Design and Implementation of Automated IoT Security Testbed

Omnia Abu Waraga , Meriem Bettayeb , Qassim Nasir , Manar Abu Talib

PII: S0167-4048(19)30192-0
DOI: https://doi.org/10.1016/j.cose.2019.101648
Reference: COSE 101648

To appear in: *Computers & Security*

Received date: 28 April 2019
Revised date: 22 September 2019
Accepted date: 12 October 2019

Please cite this article as: Omnia Abu Waraga , Meriem Bettayeb , Qassim Nasir , Manar Abu Talib , Design and Implementation of Automated IoT Security Testbed, *Computers & Security* (2019), doi: https://doi.org/10.1016/j.cose.2019.101648

Article

Design and Implementation of Automated IoT Security Testbed

Omnia Abu Waraga[1,*], Meriem Bettayeb[2], Qassim Nasir[3] and Manar Abu Talib[4]

[1] Department of Computer Science, University of Sharjah; u17105683@sharjah.ac.ae
[2] Department of Electrical and Computer Engineering, University of Sharjah; u17105766@sharjah.ac.ae
[3] Department of Electrical and Computer Engineering, University of Sharjah; nasir@sharjah.ac.ae
[4] Department of Computer Science, University of Sharjah; mtalib@sharjah.ac.ae
* Correspondence: u17105683@sharjah.ac.ae

Abstract: The emergence of technology associated with the Internet of Things (IoT) is reshaping our lives, while simultaneously raising many issues due to their low level of security, which attackers can exploit for malicious purposes. This research paper conducts a comprehensive analysis of previous studies on IoT device security with a focus on the various tools used to test IoT devices and the vulnerabilities that were found. Additionally, the paper contains a survey of IoT-based security testbeds in the research literature. In this research study, we introduce an open source platform for identifying weaknesses in IoT networks and communications. The platform is easily modifiable and extendible to enable the addition of new security assessment tests and functionalities. It automates security evaluation, allowing for testing without human intervention. The testbed reports the security problems of the tested devices and can detect all attacks made against the devices. It is also designed to monitor communications within the testbed and with connected devices, enabling the system to abort if malicious activity is detected. To demonstrate the capabilities of the proposed IoT security testbed, it is used to examine the vulnerabilities of two IoT devices: a wireless camera and a smart bulb.

Keywords: Internet of Things; IoT Testbed; Vulnerability Assessment; Automated Testbed Architecture

1. Introduction

The Internet of Things (IoT) is a recent evolution in communication technology that is rapidly reshaping our future. This technology enables communication and interaction between small embedded devices, improving the ability of such devices to better serve our needs [1]. In the future, IoT will be a key technological solution for many sectors including health care, agriculture and manufacturing [2], [3]. For example, in the field of health care, IoT can monitor and control human health indicators and rapidly deliver reports and alarms to medical personnel. The application of these devices is saving many lives. According to [4], the total worth of all existing IoT devices is valued at around $6.2 trillion, most of which is deployed in healthcare applications.

Moreover, IoT technology is considered to be one of the main components in the up-and-coming trend of smart cities. Many studies have discussed the various uses of IoT in shaping healthier building structures, managing waste, monitoring noise, controlling smart lighting and even relieving traffic [5]. The concept of smart cities is emerging as a result of the perceived benefits to citizens, government and the environment.

However, due to the limited capabilities of IoT devices, many of them have vulnerabilities that make them prone to various attacks. A vulnerable IoT device can be a dangerous hole in any network, regardless of its security level [6]. Many attacks have involved leveraging the

vulnerabilities of IoT devices, including actions such as replay attacks, zero-day attacks, impersonation attacks and spoofing attacks. An increase in botnet attacks has also been observed. The Mirai botnet is a well-known example; it attacks devices by exploiting default credentials [7], [8]. According to Proofpoint, more than 25% of the botnet's targets were smart TVs, baby monitors and other smart home devices [9]. Hundreds of IoT devices have been corrupted and forced to launch Denial of Service (DoS) attacks on critical servers. These attacks use Domain Name Service (DNS) and Network Time Protocol (NTP) as a form of distributed DoS (DDoS) attack. One study reported that the main reason the Mirai botnet is so effective is the use of low-cost, easy-to-install IoT devices, developed with little or no concern for security [10].

Testing the security of IoT devices before introducing them to the market is an important step in product development, and this is a field in which testbeds can be extremely useful. A security testbed is a predefined testing environment in which all triggers, tests, attacks and devices are controlled [11]. Testbeds are isolated to prevent interference from surrounding noise. They perform comprehensive vulnerability assessments on devices using penetration testing tools within certain environmental conditions. Generally, testbeds consist of an array of software and hardware tools working with simulators to change environmental settings such as light, time, GPS location, etc. They assess the device's vulnerabilities under real-world conditions and analyze its behavior to detect any malicious applications. Testbeds can specify various parameters to assess different security aspects. They examine the IoT device's response to each test in order to draw conclusions about the device's weaknesses and vulnerabilities.

According to Murad et al. [12], testing IoT devices can be challenging due to the characteristics and limitations of these devices. The next section of this paper is a comprehensive literature review presenting studies that attempt to analyze IoT device vulnerabilities and discussing the tests developed for each product. Some researchers have introduced structures for IoT security testbeds, but few of these designs were implemented. To the best of our knowledge, one of the most comprehensive IoT security testbeds was implemented and developed by Siboni et al. [13]. They introduced a testbed structure and implementation plan for testing IoT devices, using a closed-source tool as a testbed orchestra. However, their testbed lacks scalability, making it difficult to add more tests. The aim of this paper is to design an automated IoT security testbed that is comprehensive, easy to use and repeatable, using only open-source tools. The testbed has a modular structure so that tests can be added without affecting the testbed's structure and behavior. This testbed will assess the security of IoT products that are fully functional and ready to be used. The main goal of this testbed is to identify the minimum security level of IoT products.

The practical implications of our product are that it can be used by IoT pen-testers and product manufacturers to assess the security of IoT devices before they are distributed. It can also be used by market regulators to set a minimum level of security for IoT devices sold on the market. The modular nature of our software also allows researchers to extend the system and add their own test cases to the IoT testbed, making it a powerful tool for research and experimentation. We are providing the IoT security testbed as a service for individuals from academia and industry, and for smart home IoT end users. The implementation results in Section 5 show the testbed's effectiveness at detecting the vulnerabilities of IoT devices.

The main contributions of this paper are to:

- Conduct a comprehensive analysis of previous studies on IoT device security stating what tools were used on which devices and what vulnerabilities were found.
- Introduce a survey of IoT-based security testbeds introduced in the research literature.
- Define a structure for building an IoT security testbed to assess the vulnerabilities of IoT devices using open source tools.
- Introduce an automated testbed that reduces user interaction. This will guarantee that all connected devices are authenticated in order to meet security requirements. It will report attacks against devices as well as against the testbed itself. In addition, it is designed to monitor communication within the testbed and with outboard connections. It aborts upon detecting malicious activity.

- Demonstrate the functionality of the fully implemented automated testbed by testing two IoT devices: a wireless camera and a smart bulb.

Our automated testbed is used on two IoT devices: a wireless camera and a smart bulb. The wireless camera is an example of an IoT device that hosts a web server to provide its services. The device is configured via a web page hosted in the web server. In contrast, the smart bulb is an example of an IoT device that publishes its updated status in the network (advertisements). Such devices can usually be configured by using a mobile application that connects directly to the IoT device through a Wireless Local Network (WLAN) or by connecting to the vendor server. Due to the different structures of the two IoT devices, the tests conducted by the proposed testbed are different as well. It is the role of the automated testbed to identify the device type and the services hosted in every port in order to launch the appropriate test attack. In our experiment, we reported that the wireless camera is vulnerable due to the fact that it sends user credentials in plain text with no encryption, and due to the fact that it does not use certificates. As for the smart bulb, it is vulnerable to replay attacks, as it accepted repeated packets from nodes in the network other than the authenticated user.

The structure of this research paper is as follows: Section 2 is a comprehensive review of the literature on IoT security and mitigation attempts including testbeds. Section 3 presents the requirements and structure of our proposed IoT security testbed. The setup for the proposed testbed is shown in Section 4. Section 5 demonstrates a full implementation of our testbed and shows its capabilities by testing two IoT devices and analyzing the results. Finally, some recommendations and future plans are suggested in Section 6.

2. Comprehensive Study on IoT Security Analysis

Markets nowadays promote various types of IoT devices and products—smart cameras, smart plugs, etc.—some of which have severe security issues. Many security researchers have conducted vulnerability assessments for IoT products, which we discuss in this section.

2.1 IoT Vulnerabilities

Several researchers have investigated security breaches in IoT devices in order to assess their security mechanisms and identify all potential vulnerabilities [14]–[23]. Section 2.1 concentrates on the weaknesses found in IoT products in the academic literature.

A case study on the security of the August Smart Lock was done by Ye et al. [24]. The study analyzed the device's vulnerabilities, which include exposure of the device's handshake key and the owner's account data and personal information, as well as susceptibility to Denial of Service (DoS) attacks. Methods to defend the devices against these attacks were conducted in the study in an effort to improve the device's security. In another study, Ly & Jin [14] analyzed the problem of user information leakage. They examined the firmware of tech wristbands including the Nike+ Fuelband, the Huawei band, the Xiaomi Mi band and the Codoon band and found insufficient security causing leakage of user information.

Another IoT device that has been the focus of security testing is the smart meter. Two research teams, Wurm et al. [15] and Tabrizi et al. [16], both published studies in which they simulated smart meter functionalities and launched controlled attacks to discover the device's weak points. Wurm et al. [16] proposed solutions to improve the device's security, while Tabrizi et al. [16] added an analysis tool to enable users to detect malicious activity.

Smart lock security has also grabbed the attention of researchers [17]–[20], many of whom have analyzed the various risks associated with these IoT devices. Some of the smart locks under scrutiny exposed sensitive user information, while others could be controlled by unauthorized devices. To solve the access control issue, Kim et al. [17] suggested that modern smart locks should have the following control levels: full, restricted, partial and minimal. Chistiakov et al. [20] developed a new security design for smart locks using an Electrically Erasable Programmable Read-Only Memory (EEPROM) chip. The improved design included user authentication over the Hypertext Transfer Protocol Secure (HTTPS) channel.

The Smart Nest Learning Thermostat is another smart home device that has been analyzed by researchers. In their study, Hernandez et al. [21] tested the device by booting a malicious image through a USB port. In another paper, Oren et al. [22] discovered attacks on smart TVs that targeted the devices' communication protocols.

With the emergence of IoT technology, another concept entering the market is smart home technology, which enables wireless control of doors, lights and other appliances. According to Denning et al. [23], these types of home devices are vulnerable to attacks due to the lack of a professional administrator. Studies by Denning et al. [25] and Ur et al. [26] have analyzed access control policies and threats associated with these types of devices. They also discussed possible attacks on smart home devices such as data destruction, illegal physical entry and attacks of privacy violation. They showed how such attacks could reduce the security level of home devices.

