

Security Testbed for Internet-of-Things Devices

Shachar Siboni , Vinay Sachidananda , Yair Meidan , Michael Bohadana, Yael Mathov , Suhas Bhairav , Asaf Shabtai , and Yuval Elovici

Abstract—The Internet of Things (IoT) is a global ecosystem of information and communication technologies aimed at connecting any type of object (thing), at any time, and in any place, to each other and to the Internet. One of the major problems associated with the IoT is the heterogeneous nature of such deployments; this heterogeneity poses many challenges, particularly, in the areas of security and privacy. Specifically, security testing and analysis of IoT devices is considered a very complex task, as different security testing methodologies, including software and hardware security testing approaches, are needed. In this paper, we propose an innovative security testbed framework targeted at IoT devices. The security testbed is aimed at testing all types of IoT devices, with different software/hardware configurations, by performing standard and advanced security testing. Advanced analysis processes based on machine learning algorithms are employed in the testbed in order to monitor the overall operation of the IoT device under test. The architectural design of the proposed security testbed along with a detailed description of the testbed implementation is discussed. The testbed operation is demonstrated on different IoT devices using several specific IoT testing scenarios. The results obtained demonstrate that the testbed is effective at detecting vulnerabilities and compromised IoT devices.

Index Terms—Internet of Things (IoT), IoT devices, privacy, security, testbed framework.

NOMENCLATURE

IoT	Internet of Things.
DoS	Denial-of-service.
MITM	Man-in-the-middle.
PII	Personally identifiable information.
BYOD	Bring your own device.
XSS	Cross-site scripting.
MRM	Management and reports module.
STMM	Security testing manager module.
STM	Security testing module.
MAM	Measurements and analysis module.

Manuscript received August 14, 2017; revised November 15, 2017 and May 17, 2018; accepted August 2, 2018. Date of publication December 6, 2018; date of current version February 26, 2019. This work was supported by the Singapore Ministry of Defense (MINDEF). Associate Editor: Z. Chen. (*Corresponding author: Shachar Siboni*)

S. Siboni, Y. Meidan, M. Bohadana, Y. Mathov, A. Shabtai, and Y. Elovici are with the Department of Software and Information Systems Engineering, Cyber Security Research Center, Ben-Gurion University of the Negev, Beersheba 84105, Israel (e-mail: sibonish@post.bgu.ac.il; yairme@post.bgu.ac.il; bohadana@post.bgu.ac.il; yaelmath@post.bgu.ac.il; shabtaia@bgu.ac.il; elovici@bgu.ac.il).

V. Sachidananda and S. Bhairav are with the iTrust, Singapore University of Technology and Design, Singapore 487372 (e-mail: sachidananda@sutd.edu.sg; suhas_setikere@sutd.edu.sg).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TR.2018.2864536

DUT	Device under test.
ADB	Android debug bridge.
OM	Orchestrating machine.
CCM	Control and communication machine.
AM	Analysis machine.
CVE	Common vulnerabilities and exposures.
CVSS	Common vulnerability scoring system.
NVD	National vulnerability database.
SUT	System under test.
TPR	True positive rate.
TNR	True negative rate.

I. INTRODUCTION

THE Internet of Things (IoT) consists of a combination of physical objects with sensors, actuators, and controllers with connectivity to the digital world via the Internet. The low cost of hardware, along with the prevalence of mobile devices and widespread Internet access, has made the IoT a part of modern everyday life. An exponential increase in the use of IoT devices is expected in the future; as it does, security issues must increasingly be considered given that all IoT devices are connected to the Internet, providing the means for hackers to obtain access to these devices.

SHODAN [1], the IoT search engine, shows the dark side of connected IoT devices, where several vulnerabilities have been discovered using this tool [2], [3]. Different Internet-connected devices, ranging from cameras to industrial controllers, can be easily manipulated [4], [5]. These studies confirm both the fact that IoT devices are, by their very nature, prone to attacks, and the need to seriously consider security measures for such devices. Furthermore, no common security standard exists for all IoT devices. Although there is a need to address the security challenges of the IoT ecosystem, a flexible method for evaluating the security of IoT devices does not currently exist, and there is a lack of dedicated testbeds to perform security testing and analysis on IoT devices [6].

The development of a testbed to perform comprehensive security testing and analysis for IoT devices under real conditions will help to remedy this situation. Moreover, due to the heterogeneity of IoT devices (different types of devices with different configurations, such as device drivers, hardware and software components, and more), an advanced generic security testbed is required. In this paper, we propose a fully functional IoT testbed for security analysis in which various IoT devices are tested against a set of security requirements. The proposed IoT testbed can emulate different types of testing environments that simulate the activity of various sensors (such as GPS,

movement, Wi-Fi, etc.), as presented in [7], and perform predefined and customized security tests along with advanced security testing analysis based on machine learning approaches.

The testbed consists of hardware and software components for experiments involving wide-scale testing deployments. The proposed security testbed supports a range of security tests, both standard and advanced security testing, aimed at different aspects of security requirements. Standard security tests uses standard, of-the-shelf security analysis tools that can perform vulnerability scans and penetration tests, in order to assess and verify the security level of IoT devices under test. Advanced security tests implement more complex mechanisms, such as using machine learning algorithms in order to identify the device type and to detect suspicious behavior of an IoT device under test (DUT) by analyzing its network traffic, evaluating the resilience of the IoT device to denial-of-service (DoS) attacks, or checking the management connection password complexity test in order to measure the device security strength.

Given the fact that the vast majority of security technologies adopted today are primarily focused on alerting users about specific technical aspects of an attack, rather than the root cause of an attack, an implementation of automated security testbed can be difficult. Moreover, defining the requirements for the development and implementation of such a testbed is also a challenging task. Furthermore, designing a comprehensive security testing system targeted for IoT scenarios is a challenging task. In this paper, the testbed system architecture and design is a layer-based platform model with a modular structure. Based on this architecture and design, any type of IoT device can be tested in the proposed security testbed framework, including smart appliances, smart city devices, smart wearable devices, and more. In addition, any relevant simulator and/or measurement and analysis tool can be deployed in the testbed environment in order to perform comprehensive testing in the testbed. As a modular system, the testbed also integrates different analysis mechanisms as plugins that are used to conduct advanced security testing.

The main contributions of this paper are threefold.

- 1) We provide a detailed discussion of security and privacy threats for current and future IoT devices, and present several approaches to mitigate the threats including our proposed security testbed targeted for IoT devices.
- 2) We present the system requirements and design for a novel advanced security testbed framework, and provide an in depth description of the proposed testbed mechanism, including the interactions between the relevant modules of the testbed framework that are designed to deal with the challenges that are associated with security testing for IoT devices.
- 3) We use the implementation of our testbed to analyze different types of IoT devices, using the standard security testing methodology and advanced security testing methods, in order to determine and evaluate the security level of the tested IoT devices under test.

The structure of this paper is as follows. After providing an introduction in Section I, related work is discussed in Section II. In Section III, we present different security aspects related with IoT devices. In Section IV, we describe the testbed system ar-

chitecture and design and discuss practical implementation of the testbed in Section V. Section VI provides several test scenarios conducted using the proposed testbed, and Section VII conclude this paper.

II. RELATED WORK

Several testbeds have been proposed for IoT devices [6]. In addition, there are a few labs around the world that focus on IoT security [8]. Most of the recent work on IoT testbeds tends to focus on a single technology domain [e.g., wireless sensor networks (WSNs)] [9]–[12]. Others take a more heterogeneous approach to the study of IoT testbeds [13], [14]. There are very few studies using various IoT devices and focusing on multiple technology domains [15].

MoteLab [9], which provides a testbed system for WSNs, was one of the first testbeds developed. Still in use today, it has also served as the basis for various other testbeds such as INDRIYA [16]. Kansei [10] is one of the most surveyed testbeds, providing various advanced functions, including cosimulation support, mobility support using mobile robots, and event injection possible mote level. CitySense [11] is a public mesh testbed deployed on light poles and buildings. The following two features make this testbed particularly interesting: 1) its realism and domain specificity provided by a permanent outdoor installation in an urban environment, and 2) the implementation of the control and management plane based solely on wireless links. The Senselab [12] testbed consists of more than 1000 sensor nodes with energy measurement supported for every node and repeatable mobility via electric toy trains. In [13], the testbed consists of federation architecture, cosimulation support, topology virtualization, *in situ* power measurements on some nodes, and mobility support. FIT IoT-LAB [14] provides a very large scale infrastructure facility suitable for testing small wireless sensor devices and heterogeneous communicating objects. The testbed offers web-based reservation and tooling for application development, along with direct command line access to the platform. All of the aforementioned IoT testbeds focus solely on WSNs.

The T-City Friedrichshafen [15] testbed considers various IoT devices, making it multidomain; it combines innovative information and communication technologies, together with a smart energy grid, to test out innovative healthcare, energy, and mobility services. Although the T-City Friedrichshafen testbed is multidomain, it fails to take into account security aspects.

INFINITE [17], 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. Projects such as FIESTA-IOT [18] provide experimental infrastructure for heterogeneous IoT technologies. The FIESTA-IOT project consists of various testbeds like SmartCampus [19] and SmartSantander [20]. SmartSantander proposes a unique city scale experimental research facility for common smart city applications and services. In [21], the authors propose ASSET (Adaptive Security for Smart Internet of Things in eHealth), a project to develop risk-based adaptive security methods and mechanisms for IoT

in eHealth. The project proposes a testbed to accurately evaluate adaptive security solutions in realistic simulation and use case scenarios, however, the project does not address multidomain IoT devices and security aspects.

Stanford's Secure Internet of Things Project [8] is a cross-disciplinary research effort between computer science and electrical engineering faculty at Stanford University; the University of California, Berkeley; and the University of Michigan. The research effort focuses on the following three key areas: analytics, security, and hardware and software systems. Though the project is focused on securing IoT devices, a full security testbed system has not yet been proposed in [8].

Hence, based on our knowledge, critical gaps exist, and a testbed that focuses on the security testing for IoT devices, and especially considering different context environments, has not yet been developed.

III. SECURITY ASPECTS OF IoT DEVICES

IoT devices may pose major security and privacy risks, because of their range of functionality and the variety of processes involved in their operation, including data collection, processing, storage, and transfer—by, from, and to these smart devices [22], [23]. Furthermore, these smart devices are integrated in enterprise networks, deployed on public spaces, and worn on the body and can be operated continuously in order to gather information from their surroundings; hence, they are highly visible and accessible—especially to attackers. In the following subsections, we discuss security and privacy aspects related to device architecture, network connectivity, and the type of data collected by IoT devices. In addition, we present countermeasures to reduce and mitigate the problems discussed.

A. Device Architecture

The device architecture security aspect includes hardware and software considerations as follows. Regarding hardware, IoT devices are low resource devices, in terms of power source, memory size, bandwidth communication, and computational capabilities [24], [25]. This may result in severe security flaws, as only lightweight-based encryption mechanisms and authentication algorithms can be applied in order to encrypt the data stored on, and transmitted from, the device [25], [26].

From a software perspective, open source and proprietary operating systems are in use, which can be highly exposed to known and zero-day vulnerabilities [27]. Additionally, the applications running on IoT devices are only as good as the developers who wrote them. Often, if serious bugs are identified in the software, no one is responsible for patching them [28]. Furthermore, in contrast to standard computing systems, most IoT devices are assumed to be less continuously maintained and upgraded by the manufacturers [26].

IoT devices often automate certain functionalities and require limited configuration with little intervention from the user [25]. For example, the Google Glass device enables the automatic set up of a Wi-Fi connection after viewing QR codes or sharing information on the web. This can make IoT devices more exposed to security risks than traditional computing devices.

B. Network Connectivity

IoT devices can be constantly connected to the Internet, either directly via long range connectivity (e.g., via cellular network), or indirectly using gateways via short/medium range connectivity (e.g., via Wi-Fi, Bluetooth connection, etc.) [29], [30]. However, these advanced devices are not always designed with security in mind, due to cost considerations and their limited resources [31], [32]. Consequently, IoT devices can be highly exposed to the traditional Internet attacks, such as DoS attacks, data leakage, man-in-the-middle attacks, phishing attacks, eavesdropping, side-channel attacks, and compromise attacks [24], [25]. Moreover, due to the fact that lightweight authentication algorithms are employed, it is quite possible to manipulate and control these devices at their weakest point—when data are sent from, and received by, the device [23].

Another potential security issue is network disruption and overload [26]. With the proliferation of IoT devices, especially in private enterprise networks, public spaces, and more, these smart connected devices are constantly producing and broadcasting information, and thus, unceasingly consume bandwidth. More importantly, they increase the attack surface as they become new points of entry into the network.

C. Data Collection

A major concern related to IoT devices is the type of data they collect, which potentially may lead to privacy invasion and information theft [30], [33]. As data become an increasingly valuable asset, many data brokers collect information about potential customers and organizations by any means, including vulnerable IoT devices.

