Let the Cat Out of the Bag: A Holistic Approach Towards Security Analysis of the Internet of Things

Vinay Sachidananda
iTrust, Singapore University of Technology and Design
Singapore 487372
sachidananda@sutd.edu.sg

Shachar Siboni
CSRC, Ben-Gurion University of the Negev
Beer-Sheva, 84105, Israel
sibonish@post.bgu.ac.il

Asaf Shabtai
CSRC, Ben-Gurion University of the Negev
Beer-Sheva, 84105, Israel
shabtaia@bgu.ac.il

Jinghui Toh
iTrust, Singapore University of Technology and Design
Singapore 487372
jinghui_toh@sutd.edu.sg

Suhas Bhairav
iTrust, Singapore University of Technology and Design
Singapore 487372
suhas_setikere@sutd.edu.sg

Yuval Elovici
iTrust, Singapore University of Technology and Design
Singapore 487372
yuval_elovici@sutd.edu.sg

ABSTRACT
The exponential increase of Internet of Things (IoT) devices have resulted in a range of new and unanticipated vulnerabilities associated with their use. IoT devices from smart homes to smart enterprises can easily be compromised. One of the major problems associated with the IoT is maintaining security; the vulnerable nature of IoT devices poses a challenge to many aspects of security, including security testing and analysis. It is trivial to perform the security analysis for IoT devices to understand the loop holes and very nature of the devices itself. Given these issues, there has been less emphasis on security testing and analysis of the IoT. In this paper, we show our preliminary efforts in the area of security analysis for IoT devices and introduce a security IoT testbed for performing security analysis. We also discuss the necessary design, requirements and the architecture to support our security analysis conducted via the proposed testbed.

CCS Concepts
• Security and privacy → Systems Security → Vulnerability management • Formal methods and theory of security → Security requirements; Formal security models.

Keywords
Internet of Things (IoT), Security, Privacy, Testbed Framework.

1. INTRODUCTION
The Internet of Things (IoT) are the combination of physical objects with sensors, actuators, and controllers with connectivity to the public world via the Internet. The exponential increase in the use of the IoT and the information that can be accessed via the IoT devices are susceptible to the hackers. In this regard, the security issues associated with the IoT and protecting the IoT devices will be of key importance.

IoTPTS'17, April 2, 2017, Abu Dhabi, United Arab Emirates.

DOI: http://dx.doi.org/10.1145/3055245.3055251

Currently, there are several types of IoT devices available in the market each with different capabilities. Today IoT devices are chosen based on their specs and price alone. Security has not played a major role, despite the fact that it poses a major challenge to IoT devices which by their very nature are connected to the Internet. Apparently, security is a major challenge of IoT. Since IoT devices will have: a) an internet connection, implying that a hacker can get access to the device and b) a connection to the physical devices. SHODAN [1], the IoT search engine, reveals the dark side of the connected IoT devices. Devices, ranging from cameras to industrial controllers that are connected to the Internet have proven to be easily manipulated [4] [5]. In addition, several vulnerabilities have been discovered using SHODAN [2] [3]. This indicates that IoT devices are prone to attacks, and there is a critical need to consider security measures for IoT devices.

Additional vulnerabilities have been discovered in various IoT devices, further demonstrating that security vulnerabilities are a common problem for most IoT devices [7] [11] and the associated protocols, such as ZigBee [34], Wi-Fi [35], etc. Though there has been some prior research regarding security analysis for the IoT devices, this has not provided a comprehensive approach to test and analyze all types of IoT devices, regardless of their capabilities and protocols.

Considering the above, in this paper we perform security analysis aiming towards a holistic approach. Specifically, we evaluate the security loop holes of the IoT devices by choosing the pen testing approach as our preliminary effort. We have chosen state of the art IoT devices available in the consumer market such as Amazon Echo [39], Nest Cam [40], Philips Hue [41], SENSE Mother [42], Samsung SmartThings [43], Withings HOME [44], WeMo Smart Crock-Pot [45], and Netatmo Security Camera [46] to conduct our security analysis.

To perform the security analysis in a holistic way we are also proposing an security IoT testbed in which various IoT devices, such as smart home devices, smart wearables, etc., as well as Wireless Sensor Networks (WSNs), are tested against a set of security requirements. The testbed consists of hardware and software components for experiments of wide-scale testing deployments. Variety of tests can be conducted by the proposed testbed such as standard, context-based, data, and side-channel. The IoT testbed offers different types of testing environments which simulate various sensor activity (GPS, movement, Wi-Fi, etc.) and performs predefined and customized security tests. In addition, any relevant simulator and/or measurement and analysis tool can be deployed in the testbed environment in order to perform comprehensive testing. The testbed also collects data

while performing the security analysis to conduct a security forensic analysis. Finally, a report is produced via the testbed setup, which lists the type of IoT device tested, its connectivity and the communication protocols supported, and the security test cases executed and their status (PASS or FAIL).

As we aim to perform the security analysis in a holistic way using the testbed, we also propose a set of requirements that are needed to perform such security analysis on IoT devices. We divide these requirements into functional and non-functional requirements. Furthermore, we present a novel architecture for performing security analysis via the testbed. The architecture presented in this paper is a layer-based platform model with a modular structure. Based on our approach, any type of IoT device such as smart appliances, smart city devices, smart wearable devices, etc. can be tested.

