computers

MDPI

Article

Improving Internet of Things (IoT) Security with Software-Defined Networking (SDN)

Abdullah Al Hayajneh [1,*], Md Zakirul Alam Bhuiyan [2] and Ian McAndrew [1]

1 Department of Doctoral Programs, Capitol Technology University, Laurel, MD 20708, USA; irmcandrew@captechu.edu

2 Department of Computer and Information Sciences, Fordham University, New York, NY 10458, USA; mbhuiyan3@fordham.edu

* Correspondence: asal-hayajneh@captechu.edu

Received: 6 January 2020; Accepted: 5 February 2020; Published: 7 February 2020

Abstract: There has been an increase in the usage of Internet of Things (IoT), which has recently become a rising area of interest as it is being extensively used for numerous applications and devices such as wireless sensors, medical devices, sensitive home sensors, and other related IoT devices. Due to the demand to rapidly release new IoT products in the market, security aspects are often overlooked as it takes time to investigate all the possible vulnerabilities. Since IoT devices are internet-based and include sensitive and confidential information, security concerns have been raised and several researchers are exploring methods to improve the security among these types of devices. Software defined networking (SDN) is a promising computer network technology which introduces a central program named 'SDN Controller' that allows overall control of the network. Hence, using SDN is an obvious solution to improve IoT networking performance and overcome shortcomings that currently exist. In this paper, we (i) present a system model to effectively use SDN with IoT networks; (ii) present a solution for mitigating man-in-the-middle attacks against IoT that can only use HTTP, which is a critical attack that is hard to defend; and (iii) implement the proposed system model using Raspberry Pi, Kodi Media Center, and Openflow Protocol. Our system implementation and evaluations show that the proposed technique is more resilient to cyber-attacks.

Keywords: IoT; software defined networking; HTTP security

1. Introduction

Special concerns have been raised for the security of Internet of Things (IoT) [1], these devices are uniquely identifiable, smart in analyzing data and making decisions, and have network capabilities that allow them to be connected to the Internet. IoT devices can range from simple home sensors, medical devices, cars, airplanes to even nuclear reactors. Due to the nature of IoT, communication may be done without confidentiality and authenticity, thus making it susceptible to attacks. Traditional methods of protecting networks use firewalls and intrusion detection/prevention systems on the network edge to prevent external attack. However, due to their special characteristics, such defense mechanisms do not work directly with IoT networks.

Recent advances in computer networking introduced a new technology, software defined networking (SDN), which allows a central program, called the 'SDN Controller', to control the overall behavior of the network [2]. The controller enables quick reactions to security threats, granular traffic filtering, and dynamic security policies deployment. Researchers have investigated the use of SDN to secure computer networks, such as using SDN controller and switch to build a firewall [3–6], and a few researchers studied integrating SDN with IoT. Sood et al. discussed the security and scalability of SDN and IoT networks as well as the need for deep packet investigation (DPI) in that environment [7]. Comparing with our work, we proposed a system model with SDN to protect IoT devices that use HTTP. Qin et al. proposed an SDN architecture for IoT to address the need for an SDN that works with multi-hop and heterogeneous ad-hoc path to optimize the network for the nature of IoT [8]. Comparing this proposal with our system model, we did not require any changes in the IoT network environment. Jararweh et al. proposed SDIoT, a software defined based IoT framework that simplifies the IoT management process related to network storage and security [9]. Similarly, Liu et al. also proposed a software defined based IoT architecture that focused on urban IoT sensors [10]. Comparing the previous two proposals with our system models, we did not require any modifications on the IoT devices either in the storage or network. Salman et al. suggested an identity authentication scheme for IoT with SDN in which the identity used different communication protocols that are mapped to a shared identity through the SDN controller [11]. Chakrabarty et al. suggested an SDN-based security networking mechanism called Black-SDN for IoT. Their proposed mechanism encrypted specific components of the IoT including the header, the source and destination IP addresses in addition to the payload. The header encryption showed routing challenges that were addressed through a broadcast routing protocol using the SDN controller [12]. Theodorou et al. described MINOS which is a multi-protocol SDN platform for IoT. This approach proposed an awareness service using SDN facilities and interfaces to create a centralized network control of IoT resource environments [13]. Tran et al. proposed algorithms to increase the efficiency of the SDN Controller placement with the distributed IoT networks. The SDN Controller placement has common problems with the distributed networks, and this paper proposed algorithms to solve these problems specifically for the IoT Networks that use SDN [14]. Zarca et al. proposed a cyber situational awareness security framework to sustain the authentication, authorization, and accounting (AAA) dynamic management for IoT security function networks with SDN/VNF (virtual network function) [15]. Lu et al. proposed the SDTCP, which is a software-defined network (SDN) with a high-traffic TCP control mechanism solution to solve the TCP in cast problems for IoT applications. The SDTCP implemented to reduce the bandwidth of sending rate of background flows by modifying the open gate of TCP ACK packets [16]. Zhang et al. proposed SoftThings, which is a security framework on IoT devices based on SDN to capture and prevent attacks. This proposed system uses machine learning on the SDN controller to monitor and analyze the current normal observed behavior on the IoT devices to detect any suspicious activities that might occur in the future, such as denial of service (DoS) attacks [17].

Comparing our proposal in this paper with other existing systems and proposals, we introduced a novelty solution of system model and implementation using SDN to secure IoT devices that use HTTP to mitigate and prevent security attacks without the need to modify the IoT devices.

In this paper, we further investigate the use of SDN in IoT to improve the system's performance and enhance its security. Our proposed system model for SDN with IoT helps to prevent man-in-the-middle attack. Also, we applied traffic separation techniques using both deep packet inspection (DPI) along with SDN. We implemented the proposed system using Raspberry Pi [18] as the IoT device, we run Kodi Media Center [19] on the devices as the software media center, and we used Openflow Protocol to implement SDN. Our solution provides both confidentiality and integrity and mitigates various risks without the need to modify the IoT devices.