As the number of IoT devices deployed in homes increases, controlling these devices becomes progressively more complicated because each device uses a separate mobile application. This issue can be resolved with a smart home system, such as Samsung's SmartThings or Apple's HomeKit, which controls all devices efficiently using a single app.

The analysis of Samsung SmartThings by Fernandes [27] identified four possible attacks that could be launched against IoT device control applications. These included creating backdoors in mobile apps, snooping door-lock pin codes, disabling protection setups and generating fake alarms. In addition, Gyory and Chuah [28] found security bugs in SmartThings that gave a third party privileged access to the system. The researchers solved this issue by proposing IoT ONE, an open-source automation platform developed by openHab that supports a number of IoT devices along with Z-wave, Zigbee and Wi-Fi protocols. However, openHab is not compatible with all SmartThings devices. Ammar et al. [29] 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.

Studies by Fernandez et al. [30] and Alghamdi et al. [31] examined the security drawbacks of network protocols, which have been the target of attacks in recent years. Fernandez et al. [30] studied DoS attack patterns on VoIP networks and improved the security structure of the protocol, but their improvement requires effort to be applied. Alghamdi et al. [31] examined the security drawbacks of the Constrained Application Protocol (CoAP), which is an application layer for constrained IoT devices.

Other researchers have launched attacks on IoT devices in order to investigate potential security weaknesses [15], [32]–[34]. Cyr et al. [32] conducted network analyses and firmware analyses on smart watches, while also checking for mobile app vulnerabilities. The authors traced the user's private address from the IoT device, captured the key exchange, reverse-engineered the mobile app, monitored traffic between the app and the Fitbit server and used proxy Transport Layer Security (TLS) traffic to intercept and extract data. The authors used various tools including Ubertooth, Wireshark, crackle, APK Extractor and dex2jar. Moreover, they used the Joint Testing Action Group (JTAG) for hardware analysis. Willingham et al. [57] focused on assessing the security of BLE devices. They tested the security of smart watches manually using Wireshark, Kismet and Crackle.

Table 1 shows a summary of the research conducted to assess the vulnerabilities of IoT products though attacks. The table lists the topic of each paper and the IoT products that were analyzed. The table also lists the tools used and the attacks conducted in the research papers, as well as the results and findings of each attack.

Table 1. Conducted IoT Attacks and Results

Ref.	YEAR	SUMMARY	PRODUCTS TESTED	TOOLS	ATTACKS	RESULTS
CYR et al. [32]	2014	• Analyzed smart watches by network analysis, firmware analysis. • Assessment of mobile app vulnerabilities.	• Fitbit smart watch.	• JTAG • Ubertooth • Wireshark • Crackle • APK Extractor • Backsmali • dex2jar	• Trace private addresses. • Capture key exchange. • Reverse engineer mobile app. • Monitor traffic between app and Fitbit server and intercept TLS traffic with proxy.	• MAC address is traceable. • Key exchange is not exposed. • TLS was replaced through a proxy to extract clear text credentials.
ARIAS et al. [35]	2015	• Created a Trojan Horse that exposed devices to an external IP address to be attacked by a server. • Accessed devices physically to change firmware.	• Nest Thermostat • Nike+ Fuelband	-	• Hardware access on Nike+ FuelBand. • Physical tamper for Nest to get backdoor.	• Firmware and checksum modifiable.
BACHY et al. [36]	2015	• Multiple attacks on smart TV by intercepting channel or attacking apps running on the TV.	• Smart TV – 4 types.	• Binwalk	• Compromise devices in public network ADSL to extract firmware. • Apply XSS attacks on web browser.	• Firmware is updated in an unsecured channel, making it prone to firmware modification attack.
MOODY and HUNTER [33]	2016	• Used Kiddie Scripts (tool for non-IT practitioners) to exploit devices.	• Nest thermostat.	• Kiddie scripts • Wireshark • Ettercap • Forensic Toolkit (FTK) • Autopsy	• Physical access to gain credentials. • Packet analysis.	• Failure to gain root access. • Communication was encrypted with AES128 encryption.
WURM et al. [15]	2016	• Analyzed security of Haier home systems through different attacks.	• Haier Smart Care home automation system.	• Wireshark, UART.	• Obtain password with brute force attack. • Gain root shell by accessing UART. • Analyze network analysis and reverse engineer firmware on air.	• Telnet credentials were exposed by root shell access. • Firmware updates were sent in clear text. • Reversed firmware exposed details about device's MQTT information.
RONEN and SHAMIR [33]	2016	• Analyzed smart bulb security issues and attempted to gain control from 100 meters away.	• Limitless LED • Philips Lux	• Introduced their own receiver.	• Eavesdrop control packets. • Extract secret information using API.	• Private data were exposed during MITM attack.
SIVARAMAN et al. [37]	2016	• Injected malware in an iOS mobile app to discover BLE and wireless IoT devices with a server. • Devices exposed to external IP using	• Dlink DCS-5500G camera. • WeMO plug • Netgear Nighthawk R7000 AP [Emulated]	• iOS App • a cloud-hosted server to receive scan results from the app.	• Search nearby LANs to find devices. • Expose those devices to a public IP address.	• Use SSDP to collect device responses in LAN and analyze them to check for IoT devices. • Exposed devices enabled server to attack

		UPnP were attacked by server.				devices.
MORGNER et al. [38]	2016	• Leverage insecurity of Zigbee light link (ZLL) to attack smart bulbs.	• Philips Hue • Osram Lightify • GE Link	• Ubertooth spectrum analyzer	• DoS attack. • Reset device attack. • Network hijacking. • Command injection attacks.	• ZLL devices vulnerable to command injection, DoS and device reset attacks. • New passwords injected by attackers as master keys.
LING et al. [39]	2017	• Reversed communication of smart socket.	• Socket Edimax plug.	• Special attacking scripts written in python	• Device scanning. • Brute force. • Spoofing. • Firmware modification attack	• Insecure communication protocols. • Lack of device authentication. • Weak password policy.
LING et al. [40]	2017	• Analyzed communication protocols and architecture of Edimax IP camera and extracted vulnerabilities.	• Edimax IP camera system.	-	• Scan online devices by enumerating all possible MAC combinations. • Brute force device credentials. • Emulate victim camera to fool authentication server.	• The camera exposed its connection status (online/offline). • Vulnerable to brute force. • Spoof attack can impersonate real cameras to get authentication information.
SERALATHAN et al. [41]	2018	• Analyzed IP camera traffic.	• IP Cameras	• Nmap • Wireshark	• Perform network analysis and MITM. • Brute force port RTSP to get video streams. • Reverse engineer mobile app.	• RTSP port found to expose real-time streams that can send commands. • Commands/ credentials sent in clear text. • Failed to get video streams. • Credentials in mobile app are in clear text.
HURAJ et al. [42]	2018	• Created a reflected UDP-based DoS attack using IoT devices.	• IP camera • Philips Hue Bridge • AirLive Wireless Printer • Raspberry Pi	• Hping3 tool.	• Flood UDP DoS attack using victim's IP.	• Victim device services were not affected.
SIBONI et al. [43]	2018	• Compromised smart watch to impersonate a WiFi printer.	• WiFi Printer.	• Wireshark • Printer Command Language (PCL) • Airoplay	• De-authenticate legitimate printer and host fake AP.	• Successfully received printing orders from devices in network.
XU, XU, and CHEN [44]	2018	• Used Insecam website to retrieve open cameras with live streams.	• Different IP cameras taken from Insecam website.	• Angry IP (for scanning domains).	• Checked open-access devices.	• Many IP cameras did not have passwords set.

CLASSEN et al. [45]	2018	• Analyzed many security vulnerabilities and attacks on Fitbit smartwatch.	• Fitbit smartwatch	• APKtool • Gatttool • Many others	• Leak information. • Analyze firmware and modify protocols. • Modify Fitbit mobile app to access cloud.	• Information leakage. • Injecting compromised firmware. • Modifying app to access developer mode and gain access to cloud.
WILLINGHAM ET AL. [46]	2018	• Find exploits in the BLE protocol through testing smartwatches using Kali Linux and Ubertooth	• FitBit Charge • Logitech Keyboard • LG watch	• Wireshark • Kismet • Crackle	• Packet sniffing	• Ubertooth was not able to read personal data. • Due to unawareness packet, format packets were not understandable.

2.2 Vulnerability Mitigation Techniques and Security Testbeds

To mitigate security holes, researchers have developed defense methodologies. Some researchers, such as Prokofiev, Smirnova & Surov [47], proposed tools that can detect attacks in advance. They introduced a logistic regression method that analyzes IoT devices and their network characteristics to assess the probability of botnet attacks on IoT devices. Gegick & Williams [48] compiled attack patterns that highlight security issues in software design and found that matching these patterns to security threats in the design phase helps to prevent threats early. Miettinen et al. [49] introduced a framework to secure vulnerable devices by identifying devices connected to a network using network traffic fingerprinting and machine learning techniques. This is useful in increasing or decreasing the security restriction level on connected devices.

As discussed earlier, smart home security is essential. Demetriou et al. [50] increased security in the home environment by creating a software-defined network (SDN) that categorizes IoT devices as nodes and smartphones as monitors to check node behavior. Gelenbe et al. [51] proposed SerIoT, an IoT platform based on SDN and secure routers.

Another important focus of recent research was testbed assessments. Generally, IoT testbeds analyze various aspects of IoT, but they do not specifically address device security. According to Chernyshev et al. [52] and Adjih et al. [2], sometimes testbeds are used experimentally as a substitute for IoT simulators. For example, FIT IoT-LAB is a testbed for low-power wireless devices used in conjunction with mobile robots for large-scale environment experiments. The resulting heterogeneous testing system covers many IoT case studies and applications.

Nevertheless, to gain a more general understanding of IoT device exploits and vulnerabilities, many researchers used security testbeds. Berhanu & Abie [53] illustrated a testbed for securing IoT devices in eHealth applications. For example, many low-power devices communicate by receiving and forwarding patient indicators using low-rate communication media. The researchers developed a scenario for the assessment and validation of context-aware adaptive security solutions for eHealth.

Moreover, Sachidananda et al. [54] introduced a security testbed to analyze the security issues of IoT devices. This testbed specified architecture and design requirements to support the development of penetration testing for security analysis. The penetration testing included port scanning, fingerprinting, process enumeration and vulnerability scanning. They conducted testing based on the security holes in the IoT device market (i.e. Amazon Echo, Nest Cam, Philips Hue, SENSE Mother, Samsung SmartThings, Withings HOME, WeMo Smart Crock-Pot and Netatmo Security Camera). The testbed included various IoT devices such as smart home devices, smart wearables and Wireless Sensor Networks (WSNs), which were tested according to security requirements. In terms of testbed control and management, their testbed uses NI TestStand software to manage testbed events and processes. NI TestStand is a closed source software that runs exclusively on Windows OS which is heavily restrictive and proprietary. This prevents tests from managing wireless cards, passive capture of packets and other network or low-level functionalities, which is considered a huge drawback as it limits network penetration testing capabilities.