From a user's point of view, most of the collected data are personal, and may contain sensitive information about the user's habits and behavior, and even private health details. Moreover, recently, IoT technology has also been integrated into enterprise and organizational environments in order to increase the business productivity and efficiency levels [30], [34]. As IoT devices become more commonly used in the workplace, companies might exploit them to violate employees' privacy, as employers can track and record an employee's actions—and even more worrisome—monitor a user's health condition, e.g., using smartwatches and wristbands. On the other hand, for using IoT devices on enterprise networks, sensitive corporate information might also become more accessible to outsiders and can be exposed to unauthorized individuals via these smart devices [22], [35].

Another concern associated with IoT devices centers on theft or loss of the device, as well as ransomware attacks [36]. Personally identifiable information (PII) stored on the device renders it at risk to security and privacy issues. Due to the lightweight security mechanisms that are employed, this sensitive information is readily accessible to attackers and can be used for malicious activities, such as identity theft [37].

D. Countermeasures and Mitigations

Several countermeasures can be implemented to reduce and mitigate the security and privacy risks posed by IoT devices

[7]. For example, sensitive data stored on the device should be limited and encrypted (both regarding the type, and the amount of data stored on the device) in order to reduce the possibility of personal data exposure. In addition, data scrubbing and automatic wipe features that enable remote deletion of unnecessary data from IoT devices should be employed.

From a business point of view, companies should enforce security and privacy policies, e.g., BYOD policy. This can be done using enterprise-grade encryption mechanisms for access control in order to identify any new connected device in the network, as well as to protect data from eavesdropping measures. Moreover, the rule of least privilege should be implemented to limit the capability of employees to read and/or write unauthorized data and restrict attackers from accessing sensitive corporate data from IoT devices that have been compromised. In addition, implementing further authentication, authorization, and accountability mechanisms for IoT devices that directly connect to the network is required.

As most IoT devices are wireless-based and always ON, it is preferable to turn the wireless connectivity OFF once the device is not in use. Moreover, users should be responsible for maintaining and periodically updating software versions and downloading relevant updates and patches for their IoT devices.

If, for any reason, the aforementioned security problems cannot be mitigated, IoT electronic devices will eventually need to be banned in highly sensitive places, as is the case with other commonly used mobile devices (such as laptops, smartphones, tablets, etc.), in order to provide an infrastructure solution. Such measures will be instituted in the interest of protecting the security, privacy, and confidentiality of the surroundings.

In addition to the aforementioned countermeasures and mitigations, there is a constant need to be able to evaluate the security and privacy levels of IoT devices. This should be done using a designated security testbed for the IoT, where the motivation is to perform security testing targeted specifically for IoT devices as a means of assessing their security level. Because the conditions that trigger compromised devices to execute attacks are not always known, the testbed should be able to simulate possible conditions (e.g., using different simulators) [7] in order to identify any *context-based attacks* the device may carry out under predefined conditions that an attacker may set, as well as *data attacks*, which may be achieved by sending crafted (or manipulated) context/sensor data. The issue, of security testbed for IoT devices, is discussed in more detail in this paper.

IV. TESTBED ARCHITECTURE AND DESIGN

The proposed security testbed architecture and design, presented in this section, stem from end users' needs, prioritized risk scenarios, regulation laws, and best practices and standards, as well as the system architecture, including an in depth description of the testbed's modules and the interactions between these modules as a full security testing platform.

A. Testbed Capabilities

The required capabilities for a security testbed for the IoT can be classified by and formulated on various abstraction levels as follows.

1) Initialization and Detection: 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. After initialization and activation of the IoT device, the next requirement is the detection of the IoT device present in the testbed environment. During the detection process, a log file should be created consisting of the IoT device operating system (OS), the processes running, actions being performed, etc. This information will be used for any subsequent anomaly detection.

2) Security Tests: The IoT 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 an analysis for these vulnerabilities. Accordingly, our security testbed takes into account some of the vulnerabilities from OWASP [38], including: injection, broken authentication and session management [2], cross-site scripting (XSS), security misconfiguration, sensitive data exposure, missing function level access control, and using Components with known vulnerabilities. In addition, the testbed should support templates of tests and scenarios. 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., all tests that are relevant to IP cameras). In addition, the testbed should provide a success criterion for each test (for example, 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).

3) Logging and Analysis: After conducting a series of steps associated with the functional requirements, 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. In addition, the testbed system should support intelligent analysis.

4) Usability: Usability ensures the testbed's ease of use, with minimal efforts on the part of the user. The security testbed should be easy to operate and use, with easily defined tests, easy to input configuration, and easy to interpret output.

5) Security Related:

- 1) **Reliability:** Refers to the ability of the IoT testbed to perform its required functions under the stated conditions for a specific period of time.
- 2) **Antiforensic:** Refers to the capability of the testbed to detect, and subsequently, prevent malicious applications on the IoT device (if it has been infected) from being activated.
- 3) **Security:** Refers to the ability of the testbed to ensure authorized access to the system in order to safeguard the integrity of the IoT testbed from accidental or malicious damage.
- 4) **Accountability (Including Nonrepudiation):** Refers to the capability of the testbed to keep audit records in order to support independent review of access to resources/uses of the testbed.

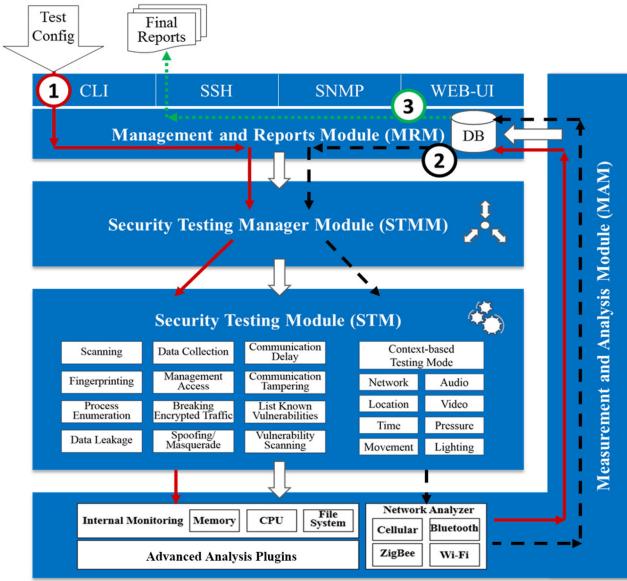


Fig. 1. Security testbed framework—Abstract functional architecture model.

6) *Adaptive*: The security testbed should be able to adapt in accordance with new application domain concepts and support various communication types.

- 1) *Scalability*: Refers to the capability of the testbed to increase the total throughput under an increased load when resources (typically software and hardware) are added to the testbed.
- 2) *Performance*: Refers to the ability of the testbed to perform well under different conditions, e.g., performance with respect to the time and user input.
- 3) *Flexibility*: Refers to the ability to modify the testbed after deployment. This includes adaptability, sustainability, and customizability.

B. System Architecture

The architecture of the security testbed, illustrated in Fig. 1, is a layer-based platform with a modular structure. This means that any type of the IoT device can be tested in the proposed security testbed framework, including smart appliances, smart city devices, smart wearable devices, and more. In addition, in order to perform the security testing under different contexts, any relevant simulators and/or stimulators can be deployed in the testbed environment, along with measurement or analysis tools used to collect and analyze test results. As a modular system, the testbed also integrates different analysis mechanisms as plugins that are used to conduct advanced security testing (mainly mechanisms based on machine learning algorithms that we developed). A detailed description of the modules that comprise the functional model and the interactions between these modules as a complete security testing system are provided. Note that the architecture model suggested here is based on our existing model [7], as this study is a continuation of research on this subject.

1) *Management and Reports Module (MRM)*: This module is responsible for a set of management and control ac-

tions, including starting/initializing the test procedure, enrolling new devices, simulators/stimulators, security tests, measurement and analysis tools, to the testbed, and generating the final reports upon completion of the test. The testbed operator (the user) interfaces with the testbed through this module using one of the communication interfaces (command line interface\secure shell (SSH)\simple network management protocol (SNMP)\web user interface (WEB-UI)) in order to initiate the test, as well as to receive the final reports. Accordingly, this module interacts with the security testing manager module (STMM) and the measurements and analysis module (MAM), respectively. The MRM contains a system database component that stores all relevant information about the tested device (including the OS, connectivity, sensor capabilities, advanced features, etc.), as well as information regarding the test itself (including config files, system snapshots, and test results).

2) *Security Testing Manager Module (STMM)*: This module is responsible for the actual testing sequence executed by the security testbed (possibly according to regulatory specifications). Accordingly, it interacts with the security testing module (STM) 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).

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 DUT. See Table I for a list of supported tests and the appropriate success criteria for each test. The STM is an operational module that executes a set of security tests as plugins, each of which performs a specific task in the testing process. This module also supports a context-based testing mode, where it generates various environmental stimuli for each sensor/DUT. Meaning, in this mode of operation, the STM simulates different environmental triggers and runs the security tests in order to simulate different contexts and working environments for the tested IoT devices, as illustrated in [7]. This is obtained using a simulator array list, such as a GPS simulator or Wi-Fi localization simulator (for location-aware and geolocation-based attacks), time simulator (using simulated cellular network, GPS simulator, or local NTP server), movement simulator (e.g., using robots), etc. See Table II for a list of supported simulators. The module interacts with the measurements and analysis module (MAM) in order to monitor the test performed and analyze the results of the test.

4) *Measurements and Analysis Module (MAM)*: This module employs a variety of measurement (i.e., data collection) components and analysis components (both software and hardware based). The measurement components include different network sniffers for communication monitoring such as Wi-Fi, cellular, Bluetooth, and ZigBee sniffers, and device monitoring tools for measuring the internal status of the devices under test, including memory consumption, CPU utilization, and file system changes in the IoT-DUT. Based on the collected information, advanced security testing is conducted in the testbed, using mechanisms based on machine learning algorithms that we developed. The analysis component processes the collected data and evaluates the results according to a predefined suc-

TABLE I
PENETRATION TESTS SUPPORTED BY THE SECURITY TESTBED

Test	Description	Test Success Criteria
Scanning (e.g., IP and port scanning, data traffic, etc.)	Investigate the detectability of IoT devices by observing wireless/wired communication channels. Attempt to identify the existence of the device. Enumerate communication channels/traffic types observed and open ports (active monitoring). Collect data traffic (passive monitoring), etc.	<i>Undetectable</i> - the IoT-DUT cannot be detected by the testbed via any communication channel; <i>Safe</i> - the IoT-DUT is detectable, but no open ports were observed; <i>Minor risk</i> - the IoT-DUT is detectable, and common ports are open, e.g., port 80 (HTTP), 443 (HTTPS), etc.; <i>Major risk</i> - the IoT-DUT is detectable, and uncommon ports for such devices are open, e.g., ports 20, 21 (FTP), port 22 (SSH), port 23 (Telnet), etc.; or, <i>Critical risk</i> - the IoT-DUT is detectable, and unexpected ports are open on the device.
Fingerprinting	By monitoring communication traffic to/from the device, attempt to identify the type of device, its operating system, software version, list of all sensors supported, etc.	<i>Unidentifiable</i> - the type of IoT-DUT cannot be identified by the testbed; <i>Safe</i> - the device provides identifiable information, but all the IoT-DUT's software versions are up-to-date; <i>Minor risk</i> - some low risk applications, e.g., calendar, etc., are out-of-date; <i>Major risk</i> - some major risk applications, e.g., navigator, mail, etc., are out-of-date; or, <i>Critical risk</i> - operating system and critical applications are out-of-date.
Process Enumeration	Lists all processes running on the device and presents their CPU and memory consumption. This can be done by monitoring the device's activities, e.g., using ADB (Android Debug Bridge) connectivity.	<i>Safe</i> - the list of processes cannot be extracted without admin privileges; <i>Moderate risk</i> - the list of processes can be extracted without admin privileges on the device only; or, <i>Fail</i> - the list of processes can be remotely extracted without admin privileges.
Data Leakage	Validate which parts of the communication to/from the device are encrypted (and how) or sent in clear text, and accordingly, check if an application leaks data out of the device.	<i>Pass</i> - traffic is encrypted, and no data leaks are detected; or, <i>Fail</i> - traffic is unencrypted and sent in clear text, therefore, data may leak from the IoT-DUT.
Data Collection	Check if an application on an IoT device collects sensor data and stores it on the device. This can be achieved by monitoring the locally stored data and correlating sensor events.	<i>Safe</i> - the tested application does not collect and store data on the IoT-DUT; <i>Minor risk</i> - the tested application collects and stores normal data, e.g., multimedia files, on the IoT-DUT; <i>Major risk</i> - the tested application collects and stores sensitive data, e.g., GPS locations, on the IoT-DUT; or, <i>Critical risk</i> - the tested application collects and stores critical information, e.g., device status (CPU, memory, sensor events, etc.), on the IoT-DUT.
Management Access	Attempt to access the management interface/API of a device using one of the communication channels. Access could be obtained by using default credentials, a dictionary attack, or other known exploits.	<i>Pass</i> - management access ports, e.g., port 22 (SSH), port 23 (Telnet), are closed; or, <i>Fail</i> - one of the management access ports is open on the tested device.
Breaking Encrypted Traffic	Apply known/available techniques of breaking encrypted traffic. For example, try to redirect HTTPS to HTTP traffic (SSL Strip) or impersonate remote servers with self-certificates (to apply a man-in-the-middle attack).	<i>Pass</i> - unable to decrypt traffic sent/received by/to the IoT-DUT with the applied techniques; or, <i>Fail</i> - able to decrypt traffic data sent/received by/to the IoT-DUT using the applied techniques.
Spoofing/ Masquerade Attacks	Attempt to generate communication on behalf of the tested IoT device. For example, determine if any of the communication types can be replayed to the external server.	<i>Pass</i> - reply attack failed; or, <i>Fail</i> - replay attack successful.
Communication Delay Attacks	Delay the delivery of traffic between the device and remote server without changing its data content. Determine which maximal delays are tolerated on both ends.	<i>Safe</i> - the time delay between two consecutive transactions of the IoT-DUT is within the <i>defined/normal</i> range; or, <i>Unsafe</i> - the time delay is greater than the <i>defined/normal</i> range.
Communication Tampering	Attempt to selectively manipulate or block data sent to/from the device. For example, inject bit errors on different communication layers or apply varying levels of noise on the wireless channel.	<i>Safe</i> - the device ignores received manipulated/erroneous data; or, <i>Unsafe</i> - the device crashes or behaves unexpectedly when manipulated/erroneous data is sent.
List Known Vulnerabilities	Given the type, brand, version of the device, running services, and installed applications—list all known vulnerabilities that could be exploited.	<i>Safe</i> - no relevant vulnerabilities were found; <i>Minor risk</i> - insignificant/low risk vulnerabilities were found; or, <i>Unsafe</i> - significant and critical vulnerabilities were found.
Vulnerability Scan	Search for additional classes of vulnerabilities by: (1) utilizing existing tools (or developing new dedicated tools as part of the ongoing research) that attempt to detect undocumented vulnerabilities such as buffer overflow and SQL injection; (2) maintaining a database of attacks (exploits) detected on previously tested IoTs or detected by honeypots, and evaluate relevant/selected attacks on the tested IoT; and (3) using automated tools for code scanning, such as the fuzzing testing technique.	<i>Safe</i> - no new vulnerabilities were found during the testing process conducted; <i>Minor risk</i> - new insignificant/low risk vulnerabilities were found; or, <i>Unsafe</i> - new significant and critical vulnerabilities were found.

