IoT Testbed Security: Smart Socket and Smart Thermostat

Meriem Bettayeb
Department of Electrical and Computer Engineering
University of Sharjah
Sharjah, UAE
u17105766@sharjah.ac.ae

Omnia Abu Waraga
Department of Computer Science
University of Sharjah
Sharjah, UAE
u17105683@sharjah.ac.ae

Manar Abu Talib
Department of Computer Science
University of Sharjah
Sharjah, UAE
mtalib@sharjah.ac.ae

Qassim Nasir
Department of Electrical Engineering
University of Sharjah
Sharjah, UAE
nasir@sharjah.ac.ae

Omar Einea
Department of Computer Science
University of Sharjah
Sharjah, UAE
u14111378@sharjah.ac.ae

***Abstract*—Internet of Things (IoT) technology is changing the shape of our lives, however they are raising many security issues. Attackers can exploit the security vulnerabilities of IoT devices for malicious ends. Assessing the security of IoT devices is important before they are distributed to reduce the attack surface. However, the security assessment could be difficult due to the wide variety of IoT devices and their functionalities. Moreover, replicating the testing scenarios and environments can be challenging without a fully documented test scenario. The aim of this research is to design a structure for an IoT security testbed that can analyze the vulnerabilities of IoT devices. The paper provides a road map for an easy-to-setup IoT Security Testbed and a general methodology for constructing the testbed. The structure includes the hardware and software setup as well as a testcase template that standardizes test cases and ensures test replication. Moreover, a simple GUI was developed for managing the test cases, launching them and reporting their results. To showcase the feasibility of the testbed structure proposed, three test cases have been launched on a smart socket and smart thermostat followed by a discussion on assessment results.**

***Index Terms*—Internet of Things (IoT), testbed, attacks, vulnerabilities, security, smart socket, smart thermostat**

I. Introduction

The Internet of Things (IoT) is a recent technology solution for many sectors. It enables communication between embedded devices to meet our requirements. IoT technology is considered to be one of the main components for so-called smart cities. IoT is rapidly emerging because of the benefits to citizens, government and the environment [1].

As the importance of these devices increases, security issues increase as well. Many kinds of attacks have been discussed in the IoT literature. One of the most important is the man-in-the-middle (MITM) attack. The MITM attack is about gaining non-authorized access over a communication channel between two entities, enabling unauthorized session and communication modification. A survey was conducted and provided a detailed analysis and comparison of most of the MITM techniques and solutions in [2]. This attack was used in our research into IoT testbed security.

Thanks to Dubai Electronic Security Center for funding this research

The contribution of this paper is to define a testbed for building and testing selected IoT devices, using suitable penetration tools, so as to discover the security issues in IoT devices and detect their vulnerabilities. Penetration testing was conducted on the smart socket to determine its vulnerability to specific kinds of attacks. Our main goal was to construct a detailed, step-by-step road map for penetration testing. This was subsequently used to test another smart IoT device, the smart thermostat.

The testbed proposed serves as a testing software to organize penetration testing process. It enables developers to easily set up the hardware component and software tools. The test cases are formulated to list an informative description of the test case along with the descriptive explanation for the expected results and outcomes of the test. Test case also defines the testing environment such as network and hardware structure, software tools used as well as listing the devices involved with their roles. Furthermore, it describes clearly what the initial status of the system and devices involved before launching the test. All of this helps the developer in recreating the same environment and easily compare the results obtained with the expected results mentioned in the test case description.

The structure of the paper is as follows: Section 1 presents an introduction to IoT security and Section 2 shows the state of art for IoT vulnerabilities. Section 3 shows the methodology while Section 4 details the experimental setup for testing the IoT devices and the results are discussed in Section 5. Finally, the conclusion is section 6.