To perform the security analysis using the testbed, we address the following questions: In what ways are the state of the art IoT devices vulnerable and what are the loopholes that cause IoT devices to become vulnerable? How can we build a holistic solution to perform a complex security analysis for IoT devices? It is necessary to understand and answer these questions before IoT devices become a part of every household. Though this paper represents our initial efforts to build a thorough comprehensive solution for complex security analysis, the main contributions of this paper are threefold:

- We propose an IoT security testbed to perform the security analysis.
- We propose a set of requirements and a novel architecture which is modular and adaptable for the security analysis of diverse types of IoT devices.
- We conduct security analysis for state of the art IoT devices.

The structure of the paper is as follows: after introducing our concepts in Section 1, related work is discussed in Section 2. In Section 3, we introduce our system design. We discuss the necessary requirements for performing security analysis in Section 4, and introduce the system architecture in Section 5. In Section 6, we discuss detailed security analysis for chosen IoT devices. In Section 7 we discuss our findings, and conclude in Section 8.

2. RELATED WORK

Although the IoT is an exponentially growing research field, there has been less emphasis on the security issues surrounding the IoT, with just a few state of the art works dedicated to security analysis for IoT devices [7] [11] [12] [17] [18] [19] [20].

In [7] the authors analyze and identify a backdoor for both consumer and industrial IoT devices, however the security analysis focuses only on home automation system and smart meter. In [11], the authors perform an empirical security evaluation of the popular SmartThings framework for programmable smart homes. They also conduct a market-scale over privilege analysis of existing apps to determine how well the SmartThings capability model protects physical devices and associated data. While both [7] and [11] perform security analysis on IoT devices, they each focus on specific IoT devices, and a logical holistic approach aimed at providing a comprehensive solution for security analysis is missing in the above mentioned work.

The authors in [12] present a security analysis of Arduino Yun to show that Arduino Yun is vulnerable to a number of attacks, while in [17], researchers examine the security of smart locks for the home and examine two classes of attacks to show that existing smart locks are vulnerable to attack. However, in both of the papers mentioned above, the research is again targeted at just a specific IoT device and without consideration of the security issues facing IoT devices as a whole.

The authors in [18] investigate the Constrained Application Protocol (CoAP), an application layer protocol for constrained devices in the IoT. Their analysis highlights the main security drawbacks and supports the need for a new integrated security solution. The security of the CoAP is also explored in [36] via the Datagram Transport Layer Security (DTLS) communications protocol; this research discusses the many issues incurred and proposes solutions. In [19], the authors analyze a specific authentication and access control protocol and find that the protocol is vulnerable to compromised device attacks and replay attacks. They also provide enhancements for different aspects corresponding to the security gaps found in the protocol. In [20], the authors provide a threat model based security analysis which can be used to determine where efforts should be invested in order to secure IoT systems. However, in all the work mentioned above, the key point has been to focus on the security analysis of the protocols for the IoT, as opposed to achieve a holistic solution for advanced security analysis.

As we are proposing security IoT testbed, there has been few testbeds for IoT devices proposed in current state of the art research [6], however most of the recent work on IoT testbeds focuses on a single technology domain (e.g., WSNs) [8] [9] [10] [15]. Other research takes a wider approach to the study of IoT testbeds and focuses on multiple technology domains [13] [14].

MoteLab [8] was one of the first testbed systems for WSNs. Still in use today, it has also served as the basis for various other testbeds such as Indriya [15]. The Kansei testbed [9] is one of the most surveyed testbeds, providing various advanced functions, including co-simulation support, mobility support using mobile robots, event injection capabilities, and more. CitySense [10] is a public mesh testbed deployed on light poles and buildings. FIT IoT-LAB [13] provides a very large scale infrastructure facility suitable for testing small wireless sensor devices and heterogeneous communicating objects. However, all of the above mentioned IoT testbeds focus solely on WSNs. The T-City Friedrichshafen [14] testbed, operated by Deutsche Telekom, combines innovative information and communication technologies, together with a smart energy grid, to test innovative healthcare, energy, and mobility services. Although it considers various IoT devices, making it multi-domain, the T-City Friedrichshafen testbed fails to take into account security aspects. INFINITE [16], the Industrial Internet Consortium approved testbed, encompasses all of the major technologies, domains, and platforms for industrial IoT environments, covering the cloud, networks, mobile, sensors, and analytics perspectives of IoT. However, INFINITE overlooks the aspect of security.

Based on our review of the existing related research in the IoT realm and the observed gaps in the area of security analysis in this domain, a more logical holistic approach to security analysis is required. Such an approach, based on a dedicated testbed and novel architecture, will lead to a comprehensive and effective solution for the advanced security analysis of any IoT device and protocol.

3. SYSTEM DESIGN

In this section we introduce our system design to perform security analysis using the testbed. The system design involves the security IoT testbed setup which consists primarily of three machines which are used to run and support the security analysis. The three machines interact with each other and are used to ensure the testbed's functionality. The IoT devices, measurement tools,

access point and the shielded room, are also part of the comprehensive testbed setup.