TABLE II
SIMULATORS SUPPORTED BY THE SECURITY TESTBED

Simulator	Description
Network	The testbed uses network simulators to simulate different network environments, such as Wi-Fi, Bluetooth, ZigBee, and more, in order to support different network connectivity in the testbed.
Location	The testbed simulates different locations and trajectories using the GPS generator device, in order to test the behavior of the IoT device under test in different locations/trajectories.
Time	The testbed simulates different days of the week and times of day using either the GPS generator device, internal NTP server, or internal cellular network, in order to test the behavior of the IoT device under test at different times.
Movement	The testbed simulates different movements using either robots or human testers, in order to test the behavior of the IoT device under test while performing different movements.
Lighting	The testbed simulates different lighting levels, in order to test the behavior of the IoT device under test in different lighting scenarios.
Audio	The testbed simulates audio using a voice simulator, in order to test the behavior of the IoT device under test in different sound environments.

cess criterion. Note that most of the predefined success criteria are not generic and are defined for a specific tested IoT device and/or tested scenario. In some cases, a success criterion cannot be clearly defined, and therefore, advanced analysis tools and mechanisms will be deployed in the testbed (for example, a network-based anomaly detection tool will be employed to process the recorded network traffic of the tested IoT device in order to detect anomalous events in the system). In this case, the pass/fail decision will be based on a predefined threshold provided by the system operator in advance. The detected anomalies should then be investigated and interpreted by the system operator using dedicated exploration tools that are part of the user interface.

5) *Testing Process*: The testing process shown in Fig. 1 starts by loading a configuration file (by the user/testbed operator) in the testbed via the MRM component. Based on the configuration loaded, a set of security testing is conducted in the testbed (indicated by the red line in Fig. 1) using the STM component. The results are then stored in the system database component. Next, context-based security testing is performed using the STM component (indicated by the black dashed line in Fig. 1), by selecting the appropriate simulators for the test. In this phase, different simulators are employed in order to realistically simulate the environment in which IoT devices operate, and the same set of security tests are conducted (again, based on the configuration file loaded in advance). The results obtained are then stored in the system database component. Both of these testing phases are controlled by the STMM component. Note that during the execution of the testing process, 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.). Finally, a forensic analysis is performed by the MRM com-

ponent, based on the results obtained from both phases and the information collected during the testing process. The final results of the overall testing process are then generated and sent to the user/testbed operator (indicated by the green dashed line in Fig. 1).

V. PRACTICAL TESTBED IMPLEMENTATION

In this section, the practical testbed implementation is presented, including a detailed description of the system's structure and components and the testbed's infrastructure.

A. System Structure and Components

The testbed environment, illustrated in Fig. 2, includes both software and hardware system components. From the internal software system component perspective, this includes the user interface and several testbed manager modules, each responsible for a specific task. From the environmental system component point of view, this includes the IoT DUT (IoT-DUT), the set of security test tools, measurement, and analysis tools, and a set of simulator/stimulator devices employed in the testbed.

1) *Internal Software System Components*: The internal software system components of the security testbed include the user interface (GUI/Remote), testbed manager, test manager, element manager, and storage manager elements.

- 1) *User interface—GUI/Remote*: The user interface component is used for sending and receiving commands and test results to/from the testbed, respectively. This can be handled locally (e.g., using a GUI) or remotely (e.g., via REST API). SSH and Telnet connectivity are supported as well.
- 2) *Testbed manager*: The testbed manager component acts as an orchestrator in the system. It is responsible for managing the workflow between the software system components of the testbed (including the underlying managers: element manager, test manager, etc.), as well as the hardware system components, and the user interface.
- 3) *Test manager*: The test manager component is responsible for the creation and execution of testing scenarios. A scenario defines a testing process in the testbed, including creation and execution of security tests, each composed of a set of security testing actions. In addition, the test manager enables to generate templates of testing scenarios for future use.
- 4) *Element manager*: The element manager component is responsible for provisioning and deleting elements from the testbed. An element is a general term used in the testbed that applies to both software and hardware. Each element is defined by its driver. A driver is a programmable component that exposes the element's capabilities, either to the user or to other elements of the testbed. Examples of types of elements used in the testbed are: IoT-DUTs, simulators/stimulators, measurement and analysis tools, and security tests.
- 5) *Storage manager*: The storage manager component is a repository of system elements. In addition, it is responsible for logging different events occurring in the sys-

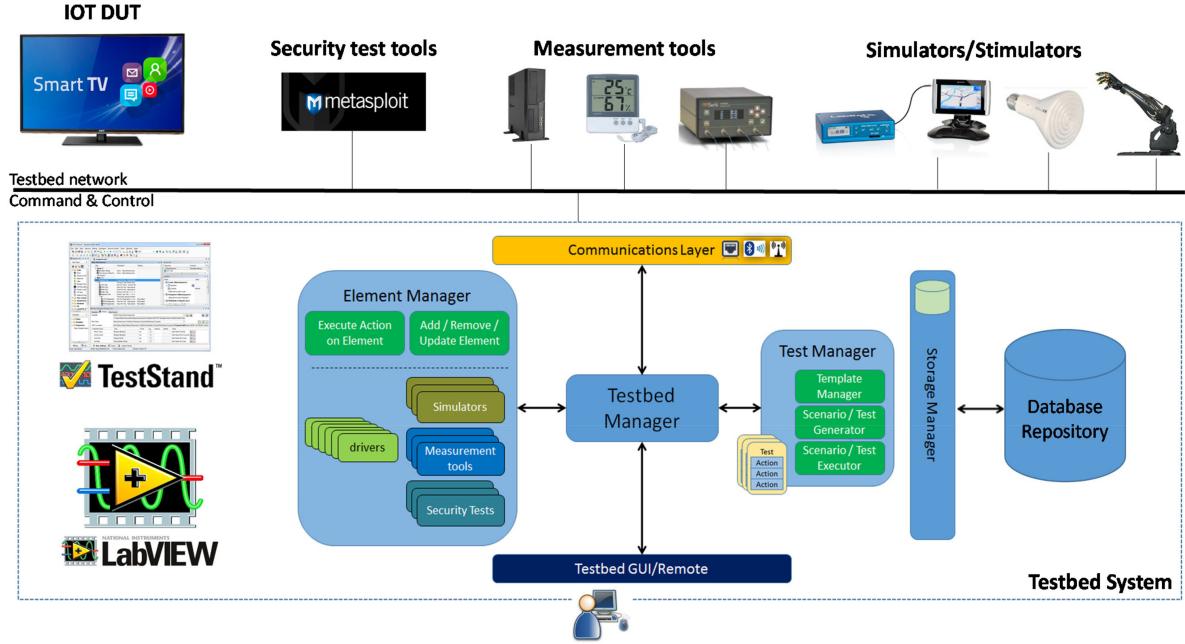


Fig. 2. IoT security testbed system components.

tem, before, during, and after the test is conducted (e.g., registering simulator, driver event, test action being run, test results, etc.).

2) *Environmental System Components*: The environmental system components include both hardware and software components, including: the IoT-DUT, a set of security tools, environmental simulators and stimulators, and different types of measurement and analysis tools, as discussed next.

- 1) *IoT-DUT*: The security testbed is designed and implemented to support examination of a wide range of IoT devices, including different categories such as: smart home appliances, smart industrial equipment, smart city devices, wearable devices, and more.
- 2) *Security test tools*: The security testbed utilizes different security testing tools available online, including the Nmap security scanner tool for the network discovery and security auditing [39], the Wireshark tool for network protocol analysis [40], Aircrack-ng [41] to assess Wi-Fi networks, and Metasploit, which is used for penetration testing [42]; all of these tools run under the Kali Linux penetration testing environment [43]. Other security tools, such as Nessus [44], OpenVAS [45], Cain and Abel [46], and OS-SEC [47], can be employed in the testbed as well.
- 3) *Measurement and analysis tools*: The security testbed uses different types of measurement and analysis tools, including: data collection modules, analysis and security rating modules, data analysis modules, and more. These modules are developed in order to enhance the testbed capabilities; for example, the anomaly detection model is used in the testbed in order to automatically identify and detect anomalies in the network traffic of the IoT-DUTs.
- 4) *Simulators and stimulators*: The security testbed employs different types of environmental simulators and stimulators (e.g., a GPS simulator that simulates different locations and trajectories, movement simulators such as

robotic hands, etc.). Using the set of simulators (simulator array), the testbed realistically generates arbitrary real-time stimulations, ideally for all of the sensors of the tested IoT devices. See Table II for a list of the simulators supported by the testbed.

B. Testbed Infrastructure Implementation

Due to the diversified nature of the IoT, it is a huge challenge to develop a generalized security analysis testbed. The capabilities of the testbed need to cover various communication standards such as Wi-Fi, Bluetooth, Zigbee, etc., and also need to address various other issues related to other protocols (focusing on IoT communication and applications protocols). In addition, the system on chip used by IoT devices varies and their functionalities range from simple to complex. Hence, to have the ability to test any IoT device, regardless of its capabilities and specs is a challenge. Furthermore, our testbed is just not focused on a single domain (such as a testbed dedicated to wireless sensor networks); we designed and developed a multidomain testbed to test various types of IoT devices. Moreover, our testing capabilities are not just limited to penetration testing, but also cover various other analysis methods (for example, our tools for anomaly detection, IoT DoS resilience tool, etc.) making our IoT testbed even more innovative and able to solve other challenging issues. These innovative security testing tools demonstrate the challenges and complexity of testing the security of IoT devices. As for the future, we plan to make the use of testbed capabilities available outside our lab, in the outside world where the testbed can be used as a service by any individual user or enterprise to evaluate their IoT environment and obtain a metric score of the environment's level of security as it relates to the IoT.

We deployed the testbed system in a shielded room inside our lab (shown in Fig. 3), which provides a testing environment with minimal external disruptions. The IoT testbed setup consists



Fig. 3. Shielded room setup in the iTrust lab at SUTD.

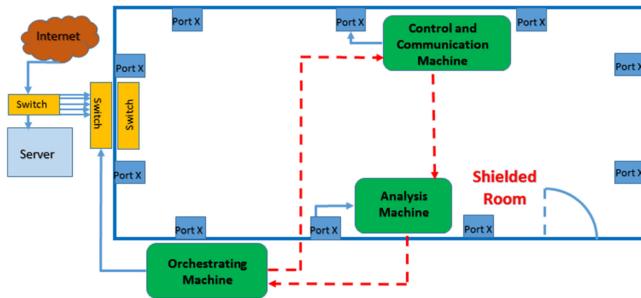


Fig. 4. Physical design of the testbed with the OM, CCM, and AM.

primarily of three machines that 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 our comprehensive testbed setup.

We established an access point within the shielded room, to ensure that all of 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, and maintain project details.

