Toward Consumer-Friendly Security in Smart Environments

Ruth M. Ogunnaike*, Brent Lagesse†

Computing and Software Systems
University of Washington Bothell
Email: *tunrayo@uw.edu, †lagesse@uw.edu

***Abstract*—The use of Internet of Things (IoT) devices has grown significantly in the past decade. While IoT is expected to improve life for many by enabling smart living spaces, the number of security risks that consumers and businesses will face is also increasing. A high number of vulnerable IoT devices are prone to attacks and easy exploit. Existing research has focused on security that must be implemented by administrators and manufacturers to be effective. Our work focuses on a system that does not rely on best practices by IoT device companies, but rather allows inexperienced users to be confident about the security of the devices that they add to their network. We present an implementation of an IoT architectural framework based on Software Defined Networking (SDN). In this architecture, IoT devices attempting to join an IoT network are scanned for vulnerabilities using custom vulnerability scanners and penetration testing tools before being allowed to communicate with any other device. In the case that a vulnerability is detected, the system will try to fix the vulnerability. If the fix fails, then the user will be alerted to the vulnerability and provided with suggestions for fixing it before it will be allowed to join the network. Our implementation demonstrates that the approach works and causes minimal overhead to the network once the device is deemed trustworthy.**

Keywords: Internet of Things (IoT); Software Defined Networking; Vulnerability detection; Security

I. Introduction

Internet of things (IoT) is a collection of interconnected embedded computing devices that can communicate to provide data and services for a variety of applications [1] that drive smart living spaces. Much of the software installed on IoT devices has focused primarily on functionality and has not undergone significant security review [2]. As IoT-connected devices become an integral part of our daily lives, it is crucial that these devices undergo thorough testing and establish minimum baseline for security; however, many devices have already been deployed insecurely as demonstrated by a number of recent distributed denial of service attacks that leverage IoT devices [3]. Furthermore, many manufacturers currently fail to implement well-established security standards correctly or emerging techniques at all. While it is critical to engineer secure systems [4], the focus of our work is to make it easier for consumers to deploy their IoT devices without having to worry about their devices being made part of an attack or revealing personal information about them [5] due to negligence by the manufacturer. The work described in this paper focuses on the system we have implemented and tested for detecting vulnerable devices, preventing them from being available on the network, and automatically fixing vulnerabilities in the device.

Techniques such as the static analysis [2], [6] of source code that runs on the IoT device and dynamic analysis [2] that examines the physical response of embedded systems in the IoT devices to variables that change with time (e.g. location, user behaviour) have been used to mitigate threats; however, in many cases these tests are not sufficiently performed. As a result, our goal is to protect users against devices where the manufacturer has failed to perform proper security practices.

This paper presents an implementation of a SDN-based security framework whereby IoT devices are scanned for vulnerabilities when they are first added to the network. Figure 1 shows the process flow of the proposed framework.

In this paper, we discuss the vulnerabilities existing in IoT devices and how the security framework mitigates these vulnerabilities. We describe the system design and implementation of the proposed security framework and how it improves IoT security while utilizing SDN architecture, custom vulnerability scanner and penetration testing tools. Furthermore, we discuss our work in automated vulnerability patching.

A. Major Contributions

This research work prevents vulnerable devices from gaining access to an IoT network and blocks communication to and from such devices. That is, the ability of the IoT device to share or receive data with other hosts/devices is disabled. The framework is scalable and adaptable which enables easy integration of varieties of penetration testing tools. While the focus of our research is to enable non-technical users to securely add IoT devices to their network, the framework can also be used by corporate organizations to secure and mitigate against vulnerable devices in their IoT networks.

