Advanced Security Testbed Framework for Wearable IoT Devices

SHACHAR SIBONI and ASAF SHABTAI, Ben-Gurion University of the Negev, Israel
NILS O. TIPPENHAUER, Singapore University of Technology and Design, Singapore
JEMIN LEE, Daegu Gyeongbuk Institute of Science and Technology, South Korea
YUVAL ELOVICI, Ben-Gurion University of the Negev, Israel; Singapore University of Technology and Design, Singapore

Analyzing the security of Wearable Internet-of-Things (WIoT) devices is considered a complex task due to their heterogeneous nature. In addition, there is currently no mechanism that performs security testing for WIoT devices in different contexts. In this article, we propose an innovative security testbed framework targeted at wearable devices, where a set of security tests are conducted, and a dynamic analysis is performed by realistically simulating environmental conditions in which WIoT devices operate. The architectural design of the proposed testbed and a proof-of-concept, demonstrating a preliminary analysis and the detection of context-based attacks executed by smartwatch devices, are presented.

CCS Concepts: • **Security and privacy → Vulnerability management**

Additional Key Words and Phrases: Internet of things (IoT), security, privacy, testbed framework, wearable devices

ACM Reference Format:
Shachar Siboni, Asaf Shabtai, Nils O. Tippenhauer, Jemin Lee, and Yuval Elovici. 2016. Advanced security testbed framework for wearable IoT devices. ACM Trans. Internet Technol. 16, 4, Article 26 (December 2016), 25 pages.
DOI: http://dx.doi.org/10.1145/2981546

1. INTRODUCTION

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. The application domains of the IoT are diverse, spanning from smart cities, building and home automation, transportation and logistics, and environmental monitoring, to smart enterprise environments, connected home appliances, and smart wearable devices [Atzori et al. 2010; Gubbi et al. 2013; Perera et al. 2015].

Wearable computing is an emerging ubiquitous technology in the IoT ecosystem, where new products are entering the market at an ever increasing rate. Wearables

Authors' addresses: S. Siboni (corresponding author) and A. Shabtai, Department of Information Systems Engineering, Cyber Security Research Center, Ben-Gurion University of the Negev (BGU), Beer-Sheva, 84105, Israel; email: sibonish@post.bgu.ac.il; N. O. Tippenhauer, Pillar of Information Systems Technology and Design, Singapore University of Technology and Design (SUTD), 8 Somapah Road, 487372, Singapore; J. Lee, Department of Information and Communication Engineering, Daegu Gyeongbuk Institute of Science and Technology (DGIST), Daegu, 42988, South Korea; Y. Elovici, Department of Information Systems Engineering, Cyber Security Research Center, Ben-Gurion University of the Negev (BGU), Beer-Sheva, 84105, Israel, and iTrust, Center for Research in Cyber Security at Singapore University of Technology and Design (SUTD), 8 Somapah Road, 487372, Singapore.

DOI: http://dx.doi.org/10.1145/2981546

are smart embedded devices comfortably worn directly on the user's body, which define a new Wearable IoT (WIoT) segment as a user-centered environment [Hiremath et al. 2014]. These smart devices are naturally integrated into daily life by transforming ordinary wearable and health-related products, such as: bracelets, wristwatches, glasses, clothes, shoes, contact lenses (see Google's "Smart Contact Lens"), and even bandages, into smart digital wearable devices with the capability to sense, compute, and communicate with their surroundings [Swan 2012].