The three machines are as follows. 1) The orchestrating machine (OM) is located outside the shielded room. The OM runs National Instruments' (NI) TestStand [48], 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 [49], 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, such as Nmap, Wireshark, etc., needed to support various test cases. All three machines are interconnected and can speak to each other. The physical design of the testbed is as shown in Fig. 4.

The security testing tools and mechanisms used in the security testbed (both the open source tools, such as Metasploit, and the mechanisms we developed) generate reports in various data formats such as *.csv, *.txt etc.; the format of the report depends on the tool used. The OM consolidates all of the reports to generate a single report in one data format (in our case, a *.pdf

format of the report), which is accomplished by our own parser. The user has access to the report at any time and can also get the report in various other data formats.

C. Testbed Operation

In general, the OM (running TestStand and the MRM) starts the test. More specifically, there are a sequence of steps written in TestStand that initiates the test by asking the CCM (running LabVIEW and the 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; the results will consist of 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 select 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, the AM will 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 view the detailed report. The detailed report is displayed on the OM. Since the report is present on the local server, the user can access the report anytime.

VI. SECURITY ANALYSIS USING THE TESTBED

In this section, we describe the testbed operation for several IoT use cases. We demonstrate the use of the testbed for conducting both standard and advanced security testing.

A. Standard Security Tests

In this section, several standard security testing scenarios based on vulnerability scans and penetration test methodology are performed using the security testbed, in order to assess and verify the security level of IoT devices under test.

1) Test Scenario 1. Security Testing for IoT Devices: The security analysis is conducted via the testbed and by considering the requirements and architecture explained in Sections IV and V. We have chosen several IoT devices to be tested in the testbed in this phase, such as Amazon Echo, Nest Cam, Philips Hue, SENSE Mother, Samsung SmartThings, Withings HOME, WeMo Smart Crock-Pot, Netatmo Security Camera, Logitech Circle, D-Link Camera, and HP Printer. Four use cases for security testing [50], i.e., port scanning, fingerprinting, process enumeration, and vulnerability scanning, were conducted as follows.

a) 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 previously, the AM will run Nmap to discover the open ports via the SSH setup on a 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 [51] 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.

We have established a metric score based on [51] 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 contain 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: 80—A web server was running on this port with a score of 3, 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 determines the risk from a predefined risk margin. In the case of the Samsung SmartThings home monitoring kit, the risk level is safe and the metric score is 3; the detailed report is shown in Fig. 5.

b) Fingerprinting: The goal of fingerprinting is to identify the device’s IP and MAC addresses, as well as the type of device, manufacturer, operating system, etc., by monitoring communication traffic to/from the device.

In order to successfully fingerprint for a specified IoT device, the AM uses Nmap, dhcpcdump, and the Scapy Python library. We performed fingerprinting for every IoT device mentioned previously, 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 that 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 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. While the dhcpcdump process is still running, the deauth() function is tasked with forc-

Port Scanning Results

All available ports discovered:

```
112/udp open|filtered mcidas
136/udp open|filtered profile
5353/udp open|filtered zeroconf
407/udp open|filtered timbuktu
9370/udp open|filtered unknown
17219/udp open|filtered chipper
18985/udp open|filtered unknown
20120/udp open|filtered unknown
20279/udp open|filtered unknown
20560/udp open|filtered unknown
21131/udp open|filtered unknown
21568/udp open|filtered unknown
22341/udp open|filtered unknown
34422/udp open|filtered unknown
42431/udp open|filtered unknown
```

Ports that are considered vulnerable:

5353: Multicast DNS (mDNS)

Score: 1

Risk Level:

Safe

Metric Score:

1

Fig. 5. Port scanning report for the Samsung SmartThings home kit.

ing DHCP requests, which will result in inputs for the dhcpResults.txt file. The deauth2.py uses a Scapy Python library that 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 [52] 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 result of this entire process is contained in an output file called dhcp_fingerprinting_results.html. The fingerprinting report shown in Fig. 6 is for the Nest Cam IoT device.

c) 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

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:
Nest Cam IP Camera

Fig. 6. Fingerprinting report for the nest cam device.

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

Service: upnp
State: open|filtered
Port Number: 1900
Protocol: udp

Fig. 7. Process enumeration reports for the philips hue, withings home, Samung SmartThings, Amazon echo, and D-link camera.

enumeration the AM runs the nmapScan Python script, which conducts an intense scan on the selected IoT device to reveal any open user datagram protocol (UDP) or transmission control protocol (TCP) ports. We performed process enumeration for all of the IoT devices mentioned previously, however, the report

Vulnerability Scanning Results

CVE Number:
CVE-2008-2940

Description

The alert-mailing implementation in HP Linux Imaging and Printing (HPLIP) 1.6.7 allows local users to gain privileges and send e-mail messages from the root account via vectors related to the setalerts message, and lack of validation of the device URI associated with an event message.

Impact

CVSS Severity (version 2.0):
CVSS v2 Base Score:
7.2 HIGH

Vector:

(AV:L/AC:L/Au:N/C:C/I:C/A:C) (legend)

Impact Subscore: 10.0

Exploitability Subscore: 3.9

CVSS Version 2 Metrics:
Access Vector: Locally exploitable

Access Complexity: Low

Authentication: Not required to exploit

Impact Type: Allows unauthorized disclosure of information; Allows unauthorized modification; Allows disruption of service.

Fig. 8. Vulnerability scan report for the HP printer.

presented in this paper is based on the following devices: Philips Hue, Withings HOME, Samsung SmartThings, Amazon Echo, and D-Link Camera.

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, the output can be formatted into HTML format, with the different services highlighted. Finally, the results are provided as an output file in ProcessEnumerationResults.html.

The results only contain the known ports, ignoring the unknown ports as their vulnerabilities are also unknown. Fig. 7 contains a process enumeration report for the following IoT

TABLE III
OVERALL RESULTS OF SECURITY ANALYSIS WITH SELECTED IoT DEVICES

IoT Device	Port Scanning (RL-Risk Level, MS-Metric Score)	Fingerprinting (IP, OS)	Process Enumeration (SR-Service Running, Port, P-Protocol)	Vulnerability Scanning (IS-Impact Subscore, ES-Exploitability Subscore)
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: tcprwrapped, 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: 4.6
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
Netatmo 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: svrlc, Port: 427, P: udp	IS: 6.4, ES: 5.5
Logitec Circle	RL: Minor Risk, MS: 9	IP: 192.168.2.121, OS: Unknown	SR: http-rpc-epmap, Port: 593, P: udp	IS: 6.3, ES: 4.9
D-Link Camera	RL: Major Risk, MS: 12	IP: 192.168.2.165, OS: Unknown	SR: upnp, Port: 1900, P: udp	IS: 6.3, ES: 3.1
HP Printer	RL: Minor Risk, MS: 6	IP: 192.168.2.102, OS: Linux Kernel	SR: commplex-link, Port: 5001, P: udp	IS: 10.0, ES: 3.9

devices: Philips Hue, Withings HOME, Samsung SmartThings, Amazon Echo, and D-Link camera.

d) Vulnerability scan: The goal of vulnerability scanning is to search for additional classes of vulnerabilities by understanding and measuring the CVE and CVSS [53]. The NVD [53] has been maintaining a list of vulnerabilities from 2005 onwards, including metric scores that helps us determine impact and exploitability subscores, 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 a vulnerability scan, a fingerprinting output (i.e., the OS) is provided as input. We ran the vulnerability scan for each IoT device mentioned previously, however, the report presented in this paper is based on the HP printer.

The checkCVE function utilizes multiple Python libraries to check the vulnerabilities from [53]. The querier() function creates a string that contains appropriate HTML formatting, and then, opens the allitems2005.csv, which contains all of the CVE and vulnerabilities from the year 2005. The function querier() also goes through the CSV file line by line and searches for 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. Fig. 8 presents the report for the HP printer.

Table III presents an overview of the test results for each of the IoT devices tested so far. Our testing efforts and findings for the selected IoT devices have demonstrated the vulnerability level of IoT devices.

2) Test Scenario 2. Fuzzing for IoT Devices: The increasing demand for improved cyber security protection has necessitated the involvement of the penetration testing field. Fuzz testing, or fuzzing, is an advanced and popular pen testing technique. In order to reveal unknown vulnerabilities and security holes in a software program, the fuzzer (fuzzing tool) sends malformed input data to the system under test (SUT). Fuzzing can be performed on a variety of input types, including protocols, file format, etc. In general, the fuzzing process includes input interface identification (the target to be tested), test case generation, connecting and fuzzing the SUT, and monitoring for exceptions in order to identify abnormal behavior of the SUT due to the fuzzing process.

In this section, we perform fuzz testing on several IoT devices using the proposed security testbed. Fuzz testing is conducted

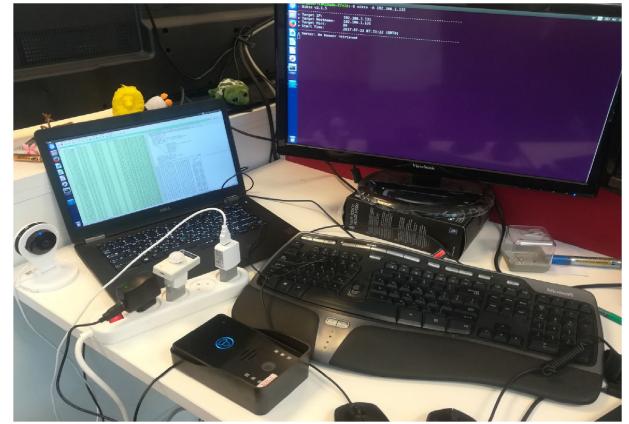


Fig. 9. Fuzzing setup environment in the security testbed.

as part of the standard vulnerability testing process presented in Section IV, and is based on the scanning information gathered in the testbed.

Regardless of whether an attacker's aim is to expose the user name and password, disrupt the normal activity of the IoT device, or to achieve any other malicious goal, the protocol input type is the first input interface that a remote attacker can exploit. Therefore, in this scenario, we focused on finding automatic vulnerabilities in several IoT devices by utilizing protocol fuzzers.

The fuzzing process conducted in the security testbed is as follows. We connected several IoT devices to the testbed environment, including the Ennio doorbell, Proteus motion detector, and Provision ISR security camera, as shown in Fig. 9. The fuzz testing is based on the information that was collected by the scanning capabilities of the testbed (e.g., port scanning, OS fingerprinting, etc.). After connecting the tested IoT device (IoT-DUT) to the testbed, we let it run without any interference with its natural behavior. Several minutes after this, we ran the automatic fuzzing tool. According to the configuration of the testbed, the fuzzer tested and monitored the device using the fuzzer's functionality. When the fuzzer finished its testing and presented the results that were calculated based on the tool's monitoring process, the testbed allowed the IoT-DUT to run for several more minutes, again without any interference. After examining the fuzzer's output, we reproduced the errors reported in order to better understand the results and filtered out false positive alerts.

Sta...	Me...	Headers	Cookies	Params	Response	Timings	Preview						
500	GET	Request URL: http://192.168.1.134:81/											
Request method: GET													
Remote address: 192.168.1.134:81													
Status code: 500 Web Error [Learn More] Edit and Resend Raw headers													
Version: HTTP/1.1													
Filter headers													
Response headers (148 B)													
Server: "GoAhead-Webs"													
Date: "Tue Jul 15 06:46:35 2014"													
Pragma: "no-cache"													
Cache-Control: "no-cache"													
Content-Type: "text/html"													
Request headers (328 B)													
Host: "192.168.1.134:81"													
User-Agent: "Mozilla/5.0 (X11; Ubuntu; Lin... Gecko/20100101 Firefox/54.0"													
Accept: "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"													
Accept-Language: "en-US,en;q=0.5"													
Accept-Encoding: "gzip, deflate"													
Connection: "keep-alive"													
Upgrade-Insecure-Requests: "1"													

Fig. 10. Error 500 shown from fuzz testing the Ennio doorbell.

The fuzzer we chose to work with is Nikto [54], a fully automated tool that conducts an extensive test on a web server by sending packets to the SUT, in our case, IoT device, and monitors the network. Like other HTTP-based fuzzers, Nikto performs its testing by sending various types of user data to the SUT in order to find vulnerabilities, but unlike the traditional fuzzers, it sends a fixed set of packets instead of malformed input every time. By sending different packets to the tested device, Nikto can identify dangerous files and common gateway interface, outdated and vulnerable server versions, and more.

The Ennio Doorbell was the first device to be tested. After receiving the information from the scanning process in the testbed for that device, we learned that a GoAhead HTTP web server was running behind port 81. Nikto discovered the web server correctly. After examining the tool's log file, we found that the device is vulnerable to clickjacking and also found that a citrine GET request (%5c/) causes error 500. We successfully reproduced both scenarios and showed that although most GET requests to the device cause error 401, the %5c command causes error 500 ("invalid URL"), as shown in Fig. 10.