B. Process Flow

Figure 1 shows the process flow of the proposed security mechanism. For an IoT device to join the network, the device sends a DHCP request to the DHCP server to be assigned an IP address. The DHCP server leases an IP address to the device and initiates a vulnerability scan. This is done by invoking the scan server which in turn connects to the integrated penetration testing tool via a REST API and connects to the custom

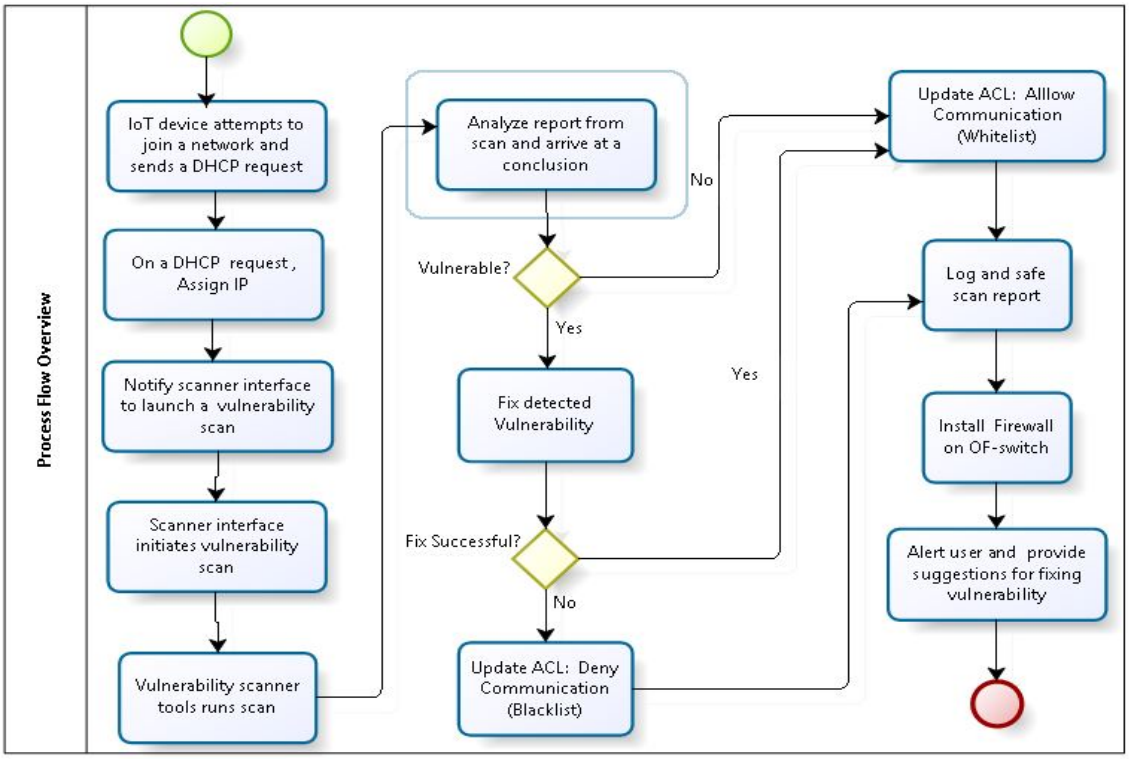

Fig. 1. Process Flow of Proposed Framework

vulnerability scan via remote procedure call (RPC). For our implementation, the Nessus penetration testing tool and a custom scanner that scans for weak/default password were integrated with our framework. We have also demonstrated the ability to implement custom scanners that provide efficient vulnerability scanning for issues that are most common to IoT devices [7].

The scan server launches the vulnerability scan and analyzes the results to determine if the IoT device is vulnerable. IoT devices that are perceived to be vulnerable are blacklisted and non-vulnerable devices are whitelisted. The scan server updates an Access Control List (ACL) managed by the DHCP server. The ACL is utilized by the firewall component of the framework to define rules and update the flow tables of the OpenFlow switch. Our approach also works in the case of static IP addresses as those devices will not be whitelisted yet, so all traffic to or from that IP address will be blocked until the device is whitelisted. On the first attempt of an IoT device with a static IP address to communicate in a network, the device undergo all required scans.

After the scan is complete, the system examines the vulnerabilities and then determines if it knows any methods for fixing those vulnerabilities. If it does, it will apply the fix and if the fix is successful for all the vulnerabilities on a device, the device will be whitelisted and granted access to the network. When a device is deemed vulnerable, an email will be sent to the user explaining the vulnerabilities and offering suggestions to the user for fixing the vulnerabilities. This paper further discusses the process flow and other relevant design decisions in sections III and IV.