The shielded room is as shown in Figure 1. The communication capabilities of the testbed are via Wi-Fi, Bluetooth and ZigBee. We have established an access point within the shielded room, to ensure that all the IoT devices can connect to the Internet without interference from any signals outside the shielded room. The server has been configured to store test results, reports, conducts, and maintain project details.

Figure 1. Shielded Room Setup in the iTrust Lab at SUTD.

The three machines are as follows: (1) The Orchestrating Machine (OM) is located outside the shielded room. The OM runs National Instruments' (NI) TestStand [31] which acts as an orchestrator to run and generate the report following a test. (2) The Control and Communication Machine (CCM) which is located within the shielded room, controls and connects the measurement tools and any IoT devices. The CCM runs NI's LabVIEW [32], and the IoT devices are connected to the CCM for purposes such as turning the IoT device ON/OFF, power control, measuring power consumption, etc. (3) The Analysis Machine (AM) is also located inside the shielded room. The purpose of the AM is to run the testing tools [22] – [27] needed to support various test cases. All three machines are interconnected and can speak to each other.

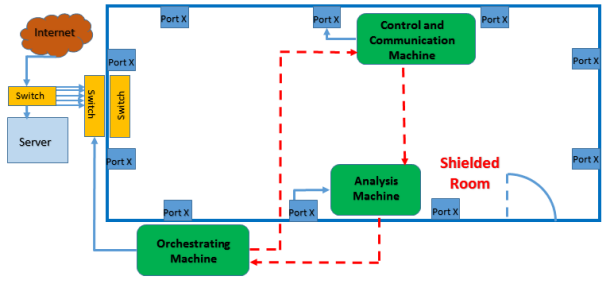

Figure 2. Physical Design Setup of the Testbed with the OM, CCM, and AM.

4. SYSTEM REQUIREMENTS

The requirements of the security analysis for the IoT can be classified and formulated on various abstraction levels. The highest abstraction level reflects the security objectives.

This section provides an overview of the functional and non-functional requirements. The functional requirements include the behavioral requirements for a system to be operational, while the non-functional requirements describe the key performance indicators. However, these requirements are general for performing holistic security analysis in a testing environment, in our case the testbed.

4.1 Functional Requirements

The functional requirements are the conditions or capabilities needed in the system, such as testbed. For example, the tests supported, test definition, analysis of the test results, etc. Moreover, these requirements describe the series of steps that are needed in order to perform a holistic security analysis of any IoT device. Table 1 presents a concise list of functional requirements for security analysis.

Functional Requirements	Description
Action initialization	Ability to simulate real world conditions and initialize the testing process.
Detection and Identification of IoT devices	Ability to detect and identify all the IoT devices.
Adding/removing a test case	Ability to add/remove a test case (test cases relate to different types of security analysis).
Automatically running a test case	Ability to run the test case automatically with minimal or no intervention for all connected devices.
Logging the status of each test case	Ability to log the status of each test case in real-time.
Report generation	Ability to generate a report for all test cases executed.

Table 1. Functional Requirements for Security Analysis

4.1.1 Initialization, Detection and Identification

One of the primary functional requirements is to establish a realistic environment for the various tests performed. By using the simulators, stimulators, and any other tools needed, the testbed should simulate real world conditions in order to test the IoT devices in different contexts and in the settings in which they operate in the real world. The next requirement is the detection and identification of the IoT devices present in the testbed environment. During the detection and identification process, a log file should be created consisting of the IoT device OS, the processes running, actions being performed, etc. This information will be used for any subsequent anomaly detection.

4.1.2 Security Tests

We take a broad and flexible holistic approach to security analysis, looking beyond individual devices at the big picture and the complexity of the IoT landscape. The testbed must support a range of security tests, each targeting a different security aspect. The testbed should detect various vulnerabilities that IoT devices can be prone to and provide analysis and reports regarding these vulnerabilities. For example, a security testbed should be able to deal with some of the vulnerabilities presented by the OWASP IoT project [21]. The testbed should be capable of running automated tests based on specific requirements (e.g., extract all tests that are relevant to the accelerometer sensor) or the device type (e.g., perform all tests that are relevant to IP cameras). Furthermore, the testbed should provide success criteria for each test (e.g., binary pass/fail or a scale from 1 [pass] to 5 [fail]), which may be based on a predefined threshold provided by the system operator in advance.

4.1.3 Logging and Analysis

Once the security tests are concluded, the testbed should be capable of logging the tests. The system collects various data during the test execution, including network traffic information (e.g., about Wi-Fi, Bluetooth, and ZigBee operation), IoT device internal status information (e.g., CPU utilization, memory consumption, and file system activity), etc. This information

should be stored as a log file for further analysis. For some tests the system operator should be able to define a decision rule specifying whether the device passed the test or not.

4.2 Non-Functional Requirements

The non-functional requirements are the set of attributes which characterize the testbed. The non-function requirements are as follows.

4.2.1 Usability

Usability ensures the testbed's ease of use (in terms of defining tests, configuring input, interpreting output, etc.) without the need for extensive efforts on the part of the user.