The subsequent sections of this paper will be organized as follows. In Section 2, we introduce the concepts of IoT and SDN and analyzed their structure and mechanisms. Next, in Section 3, we present the system model, system implementation, and man-in-the-middle attack on IoT. Then in Section 4, we show our system evaluation including the attack results and performance evaluation. In Section 5, we compare our system with other systems and discuss our future work. Finally, we conclude the paper in Section 6.

2. IoT and SDN

This section presents the definition, applications, architecture, and general security issues of Internet of Things (IoT). We then provide an overview of software defined networking (SDN), describing both the structure and integration between IoT and SDN. Then we present a real-world scenario, which is the motivation of this work.

According to the International Telecommunication Union, IoT is "(an) objects of the physical world (physical things) or of the information world (virtual world) which are capable of being identified and integrated into communication networks" [20]. With IoT technology, common physical objects may interact virtually, enabling them to be aware of events occurring at a far distance or respond to an event that it cannot detect physically.

Compared to personal mobile devices, IoT has limited resources in terms of processing power, storage, and volatile memory. Thus, it may need to connect to a Cloud Platform or Fog Platform for further data processing.

Three major components enable IoT functions:

(i) Hardware—the network of connected, sensors embedded objects/devices;
(ii) Software—program used for data collection, storage, transporting, processing, devices instructions;
(iii) Data communication—the protocols and technologies for exchange data.

2.1. *Applications of Internet of Things*

IoT is a part of our everyday life, from simple devices such as baby monitors, health bracelets, smartwatches, voice-activated speakers, smart refrigerators to more sophisticated devices such as self-driving cars and low-resource devices including wireless body sensor networks (WBAN) and medical wireless sensor networks (MWSN). Simply, IoT connected devices are all around us, Figure 1, in our homes, businesses, streets, neighborhoods and cities, and even in our bodies.

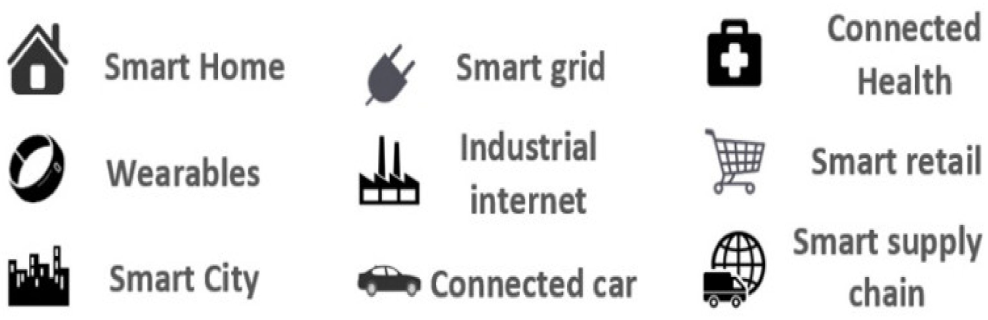

Figure 1. Forms of IoT.

As much as IoT plays an important role in our current lives, it is evident that the use of IoT will play a critical part in the infrastructure of technology in the coming years [21]. According to a recent prediction, in 2025 the total number of connected devices in the world will approximately be 75.44 billion, Figure 2 [22]. While companies are racing to produce new IoT devices with creative applications, in many cases, unfortunately, security comes as an afterthought. Companies may use dated security standard, if any.

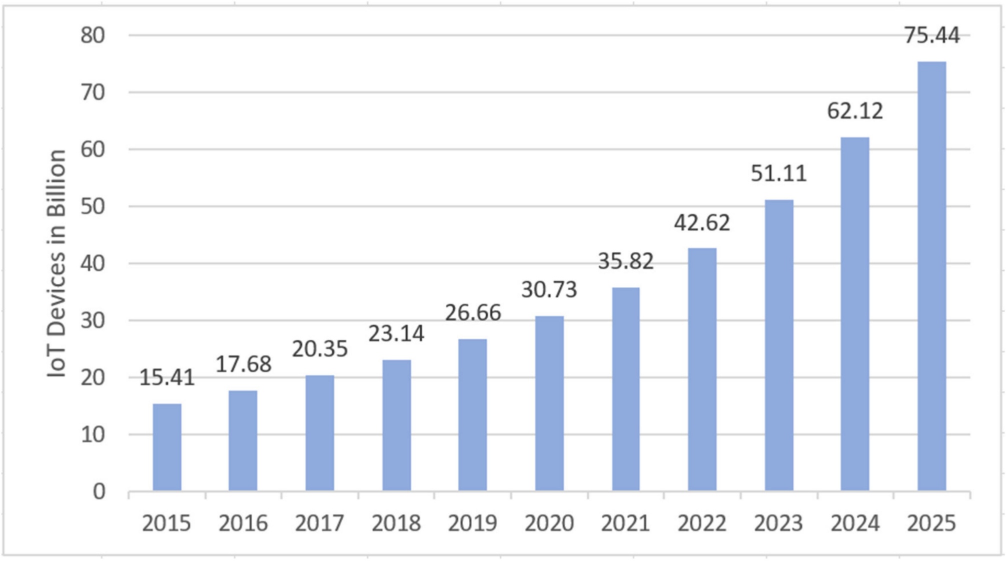

Figure 2. Internet of Things prediction from 2014 to 2025 (in billions).

2.2. Architecture of Internet of Things

There is no official standard architecture or design for an IoT network yet. However, based on the flow of data and different functions, IoT network can be divided into four main layers (Table 1):

1. Perception/sensing layer;
2. Networking/transport layer;
3. Service/management layer;
4. Application/interface layer [23,24].

Table 1. IoT network layers.