II. Related Work

Researchers have deployed several measures to address security issues in each layer of the IoT framework. Xin M. proposed a hybrid encryption techniques (cryptographic paradigm) that provides the benefit of the symmetric key and asymmetric key performance [8] for IoT security. The approach focus on securing the application layer of the IoT to ensure information integrity, confidentiality, non-repudiation on the data transmitted in IoT by using a mixed encryption algorithm; Advance Encryption Standard (AES) and Elliptic Curve Cryptography (ECC) algorithm. Messages and data sent and received over the IoT network are encrypted. The ECC algorithm was used as digital signatures and AES was used to encrypt the data.

Padraig Flood proposed a protocol that combines zero-knowledge proofs and key exchange mechanisms to provide secure and authenticated communication in static machine-to-machine networks [9]. The protocol requires a-priori knowledge about the network setup and structure, and it guarantees perfect forward secrecy. Zero-knowledge proofs (ZKP) are challenge/response authentication protocols, in which parties (each IoT network) are required to provide the correctness of their secrets without revealing any information which could be used to help another party deduce these secrets.

Quangang Wen [10] , implements the use of cipher security certificate. The certificate provides a method of one time one cipher between communicating parties (sensor nodes in IoT). It uses a lightweight encryption or decryption method, using time stamp technology, timeliness in the two communication nodes is guaranteed.

[1] presents the concept of mutual trust for inter-system security in IoT by creating an item-level access control framework. Trust is established from the creation to operation and transmission phase of the IoT. The trust is established by the creation key and the token mechanism. A new IoT device added to the network is assigned a creation key by an entitlement system, and the token is created by the system administrator (the token is combined with the unique identifier of the device).This mechanism ensures the change of permissions by the device itself if it is assigned a new owner.

To overcome the heterogeneity of various IoT devices, software and protocols, Anggorojati [11] proposed a model that depicts access delegation realized by means of a capability propagation mechanism, and the incorporation of contextual information as well as secure capability propagation under federated IoT environments. It uses identity-based capability-based access control approach in securing IoT. The model takes into consideration the flexibility and scalability that are key features in IoT systems.

Leo M., [12] defines the security needs proposing a federated model to design. The models aims at securing the authenticity and integrity of the software installed on the IoT device by defining policies and standards to ensure security; and provides mechanisms to enforce such policies are followed. E.g. Software must be authorized to run on the devices and has to be signed by an entity that authorized for it, creation of policies that can limit privileges of device, components and applications so that they only have access to the needed resources.

Michael W. et al [13] proposed the use of smart edge IoT devices for safer, rapid response with industry IoT control application in which the control systems are remotely managed. The control system leverage user client devices connected to the Internet using a gateway device that can issues suitable control sequences and sends notifications of unusual usage critical events.

A similar approach has been used in detection of malware in USB devices. [14] forces a USB drive that is plugged in to undergo a series of virus scans prior to it being made available to the user or any other part of the system. Our system obviously differs in that we are concerned with network access rather than auto-running malware, but also in that we are also focused on healing the vulnerabilities and providing useful information to the user.

Other approaches include the use of protocol simulators [15] where there is huge variety of device end-points and interfaces to validate. Data recorders [16] used for smart validation across device sets whereby the recorded data can be played across different device end-points automatically, which in turn can be a great enabler in compatibility testing of apps across different device sets and communication layers

In all of this work on IoT security, the security provided requires that the manufacturer, architect, or administrator performs time-intense and highly technical tasks. In many cases, the person responsible will not perform these due to factors such as cost or inability. Our approach does not rely on the expertise and diligence of users or manufacturers and provides confidence to the user that their system is unlikely to be compromised easily.

III. System Design

The research aim at securing the IoT network by preventing vulnerable devices from joining the network to avoid providing attackers the opportunity to launch an attack. The main function of the framework is to scan for and fix vulnerabilities on IoT devices as shown in figure 1. Note that while we used an SDN for its ease of deployment and testing, it would be possible to create a similar system using traditional routers.

According to OWASP, the following are some of the most common IoT vulnerabilities. We have developed or are developing scanners that focus on each of these items.

- **Insufficient Authentication/Authorization** many IoT devices are secured with low quality passwords, send credentials without using encrypted transport, or require no passwords at all.
- **Insecure Network Services** some IoT devices enable insecure services like Telnet, FTP etc.
- **Lack of Transport Encryption** failure to use transport encryption to protect the data and credentials sent over the network.
- **Insufficient Security Configurability** configuration of IoT devices is made difficult by poor UI design and a lack of traditional I/O interfaces.
- **Insecure Software/Firmware** Software updates are often not digitally signed. This can allow an attacker to install code on the device including backdoors or the data collection malware.