Hale et al. [55] proposes an open source platform called SecuWear which identifies the vulnerabilities of commercial hardware. The SecuWear testbed captures the information necessary for identifying different attacks, thereby assessing the security of wearable devices. Moreover, it provides to the security community a process for performing attacks and mitigating information. The disadvantage of SecuWear is that identified vulnerabilities on Metawear must still be investigated in commercial IoT devices to determine if they apply. Furthermore, vulnerabilities may be specific to certain open source components, causing false positives when identifying security issues as common problems.

Table 2 summarizes the findings of our research on IoT security testbeds and compares it with our IoT security testbed. This comparison is based on the testbed's approach, the hardware setup required to build the testbed, the devices tested, the attacks performed by the testbed and the software tools used. Information is also included about whether or not the testbeds are automated, the availability of a Management System (MS) that controls the testbed, and whether or not the MS is open source (OS), as well as the existence of Wi-Fi and BLE options.

Table 2. Literature review of IoT security testbeds

Ref.	Year	Testbed approach	Hardware Setup	Devices tested	Attacks and experiments covered	SOFTWARE	Auto-mated	MS	OS	Wi-Fi	BLE
Tekeoglu and Tosun [56]	2016	• Analyze captured packets from network layers 2 and 3.	• A hub that connects 2 access points • Kali Linux machine • Ubertooth with Wireshark in another machine • Smartphones to control the IoTs from a different WLAN	• HDMI sticks • Wireless cameras • Drones • Activity trackers • Smart watches	• Nmap • Cipher suit checks • Firmware updates in clear text • Weak password checks • Brute force detection checks • Extracts video streams from cameras	• Iptables • Ebtables • Wireshark • Kismet • OpenWrt • OpenVAS • Binwalk	✖	✖	-	✓	✓
Sachidananda et al. [54]	2017	• Penetration testing for security analysis.	• Uses the closed source NI TestStand tool.	• Nest Cam • Philips Hue • Amazon Echo • SENSE Mother • Samsung SmartThings • Others	• Port scanning • Fingerprinting • Process enumeration	• NMAP • Wireshark • Aircrack • Nessus • OpenVAS • Cain & Abel • OSSEC • Tenable	✓	✓	✖	✓	✓
Hale et al. [55]	2018	• Identify security risks in wearable IoT devices by using Metawear.	• SecuWear with Metawear chip • Kali Linux • Ubertooth	• Metawear only (development chip simulating BLE devices)	• Eavesdropping attack • DoS attack	• Wireshark	✖	✖	-	✓	✓
Our Proposed IoT Testbed	2019	• Develop an IoT penetration testing platform to assess risks and vulnerabilities of IoT devices	• Multiple modules: o GUI o Testing o Network monitoring o Reporting o Storage	• Smart Bulb • IP camera	• Port scanning • Vulnerability scans • Downgrading attack • Search exploits • Brute force directories, passwords and port services • Testing SSL configuration.	• Nmap • Tshark • Metasploit • WAFW00F • SQLmap • SSLStrip • Dirb • SSL Scan • Nikto • TLS proper	✓	✓	✓	✓	✓

2.3 Our Testbed Contribution

In this paper, we propose an automated IoT security testbed that can evaluate the security of IoT devices. We also define its main components and structure. As the testbed leverages open source tools, it is easily modifiable and extendible. The model will be tested on two off-the-shelf IoT devices. Later in this paper, we analyze the results and discuss vulnerability reports. Our IoT security testbed has the following features:

1. The implemented IoT security testbed is based on open source tools controlled by an open source MS.
2. The IoT security testbed consists of an interface module, a testing module, a network module, a report module and a storage module. All modules interact to perform as a complete security testing software. They are controlled by the MS and their updates are displayed to the user via an easy-to-use GUI.
3. The modular structure and architecture of the testbed allows other researchers to use it to build their own testing tools. It is a flexible and extendible system, meaning that researchers can adopt the initial structure and add to the modular design.
4. The IoT testbed lists all exploits and CVEs found for the device tested, as well as for the services the device hosts in each port. For example: OpenSSH 7.6 service on port 22 (SSH).
5. The IoT testbed automatically generates formal word reports containing the results of all devices.

3. Testbed Requirements, Structure and Components

Building a security testbed for IoT devices requires defining the main area of interest and developing a roadmap for the analysis process. In this section, we propose an automated IoT testbed structure to assess the vulnerabilities of IoT devices. This structure automates the penetration testing task, thereby reducing user intervention. Our objective is to build a secure IoT testbed that tests devices from various security aspects. The testbed should:

- Establish secure communication between testbed components
- Authenticate all nodes in the network
- Control test modules and test sequencing
- Record events and test results
- Establish reusable tests and testbed components
- Ensure scalability of the testbed, enabling more tests to be added

3.1 Testbed Structure

Our testbed uses a modular architecture, whereby every part of the testbed is made of modules that can be extended or even replaced completely. The structure also allows for the easy addition of more security tests.

The initial testbed structure consists of five modules, as shown in Figure 1.

- **Interface Module**: This module acts as an I/O interface. It consists of two units: a Graphical User Interface (GUI) unit and an Output unit. The GUI takes input feeds from users (when required) and delivers them to the Testing Module for analysis. This method reduces user intervention during the testing process. When the analysis is complete, a summarized report is generated by the Report Module and sent to the user.
- **Testing Module:** This module manages the test cases and launches them in order. All test cases and scripts are saved in the Storage Module. Once the testbed is in operation, it calls up general scripts from the database to examine the general network characteristics of the IoT devices. Based on the device's response, the testing module launches more advanced test cases to tackle security issues. For example, after recognizing any open ports in the IoT device, dedicated test cases will be launched to test the vulnerability of those ports. Such vulnerabilities can include outdated services or low-security configuration or authentication. Moreover, IoT device responses will be checked to determine whether each test case passed or failed.

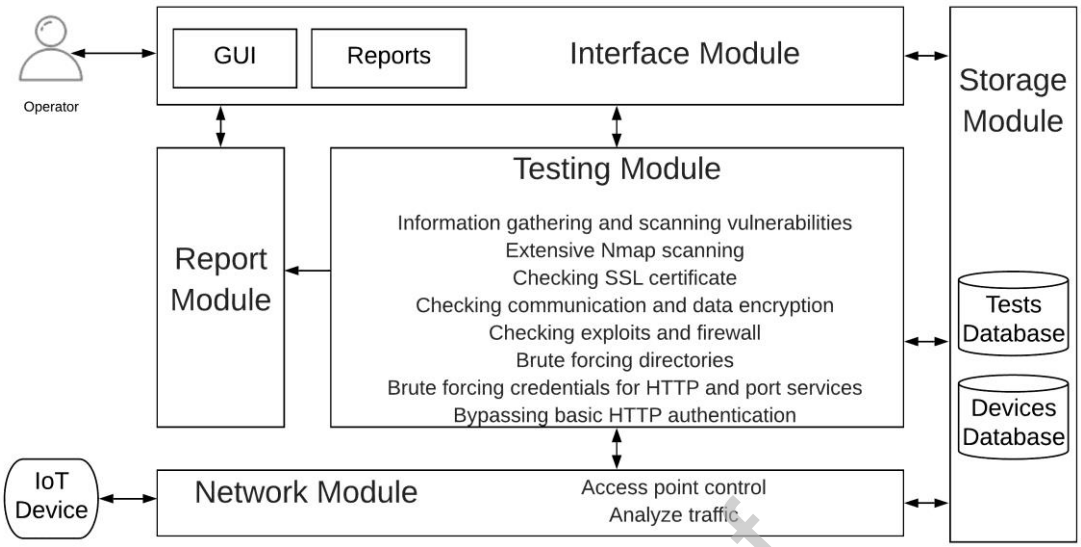


Figure 1. Structure of the proposed IoT Testbed

- **Network Module:** This module controls network activities and communication with IoT devices. It creates and monitors the Network Access Point (AP) and will be further discussed in Section 3.2.
- **Report Module:** This module generates a final report of the security assessment results for the device. It is compiled from test results and logs.
- **Storage Module:** This unit stores all events initiated by the different modules for later retrieval and for final report generation once the assessment is complete. It saves all information about the tested devices and stores the test case scripts.

3.2. Testbed Main Components

In terms of function, our proposed automated testbed relies on five components that use the testbed structure modules. Figure 2 summarizes the testbed components and their roles.

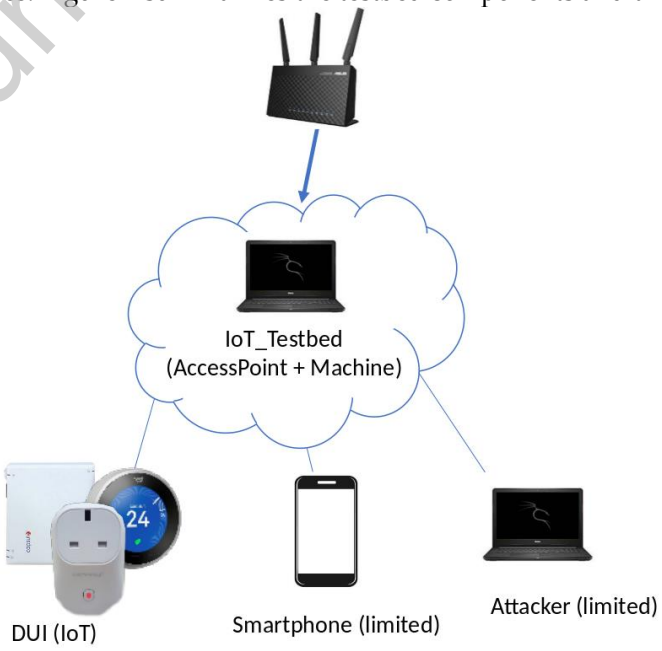


Figure 2. IoT testbed components

The testbed has the following five main functional components:

- **IoT Device Under Investigation (DUI).** This is the IoT device to be tested, such as a smart socket, smart wireless camera, etc. It will be connected to the wireless network of the testbed.
- **Admin Machine.** This is the main component of the security testbed that runs the Kali Linux operating system. Using the Network Module, it audits all network traffic and examines packets in the network. It also sends an alert if any type of attack is detected using python scripts and Tshark. Any IP address requested by the DUI will be checked against IP blacklists to determine whether or not it is malicious. Malicious calls will be blocked and reported. Moreover, the admin machine acts as an orchestra, using the Testing Module to launch and coordinate test scenarios. Once a test is launched, its results are analyzed to determine which test should be run next. The storage module stores information about all registered devices in the testbed, blocking unregistered devices from the network.
- **Private Wireless Network**. Usually, testbeds use wireless routers to simulate network environments. Sniffing software is then deployed to collect data broadcast over the local network. To acquire more information or packets between two nodes in the network, interception tools such as Address Resolution Protocol (ARP) spoofing are used to launch an attack. Based on our experiments, new IoT devices detect ARP spoofing attacks and disconnect automatically from the network once they are discovered. Using the admin machine, a Wireless hotspot is created using a virtual AP tool to create a Wireless Local Access Network (WLAN). This method is preferred over using a physical router, as it gives the testbed automation system privileged access to Dynamic Host Configuration Protocol (DHCP) server services and other functionalities in the AP. It therefore allows the testbed to audit network traffic with administrative privilege to circumvent the need for ARP Spoofing. The virtual AP tool provides communication encryption and security using the WPA2-PSK Wi-Fi key managed by the network module. In addition, it monitors the outboard connection of the DUI. In other words, the module checks all external IP addresses requested by the DUI against a collection of blacklisted IPs to prevent malware from attempting to connect to Command and Control (CNC) servers. If the device is already infected by malware, it will be detected and the device will be excluded from the network. This countermeasure fulfills the security requirements of the testbed.
- **Controlling Applications**. Some IoT devices can only be controlled through their associated mobile application. The testbed therefore includes a smartphone device that is equipped with authorized mobile applications to control IoT devices under testing. It generates traffic and packets with controlling commands on the network, and can be used to check whether or not the controlling commands and messages are sent in clear text, i.e. readable by attackers. In addition, it replicates packets to form replay attacks, thereby revealing weaknesses in IoT devices.
- **Attacking Machine.** The attacking machine (Kali Linux) is used to launch attacks against the DUI to uncover weaknesses. For instance, it can launch replay attacks or brute force password attacks. In addition, it runs DoS attacks against the testbed and the DUI to test their resistance and ability to block such attacks.