Layer	Function
Application, interface layer	Presenting, user's interaction Business applications
Service, management layer	Data processing, analyzing Generating useful information
Networking, transport layer	Data transmission over wire or wireless network
Perception, sensing layer	Hardware integration Identifying, collecting data

The perception/sensing/data collection layer includes perception nodes and networks. The networking/ transport layer has several sub-layers such as access network, core network, and LAN. The service/management layer is where data is processed and analyzed while the application/interface layer is where data is collected. Processed information gets consumed and is presented in business applications to support decision making.

2.3. Internet of Things Security Concerns

Most of the IoT devices share a simple design that is based on the premise that they can be operated quickly and simply, or that everyday devices can be converted to IoT with the addition of Internet connectivity. "The pressures of releasing a product quickly can sometimes lead to skipping non-visible aspects like security and reliability" [25].

It is obvious that security concerns are not always considered as part of the IoT device production cycle from hardware/software applications to the frameworks. Security researchers confirm that most recent IoT devices rely on cloud infrastructure for communication, which already have known security issues, and may cause IoT devices to become easy targets. Most IoT devices run on new development, nascent platforms that may have security vulnerabilities. To make matters worse, many IoT devices do not have the ability to update their firmware and software, making them extremely vulnerable and exposed to future exploits and attacks [26,27].

Attacks on the IoT can be divided into two main categories: attacks on architect layers, and attacks on data phases [28]. Yet another difference between IoT and the traditional Internet is that in the traditional Internet, the content and data are generated by human interactions. In the world of IoT, it is common that the data is collected and generated by smart machines (sensors, actuators). Machines do not lie, but they can be tricked to send or receive tainted information. The OWASP (2018) Internet of Things Project has laid out the top 10 security problems with IoT devices [29]:

1. Weak guessable, or hardcoded passwords;
2. Insecure network services;
3. Insecure ecosystem interfaces;
4. Lack of secure update mechanism;
5. Use of insecure or outdated components;
6. Insufficient privacy protection;
7. Insecure data transfer and storage;
8. Lack of device management;
9. Insecure default settings;
10. Lack of physical hardening.

To protect our privacy, we must ensure that the collected data and data communication meets the following requirements: (i) confidentiality—transmitted data, communication among endpoints, sensors and readers are secured and encrypted; (ii) integrity—transmitted data is accurate and complete; (iii) authenticity—transmitted data is verified and come from authorized sensors, endpoints, and readers.

2.4. *Software Defined Networking (SDN)*

SDN is a network architecture approach in which programmable switches are put between the data and control planes so that data forwarding can be managed and manipulated. SDN is a good alternative to traditional networks for several reasons. First of all, traditional networks are hardware-based so their infrastructure requires physical devices such as switches and routers, causing them to have some limitations in terms of speed and manipulability. However, SDN is software-based, so it can be managed virtually through the control plane. Rather than having fixed-functions, the software of SDN can be quickly and easily modified as needed. Secondly, the routers of traditional networks require high-level algorithms in order to determine the destination of the packets. When it comes to SDN, the SDN controller communicates with the network devices to centrally manage the flow of packets according to its configuration [30,31].

3. System Model

The IoT field is one of the fastest growing technology nowadays and new IoT devices are introduced every day. These devices are smart and connected to the Internet. Due to this dynamic type of environment, traditional networks are not the best option to meet the requirements of IoT. There is a need for a more dynamic and secure network infrastructure to support IoT operations. As elaborated earlier, SDN is a new technology that allows full control of the overall behavior of the network and prevents network overload. Additionally, the SDN controller provides useful debugging tools that can be used by the IoT environment to enhance security [32,33].

Figure 3 illustrates the IoT with SDN structure, the SDN controller allows up to separate the network into isolated subnets [30]. Furthermore, the SDN Controller interacts with the IoT application using a special application programming interface (API), known as 'Northbound API'. The later analyzes the network traffic and performs actions based on the configured rules. On the other hand, the controller uses another API (referred to as 'Southbound API') to communicate with the network switches based on configured rules [30]. Overall, the merge of IoT-SDN enhances IoT operations and security as it allows full and remote control of the network configuration without direct interactions with the IoT devices.

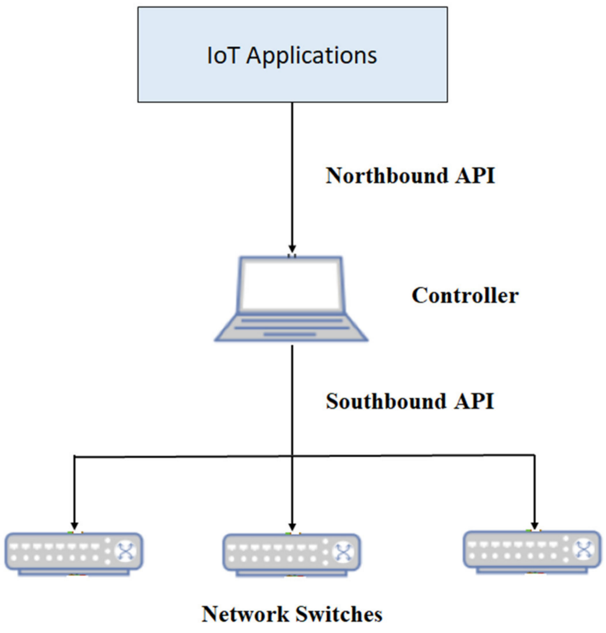

Figure 3. SDN and IoT Structure.

In our proposed system we used Openflow Protocol to implement SDN. The SDN switch operates using a flow table, similar to the routing table with traditional routers. However, it supports chaining and allows matching of a wider range of fields with each flow having associated actions. When a packet arrives at a switch, it is matched against the flow table, if a match is found, then the corresponding actions will be executed. However, if no match is found, which is likely to happen for a newly added device, the received packet is forwarded to the SDN controller through the Southbound API. The controller then inspects the packet and acts accordingly. It may install a new flow on the switch to allow future packets to be routed without having the controller involved. The SDN application will be informed through Northbound API.