In this paper, we discuss how enhance security of IoT by implementing security service (a scan server) that scans for vulnerabilities in an IoT device. Our attacker model includes both malware running on existing systems in the network and outside attackers that would try to manipulate the IoT devices. The IoT devices that are being installed are assumed to be potentially vulnerable, but not actively malicious. In other words, the IoT device may be running software with a known vulnerability, but it will not actively lie about its MAC address in order to subvert our system.

Our system consists of the following components: SDN Controller, DHCP Server, Scan Server, Host Tracker, Access Control List, and Firewall. The following subsection discusses each of these components.

A. SDN Controller

The SDN controller can be centralized or distributed. It exchanges protocol updates and maintains the routing table through exchanging updates between other controllers. The centralized controller has network intelligence and a global view of the network and can manage the entire network. The controller consists of components that are used to control and manage the network to enhance security. In this paper, we configure the DHCP server sub-component of the SDN controller to initiate vulnerability scan and implement a firewall component to control the communication of hosts.

B. DHCP Server

The DHCP subcomponent handles DHCP requests from IoT devices. The DHCP server leases an IP address to the device and initiates a vulnerability scan via an interface provided by a scan server.

C. Scan Server

This component provides an interface for the DHCP server component to send requests to launch a vulnerability scan on an IoT device/host in the network. It is a service in the application layer of the architecture provided to the SDN controller in the control plane of the network. The server connects to the integrated penetration testing/vulnerability scanning tools via a REST API or RPC to launch the vulnerability scan. The vulnerability scan can run iteratively for multiple penetration testing tools. It also analyzes the result of the scan to reach a conclusion about the risk status of the IoT i.e. deduce if the IoT device should be flagged as vulnerable or non-vulnerable. The scan server initiates a fix for detected vulnerabilities and alerts the user to the vulnerability and provide suggestions for fixing the vulnerability if the fix fails. The access control list is updated accordingly based on the feedback of the scan server. IoT device perceived to be vulnerable are blacklisted and communications are blocked to and from such device. The non-vulnerable device are whitelisted and granted access to the network. In addition, the detailed report of the scan is saved in the network server for audit.

D. Host Tracker

This is a sub-component of the POX controller. It tracks hosts in the network and detects host that has not been previously scanned in the network. When a host with a static IP address attempts to send a packet to another host in the network, this component is configured to check if the host has been previously scanned. A vulnerability scan is launched via the scan interface for hosts that have not been scanned for vulnerabilities.

E. Access Control List

The access control list is used to define rules and it is utilized by the firewall. It helps the firewall to identify vulnerable devices in the network. It provides a pair of source and destination MAC address that identifies where to restrict communication. Blacklisted devices are restricted from communicating with other hosts on the network.

F. Firewall

The firewall is a sub-component of the SDN controller that enforce security policies and restricts unauthorized network access in the IoT network by filtering network traffic based on the predefined rules in the access control list. The firewall prevents all forms of communication to vulnerable IoT devices in a network. This component inserts rules in the controller; and the controller updates the flow table of the OpenFlow switch. The flow table entry contains a set of rules (e.g. IP source) to match and an action list (e.g. forward to port, drop etc.) to be executed in case of a match.

IV. Implementation

To validate the proposed architecture, we implemented the main modules[1] and analyzed their performance. The modules were integrated and installed in a virtual machine (VM) and performance evaluations were done on the VMs based on our implementation.

Mininet simulation tool was used to create a realistic virtual IoT-network that mimics the properties and functionalities of a real network. We set up a logical test bed of 3 hosts with no assigned IP address connected to an OpenFlow (OF) switch and a remote controller. POX was used as the remote controller and the mininet simulation tool makes it possible to connect the OF switch with POX remote controller.

Besides using the functions provided by POX, such as network topology maintenance, routing calculation, DHCP request handler, interacting with switches through OpenFlow protocol, we implemented the firewall component on POX, which communicates with the forwarding function of the controller to configure the flow tables of the OF switch.