To give a better idea of how the testbed modules cooperate, Figure 3 summarizes the scenario for a DUI connected to the testing network. As shown in Figure 3, once a new device tries to connect to the network, the testbed will check its identity by looking it up in the testbed local database records. The testbed operator registers the devices to be tested in the database of the program before starting the test. If the network module does not find the device in the database, the testbed will reject the device from the network.

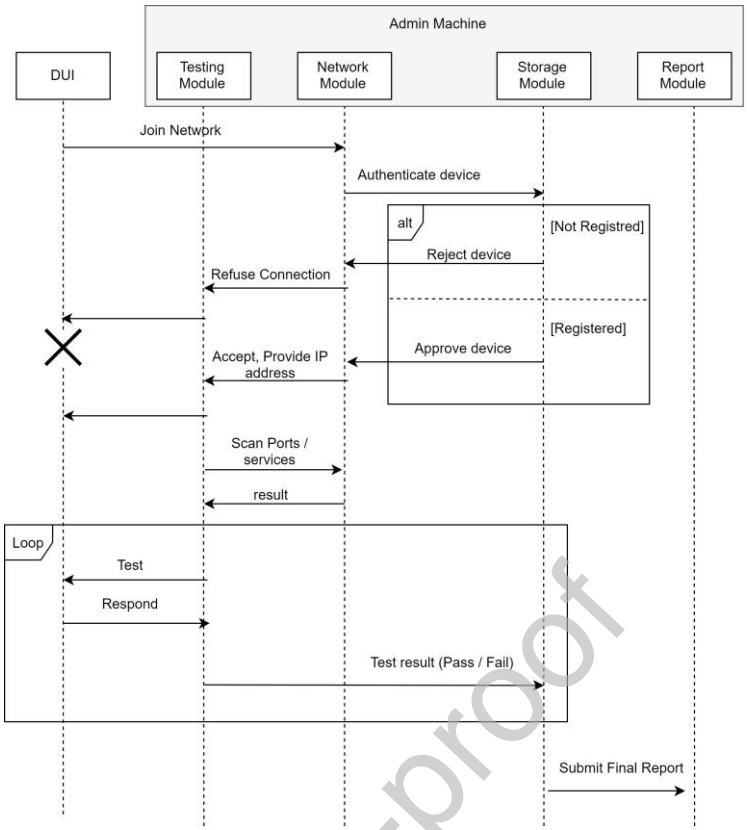
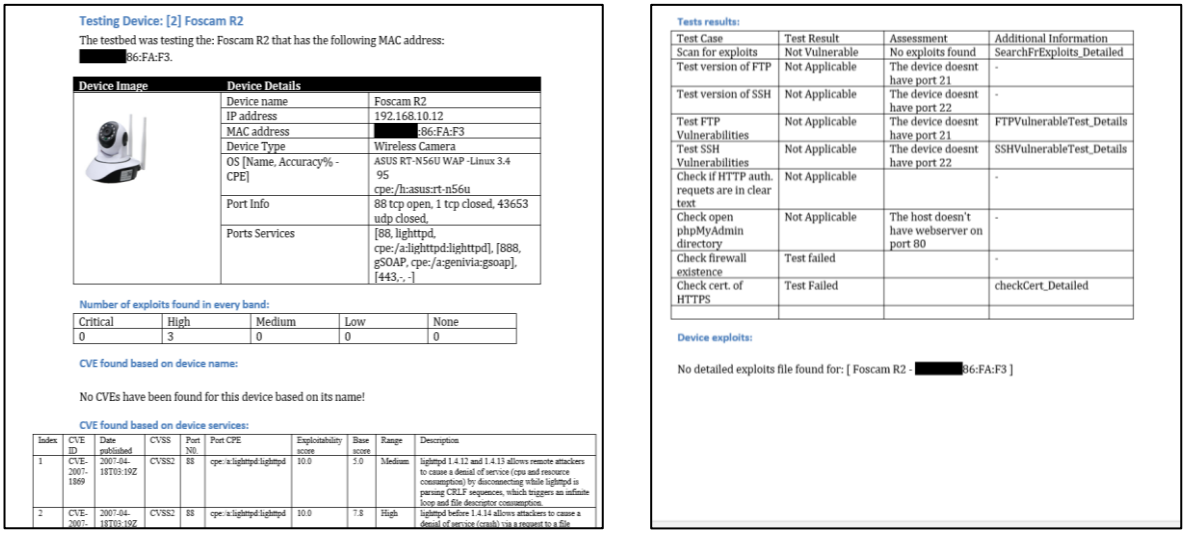


Figure 3. Communication between testbed modules

After the authorization step, the testing module launches a list of test cases on the DUI, analyzes the DUI test results and stores the results in the storage module. An example of a test case is the scan test script, in which the testbed software scans DUI directories, seeking open directories with no authentication. This vulnerability could lead to system intrusion and data leakage. After all tests have been run, test results will be listed in a formal report.

The formal report is a word file generated by a python script listing the test results in a format that is easy for a human operator to understand. A sample report is shown in Figure 4. The report lists device information in a table, including its detected Operating System and port services. It also lists all CVEs and exploits found using the device model number or device name. Additionally, as all ports are scanned during the testing process, any services found in the device's open ports will be checked on the CVE and exploits databases. The test results are then listed in the results section of the report. The **Test Case** column lists all tests launched against the IoT device, while the **Test Result** column informs the reader of the results of each test. The test result is listed as **Not Vulnerable** if the device was not found to be vulnerable to the test and **Vulnerable** if the device was vulnerable. The **Assessment** column includes short comments generated by each test. Furthermore, some of the complex tests generate extra logs and save them in a separate text file to be reviewed later by the operator. Some tests don't generate any logs; these simply enter a dash (-) in the **Additional Information** column. If necessary, tests can be modified in the future to include more comments.

In Section 4, we will focus on the testing mechanisms and tools that check the security aspects of IoT devices.



Testing Device: [2] Foscam R2

The testbed was testing the: Foscam R2 that has the following MAC address: [redacted]86:FA:F3.

Device Image	Device Details	
	Device name	Foscam R2
	IP address	192.168.10.12
	MAC address	[redacted]:86:FA:F3
	Device Type	Wireless Camera
	OS [Name, Accuracy% - CPE]	ASUS RT-N56U WAP - Linux 3.4 - 95 cpe:/h:asus:rt-n56u
	Port Info	88 tcp open, 1 tcp closed, 43653 udp closed,
	Ports Services	[88, lighttpd, cpe:/a:lighttpd:lighttpd], [888, gSOAP, cpe:/a:genivia:gsoap], [443,-,-]

Number of exploits found in every band:

Critical	High	Medium	Low	None
0	3	0	0	0

CVE found based on device name:

No CVEs have been found for this device based on its name!

CVE found based on device services:

Index	CVE ID	Date published	CVSS	Port NO.	Port CPE	Exploitability score	Base score	Range	Description
1	CVE-2007-1869	2007-04-18T03:19Z	CVSS2	88	cpe:/a:lighttpd:lighttpd	10.0	5.0	Medium	lighttpd 1.4.12 and 1.4.13 allows remote attackers to cause a denial of service (cpu and resource consumption) by disconnecting while lighttpd is parsing CRLF sequences, which triggers an infinite loop and file descriptor consumption.
2	CVE-2007-	2007-04-18T03:19Z	CVSS2	88	cpe:/a:lighttpd:lighttpd	10.0	7.8	High	lighttpd before 1.4.14 allows attackers to cause a denial of service (crash) via a request to a file

Tests results:

Test Case	Test Result	Assessment	Additional Information
Scan for exploits	Not Vulnerable	No exploits found	SearchFrExploits_Detailed
Test version of FTP	Not Applicable	The device doesnt have port 21	-
Test version of SSH	Not Applicable	The device doesnt have port 22	-
Test FTP Vulnerabilities	Not Applicable	The device doesnt have port 21	FTPVulnerableTest_Details
Test SSH Vulnerabilities	Not Applicable	The device doesnt have port 22	SSHVulnerableTest_Details
Check if HTTP auth. requets are in clear text	Not Applicable		-
Check open phpMyAdmin directory	Not Applicable	The host doesn't have webserver on port 80	-
Check firewall existence	Test failed		-
Check cert. of HTTPS	Test Failed		checkCert_Detailed

Device exploits:

No detailed exploits file found for: [Foscam R2 - [redacted]86:FA:F3]

Figure 4. Sample of report generated by the IoT testbed

4. Experimental Setup

In this section, we demonstrate how the testbed architecture and components are used to test the security of IoT devices. Our experiments are conducted in two phases: a semi-manual Extensive Analysis Phase and an Automated Testing Phase. Analyzing the threads of IoT devices is a very complicated task, as pen-testing the devices involves testing the security of communications between IoT devices and smartphones, as well as between IoT devices and the cloud. It also requires testing the vulnerabilities of the IoT device itself and testing the effect of physical tampering. The process by which communications are sent and received by IoT devices—a potential source of vulnerability—is shown in Figure 5. The devil image represents hackers and their possible points of attack.

The first phase is conducted to understand the nature of the IoT device, its communication characteristics, its possible vulnerabilities and the used tools to detect them. This information is then used to shape the second phase: an extensive automatized security analysis of the IoT device. Table 3 sets a comparison between these two phases. The comparison is based on testing setup, expected results in each phase and the method used to obtain the results.