3.1. System Implementation

To implement the proposed system, we used Raspberry Pi [18] as our IoT device and we run Kodi Media Center [19] on the devices as the software media center. A recently reported vulnerability of Kodi indicates that it uses Hypertext Transfer Protocol (HTTP) for downloading add-ons, thus making it susceptible to man-in-the-middle (MITM) attack. In [34], Liviu Arsene, demonstrated a successful malicious script injection on Kodi.

Besides serving as the primary delivery method of hypertext markup language (HTML) documents and multimedia, HTTP is also used as the basis of other protocols such as simple object access protocol (SOAP) and representational state transfer (REST). A recent survey [35] on IoT listed 7 application protocols for IoT, out of which 2 are based on HTTP, and 14 publicly-available cloud platforms for IoT do support REST.

Our system implementation consists of three entities: attacker *A*, server *S* and client *C*., as shown in Figure 4. The client models the Kodi Media Center, the server models the web servers hosting add-on files and the attack models the potential adversary for a MITM attack. The attacker will retrieve the file f_S and the certificate C_S from the server, then the attacker will hash the file $h_S = H(f_S)$, the hash function H is assumed to be collision resistant. On the other hand, the attacker then modifies the original file f_S and obtains f_A, calculates the new hash value $h_A = H(f_A)$ and generates a certificate c_A from private key p_A.

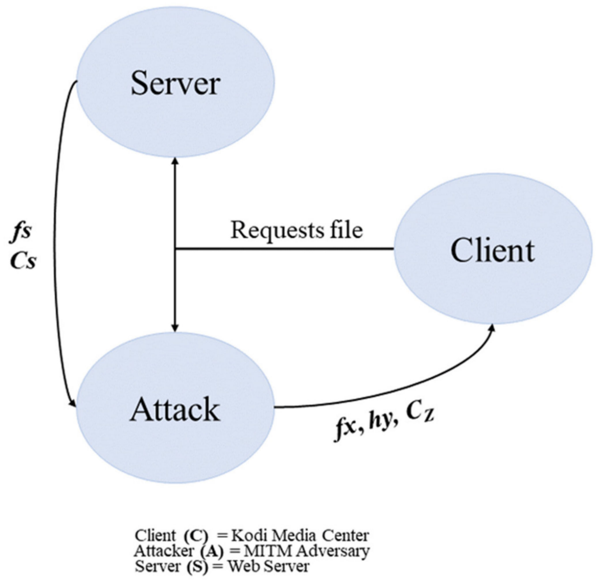

Figure 4. System implementation.

The client receives a list of trust certificates t then requests the file, hash of the file, and optionally the certificate from the attacker. The attacker then returns f_x, h_y and c_z respectively, where x, y, $z = \{S, A\}$. In other words, for each item the attacker retrieves from the server, the attacker can choose to return the original file. If a certificate is requested from the client, he/she can use the certificate c_z to send challenges to the attacker to verify that it holds the corresponding private key. The client then must answer '*TRUE*' when the received file has been modified, or when $f_x = f_A$. If the attacker is able to trick the client to answer '*FALSE*' when it sends f_A, then the attack is successful.

3.2. Man-in-the-Middle: Problem and Solution

The HTTP is inherently susceptible to a MITM attack due to the lack of authentication. One goal of the proposed system is to solve this problem. The solution to this problem is to use HTTP Secure (HTTPS), adding a transport layer security (TLS) on top of transport control protocol (TCP) to provide confidentiality as well as integrity.

While HTTPS could solve the problem, there are two difficulties that have to be considered before using HTTPS on IoT: code upgrade, and resource constraints. Using HTTPS instead of HTTP on the IoT will likely require changes in the IoT. In order for the changes to be made, the manufacturer can either issue a software/firmware update over the internet or recall the IoT and update it manually. Neither of these choices are ideal since a device recall will be costly and updates done over the internet will likely be done through HTTP anyway, thus creating an even bigger risk as an adversary can potentially gain control of the entire operating system on the IoT with a MITM attack. On the other hand, when compared to devices such as personal computers, IoT devices have more constraints in terms of their resources, namely the processing power, storage, volatile memory and energy, thus it is challenging for an IoT device to use HTTPS.

To address the problem of code upgrade and resource constraints, we propose using a TLS/SSL upgrading proxy (referred to as proxy or upgrading proxy) in transparent mode to handle the security layer. Using a proxy, the resources required for the TLS/SSL channel are offloaded to the proxy device and operate in transparent mode to avoid the issue of reconfiguring the IoT device. As illustrated in

Figure 5, HTTP connections are redirected by the switch to the proxy. This has the added benefit of simplifying the trusted certificate authority management process, since the proxy is the only device that needs to be aware of the complex public key infrastructure.

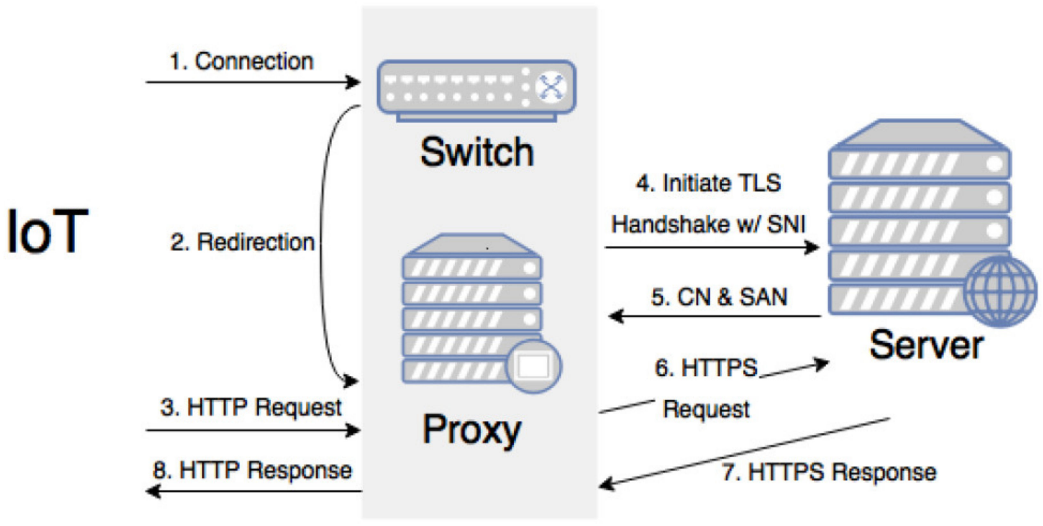

Figure 5. Operation of transparent mode.