Security experts have recently raised concerns about the security and privacy of wearable IoT devices [Lee et al. 2015; Hale et al. 2015; Cyr et al. 2014]. This is due to the following characteristics and properties of these state-of-the-art IoT devices. First, these devices are powered by advanced (and mostly standard) operating systems, enabling users to install applications on the device (specifically in smartwatch devices), and are therefore exposed to different types of security breaches and attacks, most of which are already known from other platforms (e.g., desktops/laptops, smartphones) [Donohue 2014; Migicovsky et al. 2014; Wang et al. 2015]. Second, these smart devices are equipped with advanced sensing and communication capabilities that permit the monitoring and tracking of the wearer's activity, behavior, location, and health condition in real-time [Hiremath et al. 2014; Swan 2012], all of which are attractive device features. The fact that such devices are always connected to the network greatly increases the risks of privacy violations [Lee et al. 2015; Hale et al. 2015]. Third, WIoT devices are not designed with security in mind and are designed mainly on the basis of features and cost considerations [Cooper 2015; Puckett 2014]. Unfortunately, standard security solutions are largely not applicable to such devices due to the devices' limited resources (computational, memory, power). Finally, analyzing the security of such devices is considered an extremely complex task due to their heterogeneous nature (numerous types of devices and vendors) and the fact that these devices are used in a variety of contexts and states.

Therefore, there is an urgent need to develop advanced mechanisms that, on the one hand, can determine if a wearable device complies with a set of predefined security requirements and, on the other hand, can determine if the device is compromised by malicious applications. In this article, we propose an innovative IoT security testbed whose main objective is to test the wearable IoT device's security against a set of security requirements, as well as to test its behavior under various conditions (e.g., when different applications are running). That is, the testbed is designed to simulate environmental conditions in which the tested device might be operated, such as the location, time, lighting, movement, etc., in order to detect possible context-based attacks (i.e., attacks that are designed to be triggered when the device is within a specific state) and data attacks that may be achieved by sensor manipulation. The main contributions of this article are as follows:

—We provide a detailed discussion of security and privacy threats for current and future wearable IoT devices and present several approaches to mitigate the threats posed by such devices.
—We present our 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.
—We use a prototype implementation of the testbed in order to analyze different wearable devices (Google Glass and smartwatch devices), and to detect new context-based attacks that are executed by these devices and could occur in the future.

The rest of the article is structured as follows. In Section 2, we present an overview of security and privacy aspects in wearable computing. In Section 3, we portray the design considerations for the advanced security testbed framework, including an

in-depth description of the functional architecture model. In Section 4, we present a proof-of-concept for the proposed security testbed operation using several wearable devices, followed by work related to this study in Section 5. We conclude with a discussion and possible future work in Section 6.

2. SECURITY AND PRIVACY ASPECTS OF WEARABLE COMPUTING

Wearable devices may pose major security and privacy risks [Lee et al. 2015; Hale et al. 2015], 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. Furthermore, these devices are worn on the body and can operate 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 wearable devices. In addition, we present countermeasures to reduce and mitigate the problems discussed.

2.1. Device Architecture

The device architecture aspect includes hardware and software security considerations as follows. Regarding hardware, wearable devices are low resource devices (in terms of power source, memory size, low-bandwidth communication, and computational capabilities) [Hiremath et al. 2014; Lindström 2007]. 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 [Al-Muhtadi et al. 2001; Lennon 2015].

From a software perspective, the common operating systems of wearable IoT devices (e.g., Android) are highly exposed to known and zero-day vulnerabilities [Donohue 2014; Briodagh 2015; Goodin 2015]. Additionally, the applications running on wearable 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 [Hammond 2014]. Furthermore, wearables are much cheaper than PCs and smartphone devices, and hence they are assumed to be less continuously maintained and upgraded by the manufactures [Locke 2014].

Wearable devices often automate certain functionalities and require limited configuration with little intervention from the user. For example, the Google Glass device enables the automatic set-up of a Wi-Fi connection after viewing QR codes [Rogers 2013] or sharing information on the web. This often makes wearable devices more exposed to security risks than traditional devices.

2.2. Network Connectivity

Wearable devices can be constantly connected to the Internet [Swan 2012], either directly using Wi-Fi or cellular connectivity, or indirectly via Bluetooth connection (by pairing with smartphone devices). However, these advanced IoT devices are not always designed with security in mind, due to cost considerations and their limited resources [Cooper 2015; Puckett 2014]. Consequently, they are highly exposed to traditional Internet attacks, such as denial-of-service (DoS) attacks; data leakage; man-in-the-middle (MITM) attacks; phishing attacks; eavesdropping; side-channel attacks; and compromised attacks [Blum 2015a; Bitdefender 2014]. Moreover, due to the fact that lightweight authentication algorithms are employed [Al-Muhtadi et al. 2001], it is quite possible to manipulate and control these devices at their weakest point—when data is sent from, and received by, the device.