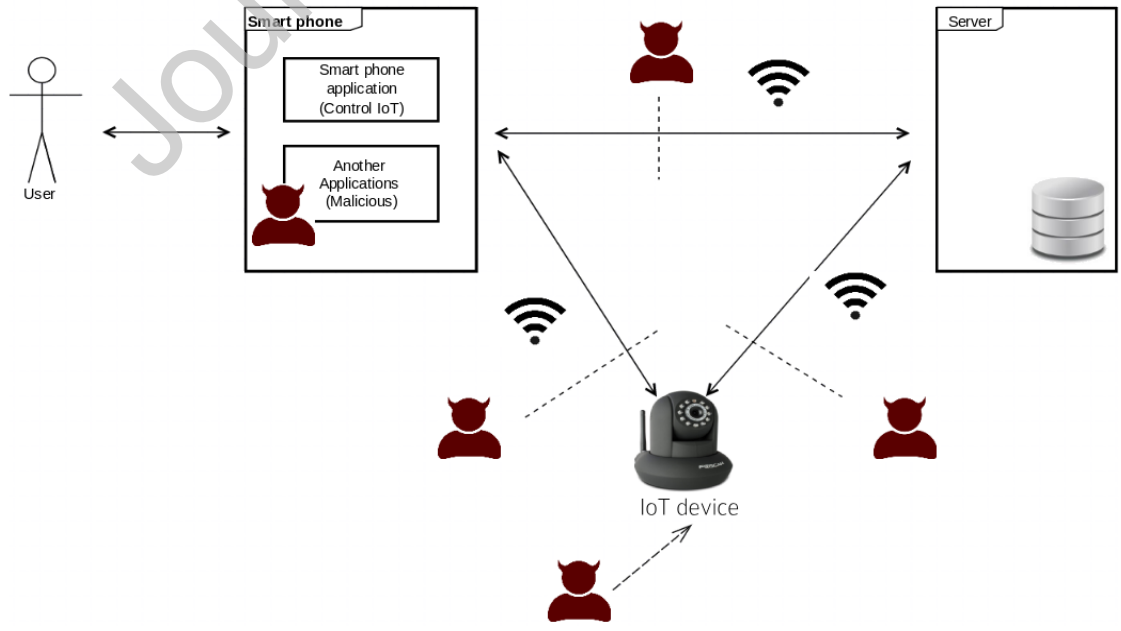


Figure 5. Points of weakness in IoT device communications

Table 3. Comparison between extensive analysis phase and automated testing phase

Research phases	Set up	Expected result	Methodology
Phase 1: Extensive Analysis	Each device is tested individually using a list of tools conducting different hacking attempts to find the IoT device's vulnerabilities.	Each test will generate different results/outcomes.	Tests are done manually. Results are obtained manually through analysis.
Phase 2: Automated Testing	The testbed system's software is based on a modular structure. Each testing module will automatically run a list of tests to check the IoT device's vulnerabilities.	The test results will be expressed as Vulnerable or Not Vulnerable. If the device passes a given test, it is not vulnerable in that area; if it fails, the device is vulnerable to attack.	The module's test cases are in Python code. The module analyzes the IoT device's responses to each test script to determine if it passed or failed the security test.

4.1 Phase 1- Extensive Analysis Phase

The first testing phase investigates the security of IoT devices using the following steps, summarized below in Table 4:

- **Gather information and scan for vulnerabilities.** Before testing the IoT device, it is necessary to search for device vulnerabilities and any related exploit attempts. This can be done using Shodan, vulnerabilities databases such as the Common Vulnerabilities and Exposures database (CVE), the National Vulnerabilities Database (NVD) and the Rapid7 Exploits Database, and tools such as Snitch, OWASP ZAP, Wascan, Skipfish or other similar tools. If any exploits are found for similar devices, whether of the same type or from the same vendor, these are tested on the DUI to assess its vulnerability.
- **Perform Nmap scanning.** The DUI and all its ports are checked using the Nmap scanner to analyze the vulnerability of any open ports. Nmap responses should be checked to ensure that the DUI doesn't expose critical information during the Nmap test.
- **Check Secure Sockets Layer (SSL) certificate.** The SSL certificate is tested to see if a DUI that hosts a web server has a reliable certificate. This can be checked using TLS-proper, SSLScan and Nikto tools.
- **Check asynchronous connection with a time server.** This test checks if the IoT device is synchronized with a time server. If this test fails, the resulting vulnerability could make it hard to track its system logs and events, and also to perform operations that require timestamps and synchronization.
- **Perform downgrading attack.** This attack focuses on reducing the level of cryptography used in the communication channels between two nodes [57]. Reducing the level of encryption used in the secure channel can result in the device sending information without any encryption at all, as is the case with HTTP. This test forces communications to downgrade from HTTPS to HTTP using SSLStrip, Ettercap, Better-cap, etc. If a conversation is successfully downgraded, critical information such as credentials and control packets may be collected by a third party. If the device refuses the downgrade and rejects any non-HTTPS connections, it is considered a secure device.
- **Perform credentials check and brute force attacks.** Another potential vulnerability that must be tested is the use of default credentials. As mentioned earlier, Mirai botnets have been known to gain control over devices with default credentials. If users are not forced to change the default password during configuration, the resulting vulnerability is a severe issue. Another potential issue is when IoT devices allow an unlimited number of false access trials. Limiting access trials prevents brute force attacks. These are all aspects that we can test using a simple python script.
- **Conduct brute force attack on directories.** If the DUI is hosting a web server, this server could have multiple directories. Even if it has credentials, it's possible that not all directories will be

protected. To check if any directories are left without credentials, the testbed uses a dictionary to conduct a brute force attack using Dirb and DirBuster tools.

- **Bypass basic HTTP authentication.** Some web servers use HTTP basic authentication to obtain user credentials. HTTP requests can use POST and GET methods. If servers are weakly configured, they may bypass HTTP authentication requests that have HTTP methods other than GET/POST. As a result, private data may be exposed or non-authenticated users may gain access.
- **Inject XSS and SQL commands.** If the DUI hosts a web server with an HTML interface, it could be vulnerable to XSS and SQL injection attacks. This can be checked using tools such as SQLmap that examine the parameters of an HTTP GET request to inject SQL commands. If the web server is not protected against this type of attack, the server's SQL service may expose critical information.
- **Check firewalls.** Some web servers have firewalls that protect them from network attacks. The firewall of the DUI web server can be tested using a WAFWoof tool.
- **Check exploits.** As shown in the literature review, researchers are interested in revealing the vulnerabilities of different types of IoT products, and many CVEs are reported every day. Some CVEs are also publicized with a python or a bash script that takes advantage of the vulnerability to perform an attack. Existing exploits on devices similar to the DUI can be checked using Metasploit and Armitage tools.
- **Analyze communication between the IoT device and the user machine.** In this task, we intercept the communication between the IoT device and the user's machine. First, traffic will be generated by using a mobile application (or the browser, if the device contains a web server) to control the IoT device. This allows us to check if communication occurs in clear text, and whether the device is vulnerable to replay, impersonation or modification attacks. In addition, this task detects if any credentials are sent in clear text.
- **Check requested external IP addresses.** In this test, the testbed will report any attempt by the DUI to connect to malicious IP addresses.
- **Disassemble mobile application.** Breaking down the mobile application can give hints about control packet creation and expose secret information. Applications can slow down reverse engineering by using obfuscation techniques, which raises the security level in smartphone applications.
- **Check firmware.** Outdated firmware is usually vulnerable. If vendors do not enforce updates on an IoT device's firmware, the device may be compromised. Analyzing firmware can highlight the existence of backdoors, hardcoded admin credentials or command injection vulnerabilities. Analyzing firmware requires experience in reverse engineering.
- **Analyze hardware.** Some IoT device vendors don't give public access to device firmware. An alternative method is extracting the firmware from the IoT device. By disassembling the device, the printed circuit board (PCB) can be checked to find universal asynchronous receiver-transmitter (UART) ports or other serial ports. If ports are found, a PC/laptop can be connected to the IoT device to analyze its binary image and extract its credentials, i.e. a physical tampering attack.

Table 4. Summary of the test cases and their expected results.

Tests	Used tools	Expected results
Gather information and scan for vulnerabilities	Snitch, OWASP ZAP, Wascan, Skipfish	Gathers information about the DUI's vulnerabilities or about previous attack attempts recorded in the CVE database. Wascam and Skipfish are security penetration testing tools that recursively crawl web pages hosted in webservers. They assess security and look for vulnerabilities such as flaws, links, email addresses and any other information that could lead to social engineering, malware injections, etc.
Nmap scanning	Nmap	Lists all open ports along with their services and DUI

		information. DUI information could include the operating system running on the device, its version number, etc.
Check Secure Sockets Layer certificate	TLS-proper, SSLScan and Nikto tools	Compares the HTTPS certificate signature to the database. This reveals information about the certificate such as the encryption used, the generation date, etc.
Check asynchronous connection with a time server	Wireshark	If the device fails to synchronize with the NTP server during multiple connection requests, it is considered vulnerable.
Downgrade Attack	SSLStrip, Ettercap, Better-cap	If the device refuses the HTTP connection/request, then it is not vulnerable. If the device accepts an HTTP request (instead of HTTPS), it is vulnerable to this kind of attack.
Credentials check and brute force attacks	Python Script	Attempts to authenticate user by sending many usernames and passwords. If the device doesn't detect the attack or if the password is found, the device is vulnerable to this kind of attack.
Brute force attack on directories	Dirb and DirBuster	Lists directories that are accessible without authentication, indicating that the device is vulnerable.
Bypass basic HTTP authentication	Web browser plugin (HTTP headers	Sends misconfigured http header to check for possible configurations that may give access.
Inject XSS and SQL commands	SQLmap, Manual	If XSS or SQL injection attempts successfully reveal hidden information, the device is vulnerable.
Check firewalls	WAFWoof	Checks whether or not firewall is used
Check exploits	Metasploit and Armitage	Reports any attacks and vulnerabilities in the IoT device found by Metasploit.
Analyze communication between IoT device and user machine	Wireshark, Manually	If control packets are sent between the user machine and the DUI in clear text without encryption, the device is vulnerable.
Check requested external IP addresses	Wireshark, manually	If the device attempts to connect to malicious servers, it is considered vulnerable.
Disassemble mobile application	Dex2jar	If the dissembled mobile application contains hard-coded credentials, it is considered vulnerable.
Firmware check	Binwalk	Outdated firmware is usually vulnerable. The firmware is therefore checked to ensure that the device is using the latest version.
Hardware analysis	UART	This test attempts to dump firmware from the hardware using UART in order to obtain root shell and access sensitive information

In this phase, the security of several IoT devices was assessed manually by using the above test cases. For an example, we will detail the assessment of two devices: a medical gateway and a wireless camera. Both devices host a webserver, but the medical gateway leverages HTTPs for communication while the wireless camera uses HTTP. The detailed test report for the two devices is shown in Appendix 1.

During the manual security assessment, the tested devices are grouped into two sets: devices that contain a web server and devices that act as hosts, connecting to a cloud or a server. The devices that host web servers can be recognized by examining the services available in port 80, 8080 or 443. The Nmap tool is capable of recognizing the services on the ports, as it has an extensive database of service signatures. The devices can therefore be categorized as webserver or host-based according to the Nmap results for the services hosted in port 80, 8080 or 443. The devices that host a web server can be attacked using the same mechanisms as those used against email servers and web servers.