4.2.2 Security-Related

- **Reliability:** is the ability to perform security analysis under the stated conditions for a specific period of time.
- **Anti-Forensic:** is the capability to detect and subsequently prevent malicious applications on the IoT device (if it has been infected) from being activated.
- **Security:** is the ability to ensure authorized access to the system.
- **Accountability (including non-repudiation):** is the capability to keep audit records to support independent review of access to resources.
- **Controlled:** is the ability of the testbed to prevent malicious IoT devices from being activated during security analysis.

4.2.3 Adaptive

To take a holistic approach for performing security analysis, the security testbed should be able to adapt to new application domain concepts and support various communication types.

- **Scalability:** is the capability of the testbed to increase total throughput under an increased load when resources (typically hardware) are added.
- **Performance:** is the speed of operation of the testbed. Performance requirements pertain to throughput requirements which define how much the testbed can accomplish within a specified amount of time.
- **Flexibility:** is the capability to modify the testbed and adapt according to any IoT device and communication type.

5. SYSTEM ARCHITECTURE

In this section, we present the system architecture for the proposed security analysis of IoT devices via a testing environment, in our case the testbed. The need for a system architecture is to make sure that the security analysis can be done in a logical and modular approach. Hence, any of the IoT devices can be tested with less modifications to the testbed setup. The architecture presented here will be adaptable and tunable to any IoT devices regardless of specification and protocols. The abstract functional architecture model, illustrated in Figure 3, is designed based on the requirements described in Section 4. We also show how the modules presented in this section will interact with the OM, CCM and AM presented in Section 3. The suggested functional model is a layer-based platform model with a modular structure as follows.

5.1 Adaptable and Tunable Modules

5.1.1 Management and Reports Module (MRM)

This module is responsible for a set of management and control actions via the OM such as starting/initializing the test, enrolling new devices, simulators, tests, measurement and analysis tools, and communication channels, and generating the final reports upon completion of the test. The operator (the user) interfaces with the system through this module using one of the user interfaces (CLI\SSH\SNMP\WEB-UI) in order to initiate the test, as well as to receive the final reports.

5.1.2 Security Testing Manager Module (STMM)

This module is responsible for the actual testing sequence executed by CCM for the security analysis (according to the requirements and specifications for an adequate security testbed).

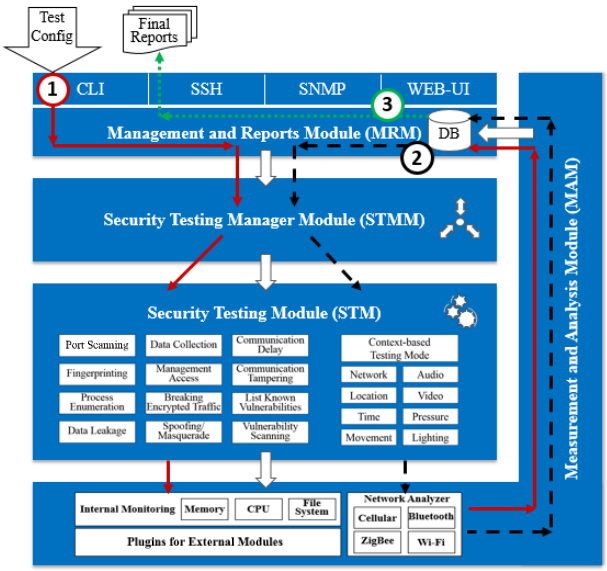

Figure 3. Security Analysis Framework - Abstract Functional Architecture Model.

The module interacts with the Security Testing Module in order to execute the required set of tests, in the right order and mode, based on predefined configurations provided by the user (based on the config file loaded in the MRM).

5.1.3 Security Testing Module (STM)

This module performs standard security testing based on vulnerability assessment and penetration test methodology, in order to assess the security level of the IoT Devices under Test (IoT DUT). The STM is an operational module which executes a set of security tests as plugins via the AM (such as port scanning, fingerprinting, list known vulnerabilities, vulnerabilities scan, and more), each of which performs a specific task in the testing/analysis process. In this regard, different security testing tools available online are utilized, including the Nmap security scanner tool for network discovery and security auditing [22], the Wireshark tool for network protocol analysis [23], Aircrack [24] which is used to assess the Wi-Fi network, Metasploit used for penetration testing [25], and all tools running under the Kali Linux penetration testing environment [26]. Other security tools, such as Nessus [27], OpenVAS [28], Cain and Abel [29], and OSSEC [30], as well as dedicated security tools, are integrated and employed in the testbed as needed.

The STM module also supports a context-based testing mode, in which it generates various environmental stimuli for each sensor/device under test. In this mode of operation, the STM simulates different environmental triggers and runs the security tests, in order to identify and detect context-based attacks that may be launched by the IoT DUT. Using the provided set of simulators and stimulators (e.g., a GPS simulator that simulates different locations and trajectories, movement simulators such as robotic hands, etc.), the testbed system realistically generates

arbitrary real-time stimulations, ideally for all sensors of the tested IoT device. Finally, the STM interacts with the Measurements and Analysis Module in order to monitor and analyze the test performed.