The Proteus Motion Sensor device was also tested in the testbed as part of the fuzzing process. Based on the scanning test results, we discovered that a tcpwrapped (HTTP) service is running on port 80. The testing process informed us that the device's response contained an uncommon header (access-control-allow-origin), and even though this is not a security hole, we confirmed this information by checking the packets on Wireshark. Based on the fuzzer's output, we also learned that the device is vulnerable to clickjacking, and we were able to confirm this. We believe that in this case clickjacking can be used to imitate the sensor's clean web application for malicious purposes. By embedding the real device's GUI in an iframe on an external website, the attacker can perform a phishing attack.

The Provision ISR T737E camera was also tested in the testbed. The information from the scanning test showed that a GoAhead server (HTTP) was running on this device. Nikto found the same information and reported it as was done in the doorbell testing. The fuzzer also showed that the web application is vulnerable to clickjacking, and we were able to confirm this as well. The fuzz testing also showed a default user name

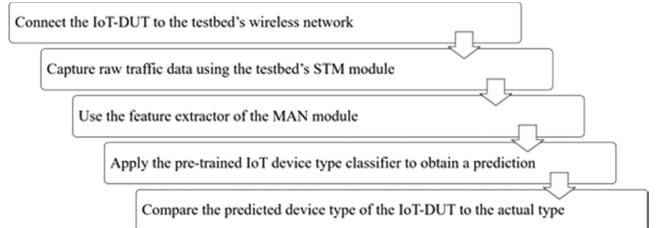


Fig. 11. Functionality of the testbed's IoT-device-type identification module.

and password for a specific path, however, we failed to connect with the given credentials. These results are important, because this shows that fuzzing tools can make mistakes. The testbed can identify those mistakes, and by reproducing the reported vulnerability, the testbed can ensure that this issue really exists. Such false positive filtering is an important capability that increases trust in the security system.

B. Advanced Security Tests

In this section, various advanced security testing scenarios, demonstrate the capabilities of the advanced analysis plugins embedded in the security testbed, are performed. This obtained using our self-developed mechanisms, including device identification and anomaly detection mechanisms based on machine learning algorithms, a security test for checking resilience to DoS attacks of IoT devices, and checking the password complexity of the management connection of the IoT devices under test.

1) Test Scenario 1. IoT-Device-Type Identification: The ever growing variety of commercially available IoT devices includes smart doorbells, smoke detectors, TVs, refrigerators, humidity sensors, glasses, light bulbs, speakers, watches, thermostats, and many other types of devices. In order to allow the user to interact with them, the devices usually have to be connected to the Internet, often via Wi-Fi. In this test scenario, we demonstrate how the traffic data emanating from IoT-DUTs connected to the testbed's network via Wi-Fi can be leveraged for the security testing. More specifically, as part of providing security analysis for a given IoT device, we analyze its traffic data and perform traffic-based IoT-device-type identification. For a connected IoT-DUT in the testbed, we address the following question: Does the IoT-DUT behave like a device of its type or does its network behavior resemble that of a different (possibly unknown) IoT device type? A mismatch between the actual and predicted device type might indicate that the IoT-DUT is compromised, e.g., deliberately forced by a hacker to behave like a different IoT device type in order to bypass a traffic-based blacklisting or whitelisting system.

Traffic-based identification of the type of the connected IoT device is a task supported by the STM and MAM modules of our security testbed. In this task, portrayed in Fig. 11, the IoT-DUT first connects (often wirelessly) to the testbed's network. Then, the connected IoT-DUT operates normally for a given duration, e.g., a smart TV may be switched ON, and have its channel changed and the volume adjusted. Using the STM module of the testbed, the resulting network traffic data are

TABLE IV
IoT DEVICES USED TO TRAIN THE DEVICE TYPE CLASSIFIER

Type of device	Number of devices	Quantities, manufacturers, and models of devices	Number of manufacturers	Number of models	Number of TCP sessions as a client	Number of recorded days
Baby_monitor	1	1 X Beseye Baby Monitor Pro Security System	1	1	51,578	9
Motion_sensor	2	2 X D-Link DCH-S150	1	1	3,834	108
Refrigerator	1	1 X Samsung RF30HSMRTSL	1	1	1,018,921	74
Security_camera	3	2 X SimpleHome XCS7-1001-WHT	2	2	14,394	70
		1 X Withings WBP02 WT9510				
Smoke_detector	1	1 X Nest Protect	1	1	369	56
Socket	2	2 X SimpleHome XWS7-1001-WHT	1	1	2,808,876	114
Thermostat	1	1 X Nest Learning Thermostat – 3 rd generation	1	1	19,015	52
TV	2	1 X Samsung UA40H6300AR	1	2	144,205	73
		1 X Samsung UA55J5500AKXXS				
Watch	4	1 X LG G Watch R W110	2	3	4,391	65
		2 X LG Watch Urbane				
		1 X Sony SmartWatch 3 SWR50				

captured and saved as a *.pcap file. Then, in order to structure this raw captured data, a feature extractor [55] is employed in the MAM module of the testbed. Given a *.pcap file, it reconstructs TCP/IP sessions, defined as 4-tuples consisting of IP addresses and port numbers of the source and destination, from SYN to FIN. The feature extractor describes each session by nearly 300 session-level features, such as the interarrival time of packets (minimum, maximum, average, variance, and entropy), ratio between incoming and outgoing bytes, time to live statistics, session duration, and more. Finally, the (structured) data are processed by a device type classifier, trained in advance on the network traffic collected in the testbed from the devices in Table IV.

Given a set of IoT device types (shown in Table IV) and a structured set of labeled traffic data, we treat the task of IoT-device-type identification as a multiclass classification problem. That is, we wish to label each IP stream with the type of the IoT device that is most likely to have produced it. To achieve this, we employ the random forest supervised machine learning algorithm for training a classifier. This is conducted by the STM, where network traffic data are first collected, and MAM modules of the testbed, in order to generate appropriate models in advance to use in the testing phase of this test scenario. The classifier is comprised of the following two steps: 1) session-level classification, where a multiclass classifier is used to classify one session at a time; and 2) sequence-level classification, which performs a majority vote on a sequence of session classifications to make a final decision regarding the IoT device type.

In step 1 (the session-level classification of the classifier operation), we train a random forest multiclass classifier on a training set. Then, we optimize its classification threshold, denoted as tr , such that it maximizes the F-measure [see (1)] on a separate validation set. This traditional metric, also known as the balanced F-score or F1 score, ranges from 0 (the worst value for the harmonic mean of precision and recall) to 1 (best value, attained when both recall and precision are high).

$$F = 2 \cdot \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}. \quad (1)$$

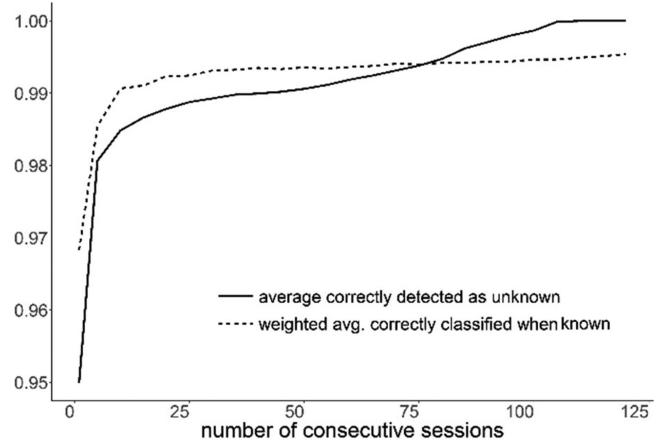


Fig. 12. Classification accuracy with the test set as a function of the length of the session sequence used for majority voting.

When applied to a single session, the classifier outputs a vector of posterior probabilities P , whose length is equal to the number of IoT device types in the training set. Each probability $p_i \in P$ denotes the likelihood of the inspected session to originate from the respective device type. Given the probability vector P , if there exists any $p_i > tr^*$, then the session is classified as originating from device i which maximizes P . Otherwise, the session is classified as “unknown.” In step 2 (the sequence-level classification of the classifier operation), the length of the sequence is optimized to reach the maximum accuracy on the validation set with a minimal sequence length.

For empirical evaluation of the testbed’s IoT-device-type identification plugin, we conducted nine experiments, corresponding to the nine device types available in our testbed. In each experiment, one IoT device type was left out of the training and validation sets, to represent an unknown IoT device type. Then, a classifier was trained on the remaining (known) IoT device types while optimizing tr for the single-session classifier and the sequence length for the majority vote. Fig. 12 and Table V summarize the nine experiments we conducted. In Fig. 12, it can be seen how longer sequences of consecutive

TABLE V
PERFORMANCE WITH THE TEST SET (CLASSIFICATION BASED ON SEQUENCES OF 20 SESSIONS EACH)

Device type left out	Number of sessions	tr^*	Weighted avg. correctly classified when known	Correctly detected as unknown
Baby monitor	1,981	0.41	1	1
Smoke detector	104	0.46	1	1
Socket	1,962	0.52	1	1
TV	1,962	0.54	0.98	0.84
Refrigerator	1,981	0.54	1	0.99
Thermostat	1,981	0.55	1	1
Motion sensor	1,239	0.68	0.99	1
Security camera	1,375	0.6	0.99	0.94
Watch	1,111	0.84	0.97	0.84
Average			0.99	0.96
Standard deviation			0.01	0.07

sessions lead to a higher classification accuracy of both known and unknown IoT device types. At the same time, however, longer sequences directly translate into longer detection times. Hence, we chose the sequence length of 20 sessions as the point that optimizes the tradeoff between classification accuracy and speed. Based on this sequence length and the optimized classification threshold tr^* for each device type, Table V shows the classification accuracies obtained with the test set. Overall, these results demonstrate the feasibility of our approach for the IoT-device-type identification: classification accuracy of 0.99 ± 0.01 for known IoT device types and 0.96 ± 0.07 for unknown types.

The final classification results obtained in the MAM module are eventually sent to the database of the testbed system in the MRM module. Then, the testbed generates the final report for this test scenario for the user. It also enables the comparison between the IoT device type predicted by the model and the actual type.

2) *Test Scenario 2. Automatic Anomaly Detection:* The always-connected nature of IoT devices, in addition to their inherent computational weakness, makes them especially vulnerable to breaches from outside attackers or compromised devices sharing the same network, potentially exposing all of the network nodes and the data they may hold to cyberattacks. In this test scenario, we demonstrate the anomaly detection capability of the security testbed as an advanced analysis module. The main goal of this module is to detect compromised IoT devices in advance, based on their traffic data alone. The entire process of this module, from the network scanning to anomaly detection conducted in the testbed, is described next and illustrated in Fig. 13.

For the testbed's anomaly detection operation, we first utilize the scanning test supported by the security testbed's STM module. More specifically, we employ a passive monitoring capability of the scanning test, by which network data traffic from the IoT-DUT is collected without any interference. This is different from active monitoring, where data about the configuration of a connected IoT device is collected by interacting with it, for example, discovering open ports with Nmap, as shown in Section VI-A Test Scenario 1. For anomaly detection, the network traffic data are passively monitored using the Wireshark [40] tool, which is embedded in the testbed (this tool is run by the testbed's MAM module). The network traffic is then saved

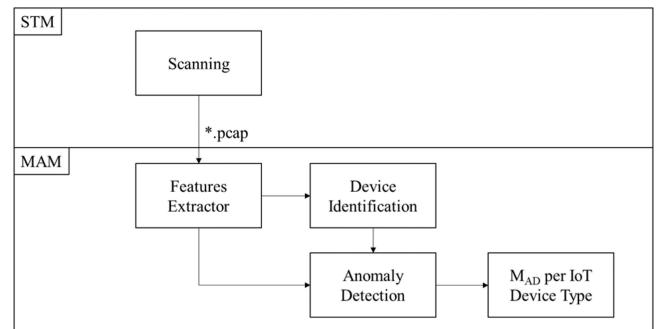


Fig. 13. The anomaly detection process in the security testbed.

as a *.pcap file and used as the raw data for the anomaly detection model of the testbed. To structure the raw network traffic data collected such that it can be processed by machine learning algorithms, we use the same process presented in the device-type identification process, (Test Scenario 1) using the STM and MAM modules of the testbed. In essence, the raw traffic data, stored in pcap format, is processed by a feature extractor tool [55], which reconstructs the TCP sessions, and then, extracts an abundance of session-level features. The original vector for each session includes 274 features. For anomaly detection, we only select the most valuable features in terms of information gain; see Table VI for the set of the 20 most valuable features extracted.

In order to train an anomaly detector for each IoT device type, we use the one-class support-vector-machine algorithm, implemented in Python's Scikit Learn library [56]. At this stage, referred to as the learning (or training) process, only benign traffic is used in the testbed so that future deviations from normal behavior are likely to be detected as anomalous. Similarly to the algorithm of the IoT-device-type classification described in Test Scenario 1, we designed the anomaly detector to operate in the following two steps: in step (1), session-level anomaly scoring and categorization (i.e., is the session normal or anomalous?) is conducted, followed by step (2), where a majority vote on a sequence of session anomaly classifications is obtained to improve the accuracy.