However, IoT devices are much weaker than the regular web servers due to their limited power and computing capabilities, which affects their security capabilities.

An automated IoT testbed is required for this step, as it can detect the vulnerabilities of a group of different IoT devices automatically without the need of human intervention.

4.2. Phase 2- Automated Testing Phase

Automating IoT vulnerabilities assessments can be a challenging task due to the limitations of IoT devices, as explained earlier. Based on the experiments conducted in Section 4.1, we propose methods with different security testing scenarios for different IoT devices. These scenarios will run automatically using the system in Figure 1. The tests are to be conducted with minimal user intervention. As shown in Figure 1, all modules work simultaneously at the back end, while the GUI shows the testbed status and related results. The various steps of the automated testing process are as follows:

As shown in Algorithm 1, once the operator chooses one or more devices to be tested, the testbed assesses each one individually in turn. To test each IoT device, the testbed first excludes all IoT devices from the network other than the DUI. The testing module then launches some initial test cases, including an extensive Nmap test to check network activity and report open ports. Based on the results of these preliminary tests, the testing module will determine whether or not there is a web server hosted in the IoT device. Based on this data, the testing module will launch the corresponding tests as shown in Figure 6.

Algorithm 1 Automated Testbed Process

Require: DUI; Testing device;
Ensure: DUI is configured with wireless network of the testbed

```
DUI connect to the wireless network
Testbed check device information in Database
if not found then
    Reject the device connection request
    return 0
Accept connection, allow device to be in the network
Testbed launch Nmap on ports 80, 443, 8080
for Nmap results in port (80, 443, 8080) do
    Check services on the port
    if port has webservices then
        DUI = webserver
    else
        DUI = host
    end if
end for
if DUI = webserver then
    results = webserver_tests( )
else
    results = host _tests( )
end if
report= Generate_Report(results)
    return report
```

In non-web server cases, a replay attack is used to replay control packets after a period of time. The control packets contain the commands that affect the status of the IoT device.

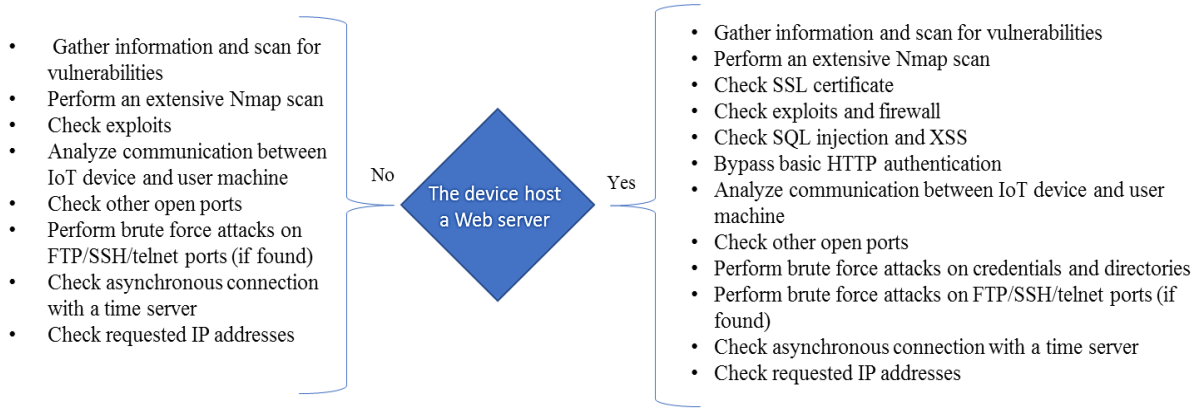


Figure 6: Tests launched for web server DUI vs non-web server DUI

Figure 7 shows the updated testbed GUI and its functionalities. Users can choose to test multiple devices, which will be listed in the left panel. The test updates will be shown in the middle panel: the upper part shows the test logs, while the lower table lists test summaries and results. In the right panel, the user can see network traffic inside the WLAN, as well as traffic to and from the DUI. In addition, the testbed lists the last connections attempted by the DUI. Using the proposed automated testbed architecture, vulnerability assessments are conducted on multiple IoT devices. In Section 5, we discuss the use of the testbed on a wireless camera and a smart bulb.

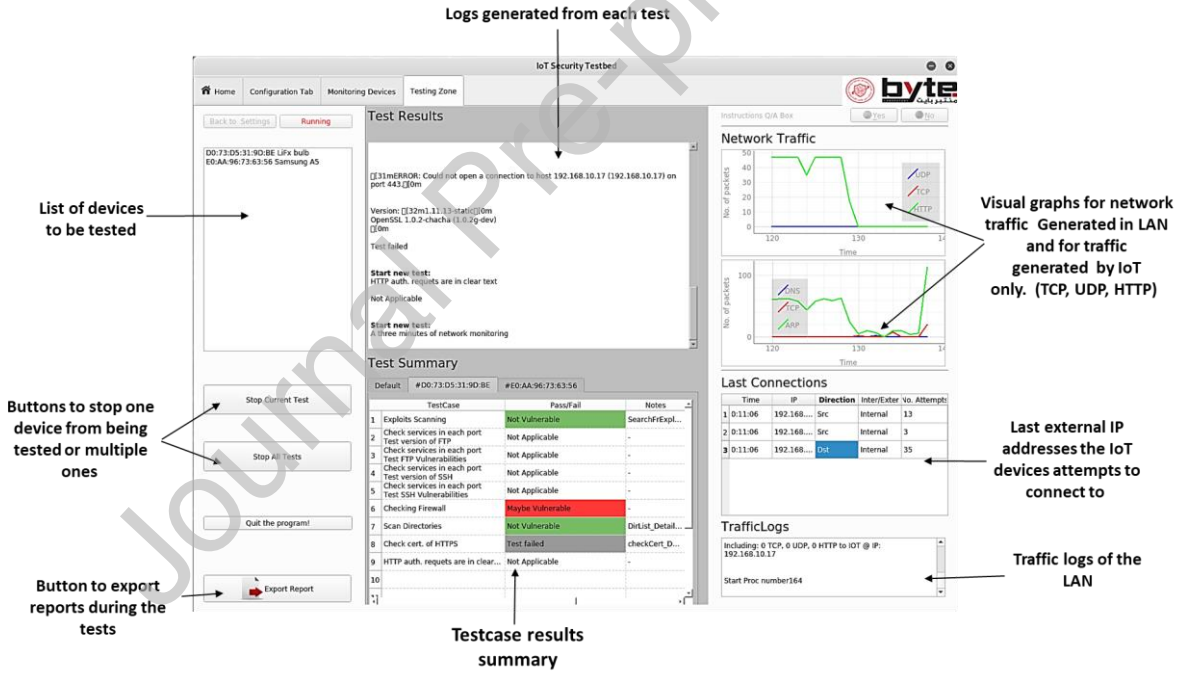


Figure 7. Graphical user interface of the testbed

To demonstrate the wireless camera security testing scenario, the steps in Figure 8 are followed. The wireless camera is contacted to be connected to the local network. If the testbed system software detects the camera, it tries to authorize and identify the device and retrieves saved information from its database. Once accepted, all traffic generated in the testbed—especially traffic originating from the device—is monitored and analyzed, as are the device's external connections. The fingerprints of all ports are checked with the Nmap tool to determine whether or not the device hosts a webserver. As the camera hosts a webserver on port 80, it is categorized as a "device with embedded webserver". The testbed runs a predefined list of tests to check open ports and discover any web server vulnerabilities. Each test assesses a specific security aspect. For example, the SQL injection test uses the SQLmap tool, which injects an SQL statement into the HTML page of the web server. If the webserver executes the statement, the device is considered to be vulnerable. Similarly, the device's

SSL, firewall and certificates are checked. Some scripts use different approaches to determine if the device is vulnerable or not. For example, the "device sync with NTP server" test entails scanning all NTP packets in the device's communications to determine whether or not it is synchronized with the NTP server. Each test case checks a specific characteristic of the device and reports if the device passed or failed this security check. Finally, once all the tests are done, all results will be reported in a word document as shown in Figure 4.

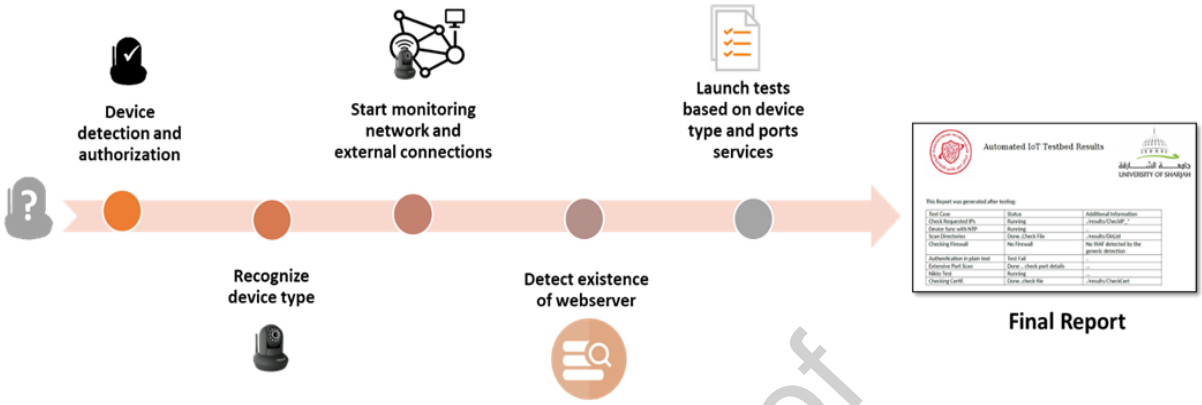


Figure 8. Testing for wireless camera assessment by the testbed software

Table 5 lists the test results generated by the proposed testbed from the wireless camera security assessment. Each test checks one vulnerability. If the vulnerability is found, the report indicates "Vulnerable" for that particular test. If the test found no vulnerability in a given domain, it is listed as a "Not Vulnerable". If a device receives "Not Vulnerable" for all tests, it means that no weak point was found, and the device is not vulnerable to the tests specified. Some tests generate detailed reports in the reporting module that must be reviewed by an operator, as shown in the "Additional information" column in Table 5. A column was added to the table to discuss the results.

From the test results, one can conclude that the wireless camera is vulnerable to attack, as it sends authentication credentials in clear text with no encryption. However, its use of an HTTP authentication mechanism means that it is not prone to SQL injection and XSS, as long as HTTPS protocol is used in communications instead of HTTP.