II. Related Work

There are many types of IoT devices in the market and multiple security analyses have been proposed to check the vulnerabilities of these devices. A case study on the security

of the August Smart Lock by Ye et al. [4] demonstrated vulnerabilities that included handshake key leakage, owner account leakage, personal information leakage, and DoS attacks

Gyory et al. [5] described the security bugs found in the SmartThings framework and proposed an IoTOne solution. The paper solved the problem of a third party that has privileged access to SmartThings devices. However, the solution is not compatible with all SmartThings devices. Ammar et al. [9] also conducted a comprehensive analysis on Samsung SmartThings and Apple HomeKit, as well as IoT frameworks such as AWS IoT Amazon and Azure IoT Microsoft. Furthermore, Prokofiev et al. [6] proposed a logistic regression method, which analyzes IoT devices and their network characteristics to provide a probability of botnet attack on IoT devices.

Sachidananda et al. [7] introduced a security testbed for IoT devices for analyzing security issues. This testbed specifies the architecture and design requirements that supports development of penetration testing for security analysis. The penetration testing included port scanning, fingerprinting, process enumeration, and vulnerability scanning.

Chistiakov et al. [8] developed a new design using an Electrically Erasable Programmable Read-Only Memory (EEPROM) chip. The improved design included authentication for the user over the HTTPS channel. It, also, included the security characteristics of Bluetooth Low Energy (BLE) for securing short range communication between the mobile and the smart lock. Moreover, Thomas et al. [10] focused on assessing the security of BLE devices. They tested the security of smart watches manually using Wireshark, Kismet and Crackle.

Ling et al. [11] reversed communication of smart socket and found insecure communication protocols, lack of device authentication, weak password policy and device scanning. This allowed them to perform the brute force, spoofing and firmware modification attack.

Ling et al. [12] analyzed communication protocols and architecture of Edimax IP camera and demonstrated their vulnerabilities. This allowed them to perform the brute force, spoofing and firmware modification attack. Moreover, Seralathan et al. [13] analyzed IP camera traffic to perform network analysis and MITM. The authors brute force port RTSP to get video streams and reverse engineer the mobile application. Whereas, XU et al. [14] used Insecam website to retrieve open cameras with live streams.

Huraj et al. [15] created a reflected UDP-based DoS attack using IoT devices. Siboni et al. [16] compromised smart watch to impersonate a legitimate Wi-Fi printer to leak secret documents. Moreover, Classen et al. [17] analyzed many security vulnerabilities and attacks on Fitbit smartwatch. They were able to inject compromised firmware and modify the Fitbit's mobile app to access developer mode and gain access to cloud.

In this paper, we propose an IoT security testbed and define its main component and structures, based on open source tools making it repeatable and adjustable. We then apply it to two off-the-shelves IoT devices by constructing test cases and scenarios. We analyze the results and report the device vulnerability. We also demonstrate a hacking attempt on the vulnerable device.

III. Methodology

Building a security testbed for IoT devices requires defining the main region of interest as well as developing a road map for the analysis process. As seen in the previous section, the literature review we conducted helped us to understand the foundation of testbed structure and testing methodology. This enabled us to define the structure and components of our IoT security testbed that fulfills the requirements and objectives of security testing.

In this section, a proposal for a testbed structure was presented based on practical experience which includes the use of open source tools. The main structural components of our proposed IoT security is described in Table I. The testbed was used to examine various types of smart devices such as the smart Socket and the smart Thermostat.

TABLE I
Components of IoT testbed

Structural components	**Description**
Testing environment	Defines the environment required to test the device under investigation.
Required devices and tools	List of all the devices and tools needed in the testbed and their role.
Testing scenarios	Defines testing scenarios with detailed tasks/tests.
Initial assumptions	Sets initial assumption about all devices involved in each test.
Audit results	Audits the results of each test.
Automatically run tests	Runs the selected test cases automatically based on test operator preference.
Report generation	Generates a report for results of selected tests.