The firewall restricts the communication between certain devices/hosts when necessary based on an access control list, a list of pairs of MAC address that defines the control for each IoT device in the network. The firewall inserts rule in the controller which will drop offending packets (packets from blacklisted host) and the controller updates the flow tables of the OpenFlow switch that filters the network traffic.

We configured the DHCP server component to launch a vulnerability scan when an IP address is leased to an IoT device. The configuration was done by importing the interface provided by the scan server. This interface allows the DHCP component to launch the vulnerability scan. The host-tracker component also launches vulnerability scans for hosts configured with static IP address. This components runs when a host attempt to send packets or when packets are sent to it.

The scan server implemented with Python connects to integrated vulnerability scanning tools via a REST API (for penetration testing tools) and RPC (for custom scanners) . Our system enables the creation of custom scanners or the ability to wrap existing scanners with the REST API and use them. To demonstrate this, we have wrapped Nessus vulnerability scanner along with a custom weak password scanner.

V. Experimental Results and Discussions

After implementing the required modules, we ran tests to ensure the framework functions as intended. The VMs run a bridged network adapter that allows external communication.

The network topology is invoked by executing the logic topology; a python script. POX is invoked by running pox.py followed by the modules required to test our framework. For our implementation, we invoked the forwarding.l2learning (forwarding function), proto.dhcpd (DHCP server), misc.firewall (firewall component), openflow.discovery (OF-switch topology discovery), and host-tracker (for tracking scanned hosts). A DHCP request is sent for each host to assign IP addresses. On an event the DHCP

[1] https://github.com/SecurityInEmergingEnvironments/IoTScanner

server leases an IP address, vulnerability scan is invoked and firewall policies are updated accordingly based on the scan result. The firewall component is invoked on an event a firewall policy update occurs.

Network simulation was carried out several times to configure the scan server to run different types of vulnerability scan. Table I shows the type of scans run to test our framework. It shows the function of the scan and the types of vulnerabilities that can be detected based on the scan type.

The first simulation is a single topology mininet network with 3-hosts, 1-OF-switch and a remote controller. For every DHCP request sent for each hosts, a "Basic Network Scan" was launched after they are assigned an IP address. We configured the DHCP server to lease an IP address range of 192.168.0.12 - 192.168.0.254. Based on the results, one of the host was perceived as vulnerable and two hosts were deduced safe based on our analysis.