5.1.4 Measurements and Analysis Module (MAM)

This module employs a variety of measurement and analysis components (both software and hardware-based), including: data collection, data analysis and security rating modules, and more. MAM is present in AM, which enables the system to interface with external modules via dedicated plugins, as part of the testbed infrastructure. The measurement components include different network sniffers for communication monitoring such as Wi-Fi, Bluetooth, and ZigBee sniffers, and device monitoring tools for measuring the internal status of the IoT DUT (e.g., CPU, memory, file system, system calls, etc.). The analysis components process the collected data and evaluate the results according to a predefined success criterion (which is defined for a specific tested IoT device and/or tested scenario).

5.2 Testing Sequence

The testing process shown in Figure 3 starts with the operator loading a configuration file into the system via the MRM component. Based on the configuration loaded, both standard and context-based security testing can be performed using the STM component, by selecting a set of penetration security tests and the appropriate simulators for the test (as illustrated by the red line in Phase 1 and the black dashed line in Phase 2 of Figure 3, respectively). During the testing process, different simulators are employed in order to realistically simulate the environment in which IoT devices operate. Also, different measurement and analysis tools are employed using the MAM component, in order to collect relevant information about the test performed (including network traffic, internal status of the IoT DUT, etc.). This testing process is controlled by the STMM component. The results obtained for the tests conducted are then stored in the system database component. Finally, a forensic analysis is performed by the MRM component, and the final results of the overall testing process are then generated and sent to the operator (as illustrated by the green dashed line in Phase 3 of Figure 3). The testing process handles the security analysis of an IoT DUT as a series of steps which is explained in the following Section 6.

6. SECURITY ANALYSIS

In this section we present our preliminary efforts in the area of security analysis for the IoT. The security analysis is conducted via the testbed and by considering the requirements and architecture as explained in Section 3, 4 and 5. We have chosen state of the art IoT devices such as Amazon Echo, Nest Cam, Philips Hue, SENSE Mother, Samsung SmartThings, Withings HOME, WeMo Smart Crock-Pot and Netatmo Security Camera. We have chosen four use cases for testing, i.e., port scanning, fingerprinting, process enumeration, and vulnerability scanning.

6.1 Use Cases

In general, the OM (running NI TestStand and MRM) starts the test. The sequence of steps written in TestStand initiates the test by asking the CCM (running NI LabVIEW and STMM) to perform an intense scan to find the IoT devices present in the shielded room. Once the scan is complete, the results are sent from the CCM to the OM, and the results will contain a list of the IoT devices and their IP and MAC addresses. The user can select any IoT device from the list for further testing. Once the IoT device is chosen, the next step in the sequence is to choose the test to be performed. The OM displays the list of tests available, e.g., fingerprinting, vulnerability scan, etc., and the user can choose one or more tests to perform with the selected IoT device.

Once the IoT device and test(s) have been determined, the OM sends the information to the CCM, and the CCM sends the information to the AM with all the relevant information (including the IP address) needed to perform the test. The AM (which runs the testing tools, STM and MAM) will perform the test, and upon completion of the test, save the report on a local server and inform the CCM that the test has been completed. The OM retrieves the report from the CCM via the FTP and gives the user the option to conclude the test or see the detailed report. The detailed report is displayed on the OM. Since the report is present on the local server, the user has access to the report anytime.

6.1.1 Port Scanning

The goal of port scanning is to investigate the detectability of IoT devices by observing wireless/wired communication channels. More specifically, port scanning attempts to identify the existence of the device and detect open and vulnerable ports. The port scanning report also provides the risk level for each port discovered.

After the initial test process as explained above, the AM will run Nmap to discover the open ports via the SSH setup on selected IoT device. We ran port scanning for each of the IoT devices mentioned in this paper, however the report presented in this paper is based on the Philips Hue device.

After Nmap finishes the port scan, the results are saved as an XML file. A custom Python script on the AM will be used to extract a list of open ports discovered from the XML file. The XML file is looped line by line, checking for the keyword Discovered. Any line containing the keyword Discovered is added to a file containing a list of open ports. Finally, a custom Python script compares the open port against a list of top vulnerable open ports [33] and identifies the vulnerable ports for reporting. If the word Discovered is not found in the XML file, the whole XML file is copied as the output result, which displays everything that is scanned.

Port Scanning Results

All available ports discovered:
80/tcp open tcpwrapped

Ports that are considered vulnerable:
80: A web server is running on this port
Score: 3

Risk Level:
Safe

Metric Score:
3

Figure 4: Port Scanning Report for the Philips Hue Device.

We have established a metric score based on [33] to evaluate the risk level of open ports. The risk level is set as: 0 – safe, <15 – minor risk, 15< && <30 – major risk, and >30 – critical risk. After obtaining the scan results from Nmap, the scan results are compared with the scores of the top vulnerable ports (which contains the list of top vulnerable ports and the port numbers, a description of the ports, and a metric score given to each port), to provide the Overall Results of the test. The Overall Results contains a list of open ports, ports that are considered vulnerable,

and the metric ratings. For example, the ports that were considered vulnerable with services running include: (1) 80 - A web server was running on this port with a score of 3, (2) 5900 - A VNC server was running on this port with a score of 3, etc. To determine the Risk Level of the IoT device, a custom Python script calls on the MetricScore file, retrieves the metric number, and determine the RISK from a predefined Risk Margin. In the case of the Philips Hue device, the Risk Level is safe and the Metric Score is 3, and the detailed port scanning report for Philips Hue device is shown in Figure 4.