A. Testing Environment and Devices

Defining the testing environment required to test the Device Under Investigation (DUI). This is due to the direct impact of the environment on the performance and behavior of the DUI. Defining the environment includes describing the network structure and the geographical context, since the DUI may be affected by location or other environmental factors. In addition, all the devices and tools involved in the testbed must be listed and described. The security testbed uses a variety of software tools for analyzing network activity, such as: Nmap network scanner, Wireshark network analyzer, Ettercap tool, and many other tools available on the Kali Linux operating system. The testbed includes several devices, including:

- Triggering device that sends the control commands and affects the device status.
- Controller and auditor device that controls the flow of test cases and audits network activities and performance as well as the DUI responses. The device also generates reports on the results of each test in a readable format for the user.

- Testing device that has the attacking tools and software. Also, the tools required, as well as their purpose, are described for each test case.

B. Scenarios and Tests

Another important component of IoT security is scenarios and tests. The IoT testbed should include various testing scenarios that can tackle different aspects of DUI security. Each scenario should have one or more tests that can break the scenario into manageable steps. A test is a simple task or step performed on the DUI or on the environment to explore the response of the DUI. Defining the tests includes defining all the tools required for it.

Moreover, for each group of tests, assumptions about the devices need to be clearly stated. Reports of the results, including possible consequences of each test needs to be clearly presented. Creating such a reporting structure facilitates the ability of other researchers to repeat the experiment, or use some components in later testing.

C. Testbed Controller Software

To control the flow of the testbed, a Graphical User Interface (GUI) was created to automate the testing process. The program was written in the open source python programming language. Many open source software tools, available as editors for python, provide convenience and ease in developing scripts. Kali Linux also has several libraries such as Nmap, Wireshark as well as the command library which enables execution of Linux command lines. PyQt5 library was used to create the GUI for the operator who selects and runs the tests. The program enables execution of one or multiple tests according to the testing requirements and generates an exportable report of the test results.

The program classifies IoT devices based on category, such as smart socket, smart thermostat, etc., where one or more IoT products are listed under each category. The smart socket and Smart Thermostat were tested using the GUI, as it will be explained in the next section. Therefore, the controlling scripts have been added as well to the GUI. The GUI structure and features are shown in Fig. 1. The GUI allows the users to choose among a list of IoT devices categories or types. Based on the category chose, another list of products of this specific category will be listed to allow the users to choose one brand. By choosing one of the brands, a list of related test cases will be shown to the user where they can choose one or more tests to be launched by the testbed.

The advantage of writing our own program rather than using conventional testing software was the ability to add our own features to the program as well as being able to modify and update it more frequently. One of the important advantages of the python language is that it can run on almost all operating systems, which gave us the freedom to choose the auditing device that best suits our test.

IV. Experimental Setup

In this section It is described how the testbed structure was utilized for a real implementation and applied it to two IoT

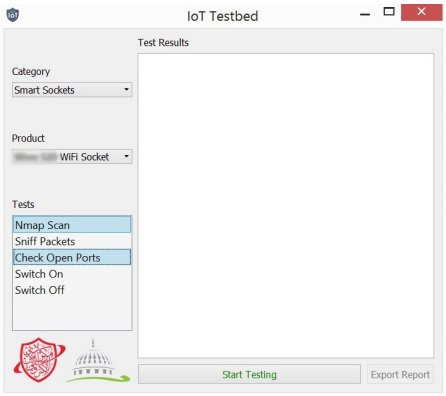

Fig. 1. Graphical user interface for IoT testbed software

devices. Our lab was configured to include all necessary tools to support the testbed during testing. The testbed components are illustrated in Fig. 2. The structure consists of the auditing machine that has the GUI which runs and controls the test cases as well as an attacking machine that is equipped with penetration testing tools such as Ettercap, Wireshark and Nmap. All the machine in addition to the IoT devices are connected to a secure WPA2 wireless network created by the router.