3.3. Traffic Separation

Not all HTTP traffic in the network needs to be upgraded to HTTPS. In our system model, we address this situation, where traffic does not need to be upgraded and avoid unnecessary resource usage on the proxy. Other devices on the network, such as personal computers, may not need an HTTPS upgrade because they already utilize HTTPS. To separate IoT traffic apart from traffic of other devices on the network, our proposed model uses SDN controller to identify the IoT traffic by inspecting the source MAC address and SDN switch to redirect the IoT traffic to proxy, as illustrated in Figure 6.

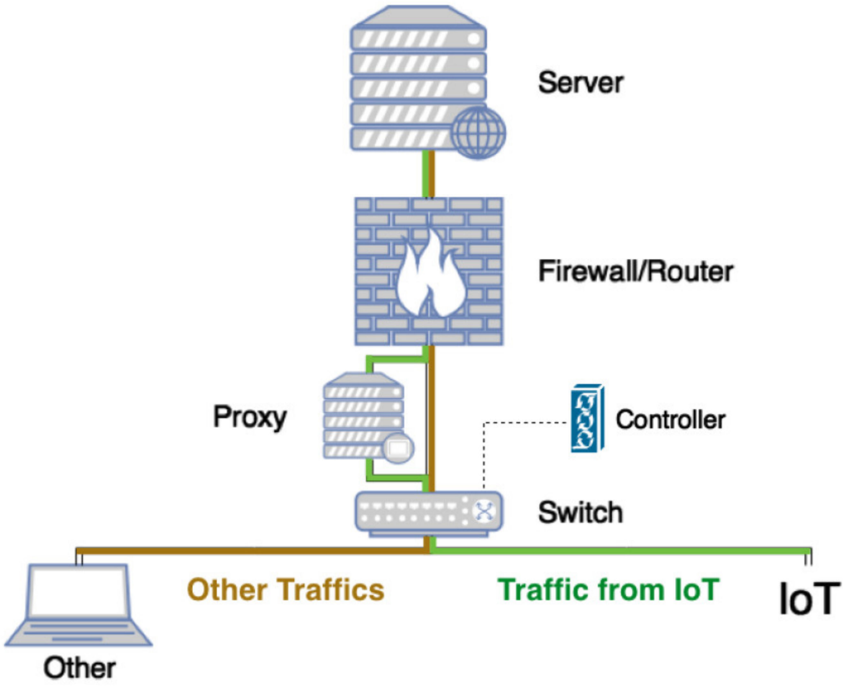

Figure 6. IoT traffic separation.

Additionally, not all the traffic coming from IoT needs to be secured and upgraded to HTTPS channel. Our model will use deep packet inspection (DPI) at the proxy to recognize security-related requests (e.g., SHA-2 hash value of add-on files) and send them through an HTTPS channel. On the other hand, it will act as a normal forwarding proxy for non-security related requests, as shown in Figure 7.

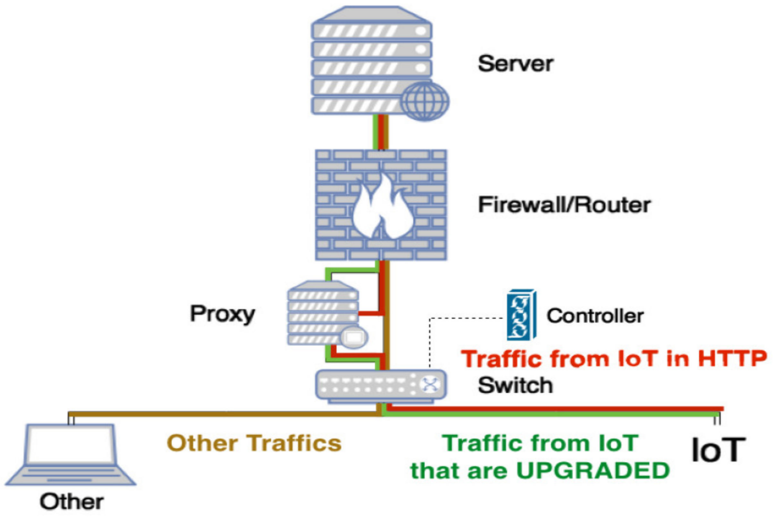

Figure 7. Separate security-related traffic.

4. System Evaluation

In this section, we present our system implementation which includes a HTTPS upgrading proxy that works in transparent mode, along with SDN that redirects IoT traffic to the proxy. Our system implementation runs on Mininet 2.2.1 [36] virtual machine, using Python 2.7.6 and POX Controller 0.2.0 which includes the virtual machine image, along with mitmproxy 0.15 [37].

The network environment, simulated with Mininet, consists of the following nodes: Server, Router, Proxy, Switch, Controller, Other, and Kodi. The Switch brings together four devices: Router, Proxy, Other, and Kodi. Controller is connected to Switch through a management interface that is separated from other devices connected to Switch. Router connects Server and Switch, which are under a different network. Figures 4 and 5 provide a logical view of the network topology which is easier to visualize the operation.