Table 5. Testbed report for wireless camera

Test case	Status	Additional information	Results discussion
Check Requested IPs	Not Vulnerable	Details of each IP are saved	During the test, the device did not connect to any malicious IP addresses; therefore, the device passed the test. The results of each IP address request have been exported in a separate file.
Device Sync with NTP	Not Vulnerable	-	The device was found to be in sync with NTP server; therefore, the DUI passed the test.
Scan Directories	Not Vulnerable	Details are saved	As no web server directory was found to be open without authentication, the device passed the test.
Check Firewall	Vulnerable	No WAF detected	The testbed did not find any firewalls in the web server; therefore, the DUI failed this security test.
Authentication in plain text	Vulnerable	User: Pass found	Authentication in plain text: When the DUI used an HTTP authentication mechanism, the authentication information (Username: Password) was sent in clear text without encryption. This is a very severe vulnerability, as an attacker could control the device as an admin using those credentials.
Extensive Port	Vulnerable	-	In this test, port 80 (HTTP and HTTPS) was found to be

Scan			open on the device, as well as port 23 (telnet) and 21 (FTP server). The fact that ports 21 and 23 were open means that the device was more prone to attacks. The testbed therefore reported that the device failed this particular test.
Nitko Test	Vulnerable	Vulnerable to cross-site request forgery and downgrade attack	The Nikto web assessment tool was used to assess the DUI. It reported some severe vulnerabilities. Therefore, the device failed the test.
Check Certificate	Vulnerable	No HTTPS	The DUI did not use a proper certificate in its communications, and port 443 was found to be open and not secured. Therefore, the device failed the test.
Check SSL	Vulnerable	No HTTPS	The DUI did not use SSL.
SQL Injection	Not Vulnerable	Details are saved	In this test, tools such as SQLmap are used to check if the device is vulnerable to SQL injection. However, the DUI used HTTP authentication rather than an HTML page and no SQL server were found. Consequently, device was not vulnerable to SQL injection, and it passed the test.

5.2. Smart Bulb Assessment

A smart bulb is an IoT device controlled by UDP packets. It receives controlling commands directly from users via a dedicated application or through a server or cloud. Information is sent using UDP or TCP packets, which are usually encrypted or ciphered. However, if the messages are not secure, the bulb may be vulnerable to replay attacks. For this reason, we test such devices against replay attacks and packet fabrication.

We tested a smart bulb controlled by a mobile application with our IoT security testbed. Only five tests were applicable, as the bulb did not have many open ports. The testbed started by checking IP addresses requested by the DUI, scanning for smart bulb exploits, replaying UDP packets, performing an Nmap scan and checking asynchronous connections with the time server. Conversely to the multiple tests run specifically for server host devices such as the smart camera, the only test uniquely dedicated to non-web server devices is the replay UDP packets test. This is because the smart bulb receives control commands through UDP packets.

The report generated from the smart bulb tests can be found in Table 6. The report lists a test as "Vulnerable" if an attack is successful; if the attack is unsuccessful, the test is listed as a "Not Vulnerable". The results show that the device is vulnerable to replay attacks, as it applied the commands received without checking the sender's MAC address.

Table 6. Testbed report for the smart bulb

Test case	Status	Additional information
Check Requested IPs	Not Vulnerable	Details of each IP are saved
Exploit Scan	Not Vulnerable	Details are saved
Replay attack – UDP	Vulnerable	..
Extensive Nmap scan	Not Vulnerable	..
Verification of asynchronous connection with time server	Vulnerable	..

6. Conclusion and Future Research

With the recent exponential increase in the use of IoT devices, security breaches associated with these devices are also on the rise. IoT device security testing is needed before the devices can be used

by the public. Assessing the security of IoT devices is difficult due to the wide variety and functionality of IoT devices. Although many research studies have explored IoT security assessment, there is an urgent need for extensive analysis and testing for vulnerabilities, and it is clear that these tasks should be automated. The goal of this research is to propose a new IoT security testbed architecture, and to present an automated IoT testbed to analyze IoT device security gaps.

Various penetration and security testing tools are leveraged to assess vulnerabilities in IoT devices. The proposed framework also secures the testbed, authenticates all devices used by the testbed and encrypts all communication between them. Furthermore, it records and logs all events that occur during the tests and generates reports informing the user if each test was passed or failed. The results provide data to inform the feasibility of practical experiments to assess common threats against these IoT devices. Two devices were successfully tested by our IoT testbed.

One of the biggest challenges in this domain is the exploding number of IoT devices being used, the great variety of IoT devices and protocols, and the lack of standardization in the field. This coupled with IoT devices interacting with each other greatly increases available attack vectors and the possibility of zero-day attacks, making it very hard for security experts and security testing tools to accurately assess the security level of different IoT devices.

We believe that an adequate testing architecture—one that is comprehensive enough to manage the abovementioned challenges and able to handle the evolving complexity of the IoT ecosystem—is yet to be developed. It will be interesting to see what developments take place in that direction. However, in designing our testbed, we think that we have taken a step in the right direction in helping to solve this difficult problem. The modular nature of our testbed and the ability to easily add new tests and change existing ones gives it the flexibility it needs to stay relevant as a security solution and to keep up with the demands of the growing IoT ecosystem.

Future work will include additional automated test cases and scenarios that tackle different aspects of IoT device security. More IoT devices need to be analyzed in order to increase the scope of our IoT testbed test case database. We are also looking forward to employing artificial intelligence to improve our methods for analyzing IoT devices and their vulnerabilities.

Funding: This research was funded by Dubai Electronic Security Center (DESC) and the University of Sharjah.

Acknowledgments: We are grateful to the Dubai Electronic Security Center (DESC) for funding this research project. We also express gratitude to the OpenUAE Research and Development Group for their support.

Appendix A

Table A1. Extensive Analysis Phase: First Device - Medical Gateway

Test case	Description	Test result	Notes
Check SSL certificate weakness	Tools test the existence of SSL certificate and gain more information.	Not vulnerable	SSL certificate uses OpenSSL and get TLS 1.2.
Downgrade attack	Force use of HTTP over HTTPs.	Not vulnerable	Downgrading the communication from HTTPs to HTTP doesn't work. The device refuses the connection request.
Break the password	Attempt to brute force the password.	Not vulnerable	The process takes a very long time.
Multiple logins at the same time	Attempt to login as admin using different devices at the same time.	Vulnerable	The device doesn't reject the second access, nor does it notify admin of the existence of another admin.
Directory access	List directories that are accessible without authentication.	Not vulnerable	No directories are open.
HTML analysis	Check vulnerabilities in html code.	Not vulnerable	No HTML
Inject JavaScript in the URL	Injecting JavaScript commands in the URL can give indirect access to information.	Not vulnerable	The test is not applicable for this device.
SQL injection in HTTP request	Use SQL injection in HTTP requests to gain unauthorized access to saved data in the server's database.	Not vulnerable	The test is not applicable for this device.
Bypass base authentication	Send misconfigured HTTP header to check if misconfigurations exist, which might give access to authorized information.	Not vulnerable	The device doesn't respond to misconfigured HTTP requests.
Firewall information	Tool to check the firewall used.	Not applicable	The web server rejects all connections.
Check Metasploit / Armtage for possible exploits	Metasploit / Armtage checks if an attack is possible against the device.	Not vulnerable	No exploits
Key installation attack (KRACK)– Proof of concept	The KRACK breaks the WPA2 protocol by forcing devices to reuse nonce during WPA2 handshake.	Vulnerable	The device uses another layer of encryption, as it uses TLS.
Optional encryption effects	The attack tests if confidential information (i.e. admin password) is exposed.	Vulnerable	The admin has the option of using HTTP or HTTPS in the configuration page. Once it is chosen, credentials are sent in clear text.

Table A2. Extensive Analysis Phase: Second Device – Wireless Camera

Test case	Description	Test result	Notes
Multiple logins at the same time	Attempt to log in as admin using different devices at the same time.	Vulnerable	The device doesn't reject the second access, nor does it notify the admin with the existence of another admin.
Multiple access attempts	Try multiple passwords, which results in multiple failed attempts.	Vulnerable	DUI doesn't block attempts, which can lead to brute force or dictionary attack.
Breaking the password	Attempt to get the password using dictionary attack.	Vulnerable	As the size of the password increases, the time it takes to break the password increases.

Directory access	Check for directories that are accessible without authorization.	Not vulnerable	No directories found to be accessible without authentication.
HTML analysis	Check vulnerabilities in HTML code.	Not vulnerable	No HTML
Inject JavaScript in URL	Injecting JavaScript commands in the URL can give indirect access to information.	Not vulnerable	The test is not applicable for this device.
SQL injection in HTTP request	Injecting SQL requests in HTTP to gain unauthorized access to data saved in the server database.	Not vulnerable	The test is not applicable for this device.
Bypass the base authentication	Send misconfigured HTTP header to check if misconfigurations exist, which might give access to authorized information.	Not vulnerable	The device doesn't respond to misconfigured HTTP requests.
Firewall information	Tool to check Firewall used.	Not available	The web server rejects all connections.
Check Metasploit / Armitage for possible exploits	Metasploit / Armitage will check if attacks are possible against the device.	Not vulnerable	No exploits are found.
Key installation attack (KRACK)– Proof of concept	The KRACK breaks WPA2 protocol by forcing devices to reuse nonce during the WPA2 handshake.	Not vulnerable	The device doesn't reuse nonce.
Man in the Middle (MITM) attack	The attack tests if confidential information (i.e. admin password) is exposed.	Vulnerable	The device doesn't use HTTPs. Device credentials are sent in clear text with no encryption during MITM attack.
Deauthentication attack	This attack tests if the camera can be disabled from the wireless.	Vulnerable	The device is disassociated from the network successfully.
Obtaining firmware	This tests if the firmware of the IP camera is found in online resources.	Applicable	The firmware of the wireless camera is found in online resources.
Reverse engineering	This test attempts to dump firmware from the hardware using UART in order to obtain root shell to access sensitive information.	Vulnerable	The camera is accessed through the UART. All files have been sent to another PC by using FTP server for later revision. Attackers are also able to write in the memory of the camera and change the password.
Cross domain attack	The attack tests if the camera has a file containing weak or improper configurations.	Vulnerable	Both firmware versions (1.02 and 1.16) are vulnerable to this attack.

☒No conflict of interest exists.

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

References

[1] V. A. Memos, K. E. Psannis, Y. Ishibashi, B. G. Kim, and B. B. Gupta, "An Efficient Algorithm for Media-based Surveillance System (EAMSuS) in IoT Smart City Framework," *Futur. Gener. Comput. Syst.*, vol. 83, pp. 619–628, 2018.

[2] C. Adjih *et al.*, "FIT IoT-LAB: A large scale open experimental IoT testbed," in *Proceedings of IEEE World Forum on Internet of Things, WF-IoT 2015*, 2015, pp. 459–464.

[3] A. Tewari and B. B. Gupta, "Security, privacy and trust of different layers in Internet-of-Things (IoTs) framework," *Futur. Gener. Comput. Syst.*, 2018.

[4] A. Tewari and B. B. Gupta, "A lightweight mutual authentication protocol based on elliptic curve cryptography for IoT devices," *Int. J. Adv. Intell. Paradig. Inderscience Publ.*, vol. 9, no. 2–3, pp. 111–121, 2017.

[5] M. Zanella, Andrea and Bui, Nicola and Castellani, Angelo and Vangelista, Lorenzo and Zorzi, "Internet of Things for Smart Cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, 2014.

