoS parameters of Floodlight and no communication is permitted with a microwave oven. On the other hand, when a microwave oven first connects to the network, an authentication code is sent to a registered phone number. This code, in addition to a password, will serve as the authentication token. A microwave is permitted to communicate with a refrigerator and it is offered the lowest bandwidth in the network due to potential ramifications from its exploit.

The prototype demonstrated that this approach to security offers flexibility and control over the entire network topology. In addition, the architecture does not place unreasonable requirements on a novice user and has proved to be fairly user friendly. We are currently working on identifying, designing and deploying appropriate security policies in similar IoT environments.

VII. Conclusion

In this paper, we have outlined the issues of a generic approach to IoT security. As more devices with unique features and capabilities become increasingly popular, tailored security policies must be adopted.

We have argued for a more meticulous, fine-grained approach to securing IoT devices by leveraging SDN. We have proposed a novel architecture where devices are evaluated to certify that they are competent to access the network, classified into categories based on security policy requirements, and continuously monitored for suspicious behavior.

We have demonstrated and validated the idea using a Netgear W3800 running OpenWRT and OpenVSwitch connected to an SDN network hosted by a Floodlight controller. We found that such an approach greatly enhances the security of a home network and ensures that such devices cannot be exploited by DDoS attacks. The focus of our approach has been a balanced one in regards to convenience and security. As illustrated, an overly aggressive approach could prove to be detrimental and a highly accommodating one could leave devices open to attacks. The challenge is to find the right balance and we would like to believe that we have found such an approach for IoT security.

Acknowledgment

The authors would like to acknowledge the editorial contributions of Mr. David Struble, former Senior Software Technologist at Raytheon's Net-Centric Systems Division. This work is supported in part by the NSF Net-Centric and Cloud Software and Sytems Industry/University Cooperative Research Center and its industrial members, including Lockheed Martin Missile and Fire Control and Ashum Corp.

References

[1] "The Internet of Things 2017 Report: How the IoT is Improving Lives to Transform the World," 2017, URL: http://www.businessinsider.com/the-internet-of-things-2017-report-2017-1 [accessed: 2017-05-04].

[2] "The 10 Most Popular Internet of Things Applications Right Now," 2015, URL: https://iot-analytics.com/10-internet-of-things-applications/ [accessed: 2017-06-23].

[3] "How IoT & Smart Home Automation Will Change the Way We Live," 2016, URL: http://www.businessinsider.com/internet-of-things-smart-home-automation-2016-8 [accessed: 2017-05-21].

[4] "Sitewhere IoT Framework," 2017, URL: http://www.sitewhere.org/ [accessed:2017-06-16].

[5] "Comply to Connect Solution Overview," 2012, URL : https://trustedcomputinggroup.org/comply-connect-solution-overview/ [accessed:2017-06-10].

[6] "The Open Web Application Security Project," 2017, URL: https://www.owasp.org/index.php/ [accessed: 2017-03-20].

[7] "DDoS attack that disrupted internet was largest of its kind in history, experts say," 2016, URL: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet [accessed: 2017-03-26].

[8] S. Kotikela, M. Gomathisankaran, T. Shah and G. Taban, "Race free on demand integrity measurement architecture," International Conference on Privacy Security Risks and Trust (PASSAT) ASE., 2014.

[9] M. Nawir and A. Amir and N. Yaakob and O. B. Lynn, "Internet of Things (IoT): Taxonomy of security attacks,," in 3rd International Conference on Electronic Design (ICED), August 2016, pp. 321–326.

[10] "Dyn Analysis Summary Of Friday October 21 Attack," 2016, URL: https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/ [accessed: 2017-05-13].

[11] "Lessons From the Dyn DDoS Attack," 2016, URL: https://securityintelligence.com/lessons-from-the-dyn-ddos-attack/ [accessed: 2017-06-2].

[12] "WannaCry Ransomware Statistics: The Numbers Behind the Outbreak," 2017, URL: https://blog.barkly.com/wannacry-ransomware-statistics-2017 [accessed: 2017-05-06].

[13] "Children's Clinic Hit by Ransomware," 2016, URL: http://www.healthcareitnews.com/news/childrens-clinic-hit-ransomware [accessed: 2017-03-22].

[14] C. Li, Z. Qin, E. Novak and Q. Li, "Securing sdn infrastructure of iot-fog network from mitm attacks," IEEE Internet of Things Journal, vol. PP, no. 99, 2017, pp. 1–1.

[15] L. Xiao and Y. Li and G. Han and G. Liu and W. Zhuang, "Phy-layer spoofing detection with reinforcement learning in wireless networks," IEEE Transactions on Vehicular Technology, vol. 65, no. 12, December 2016, pp. 10 037–10 047.

[16] W. Meng, W. Li and L. F. Kwok, "Towards effective trust-based packet filtering in collaborative network environments," IEEE Transactions on Network and Service Management, vol. 14, no. 1, March 2017, pp. 233–245.

[17] "Cisco Firewall Best Practices Guide," 2017, URL: http://www.cisco.com/c/en/us/about/security-center/firewall-best-practices.html [accessed: 2017-06-13].

[18] O. Flauzac and C. Gonzlez and A. Hachani and F. Nolot, "SDN Based Architecture for IoT and Improvement of the Security," in 2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, March 2015, pp. 688–693.

[19] C. Aggarwal and K. Srivastava, "Securing IOT devices using SDN and edge computing," in 2nd International Conference on Next Generation Computing Technologies (NGCT), October 2016, pp. 877–882.

[20] P. Bull, R. Austin, E. Popov, M. Sharma and R. Watson, "Flow Based Security for IoT Devices Using an SDN Gateway," in IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), August 2016, pp. 157–163.

[21] "Software-Defined Wi-Fi Networks with Wireless Isolation," 2017, URL: https://wiki.helsinki.fi/display/WiFiSDN/Software-Defined+Wi-Fi+Networks+with+Wireless+Isolation [accessed: 2017-04-12].