Two files with a random data are generated, one as f_S and the other as f_A in the model and the SHA-2 hash values h_S and h_A for these files are calculated, respectively. Two private keys and their corresponding certificates were generated using OpenSSL, one pair as c_S and p_S and the other as c_A and p_A.

In our implementation, we demonstrated the attack using a simplified approach. The MITM attack was launched by simply instructing the server a specific file to be served. To mimic the situation of a genuine server, it will have access to the private key p_S, server certificate c_S, file f_S and hash h_S. To mimic the situation of a MITM attack, the server will not have access to the private key p_S, and can serve either c_S or c_A as certificate, f_S or f_A as files and h_S or h_A as hashes; in this situation, p_A is always used as the private key no matter which certificate is served.

4.1. Attack Test Results

We have implemented the system as described in Section 3 and demonstrated in Figure 4. In our implementation, we have experimented all possible cases where an attacker: may or may not issue a fake certificate; may or may not create a fake file, may or may not generate a fake hash. The results of all the cases are shown in Table 2, where "HTTP" refers to an IoT device that uses http without a server and "HTTPS W/ Proxy" refers to an https proxy that is used by the IoT device. "Failed", in Table 2, refers to the case that the IoT device or Proxy server interpreted the certificate match as 'True' and the attacker failed to deceive the server. On the other hand, Succeed, in Table 2, refers to the case that the IoT device or Proxy server interpreted the certificate match as 'False' and the attacker succeeded to deceive the server. The results show that using a hash value can provide protection in many scenarios but the attacker can still succeed (case 2). There are two situations (cases 7 and 8) where an attack can trick the IoT into downloading a modified file while convincing the device that the file is genuine. On the other hand, we see that using an upgrading proxy protects the IoT in all the cases, hence, the proposed approach succeeded to prevent the attack in all the cases.

Table 2. System under attack scenarios.

	Case No.	Fake Certificate	Fake File	Fake Hash	HTTP	HTTPS W/Proxy
From Attacker	1	No	No	No	Failed	Failed
	2	Yes	No	No	Succeed	Failed
	3	No	Yes	No	Failed	Failed
	4	No	No	Yes	Failed	Failed
	5	Yes	Yes	No	Failed	Failed
	6	Yes	No	Yes	Failed	Failed
	7	No	Yes	Yes	Succeed	Failed
	8	Yes	Yes	Yes	Succeed	Failed

4.2. Performance Evaluation

The performance of the implemented system is evaluated based on the transfer speed. The transfer is evaluated against the requests sent to the network per second. The results are summarized in Figure 7 for the evaluation of the four causes of overhead: switch with and without proxy, deep packet inspection (passthrough), and ungraded to TLS (upgrade).

With "Bypass", the switch needs to examine the packet headers as compared to a switch that only does a simple L2 pairing "No Proxy". The Bypass switch aims to further differentiate traffic origin and the results clearly show that the switch overhead does not have a noticeable impact on the performance.

On the other hand, when the HTTP traffic from Kodi is associated with security related data, the proxy has to look into the HTTP payloads. In this case, referred to as "Passthrough", the proxy will perform deep packet analysis (DPI) which is obviously a process that is expected to add delays. Hence, as shown in Figure 7, there is a significant degradation in the transfer rate with Passthrough as compared to No Proxy and Bypass.

To summarize, with No Proxy mode, the Proxy was not present in the network and the Switch does not differentiate where traffic is coming from. In Bypass mode, the SDN switch recognizes the source of traffic as coming from Other and by pass the Proxy. With Passthrough mode, the traffic was redirected to the Proxy but was not upgraded to HTTPS. Lastly, in the Upgrade mode, the traffic was also redirected to the Proxy and upgraded to HTTPS. In this case, the encryption and authentication of plaintext HTTP payload may also add to the overhead. However, the results in Figure 8 shows that there are no significant differences in performance between traffic that was kept in HTTP (Passthrough) and traffic that was upgraded to HTTPS (Upgrade).

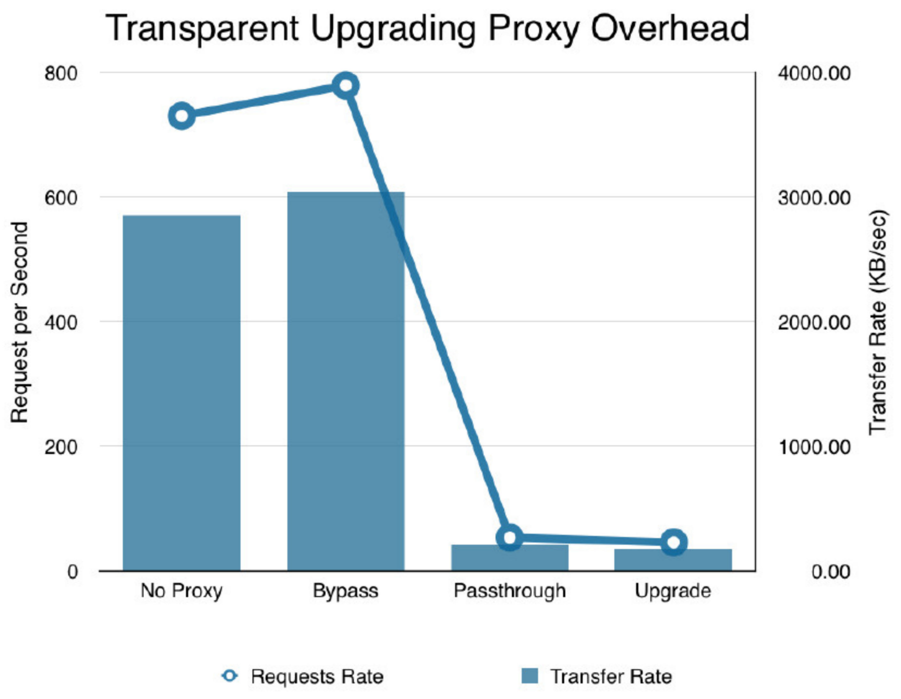

Figure 8. Performance comparison.