There are two objectives for the testbed: first, to analyze some of the network characteristics of the DUIs. This required examining the devices using Nmap to scan the open UDP and TCP ports as well as investigating the services hosted on them. Nmap reports information about the OS of a device and its particular version. In addition, Wireshark tool was used to analyze and classify the packets in the network. It also provided a flow tracing whether for TCP session or UDP packets. The second objective was to attempt to extract the control messages sent to the IoT through the network. As for the network structure, the trigger and smart device were connected to the same network so they communicate using MAC addresses rather than having to identify them using IP addresses. Wireshark might fail to examine packets between two entities if it is not be able to sniff the packets generated. Therefore, Ettercap was used to intercept the communication between the two devices and applied ARP spoofing.

ARP stands for Address Resolution Protocol. ARP spoofing or ARP poisoning is a type of attack where a malicious party sends wrong ARP messages through a Local Area Network (LAN). The attacker's MAC address can then attach to the IP address of a legal computer on the network. Hence, the attacker will receive any data intended for the authentic IP address. ARP spoofing allows illegal actors to intercept, modify or even stop data in-transit. This attack can occur on LANs that employ ARP. Ettercap tool was used for this attack. It is a free open source tool used to perform MITM and other spoofing attacks on a local area network. It runs on various Unix-like operating systems such as Linux, Mac OS X, Solaris and Windows.

Using the testbed, the two objectives of this paper were

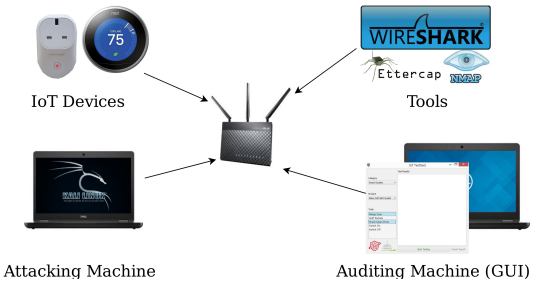

Fig. 2. IoT testbed components

fulfilled using three testing scenarios which are described in Table II.

1) **Test case 1: scanning ports**
In this test case, the network characteristics of the DUI were analyzed by scanning its ports. This test reported all the open Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) ports in the device. Network ports in IoT devices are used to carry out a specific task such as connecting to a web server or transferring files, etc. Also, they are used to provide a service such as broadcasting certain messages. If a port is open to the network without a firewall, it is vulnerable to attack.
IoT devices can create encrypted communication sessions while communicating to the server or cloud over TCP ports. TCP ports must be well secured by a firewall to prevent attacks. Many low price IoT devices have weak security capabilities and where their communication depends on UDP ports, do not use TCP ports at all. UDP ports do not process received UDP packets on that port if they don't match a certain structure. For example, if the packet received is not one of the expected commands it will be dropped automatically therefore making it difficult to inject malicious commands.
Testing was initiated by configuring a device (laptop) with our IoT testbed GUI. As mentioned above, the program uses the Nmap command which can report the test results. The laptop therefore assumed the role of testing machine (attacker) and auditor, which makes testbed implementation easier. Then, the IoT device and the testing machine were connected to a router to create a local network, which fulfilled the required test case environment.
2) **Test case 2: Network activity and packets analysis**
In this test, the testbed was configured to capture packets from the network. This configuration provided a better understanding of the network activity of the IoT device and analyzed the packets send to/received by it. Network activity of the IoT device means how often it sends packets, what the contents of the packets consist of and whether it is encrypted or not. Since Wireshark was not on the gateway, the packets sent as a broadcast of the network were able to be investigated.
This configuration does not necessarily define whether the device is vulnerable or not. Nevertheless, it provided a better understanding of the device network activity, which can help with inspections for the existence of certain attacks on those devices, i.e. when at some future point the IoT device changes its behavior.
3) **Test case 3: intercepting communication (MITM)**
As described in test case 2, devices use MAC addresses for communication within the local network. Two directions of communication were examined: one between the IoT device and router and the other between the IoT device and the controller. Therefore, there were two sub-tests for this test case:
   a) First: Examination of traffic sent between the IoT device and the router. In this test, more information about the control messages sent from the device to servers in the network were obtained (e.g. Cloud, STMP, etc.) and vice versa.
   b) Second: Examination of traffic sent between the IoT device and the controller. In this test, those packets were captured with the control data.