6.1.2 Fingerprinting

By monitoring communication traffic to/from the device the goal of finger printing is to identify the device's IP and MAC addresses, as well as the type of device, manufacturer, operating system, etc.

In order to successfully fingerprint for a specified IoT device, the AM uses Nmap, dhcpdump, and Scapy python library. We performed fingerprinting for every IoT device mentioned above, however the report presented in this paper is based on the Nest Cam device.

We begin the fingerprinting process by creating a subprocess in the shell using the subprocess.Popen() function in Python. The output is dhcpResults.txt which contains the DHCP dump of any IoT device that has made a DHCP discovery or DHCP request. This process continuously runs in the background while the script is still being executed. The nmap_done_checker() function checks whether Nmap has completed the process by constantly checking the output nmapResults.txt for the key phrase "Nmap done." In addition, nmap_done_checker() also identifies the MAC and IP addresses of the IoT DUT, which will be used later during the deauthentication step.

Fingerprinting Results

Device IP Address:
192.168.2.141

Device MAC Address:
18:b4:30:53:18:42

Manufacturer:
Nest Labs

OS Information:
Description = LaCie NAS

Additional OS Information:
[ ]

Possible Device:
NestCam IP Camera

Figure 5: Fingerprinting Report for the Nest Cam Device.

While the dhcpdump process is still running, the deauth() function tasked with forcing DHCP requests, which will result in inputs for the dhcpResults.txt file. The deauth2.py uses a Scapy library which allows for the deauthentication of a device with the specified MAC address. The mac_catcher() function opens up the text file nmapResults.txt and identifies the MAC address that exists in the text file itself. The mac_finder() function searches for the DHCP dump for the MAC address in the text file in order to get the "Parameter Request List" of the IoT device itself. The Parameter Request List is helpful in obtaining the device's OS fingerprint.

The chunk_siever() function creates a list of numbers from the Parameter Request List, which will be used later for comparison against the OS fingerprint list provided by PacketFence's [38] DHCP fingerprints. The Comparator() function compares the list obtained in the previous function against the dhcp_fingerprints.txt. This comparison allows the system to identify which OS the IoT device is using. Finally, the end result of this entire process is contained in an output file called dhcp_fingerprinting_results.html. The report shown in Figure 5 is the fingerprinting report for the Nest Cam IoT device.

6.1.3 Process Enumeration

The goal of process enumeration is to monitor the device's activities and list all services running on the device, in order to understand the state of the device and identify the protocol used and port number.

To start the process enumeration process the AM runs the nmapScan Python script, which conducts an intense scan on the selected IoT device to reveal any open UDP or TCP ports. We performed process enumeration for all the IoT devices mentioned above, however the report presented in this paper is based on the following devices: Philips Hue, Withings HOME, Samsung SmartThings and Amazon Echo.

Process Enumeration Results

Service: tcpwrapped
State: open
Port Number: 80
Protocol: tcp

Service: snmp
State: open|filtered
Port Number: 161
Protocol: udp

Service: svrloc
State: open|filtered
Port Number: 427
Protocol: udp

Service: ms-sql-s
State: open|filtered
Port Number: 1433
Protocol: udp

Figure 6: Process Enumeration Reports for the Philips Hue, Withings HOME, Samsung SmartThings and Amazon Echo.

The custom Python script nmapScan creates an output called ScanResults.xml which is used by the processEnumeration() function. First, this function filters the port numbers and various types of services, states, and protocols from the ScanResults.xml. Once filtered, then it is allowed to be formatted into html format, with the different services highlighted. Finally, the results are given as an output file in ProcessEnumerationResults.html. The results only contain the known ports, ignoring the unknown ports as their vulnerabilities are also unknown. Figure 6 contains a process enumeration report for the following IoT devices: Philips Hue, Withings HOME, Samsung SmartThings, and Amazon Echo.

IoT Device	Port Scanning (Risk Level: RL and Metric Score: MS)	Fingerprinting (Detection Criteria: IP, MAC, Manufacturer, OS, Device)	Process Enumeration (Service Running: SR, Port and Protocol: P)	Vulnerability Scanning (Impact Subscore: IS and Exploitability Subscore: ES)
Amazon Echo	RL: Safe, MS: 3	IP: 192.168.2.115, OS: AWS	SR: ms-sql-s, Port: 1433, P: udp	IS: 6.4, ES: 8.6
Nest Cam	RL: Safe, MS: 4	IP: 192.168.2.141, OS: LaCie NAS	SR: freeciv, Port: 5555 , P: tcp	IS: 4.9 ,ES: 8.6
Philips Hue	RL: Safe, MS: 3	IP: 192.168.2.139, OS: Linux Kernel	SR: tcpwrapped, Port: 80, P: tcp	IS: 2.9, ES: 10.0
SENSE Mother	RL: Minor Risk, MS: 10	IP: 192.168.2.194, OS: Unknown	SR: krb524, Port: 4444, P: udp	IS: 10.0, ES: 3.9
Withings HOME	RL: Safe, MS: 5	IP: 192.168.2.156, OS: LaCie NAS	SR: snmp , Port: 161, P: udp	IS: 2.9, ES: 8.6
WeMo Smart Crock-Pot	RL: Minor Risk, MS: 9	IP: 192.168.2.182, OS: Unknown	SR: zeroconf, Port: 5353, P: udp	IS: 2.9, ES: 4.9
Netamo Security Camera	RL: Minor Risk, MS: 8	IP: 192.168.2.123, OS: Unknown	SR: nat-t-ike, Port: 4500, P: udp	IS: 4.9, ES: 8.6
Samsung SmartThings	RL: Safe, MS: 3	IP: 192.168.2.190, OS: Unknown	SR: svrloc, Port: 427, P: udp	IS: 6.4, ES: 5.5