5. Comparison and Future Work

While firewalls such as iptables can be used to replace our proposed SDN to recognize traffic from IoT. Using SDN has two advantages: flexibility and scalability. SDN provides better flexibility, as it has a standard set of APIs that are supported by SDN Controller and switches, thus making it ready for being manipulated programmatically. An example where SDN stands out against firewall is where there is a database of MAC addresses of the IoT devices readily available. With SDN, we can allow the controller to query the database and install a rule on how to direct the traffic. As packets from new devices arrive, the switch will have an optional expiration time for the rule designated by the controller.

On the other hand, using a firewall administrator requires either having to lookup and add a new redirection rule to the firewall or simply adding the IoT MAC Addresses all at once. Neither of the two is ideal since adding a rule manually is error-prone and adding the entire database may slow the matching process and cause additional delays.

Kodi is an open-source media center that allows users to access videos and music on almost any device. The version of Kodi add-ons that is used in this work only supports HTTP [38]. While not all traffic sent to or from IoT devices is in cleartext only when using HTTP, HTTP is more likely to become a vehicle of various functionality, and not limited to be used to exchange plain-text or media. Protocols such as simple object access protocol (SOAP) and representational state transfer (REST) based protocols were usually built on top of HTTP. Hence, although an HTTPS upgrading proxy does not mitigate all possibility of an MITM attack on IoT, it should cover the majority of the cases.

While in Section 4.2, it was shown that DPI on proxy incurred a significant degradation on the performance, switch overhead and TLS overhead remain insignificant. It is possible that computational efficiency may play a major role. Both the switch (Open vSwitch [39]) and the TLS layer (handled by OpenSSL [40]) were done by programs written in C, the DPI is done by mitmproxy's scripting framework, using Python. If the performance sensitive code of mitmproxy was rewritten in C, it may allow the prototype for a wider application. This is something we would be looking to investigate further in our future work.

We believe that SDN has the potential to handle the issue of scalability and limited size of the ternary content addressable memory (TCAM) better than the other systems. Mainly, because it is software based and with the recent hardware improvements, in terms of processing and memory, it makes SDN more resilient to limited size issues and capable of managing the MAC addresses for large number of IoT devices. In our future work, we plan to further analyze the scalability of the system and investigate the use of a dynamic classification solution, such as the one proposed by Polverini et al. [41]. It is likely that our system, Figure 7, will benefit from a dynamic classification to help separating security-related traffic.

6. Conclusions

In the past few years, the Internet of Things (IoT) has become an integral part of our everyday lives. There is a significant demand to supply the world with more smart devices. As more devices become connected to the internet, there will be an increasing demand for information security since sensitive information can be intercepted. We investigated the integration of SDN with IoT to enhance the system's security. Then we further examine SDN as a new network technology and the combination structure of IoT with SDN. With implementation and testing, we demonstrated that it is possible to protect IoT devices that can only use HTTP because of resource constraints without device modification. With the combination of SDN and upgrading proxy, integrity and authenticity is provided with the use of SSL/TLS layer, and our results demonstrated that MITM attack on our system model is mitigated.

Author Contributions: Formal analysis, A.A.H.; Investigation, A.A.H.; Methodology, A.A.H.; Project administration, M.Z.A.B.; Software, A.A.H.; Supervision, M.Z.A.B. and I.M.; Writing—original draft, A.A.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gan, G.; Lu, Z.; Jiang, J. Internet of Things Security Analysis. In Proceedings of the 2011 International Conference on Internet Technology and Applications, Wuhan, China, 16–18 August 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 1–4.
2. Software-Defined Networking: The New Norm for Networks. Available online: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf (accessed on 5 January 2020).
3. Tariq, J.; Riaz, T.; Rasheed, A. A Layer2 Firewall for Software Defined Network. In Proceedings of the 2014 Conference on Information Assurance and Cyber Security (CIACS), Rawalpindi, Pakistan, 12–13 June 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 39–42.
4. Michelle, S.; Park, S.H.; Lee, B.; Yang, S. Building Firewall over the Software-Defined Network Controller. In Proceedings of the 16th International Conference on Advanced Communication Technology, Chennal, India, 27–28 February 2013; IEEE: Piscataway, NJ, USA, 2014; pp. 744–748.
5. Pena, J.G.V.; Yu, W.E. Development of a Distributed Firewall Using Software Defined Networking Technology. In Proceedings of the 2014 4th IEEE International Conference on Information Science and Technology, Busab, Korea, 26 February–1 March 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 449–452.
6. Rolbin, M. Early Detection of Network Threats Using Software Defined Network (SDN) and Virtualization. Master's Thesis, Carleton University, Ottawa, OT, Canada, 2013.
7. Sood, K.; Yu, S.; Xiang, Y. Software-Defined Wireless Networking Opportunities and Challenges for Internet-of-Things: A review. *IEEE Int. Things J.* **2015**, *3*, 453–463. [CrossRef]
8. Zhijing, Q.; Denker, G.; Giannelli, C.; Bellavista, P.; Venkatasubramanian, N. A Software Defined Networking Architecture for the Internet-of-Things. In Proceedings of the 2014 IEEE Network Operations and Management Symposium (NOMS), Krakow, Poland, 5–9 May 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 1–9.
9. Yaser, J.; Al-Ayyoub, M.; Benkhelifa, E.; Vouk, M.; Rindos, A. SDIoT: A software defined based internet of things framework. *J. Ambient. Intell. Humaniz. Comput.* **2015**, *6*, 453–461.