Another potential security issue is network disruption and overload [Blum 2015a]. With the proliferation of wearable devices, especially in corporate networks, these

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.

2.3. Data Collection

A major concern related to wearable devices is the type of data they collect, which potentially may lead to privacy invasion and information theft [Hiremath et al. 2014; Lee et al. 2015; Kaspersky Labs 2014]. As data becomes an increasingly valuable asset, many data brokers collect information about potential customers and organizations by any means, including vulnerable wearable devices [Hammond 2014].

From a user's point of view, most of the collected data is personal, and may contain sensitive information about the wearer's habits and behavior, and even private health details [Hiremath et al. 2014; Swan 2012]. Moreover, recently, wearable technology has also been integrated into enterprise and organizational environments in order to increase business productivity and efficiency levels [Perera et al. 2015]. As wearables 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 [Upton 2014]. On the other hand, sensitive corporate information might also become more accessible to outsiders and can be exposed to unauthorized individuals via these devices [Puckett 2014].

Another concern associated with wearable devices centers on theft or loss of the device [Blum 2015a], as well as ransomware attacks [Jain 2015]. 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.

2.4. Countermeasures and Mitigations

This section discusses several countermeasures that can be implemented to reduce and mitigate the security and privacy risks posed by wearable devices. For example, sensitive data stored on the device should be limited and encrypted [Tolentino 2013] (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 wearable devices should be employed [Blum 2015b].

Companies should enforce Bring-Your-Own-Device (BYOD) security and privacy policies [Puckett 2014; Almasy 2015], especially for wearable devices. This can be done using enterprise-grade encryption mechanisms for access control [Upton 2014] 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 wearable devices that have been compromised [Blum 2015a]. In addition, implementing further authentication, authorization, and accountability mechanisms for wearable devices that directly connect to the network is required [Blum 2015b].

As most of the wearable devices are Bluetooth-enabled, it is preferable to turn the Bluetooth connectivity off once the device is not in use [Nguyen 2014]. Moreover, users should be responsible for maintaining and periodically updating software versions and downloading relevant updates and patches for their wearable devices [Locke 2014].

If, for any reason, the above security problems cannot be mitigated, wearable electronic devices will eventually need to be banned in highly sensitive places [Brandon 2014], as is the case with common 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 wearer, as well as the people in his/her surroundings.

In addition to the above countermeasures and mitigations, there is a constant need to be able to evaluate the security and privacy levels of wearable IoT devices without violating the user's privacy. This should be done using a designated security testbed for wearable devices, where the motivation is twofold: (1) the security testbed will perform security testing for new devices (running known applications) as a means of assessing their security level, and (2) the security testbed will also execute security testing for wearable IoT devices that are suspected of having been compromised by malicious applications. Because the conditions that trigger these malicious applications to execute attacks are not always known, the testbed should be able to simulate possible conditions (e.g., using different simulators) 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. This issue is discussed in more detail in this article.

3. ADVANCED SECURITY TESTBED FRAMEWORK FOR WEARABLE IOT DEVICES

In this section we discuss the design considerations for an advanced security testbed framework for the wearable IoT devices environment. We list the fundamental design requirements and security tests needed from a comprehensive security testbed. Based on these requirements, we define the functional architectural model for the proposed security testbed.

3.1. Design Requirements

The design requirements for the advanced security testbed are listed below. The specifications include different aspects, ranging from the type of device under test (DUT), testing environment, and simulators array, to communication channels, protocols, data forensic analysis capabilities, management, and report tools, etc.

Device under Test (DUT). The testbed should be able to examine a wide range of wearable devices from different categories, including: activity trackers, smartwatches, smart glasses, smart clothes, smart shoes, and smart healthcare devices.