When in operation mode (as opposed to training mode), before looking for anomalous patterns in the traffic data of an

TABLE VI
THE 20 MOST VALUABLE FEATURES IN TERMS OF INFORMATION GAIN

Feature	Description
ttl B min	TCP packet time-to-live sent by server
packets	Total packets
packets' A B ratio	Ratio between packets sent by client and sent by server
packet size firstQ	Packet's size of the first quartile
bytes A B ratio	Ratio between number of bytes sent and number of bytes received
packet size A sum	Packet's size sent by the client - sum
packet inter-arrival min	Packet's inter-arrival time – minimum
packet inter-arrival sum	Packet's inter-arrival time - sum
packet inter-arrival median	Packet's inter-arrival time - median
ttl B firstQ	First quartile of TCP packet time-to-live sent by server
duration	Session duration
bytes A	Number of bytes sent by client
ttl stdev	Standard deviation of TCP packet time-to-live
packet size A stdev	Standard deviation of client's packets size
bytes	Number of bytes sent and received
ttl min	TCP packet time-to-live - minimum
ttl B entropy	Entropy of TCP packet time-to-live sent by server
packet size A thirdQ	Client packet's size sent by the third quartile
bytes B	Number of bytes sent by server
ttl B median	TCP packet time-to-live sent by server - median

IoT-DUT, it is important to first identify its device type, and only then, apply the corresponding anomaly detection model that has been trained specifically for this device type. Accordingly, an IoT-device-type identification operation is conducted in the testbed as a preliminary process, as presented in Section VI-B Test Scenario 1. In practice, the testbed's STMM executes the scanning test in the STM to collect the network traffic data produced from the IoT-DUT. Once enough data are captured (e.g., 10 min of device operation), the feature extractor is executed on the collected data, and device-type identification is performed in order to select the proper model to apply for anomaly detection in the MAM.

To evaluate the performance of the anomaly detection module described previously, which is operated as a plugin by the testbed, we used an IP camera (SimpleHome XCS7-1001-WHT; 5,573 collected sessions) and a smartwatch device (Sony SmartWatch 3 SWR50; 2,005 collected sessions). Accordingly, we trained two respective anomaly detectors on these devices' benign data, and then, compromised the devices under test (the IP camera and the smartwatch device) to test the ability of our anomaly detectors to identify them as compromised devices. In order to compromise the smartwatch device, we gained shell access via its Android Debug Bridge (ADB) connectivity. The IP camera provides Telnet access with default credentials, allowing us remote control. Having gained shell access, we infected

TABLE VII
TPR AND TNR FOR THE IP CAMERA

# Sessions	TNR	TPR
1	82.5%	100%
3	91.2%	100%
5	95.8%	100%
7	98%	100%
9	99.1%	100%
10	100%	100%

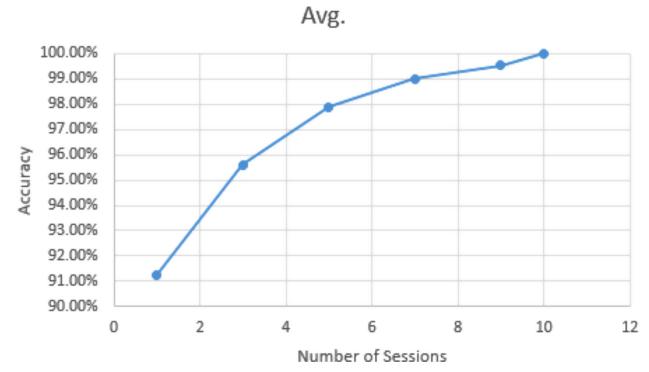


Fig. 14. TNR for the IP camera.

TABLE VIII
TPR AND TNR FOR THE SMARTWATCH

# Sessions	TNR	TPR
1	87%	85%
3	90%	87.5%
5	92%	90.5%
7	97.5%	92.5%
15	100%	98%
31	100%	100%

the IP camera and the smartwatch device with Gafgyt's C&C malware binaries collected from the IoTPOT dataset [57], and adjusted it to operate locally in our testbed. In addition, a version of Nmap that had been adjusted to Android Wear OS was used to perform portscanning and was executed via the smartwatch device to simulate a port scan attack, as shown in [7]. In the test set we used 1000 TCP sessions from each of the compromised devices. Table VII presents the TPR and TNR for the IP camera anomaly detection model as a function of the length of the session sequence for majority voting. As can be seen in Table VII, the malware-infected IP camera was immediately detected as anomalous (100% TPR), based on a single session. In addition, a sequence of ten consecutive sessions was sufficient to perfectly identify benign traffic as such (100% TNR), as can be seen in Fig. 14.

Similarly, Table VIII contains the TPR and TNR results obtained by applying the anomaly detection model on the smartwatch device. For perfect accuracy (TPR and TNR), a majority vote had to be performed on a sequence of 31 consecutive sessions, as can be seen in Fig. 15.

3) *Test Scenario 3. Resilience of Internet of Things Against DoS:* The IoT exposes various vulnerabilities at different levels. One such exploitable vulnerability is DoS. As a case in point,

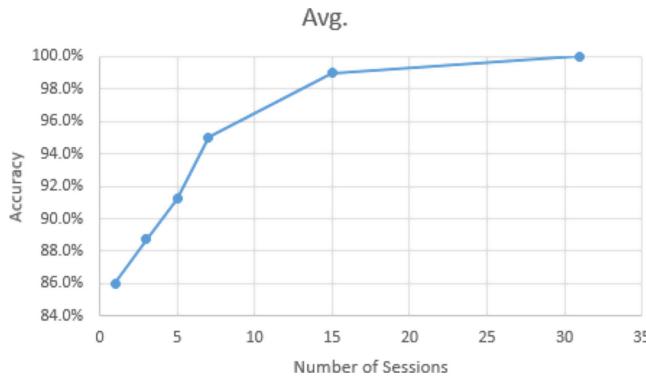


Fig. 15. TPR for the smartwatch.

IoT devices have recently been used to launch various attacks such as DoS to steal end user information [58]. Most of the recent research in the area of the IoT and DoS attacks focuses on Botnets exploiting IoT devices to launch distributed DoS (DDoS) attacks [59]. The scientific community has placed less emphasis on DoS attacks targeted against IoT devices themselves. In other words, DoS attacks are a threat to the IoT devices. Although IoT devices expose various vulnerabilities, they have less computing power compared to desktop computers and other computing devices, and thus, are susceptible and less resilient to such attacks.

In order to evaluate the resilience of IoT devices against DoS and DDoS attacks, we introduced the IoT resilience metric, referred as R_{IoT}. The IoT resilience metric is calculated based on the services running on an IoT-DUT and the security vulnerabilities exposed by that IoT device.

In order to measure the resilience of a tested IoT device to the attack, we integrated the DoS plugin to the testbed, running by STM and MAM modules of the testbed, which can perform DoS attacks against IoT devices. The DoS plugin has the capability of generating legitimate and nonlegitimate packets for various protocols such as UDP, TCP, HTTP, *Bluetooth Low Energy* (BLE), etc. and mutates those packets to cause a DoS attack on IoT devices. Furthermore, we also adopt and evaluate legacy metrics such as the allocation of resources [60]–[62] and percentage of failed transactions [63] metrics with our IoT resilience metric.

a) *IoT resilience*: Before we define the *Resilience* of an IoT device, we need to understand its *Permeance* [64]. We define the *Permeance*, referred as P_{tot} [see (2)], of an IoT device against a DoS and DDoS attack as follows.

Definition 1 (a) The total number of packets an IoT device can service over a period of time when it is bombarded with attack packets before the IoT device fails to provide service:

$$P_{\text{tot}} = S * \frac{(P_n * P_a)}{T_{\text{RRT}}} \quad (2)$$

where P_n represents the total number of normal packets, P_a represents the total number of attack packets, T_{RRT} represents the *request response time* of the IoT DUT, and S represents the *resilience constant* specific to an IoT device vulnerability. In [50], we presented a penetration testing procedure for IoT devices, and we defined a metric system for port scanning to rate the vulnerable ports of the IoT device. We make use of the

same metric system to measure our constant S . The *resilience constant* S varies as a function of the risk level of the scanned ports. The total number of open ports running specific services on each of them indicates a possibility of those services being affected when the device is under a DoS or DDoS attack. The higher the number of open ports, the higher the chances of the device being attacked. Keeping this in mind, in [50], we calculate the *exploitability score* for an IoT device. We use the same methodology to calculate the score of the IoT devices used in our experiments.

The unit of *Permeance* is $\frac{P^2}{S}$, where P denotes the total number of packets including the normal and attack packets. From the definition of *Permeance*, we can define the Resilience of an IoT device against a DoS attack, referred as R_{IoT} [see (3)], as follows.

Definition 1 (b) The resilience of an IoT device is defined as the reciprocal of its permeance:

$$R_{\text{IoT}} = \frac{1}{P_{\text{tot}}} \quad (3)$$

where R_{IoT} is the resilience of an IoT device whose unit is $\frac{S}{P^2}$.

b) *Legacy metrics*: We identify and discuss an array of DoS metrics known as legacy metrics [62], [63] and utilize them to quantify the impact of such attacks on IoT devices. We have selected some of the widely used metrics from the state-of-the-art and are as follows.

Allocation of resources: According to Bhandari *et al.* [62], allocation of resources (4) is defined as *the ratio of the bandwidth (BW) of legitimate traffic to the bandwidth of attack traffic*:

$$\text{Allocation of Resources} = \frac{\text{Legitimate traffic BW}}{\text{Attack traffic BW}}. \quad (4)$$

The percentage of failed transactions [pft, see (5)] is a DoS impact measure for the percentage of failed transactions for each application on an IoT device [63]. Formally, pft is defined as

$$\text{pft} = \left(\frac{\text{failed transactions}}{\text{total no of transactions}} \right) \times 100. \quad (5)$$

c) *Experimental methodology and results*: Our experiments were conducted in our IoT security testbed using the DoS plugin (via the STM and MAM modules of the testbed) to evaluate and calculate the *Resilience of IoT devices* against various DoS attacks.

DoS attacks: For performing DoS attacks aimed at *IP devices*, we conducted *resource exhaustion* by sending spoofed legitimate packets to the IoT devices. For example, the communication between the Android app on the mobile phone and an IP Camera are monitored, and the legitimate packets are injected into the communication network channel of the IP camera. This kind of resource exhaustion is performed for various IP-based IoT devices in the testbed in man-in-the-middle attack fashion. On *BLE devices*, we were able to carry out DoS attacks on devices such as a Fitbit, a blood pressure monitor, etc. A BLE packet injection attack involves bombarding the BLE devices with a large number of illegitimate BLE packets resulting in the devices being overwhelmed.

TABLE IX
EVALUATION OF IoT DEVICES IN OUR TESTBED

IoT Device	Resilience of IoT (R_{IoT})	Allocation of resources	Percentage of failed transactions (pft)
D-Link	1.24E-09	2.72E-06	0.8
TP-Link	3.53E-11	3.02E-06	0.7
Nest Cam	1.86E-12	2.8E-05	0.3
Philips Hue	5.09E-06	1.76E-06	0.8
HP printer	2.98E-10	3.93E-05	0.6
Amazon Echo	9.70E-09	3.96E-04	0.4
Fitbit-1	2.5E-04	4.4E-04	0.7
BLE Watch	8.18E-05	0.001	0.8

DDoS attacks: Compromised IoT devices are capable of carrying out DDoS attacks on other IoT devices, computers, or services. One such way of facilitating a DDoS is via malware, such as Mirai [65], which was used in our experiments. Mirai turns networked devices into remotely controlled Bots. We launched a DDoS attack using ten IP cameras on the victim IP camera. In our Botnet experimentation, we used a total of 11 D-Link DCS-942L [66] IP cameras, two laptops, and a dedicated access point. We monitored the network traffic on a desktop computer through a mirror port. We have evaluated our proposed R_{IoT} metric against the legacy metrics with various DoS attacks as mentioned previously. Table IX provides the comprehensive results for the relevant metrics for testing several IoT devices used in our experimentation in the testbed.

Based on our evaluation, we can conclude that the resilience of IoT metric can better calculate how resilient an IoT device is against a DoS attack than other metrics, by considering security of the IoT device and just not limiting to the resources and transactions.

4) *Test Scenario 4. Check Password Complexity:* Many of the IoT devices connected to the Internet lack basic security protection, such as secure password authentication. As a result, those IoT devices can easily become infected by a malware and get recruited into malicious botnets (for example, the infamous Mirai botnet that targeted IoT devices running the Telnet service with default credentials).

To address this issue, a new plug-in was added to the testbed as a part of the STM module. The plug-in checks the complexity of a device's credentials and, thus, evaluates the risk that an IoT-DUT may be exploited by an attacker or malware to gain remote access to the IoT-DUT. First, a simple port scan is performed in order to identify available administration services (e.g., SSH or Telnet) that may allow remote access. If such ports are found to be open, an attempt is made to log in to the device with default credentials using a dictionary of known passwords (rockyou.txt). If all login attempts fail or no such ports were found to be open, a message is displayed, informing the testbed operator that intervention with the IoT hardware is necessary in order to proceed with the test process.

In the next step of the test, the plug-in executes a black-box reverse engineering methodology according to [69] and [70]. Using this methodology, a message is displayed, asking the testbed operator to identify and connect to the universal asynchronous receiver-transmitter (UART) ports of the IoT-DUT (if such ports exist).