   For this situation, Ettercap tool was used to initiate the ARP spoofing attack between the IoT device and the router once and then between IoT device and the smartphone.
4) **Additional test: decompiling smartphone application**
The security of an IoT device is not limited to how the device is secured from attacks and has no vulnerabilities. Security also concerns the server that saves data, updates firmware, the application that gives the user control over the behavior of IoT device. Even if the IoT device is well protected against attacks, any weakness in those devices or in the communication channels between them will rescind its security.
Therefore, in this test case, IoT devices were attacked differently by reverse engineering the smartphone application dedicated to IoT. The test was not automated by the GUI, so was applied manually. It was carried out using free Android decompiling software. Decompiling the smartphone application provided us with information about the control messages used to control the IoT device, type of encryption used, etc. The amount of information that can be extracted depends on the level of encapsulation used in coding the application.

V. Implementation and Results

In this section, IoT Testbed step up and scenarios were applied, described in previous sections, on two IoT devices: smart socket and smart thermostat. The brands of the IoT devices are not exposed to keep the confidentiality of the product. First, smart socket configuration and testing procedure were explained, followed by the configuration for the thermostat.

A. Testing smart socket

A smart socket is an electric device that can be plugged into an ordinary outlet, which is connected to the wireless

TABLE II
TESTING SCENARIOS DESCRIPTION

Description	First test case	Second test case	Third test case
Test description	Scanning network ports of IoT device.	Network activity and packets analysis.	Interception of communication between the DUI and the router and the triggering device.
Testing environment	A local network is built including only involved devices and DUI. No other DUI on the network.	A local network was built including only involved devices and DUI. No other DUI on the network.	A local network was built including only involved devices and DUI. No other DUI on the network. Trigger device was connected to the network.
Devices involved	Testing and auditing device: Kali OS Laptop Trigger: None needed.	Testing and auditing device: Kali OS laptop trigger: None needed.	Testing and auditing device: Kali OS laptop Trigger: mobile application.
Tools	Nmap: To scan every UDP/TCP port and retrieve data about services provided in open ports.	Wireshark: Analyzed network packets.	Ettercap: To initiate ARP attack. Wireshark: To analyze network packets.
Initial assumptions	DUI was already ON before running the test.	DUI was already ON before running the test.	DUI was already ON before running the test.
Excepted results	Information about the OS of the DUI. List of the open UDP and TCP ports.	Capture packets sent from the DUI.	Capture of packets sent from/to the DUI.

network of the house so that the user can control devices from anywhere in the world through an application on a mobile device. Multiple steps were followed to configure the socket. Installing any smart socket would have the same scenario.

1) Operating the socket: The socket can be controlled in two ways:

- Manually, by pressing ON/OFF button. In this case, the socket toggles its state, then searches for the smartphone on the network to send a packet with updated status. If the phone doesn't exist on the network, the status information is sent to the server where it can be pushed later to the smartphone.
- Remotely, using the mobile application. In this case, the smartphone searches for the socket in the local network, if it does not exist, it sends the new status to the server. In each scenario the socket receives the status packet and updates its status, then sends a packet with the new status back to the smartphone. The application will not show the new status until it receives a packet with the new status.

2) Applying the test cases on the socket: Test cases were launched one by one and the results reported in Table III. The smart socket with UDP used port 10000 for communication after executing "nmap -sS -sU -PN IoT's IP address" command. At this stage, it was secure but prone to DoS attack if there is no firewall blocking DoS attacks. All the communication to/from the socket was in clear text. Thus, packets could be fabricated, re-sent or easily modified. It was concluded that this device was vulnerable.