Testing Environments. The testbed should be able to emulate different types of testing environments, such as indoor and outdoor, static and dynamic environments, and mobile scenarios.

Security Testing. The security testbed should support a range of security tests, each targeting a different security aspect. Standard security testing will be performed based on vulnerability scans and penetration test methodology, in order to assess and verify the security level of wearable IoT devices under test. See Table I for a list of supported tests. In addition, advanced security testing will be performed by the security testbed using different arrays of simulators. We consider two types of advanced security tests that can be executed: *context-based attacks* and *data attacks*. In *context-based attacks*, the attacker designs his or her attack to be triggered when the device is within a specific state. This is an important attack feature that enables the malicious function to evade detection. For detecting context-based attacks, the testbed will realistically simulate environmental conditions using different simulators (e.g., sending different GPS locations and times) in order to trigger the internal sensor activities of the wearable IoT devices under test. By monitoring the behavior of the tested device we will be able to identify the contexts in which different applications act. *Data attacks* are carried out by manipulating the signals and data that are sent to the sensors. This class of attack can

Table I. Vulnerability Scans and Penetration Tests Supported by the Advanced Security Testbed

Test	Description	Test/Success criteria (example)
Scanning (e.g., IP and port scanning)	Investigate the detectability of wearable IoT devices by observing wireless/wired communication channels. Attempt to identify the existence of the device. Enumerate communication channels/traffic types observed, open ports, etc.	*Undetectable*- The WIoT-DUT cannot be detected by the testbed via any communication channel; *Safe*- The WIoT-DUT is detectable, but no open ports were observed; *Minor risk*- The WIoT-DUT is detectable, and common ports are open, e.g., port 80 (HTTP), 443 (HTTPS), etc.; *Major risk*- The WIoT-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, *Critical risk*- The WIoT-DUT is detectable, and unexpected ports are open in 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.	*Unidentifiable*- The type of WIoT-DUT cannot be identified by the testbed; *Safe*- The device provides identifiable information, but all the WIoT-DUT's software versions are up-to-date; *Minor risk*- Some low risk detected applications, e.g., calendar, etc., are out-of-date; *Major risk*- Some major risk detected applications, e.g., navigator, mail, etc., are out-of-date; or, *Critical risk*- Operating system and critical applications are out-of-date.
Process enumeration	Lists all running processes on the device and presents their CPU and memory consumptions. This can be done by monitoring the device's activities, e.g., using ADB (Android Debug Bridge) connectivity.	*Safe*- The list of processes cannot be extracted without admin privileges; *Moderate risk*- The list of processes can be extracted without admin privileges on the device only; or, *Fail*- 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.	*Pass*- Traffic is encrypted, and no data leaks are detected; or, *Fail*- Traffic is unencrypted and sent in clear text, therefore data may leak from the WIoT-DUT.
Side-channel attacks	Check for side-channel attack by executing any desired measuring tool (e.g., network traffic monitoring, power consumption, acoustic or RF emanations) and analyzing the collected data while correlating it with specific events performed by/using the WIoT device under test.	The criterion is measured by the level of correlation found between the events and measurements (collected data); the weaker the correlation, the higher the pass score.
Data collection	Check if an application on a wearable 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.	*Safe*- The tested application does not collect and store data on the WIoT-DUT; *Minor risk*- The tested application collects and stores normal data, e.g., multimedia files, on the WIoT-DUT; *Major risk*- The tested application collects and stores sensitive data, e.g., GPS locations, on the WIoT-DUT; or, *Critical risk*- The tested application collects and stores critical information, e.g., device status (CPU, memory, sensor events, etc.), on the WIoT-DUT.