Table 2: Overall Results of Security Analysis with Selected IoT Devices.

6.1.4 Vulnerability Scan

The goal of vulnerability scanning is to search for additional classes of vulnerabilities by understanding and measuring the Common Vulnerability Exploit (CVE) and Common Vulnerability Scoring System (CVSS) [37]. The National Vulnerability Database (NVD) [37] has been maintaining a list of vulnerabilities from 2005 onwards, and the metric system calculated [37] helps us determine impact and exploitability sub-scores, maintain a database of attacks, and evaluate selected attacks on the tested IoT device. We run the vulnerability scan on the OS of the IoT device, and therefore to start vulnerability scan, a fingerprinting output (i.e., the OS) is provided as input. We ran the vulnerability scan for each IoT device mentioned above, however the report presented in this paper is based on the WeMo Smart Crock-Pot IoT device.

Vulnerability Scanning Results

CVE Number:
CVE-2006-5793

Impact
CVSS Severity (version 2.0):
CVSS v2 Base Score:
2.6 LOW

Vector:
(AV:N/AC:H/Au:N/C:N/I:N/A:P) (legend)

Impact Subscore: 2.9

Exploitability Subscore: 4.9

CVSS Version 2 Metrics:
Access Vector: Network exploitable
Victim must voluntarily interact with attack mechanism

Access Complexity: High

Authentication: Not required to exploit

Impact Type: Allows disruption of service

Figure 7: Vulnerability Scan Report for the WeMo Smart Crock-Pot IoT Device.

The checkCVE function utilizes multiple python libraries in order to check the vulnerabilities from [37]. The queryer() function creates a string that contains appropriate html formatting and then opens the allitems2005.csv which contains all the CVE and vulnerabilities from the year 2005. The function queryer() also goes through the csv file line by line and searches the CVE number, using the get request function to extract the vulnerability details of the specific CVE number. Finally, the htmlFormatter() function, allows the output to be highlighted where needed. Figure 7 presents the report for the WeMo Smart Crock-Pot IoT device.

7. DISCUSSION

The use cases we studied in Section 6 demonstrated a number of vulnerabilities. From the port scanning examination, we found that IoT device under test had open and vulnerable ports which would be easy for attackers to access. In addition, we identified services running on these ports that were not intended to be running, pointing to another vulnerability. We were also able to successfully fingerprint the IoT devices to understand the device type, OS, etc., which helps to obtain more information about the device vulnerability. Based on this initial testing we have labeled the tested IoT devices as safe or high risk (using our metric system), however deeper analysis is required in order to gain more insight about the IoT devices tested. The vulnerability scan provided impact and exploitability sub-scores, and the attack vector indicates that the network is exploitable and the attacker can voluntarily interact with attack mechanisms. Table 2, presents an overview of the test results for each of the IoT devices tested so far. Though we have identified other criteria of fingerprinting, we have just highlighted the IP and OS of the device in Table 2. To our understanding, our preliminary testing efforts and findings for the selected IoT devices, constituting our initial groundwork in security analysis for IoT devices, have demonstrated the vulnerability level of IoT devices.

8. CONCLUSION AND FUTURE WORK

In this paper, we describe the security analysis of IoT devices performed in the testbed using penetration testing methodologies such as port scanning, fingerprinting, process enumeration, and vulnerability scan. We introduced the security IoT testbed and provided the general requirements needed to conduct security analysis within a test environment such as the testbed. A brief description of the design and architecture needed for security analysis is also provided. Our work on security analysis was conducted with state of the art IoT devices, and the reports presented in our work show that these devices are vulnerable.

Although this represents our preliminary efforts toward a holistic approach for advanced security analysis, it is already clear that state of the art IoT devices are vulnerable; further work is required to better understand the vulnerabilities of these popular devices and improve their security. In the future, we plan to expand upon this research and conduct more complex security analysis by developing new attack and defense models. We would like to expand the testing capabilities from standard as well as to

side-channel testing (e.g., the detection of context-based attacks requires the execution of security testing in various contexts). Beyond this line of testing, we have also considered conducting deeper memory analysis with specific IoT devices. Furthermore, we intend to expose our testbed to additional IoT devices by testing other devices and inviting outside users to make use of the testbed to test their IoT devices.