The access control list generated by the scan server shows that host h2 was perceived to be vulnerable. The firewall policy list denotes communication should be disable between host h2 and other hosts in the network (h1, h3). This function is carried out by the firewall component. Figure 2 shows communication was disabled based on the firewall policy.

```
mininet> h2 ping -c1 h3
PING 192.168.0.16 (192.168.0.16) 56(84) bytes of data.
From 192.168.0.15 icmp_seq=1 Destination Host Unreachable

--- 192.168.0.16 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms

mininet> h3 ping -c1 h1
PING 192.168.0.14 (192.168.0.14) 56(84) bytes of data.
64 bytes from 192.168.0.14: icmp_seq=1 ttl=64 time=16.9 ms

--- 192.168.0.14 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 16.975/16.975/16.975/0.000 ms
mininet> h3 ping -c1 h2
PING 192.168.0.15 (192.168.0.15) 56(84) bytes of data.

--- 192.168.0.15 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

mininet> h1 ping -c1 h2
PING 192.168.0.15 (192.168.0.15) 56(84) bytes of data.

--- 192.168.0.15 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

Fig. 2. Communication flow between Blacklisted and White-listed Hosts

In the second test, we used a single network topology of 6 hosts with pre-assigned (static) IP address to test the functionality of the host tracker component. Pings were sent at random between each hosts. For every packet sent by a host to another host, the host tracker checks if both source and destination host has been scanned. This is done by checking the access control list (the whitelisted and blacklisted MAC addresses). The host tracker launches a vulnerability scan for hosts that have not been scanned for vulnerabilities and whitelist or blacklist the host based on the scan report from the scan server. To test the effect of running an extra check to confirm if a device/host has been scanned or not, we compared the time it takes to send and receive packets between hosts via the controller and without the controller. Our test shows the average response time when packets are sent via the controller is 31.7654ms and without the controller is 26.7143ms. There is a difference of 5.0511ms.

Figure 3 shows the percentage usage of the network bandwidth with and without a controller. The difference between having the controller present or not was negligible as the average percentage bandwidth usage when packets are sent via the controller is 90.75 percent and 91.15 percent when packets are sent without the controller.

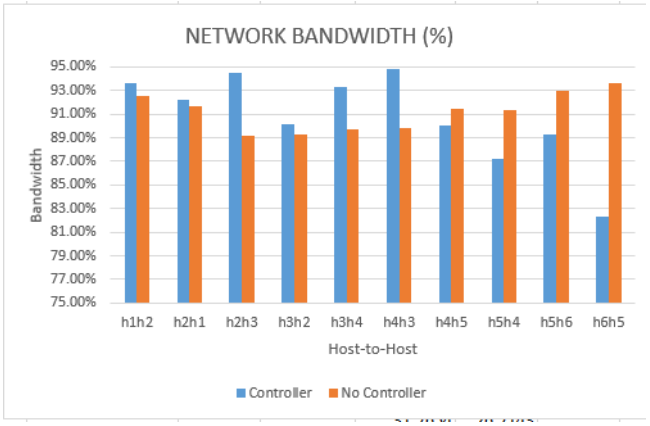

Fig. 3. Network Bandwidth between Hosts

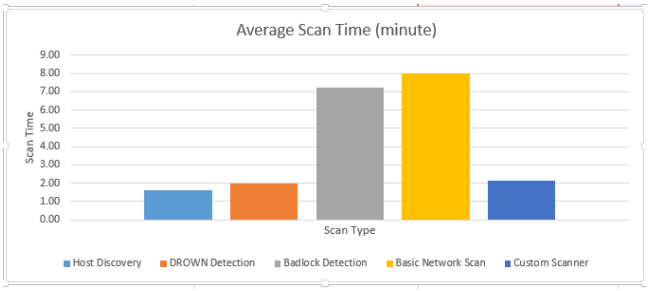

Fig. 4. Average Scan Time

We also ran a test using our custom scanner that detects common username/passwords used for login. Specific ports in the device are checked to detect if its opened or closed before the combinations are run. Devices with closed ports are flagged non-vulnerable. A total of 462 weak/default usernames and passwords combinations were run against an open port. It takes an average of 1698.352 seconds to run all 462 combinations, with an average of 3.676 seconds for one combination. In cases where there is a successful login access to the device using a weak/default username and password combination, the system changes the password on behalf of the user without the users intervention and provides the information to the user. The new password is generated by random combination of alphanumeric characters of length eight and it takes an average of 0.646 seconds to resolve a weak/default password vulnerability. This time includes the time it takes to send the user the newly generated password via email.

As shown in figure 4, it takes an average of 2.12 minutes to scan for weak/default password using a custom scanner which is significantly smaller than the time (7.25 minutes) it takes to scan for Badlock detection using the Nessus penetration testing tool. This shows we can detect IoT vulnerabilities in a reduced amount of time if we implement only custom

TABLE I
TYPES OF SCAN

Scan Type	Description	Vulnerability
Basic Network Scan	Performs a full system and network scan in a host	Insufficient security configuration
Badlock Detection	Performs checks for the weak lock vulnerability	Insufficient authentication/authorisation
DROWN Detection	Performs scan on SSL and TLS services on the host	Lack of transport encryption
Host Discovery	Performs simple scan to discover live hosts and open ports	Insecure network services, insecure software/firmware
Custom scanner	Performs scan to detect common weak/default password vulnerability	Insufficient authentication/authorisation

scanners that checks for IoT specific vulnerabilities. The scan time can further be reduced converting the scanner to a multithreaded implementation. While the delay on these scans will be noticeable by the user, we argue that the delay on the order of minutes is worth the additional security benefit since it is a one time cost for a recurring benefit.