3) Creating an attack: After examining the test case results for the smart socket, the control commands sent between the application and the IoT device were extracted and analyzed. The socket and the mobile application used Port 10000 for communication. The socket sent a discovery command every few seconds looking for other sockets or the controller smartphone in the local network. Also, using the application to toggle the status of the device, allowed us to capture a copy of the control packets and extract the control message format. Since all the data was sent in clear text by fabricating a packet with message control, the smart socket was able to be controlled. There were three main commands sent: Discover = 0x7161, Subscribe= 0x636c and Control =0x6463. At the end, It was possible to control the socket remotely and not through it's official app, but through an unauthorized third-party machine.

TABLE III
TEST CASES RESULTS FOR SMART SOCKET

Test case	Observations	Vulnerabilities
Test 1: Port Scanning	Only UDP ports are open and the rest is closed.	Secure but It is prone to DoS attack.
Test 2: Network Activity and Packets Analysis	Sends status as broadcast with clear text.	Vulnerable
Test 3: MITM – Part 1: IoT and Router	Sends status in clear text. Failed to communicate with the server.	Vulnerable
Test 3: MITM – Part 2: IoT and Application	Sends commands in clear text.	Vulnerable

B. Testing the smart thermostat

A smart thermostat is an electric device that allows control of water heater temperature. It connects to the wireless network of the house so that the user can control the device from anywhere in the world through an application on a mobile device. The following steps were carried out to configure the smart thermostat:

1) Operating the smart thermostat: The thermostat can be controlled in two ways:

- Manually from the device which can change the status in the cloud.
- Remotely, using the mobile application.

2) Applying test cases on smart thermostat: The results of launching the test cases are reported in Table IV.The setup for the IoT security testbed should be the same for each case, but unexpectedly, the thermostat did not connect to the AP dedicated for testing, nor to another virtual access point set up on the personal laptop. Fortunately, our smartphone where the application was running was able to connect to the router.

Since it was possible only to connect the smartphone to the router, the testing environment features changed and so the test cases for Nmap and MITM steps. It was planned to monitor the traffic received and sent from the smartphone and thermostat Application specifically. MITM attack was applied between the smartphone and the router with the following results:

- Data in the packets were encrypted
- IoT device was secure against ARP spoofing

The drawback of ARP spoofing is that it is easy to detect. The thermostat server detected duplication of the addresses. As a result, the app shut down immediately and disconnected the phone automatically from the router. After several minutes, it was possible to connect to the router again but the application refused to initiate any connection through the router. In decompiling the thermostat Android application, It was found that all the scripts had non-descriptive variable names and the code structure was intentionally made to keep all implementation code hidden.

TABLE IV
TEST CASES RESULTS FOR SMART THERMOSTAT

Test case	**Observations**	**Vulnerabilities**
Test 1: Port Scanning	-	-
Test 2: Network Activity and Packets Analysis	All data are encrypted.	Not vulnerable
Test 3: MITM – Part 1: Application and router	Blocked network because it detected the attack.	Not vulnerable
Test 3: MITM – Part 2: Application and server	Blocked the network because it detected the attack.	Not vulnerable
Test 4 (Manual): Decompiling application	Control commands were extracted easily.	Not vulnerable

VI. Conclusion and Future Research

This research demonstrated that IoT devices are vulnerable to attacks due to cheap or defected designs. Our objective was to demonstrate accurate techniques for investigating the weaknesses of IoT devices. Specifically, the goal of this research was to detect and test security gaps in two IoT devices, a smart socket and smart thermostat. Various security testing tools were used for vulnerability scanning, password cracking, capturing wireless and network packets, and finding open ports in the network. The results provided knowledge regarding the feasibility and practical experiments for assessing common dangerous threats against these IoT devices. Our results were limited to three IoT devices: a router, smart socket and smart thermostat.

Future studies will include additional automated test cases and scenarios that can tackle different aspects of security of IoT devices. More IoT devices need to be analyzed to enlarge the base of our IoT Testbed structure and more test cases need to be included to test these IoT devices. Also, future goals include employing artificial intelligence to improve the methods for analyzing IoT devices and its vulnerabilities.