We plan to further develop the architecture for our proposed security analysis system in order to make the architecture more adaptable and tunable. This will enable us to test IoT devices more easily with any protocol and communication capabilities. Furthermore, as we aim for modular architecture, our future work will center on using the various modules needed depending on the tested device and the test itself. In conclusion, our future efforts will center on developing methods and models for advanced security analysis.

REFERENCES

[1] SHODAN, https://www.shodan.io/.

[2] Patton, Mark, et al. "Uninvited connections: a study of vulnerable devices on the internet of things (IoT)." In Proc. of JISIC , IEEE, 2014.

[3] Linda, Markowsky, et.al, "Scanning for vulnerable devices in the Internet of Things." In Proc. of IDAACS, IEEE, 2015.

[4] Computerworld, http://www.computerworld.com/.

[5] The Next Web, http://thenextweb.com/.

[6] Gluhak, Alexander, et al. "A survey on facilities for experimental internet of things research." IEEE Communications Magazine 49.11, 2011.

[7] Wurm, Jacob, et al. "Security analysis on consumer and industrial iot devices." In Proc. of ASP-DAC, IEEE, 2016.

[8] G. Werner-Allen et.al, "Motelab: A wireless sensor network testbed." In Proc. of IPSN, IEEE, 2005.

[9] Arora, Anish, et al. "Kansei: A high-fidelity sensing testbed." IEEE Internet Computing 10.2 (2006): 35.

[10] Bers, Josh, et al. "Citysense: The design and performance of an urban wireless sensor network testbed." In Proc. of International Conference on Technologies for Homeland Security, IEEE, 2008.

[11] Earlence, Fernandes, et.al, Security Analysis of Emerging Smart Home Applications. In Proc. of IEEE S&P, 2016.

[12] Alberca, Carlos, et.al, "Security Analysis and Exploitation of Arduino devices in the Internet of Things." In Proc. of the ACM International Conference on Computing Frontiers. ACM, 2016.

[13] FIT IoT-LAB: a very large scale open testbed, https://www.iot-lab.info/.

[14] German Telekom and City of Friedrichshafen, "Friedrichshafen Smart City," 2010, http://www.telekom.com/dtag/cms/content/dt/en/395380.

[15] M. Doddavenkatappa, et.al, "Indriya: A Low-Cost, 3D Wireless Sensor Network Testbed," In Proc. of TRIDENTCOM, 2011.

[16] INFINITE Testbed, http://www.iotinfinite.org/.

[17] Ho, Grant, et al. "Smart locks: Lessons for securing commodity internet of things devices." In Proc. of ASIACCS, ACM, 2016.

[18] Alghamdi, et.al, "Security analysis of the constrained application protocol in the Internet of Things." In Proc. of FGCT, IEEE, 2013.

[19] Ndibanje, et.al, "Security analysis and improvements of authentication and access control in the internet of things." In Proc. of Sensors 14.8, 2014.

[20] Atamli, et.al, "Threat-based security analysis for the internet of things." In Proc. of SIoT, IEEE, 2014.

[21] OWASP, IoT Top Ten Vulnerabilities https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project.

[22] Nmap, https://nmap.org/.

[23] Wireshark, https://www.wireshark.org/.

[24] Aircrack-ng, http://aircrack-ng.org/.

[25] Metasploit, Penetration Testing Tool, https://www.metasploit.com/.

[26] Kali Linux, https://www.kali.org/.

[27] Nessus, http://www.tenable.com/products/nessus-vulnerability-scanner.

[28] OpenVAS, http://openvas.org/.

[29] Cain & Abel, a password recovery tool for Microsoft Operating Systems, OXID.IT, http://www.oxid.it/cain.html.

[30] OSSEC, Open Source HIDS SECurity, http://ossec.github.io/.

[31] National Instruments TestStand, http://www.ni.com/teststand/.

[32] National Instruments LabVIEW, http://www.ni.com/labview/.

[33] Tenable, http://www.tenable.com/sc-report-templates/vulnerability-reporting-by-common-ports.

[34] Behrang, Fouladi, et.al, "Honey, I'm Home!!, Hacking ZWave Home Automation Systems," Black Hat USA 2013.

[35] Egli, et.al, "Susceptibility of wireless devices to denial of service attacks." White paper, Netmodule AG, Niederwangen, Switzerland (2006).

[36] Rahman, et.al, "Security analysis of IoT protocols: A focus in CoAP." In Proc. of ICBDSC, IEEE, 2016.

[37] https://nvd.nist.gov/.

[38] https://packetfence.org/dhcp_fingerprints.conf

[39] Amazon Echo, https://www.amazon.com/Amazon-Echo-Bluetooth-Speaker-with-WiFi-Alexa/dp/B00X4WHP5E

[40] Nest Cam, https://nest.com/camera/meet-nest-cam/.

[41] Philips Hue, http://www2.meethue.com/en-sg/.

[42] SENSE Mother, https://sen.se/store/mother/.

[43] Samsung SmartThings, https://www.smartthings.com/.

[44] Withings HOME, https://www.withings.com/us/en/products/home.

[45] WeMo Smart Crock-Pot, https://www.crock-pot.com/wemo-landing-page.html.

[46] NETATMO Security Camera, https://www.netatmo.com/product/security/welcome.