[6] O. Badve, B. B. Gupta, and S. Gupta, "Reviewing the Security Features in Contemporary Security Policies and Models for Multiple Platforms," in *Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security*, no. May, 2017, pp. 479–504.

[7] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and Other Botnets," *Computer (Long. Beach. Calif).*, vol. 50, no. 7, p. 79, 2017.

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

[9] C. Stergiou, K. E. Psannis, B. G. Kim, and B. Gupta, "Secure integration of IoT and Cloud Computing," *Futur. Gener. Comput. Syst.*, vol. 78, pp. 964–975, 2018.

[10] 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 2017*, 2017, pp. 1–5.

[11] P. Cao, E. C. Badger, Z. T. Kalbarczyk, R. K. Iyer, A. Withers, and A. J. Slagell, "Towards an unified security testbed and security analytics framework," in *Proceedings of the 2015 Symposium and Bootcamp on the Science of Security*, 2015, pp. 1–2.

[12] G. Murad, A. Badarneh, A. Quscf, and F. Almasalha, "Software Testing Techniques in IoT," in *2018 8th International Conference on Computer Science and Information Technology, CSIT 2018*, 2018, pp. 17–21.

[13] S. Siboni *et al.*, "Security Testbed for Internet-of-Things Devices," *IEEE Trans. Reliab.*, vol. 68, no. 1, pp. 23–44, 2018.

[14] K. Ly and Y. Jin, "Security Studies on Wearable Fitness Trackers," in *38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society.*, 2016, p. 32816.

[15] J. Wurm, K. Hoang, O. Arias, A. R. Sadeghi, and Y. Jin, "Security analysis on consumer and industrial IoT devices," in *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*, 2016, vol. 25–28–Janu, pp. 519–524.

[16] F. M. Tabrizi and K. Pattabiraman, "Formal Security Analysis of Smart Embedded System," *Proc. 32nd Annu. Conf. Comput. Secur. Appl.*, pp. 1--15, 2016.

[17] T. H.-J. Kim, L. Bauer, J. Newsome, A. Perrig, and J. Walker, "Challenges in Access Right Assignment for Secure Home Networks.," in *HotSec*, 2010.

[18] B. Ur, J. Jung, and S. Schechter, "The current state of access control for smart devices in homes," in

Workshop on Home Usable Privacy and Security (HUPS), 2013.

[19] G. Ho, D. Leung, P. Mishra, A. Hosseini, D. Song, and D. Wagner, "Smart Locks: Lessons for Securing Commodity Internet of Things Devices," *Proc. 11th ACM Asia Conf. Comput. Commun. Secur. - ASIA CCS '16*, pp. 461–472, 2016.

[20] S. Chistiakov, "Secure storage and transfer of data in a smart lock system," 2017.

[21] G. Hernandez, O. Arias, D. Buentello, and Y. Jin, "Smart Nest Thermostat : A Smart Spy in Your Home," *Black Hat USA*, pp. 1–8, 2014.

[22] Y. Oren and A. D. Keromytis, "From the Aether to the Ethernet-Attacking the Internet using Broadcast Digital Television.," in *USENIX Security Symposium*, 2014, pp. 353–368.

[23] T. Denning and T. Kohno, "Empowering consumer electronic security and privacy choices: Navigating the modern home," in *Symposium on Usable Privacy and Security (SOUPS)*, 2013.

[24] M. Ye, N. Jiang, H. Yang, and Q. Yan, "Security analysis of Internet-of-Things: A case study of august smart lock," in *Computer Communications Workshops (INFOCOM WKSHPS), 2017 IEEE Conference on*, 2017, pp. 499–504.

[25] T. Denning, T. Kohno, and H. M. Levy, "Computer security and the modern home," *Commun. ACM*, vol. 56, no. 1, p. 94, 2013.

[26] B. Ur, J. Jung, and S. Schechter, "Intruders versus intrusiveness: teens' and parents' perspectives on home-entryway surveillance," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2014, pp. 129–139.

[27] E. Fernandes, J. Jung, and A. Prakash, "Security Analysis of Emerging Smart Home Applications," in *Proceedings of 2016 IEEE Symposium on Security and Privacy, SP 2016*, 2016, pp. 636–654.

[28] N. Gyory and M. Chuah, "IoTOne: Integrated platform for heterogeneous IoT devices," in *2017 International Conference on Computing, Networking and Communications, ICNC 2017*, 2017, pp. 783–787.

[29] M. Ammar, G. Russello, and B. Crispo, "Internet of Things: A survey on the security of IoT frameworks," *J. Inf. Secur. Appl.*, vol. 38, pp. 8–27, 2018.

[30] E. Fernandez, J. Pelaez, and M. Larrondo-Petrie, "Attack patterns: A new forensic and design tool," in *IFIP International Conference on Digital Forensics*, 2007, pp. 345–357.

[31] T. A. Alghamdi, A. Lasebae, and M. Aiash, "Security analysis of the constrained application protocol in the Internet of Things," in *Future Generation Communication Technology (FGCT), 2013 second international conference on*, 2013, pp. 163–168.

[32] B. Cyr, W. Horn, D. Miao, and M. Specter, "Security analysis of wearable fitness devices (fitbit)," *Massachusets Inst. Technol.*, p. 1, 2014.

[33] M. Moody and A. Hunter, "Exploiting known vulnerabilities of a smart thermostat," in *Privacy, Security and Trust (PST), 2016 14th Annual Conference on*, 2016, pp. 50–53.

[34] E. Ronen and A. Shamir, "Extended functionality attacks on IoT devices: The case of smart lights," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, 2016, pp. 3–12.

[35] O. Arias, S. Member, J. Wurm, K. Hoang, and Y. Jin, "Privacy and security in internet of things and wearable devices," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 7766, no. 2, pp. 99–109, 2015.

[36] Y. Bachy *et al.*, "Smart-TV security analysis: practical experiments," in *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*, 2015, pp. 497–504.

[37] V. Sivaraman, D. Chan, D. Earl, and R. Boreli, "Smart-phones attacking smart-homes," in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2016, pp. 195–200.

[38] P. Morgner, S. Mattejat, and Z. Benenson, "All your bulbs are belong to us: Investigating the current

state of security in connected lighting systems," *arXiv Prepr. arXiv1608.03732*, 2016.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] T. Willingham, C. Henderson, B. Kiel, M. S. Haque, and T. Atkison, "Testing vulnerabilities in bluetooth low energy," in *Proceedings of the ACMSE 2018 Conference*, 2018, p. 6.

[47] A. O. Prokofiev, Y. S. Smirnova, and V. A. Surov, "A method to detect Internet of Things botnets," in *Young Researchers in Electrical and Electronic Engineering (EIConRus), 2018 IEEE Conference of Russian*, 2018, pp. 105–108.

[48] M. Gegick and L. Williams, "Matching attack patterns to security vulnerabilities in software-intensive system designs," *ACM SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, p. 1, 2005.

[49] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT Sentinel: Automated device-type identification for security enforcement in IoT," in *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*, 2017, pp. 2177–2184.

[50] S. Demetriou *et al.*, "Guardian of the HAN: Thwarting Mobile Attacks on Smart-Home Devices Using OS-level Situation Awareness," *arXiv Prepr. arXiv1703.01537*, 2017.

[51] E. Gelenbe, J. Domanska, T. Czàchorski, A. Drosou, and D. Tzovaras, "Security for internet of things: the SerIoT project," in *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, 2018, pp. 1–5.

[52] M. Chernyshev, Z. Baig, O. Bello, and S. Zeadally, "Internet of Things (IoT): research, simulators, and testbeds," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1637–1647, 2018.

[53] Y. Berhanu, H. Abie, and M. Hamdi, "A testbed for adaptive security for IoT in eHealth," in *Proceedings of the International Workshop on Adaptive Security*, 2013, p. 5.

[54] V. Sachidananda, J. Toh, S. Siboni, S. Bhairav, A. Shabtai, and Y. Elovici, "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, co-located with ASIA CCS 2017*, 2017, pp. 3–10.

[55] M. L. Hale, K. Lotfy, R. F. Gamble, C. Walter, and J. Lin, "Developing a platform to evaluate and assess the security of wearable devices," *Digit. Commun. Networks*, 2018.

[56] A. Tekeoglu and A. S. Tosun, "A Testbed for Security and Privacy Analysis of IoT Devices," in *Mobile Ad Hoc and Sensor Systems (MASS), 2016 IEEE 13th International Conference on*, 2016, pp. 343–348.

[57] E. S. Alashwali and K. Rasmussen, "What's in a Downgrade? A Taxonomy of Downgrade Attacks in the TLS Protocol and Application Protocols Using TLS," in *International Conference on Security and Privacy in Communication Systems*, 2018, pp. 468–487.

Omnia Abu Waraga

Omnia Abu Waraga received her bachelor's degree (with honor) in computer engineering from University of Sharjah, UAE in 2017. Currently, she is persuading her master's degree in computer science in the same university. She is also a research assistant in OpenUAE Research and Development group. She has interests in Internet of Things security, vulnerability assessment and artificial intelligence. Omnia is a mentoring coordinator in ArabWIC UAE chapter, member of IEEE and an event organizer in Google Developer Group Sharjah branch.

Meriem Bettayeb

Meriem Bettayeb received her bachelor's degree (with first honor) in computer engineering from University of Sharjah, UAE in 2017 and received her master's degree in computer engineering with honor in the same university in 2019. Currently, she is a research assistant in OpenUAE Research and Development group. She has interests in Internet of Things security, firmware analysis, artificial intelligence and Blockchain technology. Meriem is a mentoring and event coordinator in ArabWIC UAE chapter, member of IEEE and an event organizer in Google Developer Group Sharjah branch.

Dr. Manar Abu Talib

Dr. Manar Abu Talib has interest in software engineering software measurement, software quality and testing, ISO-27001 for Information Security and OpenSource Software. Dr. Manar was involved in developing the Arabic version of ISO-19761 (COSMIC-FFP measurement method). She published +40 refereed conferences and journals, involved in +200 professional and research activities and supervised 30 capstone projects. Dr. Manar is a Co-coordinator of OpenUAE Research & Development Group and the International Collaborator to Software Engineering Research Laboratory in Montreal, Canada. Manar is the ArabWIC VP, Google WTM Lead, an executive member in UAE IEEE Section and the UAE representative for the COSMIC-FPP Education Committee.

Dr. Qassim Nasir

Dr. Qassim Nasir is currently an associate professor in University of Sharjah since 2009. He received his B.Sc., M.Sc., and Ph.D. degrees from the University of Baghdad, Iraq. He was working with Nortel Networks, Canada, as a senior system designer and then as a senior firmware system designer. His current research interests are in telecommunication and network security, CPS, IoT, drones and GPS jamming. Dr. Qassim has published over 90 refereed conferences, journals, book chapter, and technical reports. He holds professional certificate such as CISSP and Cisco trainer. He was visiting professor at Helsinki University of Technology, Finland, during the summers of 2002-2009. Dr. Qassim is a Co-coordinator of OpenUAE Research & Development Group.