Also, figure 4 shows the average scan time using the Nessus penetration testing tool is relatively high. Based on our findings, the time it takes to completely scan a device is relatively dependent on the penetration testing tool, it is necessary to develop a mechanism that reduces the scanning time to enable a near real time network access to devices that are not vulnerable. Using custom scanners will help to reduce the scan time significantly as it will only check for IoT specific vulnerabilities.

VI. CONCLUSION AND FUTURE WORK

We have proposed a system that is the first step in providing security to consumers who lack the technical expertise to configure poorly secured and difficult to use IoT devices. Whereas previous work has focused on security techniques that require manufacturers and administrators to build secure devices and systems, our approach focuses on detecting flaws, automatically correcting them, and suggesting fixes when our system cannot automatically do so. This approach adds an overhead to the initial installation period, but only on the order of minutes, so it is reasonable for a consumer who is not frequently installing new IoT devices. Once the devices have been checked and whitelisted, there is minimal impact on performance in our system, and the impact could be reduced further if the system is implemented on a traditional networking device.

As future work, we are exploring user experience enhancements that will make it easier for inexperienced users to understand what vulnerabilities exist and how they will be fixed. We are also exploring the best set of scans to use for IoT devices. In our current system we are using an extensive vulnerability checker, but we believe that we could reduce the scan time while still discovering most vulnerabilities if we implement only custom scanners that check for IoT specific vulnerabilities. We are also working to build a custom knowledge base for IoT devices to make fixing vulnerabilities and providing information to users more comprehensive and helpful.

REFERENCES

[1] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of things (IoT) security: Current status, challenges and prospective measures," in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, Dec. 2015, pp. 336–341.

[2] "Security testing the internet of things iot." [Online]. Available: http://www.beyondsecurity.com/security-testing-iot-internet-of-things.html

[3] S. Gallagher, "Double-dip Internet-of-Things botnet attack felt across the Internet | Ars Technica." [Online]. Available: http://arstechnica.com/security/2016/10/double-dip-internet-of-things-botnet-attack-felt-across-the-internet/

[4] R. Ross, M. McEvilley, and J. C. Oren, "Systems security engineering: Considerations for a multidisciplinary approach in the engineering of trustworthy secure systems," *NIST Special Publication 800-160*, Nov. 2016.

[5] M. Stanislav and T. Beardsley, "Hacking iot: A case study on baby monitor exposures and vulnerabilities," pp. 1–17, Sep. 2015.

[6] "A clustering approach for web vulnerabilities detection." [Online]. Available: https://hal-univ-tlse3.archives-ouvertes.fr/hal-00755212/document

[7] OWASP, "Iot security guidance," https://www.owasp.org/index.php/IoT_Security_Guidance, accessed November 20, 2016.

[8] M. Xin, "A Mixed Encryption Algorithm Used in Internet of Things Security Transmission System," in *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, Sep. 2015, pp. 62–65.

[9] P. Flood and M. Schukat, "Peer to peer authentication for small embedded systems: A zero-knowledge-based approach to security for the Internet of Things," in *2014 10th International Conference on Digital Technologies (DT)*, Jul. 2014, pp. 68–72.

[10] Q. Wen, X. Dong, and R. Zhang, "Application of dynamic variable cipher security certificate in internet of things," in *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, vol. 03, Oct. 2012, pp. 1062–1066.

[11] B. Anggorojati, P. N. Mahalle, N. R. Prasad, and R. Prasad, "Capability-based access control delegation model on the federated IoT network," in *2012 15th International Symposium on Wireless Personal Multimedia Communications (WPMC)*, Sep. 2012, pp. 604–608.

[12] M. Leo, F. Battisti, M. Carli, and A. Neri, "A federated architecture approach for Internet of Things security," in *Euro Med Telco Conference (EMTC), 2014*, Nov. 2014.

[13] M. W. Condry and C. B. Nelson, "Using Smart Edge IoT Devices for Safer, Rapid Response With Industry IoT Control Operations," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 938–946, May 2016.

[14] C. A. Shue, L. M. Lamb, and N. R. Paul, "United States Patent: 9081960 - Architecture for removable media USB-ARM," Patent 9 081 960, Jul., 2015, 00000.

[15] K. Zhao and L. Ge, "A Survey on the Internet of Things Security," in *2013 9th International Conference on Computational Intelligence and Security (CIS)*, Dec. 2013, pp. 663–667.

[16] G. L. d. Santos, V. T. Guimares, G. d. C. Rodrigues, L. Z. Granville, and L. M. R. Tarouco, "A DTLS-based security architecture for the Internet of Things," in *2015 IEEE Symposium on Computers and Communication (ISCC)*, Jul. 2015, pp. 809–815.