Acknowledgment

This work is an initiative of the testbed laboratory at the University of Sharjah and was sponsored by the Dubai Electronic Security Center (DESC). Our gratitude for OpenUAE Research and Development Group for their usual support.

References

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," IEEE Internet Things J., vol. 1, no. 1, pp. 22–32, 2014.

[2] B. B. Gupta, "Computer and cyber security: principles, algorithm, applications, and perspectives". CRC Press, 2018.

[3] J. A. Jerkins, "Motivating a market or regulatory solution to IoT insecurity with the Mirai botnet code," in 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, 2017, pp. 1–5.

[4] Ye. Mengmei, Jiang. Nan, Yang. Hao and Yan. Qiben, "Security analysis of Internet-of-Things: A case study of august smart lock," in 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2017, pp. 499-504.

[5] Gyory. Nathaniel and Chuah. M, "IoTOne: Integrated platform for heterogeneous IoT devices," in 2017 International Conference on Computing, Networking and Communications (ICNC), IEEE, 2017, pp. 783-787.

[6] Prokofiev. Anton O, Smirnova. Yulia S, Surov. Vasiliy A, "A method to detect Internet of Things botnets," in 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2018, pp. 105-108.

[7] Sachidananda. Vinay, Siboni. Shachar, Shabtai. Asaf, Toh. Jinghui, Bhairav. Suhas and Elovici. Yuval, "Let the cat out of the bag: A holistic approach towards security analysis of the internet of things," in Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security, 2017, pp. 3-10.

[8] Chistiakov. Sergei and others, "Secure storage and transfer of data in a smart lock system," 2017.

[9] Ammar. Mahmoud, Russello. Giovanni and Crispo. Bruno, "Internet of Things: A survey on the security of IoT frameworks," Journal of Information Security and Applications, Elsevier, vol. 38, pp. 8–27, 2018.

[10] Willingham. Thomas, Henderson. Cody, Kiel. Blair, Haque. Md Shariful and Atkison. Travis, "Testing vulnerabilities in bluetooth low energy," in Proceedings of the ACMSE 2018 Conference, ACM, 2018, p. 6.

[11] Z. Ling, J. Luo, Y. Xu, C. Gao, K. Wu, and X. Fu, "Security Vulnerabilities of Internet of Things: A Case Study of the Smart Plug System," IEEE Internet Things J., 2017.

[12] Z. Ling, K. Liu, Y. Xu, Y. Jin, and X. Fu, "An End-to-End View of IoT Security and Privacy," in GLOBECOM 2017-2017 IEEE Global Communications Conference, 2017, pp. 1–7.

[13] Y. Seralathan et al., "IoT security vulnerability: A case study of a Web camera," in Advanced Communication Technology (ICACT), 2018 20th International Conference on, 2018, pp. 172–177.

[14] H. Xu, F. Xu, and B. Chen, "Internet Protocol Cameras with No Password Protection: An Empirical Investigation," in International Conference on Passive and Active Network Measurement, 2018, pp. 47–59.

[15] L. Huraj, M. Simon, and T. Horák, "IoT Measuring of UDP-Based Distributed Reflective DoS Attack," in 2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY), 2018, pp. 209–214.

[16] S. Siboni, A. Shabtai, and Y. Elovici, "Leaking data from enterprise networks using a compromised smartwatch device," in Proceedings of the 33rd Annual ACM Symposium on Applied Computing, 2018, pp. 741–750.

[17] J. Classen, D. Wegemer, P. Patras, T. Spink, and M. Hollick, "Anatomy of a Vulnerable Fitness Tracking System: Dissecting the Fitbit Cloud, App, and Firmware," Proc. ACM Interactive, Mobile, Wearable Ubiquitous Technol., vol. 2, no. 1, p. 5, 2018.