10. Liu, J.; Li, Y.; Chen, M.; Dong, W.; Jin, D. Software-defined internet of things for smart urban sensing. *IEEE Commun. Mag.* **2015**, *53*, 55–63. [CrossRef]
11. Salman, O.; Abdallah, S.; Elhajj, I.H.; Chehab, A.; Kayssi, A. Identity-Based Authentication Scheme for the Internet of Things. In Proceedings of the 2016 IEEE Symposium on Computers and Communication (ISCC), Wrocław, Poland, 7–9 September 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1109–1111.
12. Chakrabarty, S.; Engels, D.W.; Thathapudi, S. Black SDN for the Internet of Things. In Proceedings of the 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems, Dallas, TX, USA, 19–22 October 2015; IEEE: Piscataway, NJ, USA; pp. 190–198.
13. Theodorou, T.; Violettas, G.; Valsamas, P.; Petridou, S.; Mamatas, L. A Multi-Protocol Software-Defined Networking Solution for the Internet of Things. *IEEE Commun. Mag.* **2019**, *57*, 42–48. [CrossRef]
14. Tran, A.K.; Piran, M.; Pham, C. SDN Controller Placement in IoT Networks: An Optimized Submodularity-Based Approach. *Sensors* **2019**, *19*, 5474. [CrossRef]
15. Molina Zarca, A.; Garcia-Carrillo, D.; Bernal Bernabe, J.; Ortiz, J.; Marin-Perez, R.; Skarmeta, A. Enabling virtual AAA management in SDN-based IoT networks. *Sensors* **2019**, *19*, 295. [CrossRef] [PubMed]
16. Lu, Y.; Ling, Z.; Zhu, S.; Tang, L. SDTCP: Towards datacenter TCP congestion control with SDN for IoT applications. *Sensors* **2017**, *17*, 109. [CrossRef] [PubMed]
17. Zhang, A.; Lin, X. Security-Aware and Privacy-Preserving D2D Communications in 5G. *IEEE Netw.* **2017**, *31*, 70–77. [CrossRef]
18. Raspberry pi—Teach, Learn, and Make with Raspberry pi. Available online: https://www.raspberrypi.org/ (accessed on 5 January 2020).
19. Kodi j Open Source Home Theatre Software. Available online: http://kodi.tv/ (accessed on 5 January 2020).
20. Overview of the Internet of Things. Available online: https://www.itu.int/rec/T-REC-Y.2060--201206-I (accessed on 5 January 2020).
21. Szymanski, T.H. Security and privacy for a green internet of things. *IT Prof.* **2017**, *19*, 34–41. [CrossRef]
22. Alam, T. A Reliable Communication Framework and Its Use in Internet of Things (IoT). *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.* **2018**, 2456–3307.
23. Jing, Q.; Vasilakos, A.V.; Wan, J.; Lu, J.; Qiu, D. Security of the Internet of Things: Perspectives and challenges. *Wireless Netw.* **2014**, *20*, 2481–2501. [CrossRef]
24. Ning, H.; Liu, H.; Yang, L.T. Cyberentity security in the internet of things. *Computer* **2013**, *46*, 46–53. [CrossRef]
25. Palan, V. National Privacy Day Panel: Driving Privacy and Security in IoT. Available online: https://www.intertrust.com/intertrustblog/national-privacy-day-panel-driving-privacy-and-security-in-iot/ (accessed on 5 January 2020).
26. Dhanjani, N. *Abusing the Internet of Things: Blackouts, Freakouts, and Stakeouts*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2015.
27. Leswing, K. A massive cyberattack knocked out major websites across the internet. Available online: https://www.businessinsider.com/amazon-spotify-twitter-github-and-etsy-down-in-apparent-dns-attack-2016-10?r=UK (accessed on 5 January 2020).
28. Hu, F. *Security and Privacy in Internet of Things (IoTs): Models, Algorithms, and Implementations*; CRC Press: Boca Raton, FL, USA, 2016.
29. OWASP Internet of Things Project. Available online: https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10 (accessed on 5 January 2020).
30. Bizanis, N.; Kuipers, F.A. SDN and virtualization solutions for the Internet of Things: A survey. *IEEE Access* **2016**, *4*, 5591–5606. [CrossRef]
31. Kalkan, K.; Zeadally, S. Securing internet of things with software defined networking. *IEEE Commun. Mag.* **2017**, *56*, 186–192. [CrossRef]
32. Nobakht, M.; Sivaraman, V.; Boreli, R. A Host-Based Intrusion Detection and Mitigation Framework for Smart Home IoT using OpenFlow. In Proceedings of the IEEE 11th International. Conference Availability, Reliability and Security (ARES), Salzburg, Austria, 31 August–2 September 2016; pp. 147–156.
33. Bull, P.; Austin, R.; Popov, E.; Sharma, M.; Watson, R. Flow Based Security for IoT Devices Using an SDN Gateway. In Proceedings of the 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud), Vienna, Austria, 22–24 August 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 157–163.

34. Kodi Media Center Vulnerability Exposes Users to Man-in-the-Middle Attacks. Available online: https://hotforsecurity.bitdefender.com/blog/kodi-media-center-vulnerability-exposes-users-to-man-in-the-middle-attacks-12065.html (accessed on 5 January 2020).
35. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [CrossRef]
36. Mininet: An Instant Virtual Network on Your Laptop (or Other PC). Available online: http://mininet.org/ (accessed on 5 January 2020).
37. Mitmproxy. Available online: https://mitmproxy.org/ (accessed on 5 January 2020).
38. ROOT. Available online: http://mirrors.kodi.tv/ (accessed on 5 January 2020).
39. Open vSwitch. Available online: https://openvswitch.org/ (accessed on 5 January 2020).
40. OpenSSL. Available online: https://www.openssl.org (accessed on 5 January 2020).
41. Polverini, M.; Galán-Jiménez, J.; Lavacca, F.G.; Cianfrani, A.; Eramo, V. Dynamic In-Network Classification for Service Function Chaining ready SDN Networks. In Proceedings of the 2019 10th International Conference on the Network of the Future (NoF 1019), Rome, Italy, 1–3 October 2019.