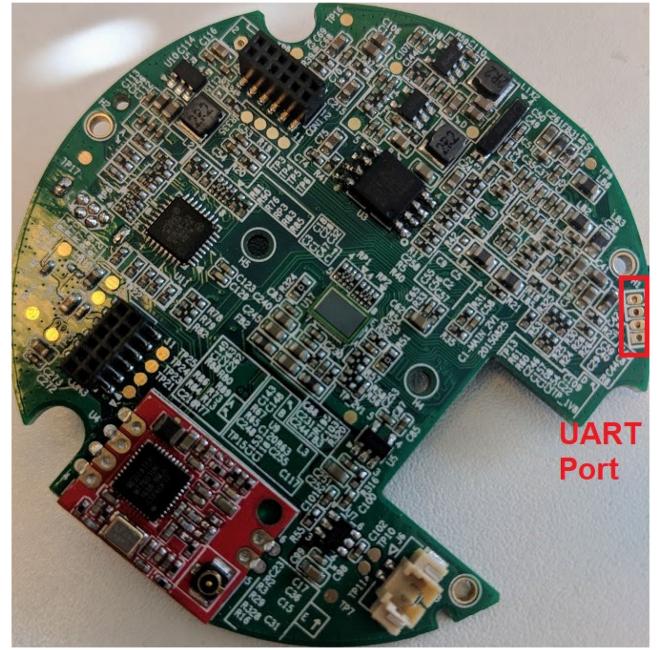


Fig. 16. Example of UART terminals of an IP camera.

UART ports are commonly used for the development and maintenance of IoT devices (an example of a UART port can be seen in Fig. 16). Connecting to UART terminals allows easy access for communication with the OS. Once a UART connection has been identified, the test proceeds with an attempt to extract the password files “/etc/passwd” or “/etc/shadow” that are stored in the OS. Then, the hashcat tool [67] is used to perform advanced password cracking on the password files based on the pattern generation guidelines and pattern list presented in [69]. Table X summarizes the results obtained in this test scenario for 16 IoT devices according to [69]. The level of complexity of a password is determined based on the time it takes to recover it. As can be seen from Table X, two IoT devices have a medium level of password complexity, five devices have a low level of password complexity, another four devices have very low complexity, four devices have unknown complexity, and one device has no password, and thus is defined as “None” in the table.

VII. DISCUSSION AND FUTURE WORK

The IoT is an emerging technology that transforms ordinary physical devices, such as televisions, refrigerators, watches, cars, and more, into smart connected devices. The potential applications associated with the IoT are seemingly infinite, with new and innovative features and capabilities being developed almost daily. However, the extensive benefits and opportunities provided by the IoT computing are accompanied by major potential security and privacy risks. Moreover, due to the heterogeneous nature of such devices (different types of devices with different software and hardware configurations installed, produced by different manufacturers, etc.) and the fact that they are used in a variety of contexts, analyzing and ensuring the security of such devices is considered a complex task. There-

TABLE X
THE PASSWORD COMPLEXITY OF THE EVALUATED DEVICES

Device Type	Manufacturer	Model	Open Services for Remote Access	Password complexity
IP Camera	Xtreamer	Cloud Camera	-	Medium
IP Camera	Simple Home	XCS7_1001	Telnet	Very Low
IP Camera	Simple Home	XCS7_1002	Telnet	Low
IP Camera	Simple Home	XCS7_1003	Telnet	Low
IP Camera	Foscam	FI9816P	-	Unknown
IP Camera	Foscam	C1	-	Unknown
IP Camera	Samsung	SNH-1011N	-	Unknown
IP Camera	Xiaomi	YI Dome	-	None
IP Camera	Provision	PT-838	-	Low
IP Camera	Provision	PT-737E	Telnet	Low
IP Camera	TP-Link	NC250	-	Very Low
Baby Monitor	Philips	B120N	SSH	Medium
Baby Monitor	Motorola	FOCUS86T	-	Unknown
Doorbell	Danmini	Wi-Fi Doorbell	Telnet	Very Low
Doorbell	Ennio	SYWIFI002	Telnet	Very Low
Thermostat	Ecobee	3 (golden firmware)	-	Low

fore, in this paper, we propose an innovative security testbed framework targeted specifically for IoT devices.

Our proposed security testbed deals with the greatest challenge in the IoT research domain, namely security testing and analysis operations, as IoT systems are considered highly complex environments due to the range of functionality and the variety of operations involved in the process. Analyzing the security and privacy risks of IoT devices and their effects on existing systems/environments is considered an extremely complex task due to their heterogeneous nature, the number of types of devices, and the many different vendors and suppliers, technologies, operating systems in use, and connectivity capabilities, and the fact that these smart devices are used in many contexts and states.

In addition, the proposed testbed deals with the challenge of testing different IoT devices and configurations, including proprietary and open-source operating systems and applications, therefore, generic and easily updatable security testing mechanisms are required, as used in our testbed. Moreover, as IoT devices are developed as closed systems (both hardware and software), the security testbed aims at dealing with the challenge of testing any embedded-based device using a combination of standard security testing mechanisms along with advanced monitoring and analysis tools, in order to examine both the internal status of the IoT-DUT (including CPU utilization, memory consumption, file system operation, and more) and the implication/impact of the device on the environment the IoT device is deployed (mainly using traffic analysis operation). The latter is done by the testbed by employing different security testing for different means of communication supported by the IoT devices, including Wi-Fi, Bluetooth, ZigBee, and others, using both standard and advanced software and hardware-based analysis tools (such as Wireshark, Ubertooth, etc.), in order to analyze the incoming and outgoing traffic patterns to/from the IoT DUT.

In order to accomplish this, the testbed employs standard and advanced security testing mechanisms in order to evaluate the security level of the IoT-DUT. The standard security testing is mainly based on penetration testing methodology, including discovery, vulnerability scans, and penetration tests

(such as port scanning, fingerprinting, process enumeration, and fuzzing), which use basic and standard testing techniques that are generic and easily updatable security testing in the testbed. In contrast, the advanced security testing is based on unique capabilities of the testbed. For example, the testbed can simulate real environments where IoT devices are deployed in order to identify possible context-based attacks by compromised IoT devices, as demonstrated in our previous work [7]. In addition, in the testbed innovative security testing mechanisms based on machine learning approaches are applied; these mechanisms are aimed at device identification, where the type of the device is identified based only on an analysis of the network traffic the device generates. Similarly, an anomaly detection mechanism to detect anomalous behavior of an IoT DUT (also based on its network traffic analysis) is also applied in the testbed. Moreover, resilience to DoS attacks test and checking the management connection password complexity test are also employed as advanced mechanisms in order to measure the device security strength. Using this methodology, of applying the standard security testing based on well-known penetration testing approaches, along with advanced security testing, which are based on innovative testing capabilities and mechanisms (including applying simulators and stimulators tools along with advanced monitoring and analysis mechanisms based on machine learning approaches), our proposed security testbed can conduct a wide range of security tests for testing and evaluating IoT devices in several contexts in order to evaluate their security level, providing a state of the art multilayered testing model. Furthermore, the testbed can be extended with new security tests, simulators, and stimulators, and monitoring and advanced analysis mechanisms that are used as internal and external plugins of the testbed.

To the best of our knowledge, we are the first to propose such an extensive security testing framework for IoT devices. Accordingly, in this paper, we emphasize the need for such a security testing platform, defining the testbed requirements and system architecture, and developing the core structure and the system infrastructure of the proposed security testbed, along with implementing a variety of security testing capabilities, based on open-source and standard tools and self-developed innovating

advanced mechanisms. As part of the building blocks in our security testing system, we integrate different peer tools as plugins to our testbed in order to meet the requirements and design for such a security testing platform, as well as to ensure that the testbed is compliant with the security testing system demands as possible. In addition, different advanced and innovative security testing mechanisms and algorithms were developed as advanced plugins as part of the testbed and are demonstrated by our security testbed, such as plugins that simulate real-time environments for IoT devices using different simulators and stimulators (as demonstrated by [7]) or employ device identification and automatic anomaly detection mechanisms based on machine learning approaches that we developed, and more. Accordingly, from our point of view, we are not competing with other peer tools, rather we are utilizing their capabilities in order to suggest as the most comprehensive security testing platform for IoT devices possible.

In future work, we intend to enhance the testbed system's capacity in order to support its full operational capability. This includes deployment of additional simulator devices, implementation of advanced measurement and analysis tools, and further automation of the testing process. Moreover, in order to extend the scope of the security testing, we intend to connect the security testbed with external testing systems targeted for IoT devices, such as a honeypot environment [68]. Based on that, additional requirements for the developing IoT security testbed, as well as its potential limitations, will be addressed. This will allow us to define which features are essential for testing various IoT devices used in different contexts and environments. Furthermore, in the future, we are planning to offer the security testbed as a service for end users from academia and industry, as well as individual smart home users; in order to do this, we plan to develop algorithms such that any individual can use our testbed capabilities to test their IoT devices. Researchers and end users will also be allowed to add their own tests to the testbed. With regard to smart home users, we are in the process of developing software that an individual can download to test their smart home network. In this case, the software will be able to scan the home network to collect the network traffic and upload a *.pcap file to our cloud service while preserving the end user's privacy. Our proposed solution will use an anomaly detection plugin and produce a report. The report will provide an assessment of how protected (or compromised) the smart home network is by assigning an overall metric score.

REFERENCES

- [1] SHODAN, "The search engine for the internet of things," 2018. [Online]. Available: <https://www.shodan.io/>
- [2] M. Patton, E. Gross, R. Chinn, S. Forbis, L. Walker, and H. Chen, "Uninvited connections: A study of vulnerable devices on the internet of things (IoT)," in *Proc. IEEE Joint Intell. Secur. Informat. Conf.*, Sep. 2014, pp. 232–235.
- [3] L. Markowsky and G. Markowsky, "Scanning for vulnerable devices in the Internet of Things," in *Proc. IEEE 8th Int. Conf. Intell. Data Acquisition Adv. Comput. Syst., Technol. Appl.*, Sep. 2015, vol. 1, pp. 463–467.
- [4] "Computerworld, a publication website and digital magazine for information technology (IT) and business technology professionals," 2018. [Online]. Available: <http://www.computerworld.com/>
- [5] "The Next Web, an online publisher of tech and web development news," 2018. [Online]. Available: <http://thenextweb.com/>
- [6] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo, "A survey on facilities for experimental internet of things research," *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 58–67, Nov. 2011.
- [7] S. Siboni, A. Shabtai, N. O. Tippenhauer, J. Lee, and Y. Elovici, "Advanced security testbed framework for wearable IoT devices," *ACM Trans. Internet Technol.*, vol. 16, no. 4, 2016, Art. no. 26.
- [8] Stanford, Secure Internet of Things Project, 2018. [Online]. Available: <http://iot.stanford.edu/>
- [9] G. Werner-Allen, P. Swieskowski, and M. Welsh, "Motelab: A wireless sensor network testbed," in *Proc. 4th Int. Symp. Inf. Process. Sensor Netw.*, Apr. 2005, pp. 483–488.
- [10] A. Arora, E. Ertin, R. Ramnath, M. Nesterenko, and W. Leal, "Kansei: A high-fidelity sensing testbed," *IEEE Internet Comput.*, vol. 10, no. 2, pp. 35–47, Mar/Apr. 2006.
- [11] J. Bers, A. Gosain, I. Rose, and M. Welsh, "Citysense: The design and performance of an urban wireless sensor network testbed," in *Proc. IEEE Int. Conf. Technol. Homeland Secur.*, Waltham, MA, USA, 2008, pp. 583–588.
- [12] C. B. Des Rosiers *et al.*, "Very large scale open wireless sensor network testbed," 2011.
- [13] I. Chatzigiannakis, S. Fischer, C. Koninis, G. Mylonas, and D. Pfisterer, "WISEBED: An open large-scale wireless sensor network testbed," in *Proc. Int. Conf. Sensor Appl., Exp. Logistics*, Sep. 2009, pp. 68–87.
- [14] FIT IoT-LAB, "IoT experimentation at a large scale," 2018. [Online]. Available: <https://www.iot-lab.info/>
- [15] "German telekom and city of friedrichshafen, friedrichshafen smart city," 2010. [Online]. Available: <http://www.telekom.com/dtag/cms/content/dt/en/395380>
- [16] M. Doddavenkatappa, M. C. Chan, and A. L. Ananda, "Indriya: A low-cost, 3D wireless sensor network testbed," in *Proc. Int. Conf. Testbeds Res. Infrastructures*, Apr. 2011, vol. 90, pp. 302–316.
- [17] INFINITE, International Future Industrial Internet Testbed. [Online]. Available: <http://www.iotinfinite.org/>
- [18] FIESTA-IoT, "Federated interoperable semantic IoT testbeds and applications." [Online]. Available: <http://fiesta-iot.eu/>
- [19] M. Nati, A. Gluhak, H. Abangar, and W. Headley, "Smartcampus: A user-centric testbed for internet of things experimentation," in *Proc. 16th Int. Symp. Wireless Pers. Multimedia Commun.*, Jun. 2013, pp. 1–6.
- [20] SmartSantander, "World city-scale experimental research facility for a smart city." [Online]. Available: <http://www.smartsantander.eu/>
- [21] Y. Berhanu, H. Abie, and M. Hamdi, "A testbed for adaptive security for IoT in eHealth," in *Proc. Workshop Adaptive Secur.*, Sep. 2013, p. 5.
- [22] R. H. Weber, "Internet of things—New security and privacy challenges," *Comput. Law Secur. Rev.*, vol. 26, no. 1, pp. 23–30, 2010.
- [23] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, 2015.
- [24] Al Ameen, Liu M., Jr., and K. Kwak, "Security and privacy issues in wireless sensor networks for healthcare applications," *J. Med. Syst.*, vol. 36, no. 1, pp. 93–101, 2012.
- [25] Q. Jing, A. V. Vasilakos, J. Wan, J. Lu, and D. Qiu, "Security of the internet of things: Perspectives and challenges," *Wireless Netw.*, vol. 20, no. 8, pp. 2481–2501, 2014.
- [26] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security challenges in the IP-based internet of things," *Wireless Pers. Commun.*, vol. 61, no. 3, pp. 527–542, 2011.
- [27] Z. K. Zhang, M. C. Y. Cho, C. W. Wang, C. W. Hsu, C. K. Chen, and S. Shieh, "IoT security: Ongoing challenges and research opportunities," in *Proc. IEEE 7th Int. Conf. Service-Oriented Comput. Appl.*, Nov. 2014, pp. 230–234.
- [28] G. M. Koien, "Reflections on trust in devices: An informal survey of human trust in an Internet-of-Things context," *Wireless Pers. Commun.*, vol. 61, no. 3, pp. 495–510, 2011.
- [29] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Comput. Netw.*, vol. 57, no. 10, pp. 2266–2279, 2013.
- [30] A. Ukil, J. Sen, and S. Koilakonda, "Embedded security for Internet of Things," in *Proc. 2nd Nat. Conf. Emerg. Trends Appl. Comput. Sci.*, Mar. 2011, pp. 1–6.
- [31] J. Granjal, E. Monteiro, and J. S. Silva, "Network-layer security for the Internet of Things using TinyOS and BLIP," *Int. J. Commun. Syst.*, vol. 27, no. 10, pp. 1938–1963, 2014.

- [32] L. Li, "Study on security architecture in the Internet of Things," in *Proc. Int. Conf. Meas., Inf. Control*, May 2012, vol. 1, pp. 374–377.
- [33] M. Abomhara and G. M. Køien, "Security and privacy in the internet of things: Current status and open issues," in *Proc. Int. Conf. Privacy Secur. Mobile Syst.*, May 2014, pp. 1–8.
- [34] K. T. Nguyen, M. Laurent, and N. Oualha, "Survey on secure communication protocols for the Internet of Things," *Ad Hoc Netw.*, vol. 32, pp. 17–31, 2015.
- [35] X. Xiaohui, "Study on security problems and key technologies of the internet of things," in *Proc. 5th Int. Conf. Comput. Inf. Sci.*, Jun. 2013, pp. 407–410.
- [36] We Live Security, "Ransomware and the internet of things," 2016. [Online]. Available: <http://www.welivesecurity.com/2016/04/27/ransomware-internet-things/>
- [37] A. W. Atamli and A. Martin, "Threat-based security analysis for the internet of things," in *Proc. Int. Workshop Secure Internet Things*, Sep. 2014, pp. 35–43.
- [38] OWASP, *The Open Web Application Security Project*, 2018. [Online]. Available: https://www.owasp.org/index.php/Top_10_2013-Top_10
- [39] Nmap, *Free Security Scanner for Network Exploration and Security Audits*, 2018. [Online]. Available: <https://nmap.org/>
- [40] Wireshark, *A Network Protocol Analyzer*, 2018. [Online]. Available: <https://www.wireshark.org/>
- [41] Aircrack-NG, *A Network Software Suite to Assess WiFi Network Security*, 2018. [Online]. Available: <http://aircrack-ng.org/>
- [42] Metasploit, *Penetration Testing Tool*, 2018. [Online]. Available: <https://www.metasploit.com/>
- [43] Kali Linux, *Penetration Testing and Ethical Hacking Linux Distribution*, 2018. [Online]. Available: <https://www.kali.org/>
- [44] Nessus, *A Network Vulnerability Scanner*, Tenable Network Security, 2018. [Online]. Available: <http://www.tenable.com/products/nessus-vulnerability-scanner>
- [45] OpenVAS, *A Framework for Vulnerability Scanning and Vulnerability Management*, 2018. [Online]. Available: <http://openvas.org/>
- [46] Cain & Abel, *A Password Recovery Tool for Microsoft Operating Systems*, OXID.IT, 2018. [Online]. Available: <http://www.oxid.it/cain.html>
- [47] OSSEC, *Open Source HIDS Security*, 2018. [Online]. Available: <http://ossec.github.io/>
- [48] TestStand NI, *A Test Management Software for Automated Test*, National Instruments, 2018. [Online]. Available: <http://www.ni.com/teststand/>
- [49] LabVIEW, *A System-Design Platform and Development Environment*, National Instruments, 2018. [Online]. Available: <http://www.ni.com/labview/>
- [50] V. Sachidananda, S. Siboni, A. Shabtai, J. Toh, S. Bhairav, and Y. Elovici, "Let the cat out of the bag: A holistic approach towards security analysis of the internet of things," in *Proc. 3rd ACM Int. Workshop IoT Privacy, Trust, Secur.*, Apr. 2017, pp. 3–10.
- [51] Tenable, *Vulnerability Reporting by Common Ports*, 2018. [Online]. Available: <http://www.tenable.com/sc-report-templates/vulnerability-reporting-by-common-ports>
- [52] PacketFence, 2018. [Online]. Available: <https://packetfence.org/dhcp-fingerprints.conf>
- [53] National Vulnerability Database (NVD)-NIST, 2018. [Online]. Available: <https://nvd.nist.gov/>
- [54] Nikto, *A Web Server Fuzzer*, 2018. [Online]. Available: <https://github.com/sullo/nikto>
- [55] D. Bekerman, B. Shapira, L. Rokach, and A. Bar, "Unknown malware detection using network traffic classification," in *Proc. IEEE Conf. Commun. Netw. Secur.*, Sep. 2015, pp. 134–142.
- [56] Scikit-Learn, *Machine Learning in Python*, 2018. [Online]. Available: <http://scikit-learn.org/stable/>
- [57] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "Iotpot: A novel honeypot for revealing current IoT threats," *J. Inf. Process.*, vol. 24, no. 3, 522–533, 2016.
- [58] Distributed Denial of Service Using Mirai, 2018. [Online]. Available: <https://www.bankinfosecurity.com>
- [59] Mirai Malware for IoT, 2018. [Online]. Available: <https://www.symantec.com>
- [60] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 3, pp. 62–73, 2002.
- [61] G. Oikonomou, J. Mirkovic, P. Reiher, and M. Robinson, "A framework for a collaborative DDoS defense," in *Proc. 22nd Annu. Comput. Secur. Appl. Conf.*, Dec. 2006, pp. 33–42.
- [62] A. Bhandari, A. L. Sangal, and K. Kumar, "Performance metrics for defense framework against distributed denial of service attacks," *Int. J. Netw. Secur.*, vol. 5, no. 2, p. 38, 2014.
- [63] J. Mirkovic *et al.*, "Towards user-centric metrics for denial-of-service measurement," in *Proc. Workshop Exp. Comput. Sci.*, Jun. 2007, p. 8.
- [64] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39–53, 2004.
- [65] *Malware Must Die - Mirai Malware*, 2018. [Online]. Available: <http://blog.malwaremustdie.org>
- [66] *Dlink IP Camera*, 2018. [Online]. Available: <http://www.dlink.com.sg/>
- [67] Hashcat, *Advanced Password Recovery*, 2018. [Online]. Available: <https://hashcat.net/hashcat/>
- [68] J. D. Guarinzo *et al.*, "Siphon: Towards scalable high-interaction physical honeypots," in *Proc. 3rd ACM Workshop Cyber-Phys. Syst. Secur.*, Apr. 2017, pp. 57–68.
- [69] O. Shwartz, Y. Mathov, M. Bohadana, Y. Elovici, and Y. Oren, "Opening Pandora's box: Effective techniques for reverse engineering IoT devices," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.* Cham, Switzerland: Springer, Nov. 2017, pp. 1–21.
- [70] O. Shwartz, Y. Mathov, M. Bohadana, Y. Oren, and Y. Elovici, "Reverse engineering IoT devices: Effective techniques and methods," *IEEE Internet Things J.*, to be published.



Shachar Siboni received the B.Sc. and M.Sc. degrees in communication systems engineering from the Ben-Gurion University of the Negev (BGU), Beersheba, Israel, where he is currently working toward the Ph.D. degree at the Department of Software and Information Systems Engineering.

For the last 15 years, he has worked in a variety of roles at leading companies in the hi-tech industry, ranging from IT Technical Leader, Communication Systems Engineer, Real-Time Embedded Software Engineer/Developer and Team Leader, to Security Researcher and Project Manager. In his most recent role, he led a joint research project collaborating with research groups from BGU's Cyber Security Research Center and the iTrust Centre for Research in Cyber Security, Singapore University of Technology and Design. His research interests include security risk analysis and machine learning approaches in the Internet of Things research domain.



Vinay Sachidananda received the master's degree from the University of Trento, Trento, Italy, and the Ph.D. degree from the Technical University of Darmstadt, Darmstadt, Germany.

He is currently a Research Scientist with iTrust, Singapore University of Technology and Design, Singapore. He has been researching in the area of cyber security focusing on Internet of Things (IoT) and has been developing various techniques on how to expose the IoT devices by finding known and unknown vulnerabilities. Apart from security, he also has been researching privacy issues related to IoT. His past research has focused on quality of information in wireless sensor networks.



Yair Meidan received the B.Sc. and M.Sc. degrees in industrial engineering and management from the Ben Gurion University of the Negev, Beersheba, Israel, where he is currently working toward the Ph.D. degree at the Department of Software and Information Systems Engineering.

Afterward, for eight years, he served in several data science roles in the industry, while teaching introductory courses on data mining at Ben-Gurion University, the Open University, and Shenkar College of Engineering. He is currently a Researcher with the Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev. His research interests include applied machine learning, Internet of Things analytics, and cyber security.



Michael Bohadana received the Master of Science and Graduate degrees from the Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Beersheba, Israel.

His research interests include machine learning, deep learning, reverse engineering, and IoT security.



Yael Mathov received the B.Sc. degree in computer science and the M.Sc. degree in software and information systems engineering from the Ben-Gurion University of the Negev, Beersheba, Israel, where she is currently working toward the Ph.D. degree.

Her work has been focusing on IoT security, reverse engineering, and machine learning. Her current research interest includes adversarial machine learning.



Suhas Bhairav received the master's degree from Technische Universität Darmstadt, Darmstadt, Germany.

He is currently a Researcher with iTrust, Singapore University of Technology and Design, Singapore. His research interests include IoT security, static analysis, programming languages, and graph theory.



Asaf Shabtai received the B.Sc. degree in mathematics and computer sciences, the B.Sc. degree in information systems engineering, the M.Sc. degree in information systems engineering, and the Ph.D. degree in information systems engineering, all from Ben-Gurion University of the Negev, Beersheba, Israel.

He is currently an Assistant Professor with the Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev. He is a recognized expert in information systems security and has led several large-scale projects and researches in this field. Since 2005, he has been a Principle Investigator of various research projects funded by Deutsche Telekom AG (Telekom Innovation Laboratories @ BGU), Israeli Ministry of Defense, Israeli Ministry of Trade and Commerce, and several leading cyber security companies. He has authored/coauthored papers in leading peer-reviewed journals and conferences. In addition, he has coauthored a book on information leakage detection and prevention. His main research interests include computer and network security, machine learning, cyber intelligence, security awareness, security of IoT and smart mobile devices, social network analysis, and security of avionic and operational technologies systems.



Yuval Elovici received the B.Sc. and M.Sc. degrees in computer and electrical engineering from Ben-Gurion University of the Negev (BGU), Beersheba, Israel, and the Ph.D. degree in information systems from Tel-Aviv University, Tel Aviv, Israel.

He is currently the Director of the Telekom Innovation Laboratories, Ben-Gurion University of the Negev (BGU), the Head of the BGU Cyber Security Research Center, and a Professor with the Department of Software and Information Systems Engineering, BGU. For the past 14 years, he has led the cooperation between BGU and Deutsche Telekom. He also consults professionally in the area of cyber security and is the cofounder of Morphisec, startup company that develops innovative cybersecurity mechanisms that relate to moving target defense. He has authored/coauthored articles in leading peer-reviewed journals and in various peer-reviewed conferences. In addition, he has coauthored a book on social network security and a book on information leakage detection and prevention. His primary research interests include computer and network security, cyber security, web intelligence, information warfare, social network analysis, and machine learning.