(Continued)

Table I. Continued

Test	Description	Test/Success criteria (example)
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.	*Pass*- Management access ports, e.g., port 22 (SSH), port 23 (Telnet), are closed; or, *Fail*- 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 (SSLstrip) or impersonate remote servers with self-certificates (to apply a man-in-the-middle attack).	*Pass*- Unable to decrypt traffic sent/received by/to the WIoT-DUT with the applied techniques; or, *Fail*- Able to decrypt traffic data sent/received by/to the WIoT-DUT using the applied techniques.
Spoofing/ masquerade attack	Attempt to generate communication on behalf of the tested wearable IoT device. For example, determine if any of the communication types can be replayed to the external server.	*Pass*- Reply attack failed; or, *Fail*- 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.	*Safe*- The time delay between two consecutive transactions of the WIoT-DUT is within the *defined*/*normal* range; or, *Unsafe*- The time delay is greater than the *defined*/*normal* 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.	*Safe*- The device ignores received manipulated/erroneous data; or, *Unsafe*- 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.	*Safe*- No relevant vulnerabilities were found; *Minor risk* - Insignificant/low risk vulnerabilities were found; or, *Unsafe*- Significant and critical vulnerabilities where found.
Vulnerability scan	Search for additional classes of vulnerabilities by: (1) utilizing existing tools (or developing new dedicated ones 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 WIoTs or detected by honeypots, and evaluate relevant/selected attacks on the tested WIoT; and (3) using automated tools for code scanning.	*Safe*- No new vulnerabilities were found during the testing process conducted; *Minor risk*- Insignificant/low risk new vulnerabilities were found; or, *Unsafe*- Significant and critical new vulnerabilities were found.

result in manipulating the normal behavior of the device (e.g., sending false GPS locations), performing a denial-of-service attack on the device by sending crafted data, or injecting code by exploiting vulnerabilities in the code that processes the sensor data. For detecting data attacks, the testbed will support the execution of a set of predefined tests, each of which involves sending crafted sensor data, which includes specific edge cases or previously observed data attacks and monitoring the behavior of the tested device.

Simulator Array. The testbed should be able to realistically generate arbitrary real-time stimulations, ideally for all sensors of the tested devices. This can be achieved using different types of simulators (e.g., a GPS simulator that simulates

Table II. Simulators Supported by the Advanced 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 WIoT 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 WIoT 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 WIoT device under test while performing different movements.
Lighting	The testbed simulates different lighting levels, in order to test the behavior of the WIoT device under test in different lighting scenarios.
Audio	The testbed simulates audio using a voice simulator, in order to test the behavior of the WIoT device under test in different sound environments.
Video	The testbed simulates images, pictures, and videos using a video simulator, in order to test the behavior of the WIoT device under test during different video changes.
Pressure	The testbed simulates different levels of pressure, in order to test the behavior of the WIoT device under test under different pressure conditions.

different locations and trajectories, movement simulators such as robotic hands, etc.). See Table II for a list of supported simulators.

Communication Channels. The testbed should be able to support the most common types of wireless communication channels, including: Bluetooth, Wi-Fi, cellular network, ZigBee, RFID, and NFC connectivity, as well as wired communication technologies such as Ethernet and USB.

Protocol Analysis. The testbed should be able to process and analyze different communication protocols, such as IPv4, IPv6, TCP, UDP, HTTP, FTP, and SNMP, to name a few, as well as security protocols, such as SSL, TLS, DTLS, and IPsec.

OS Compatibility. The testbed should provide virtual machines to natively run software related to the wearable devices. In addition, its software tools should support the following embedded operating systems: Android (Android Wear), Windows (Windows Mobile, Window 10 IoT), Linux, iOS, and others.

Data Forensic Analysis. In order to perform security forensic analysis, the testbed should be able to extract all stored data from the tested device, including system snapshots (the status of the memory and processes) and system files (e.g., config files). Data extraction could be achieved through connections such as USB and JTAG, by using different command line tools, such as ADB (Android Debug Bridge).

Management and Report Mechanisms. The testbed should be able to support management and report mechanisms in order to control and manage the testing flow, as well as to generate reports upon completion. Such report tools should include intelligent data exploration tools for manual investigation and analysis of the collected and processed data. In addition, information obtained from the security tests, as well as prior settings provided by the system operator, can be used to output the probability of an attack and its severity of impact, and consequently this can also be used to quantify risks associated with using the tested WIoT in different case scenarios.

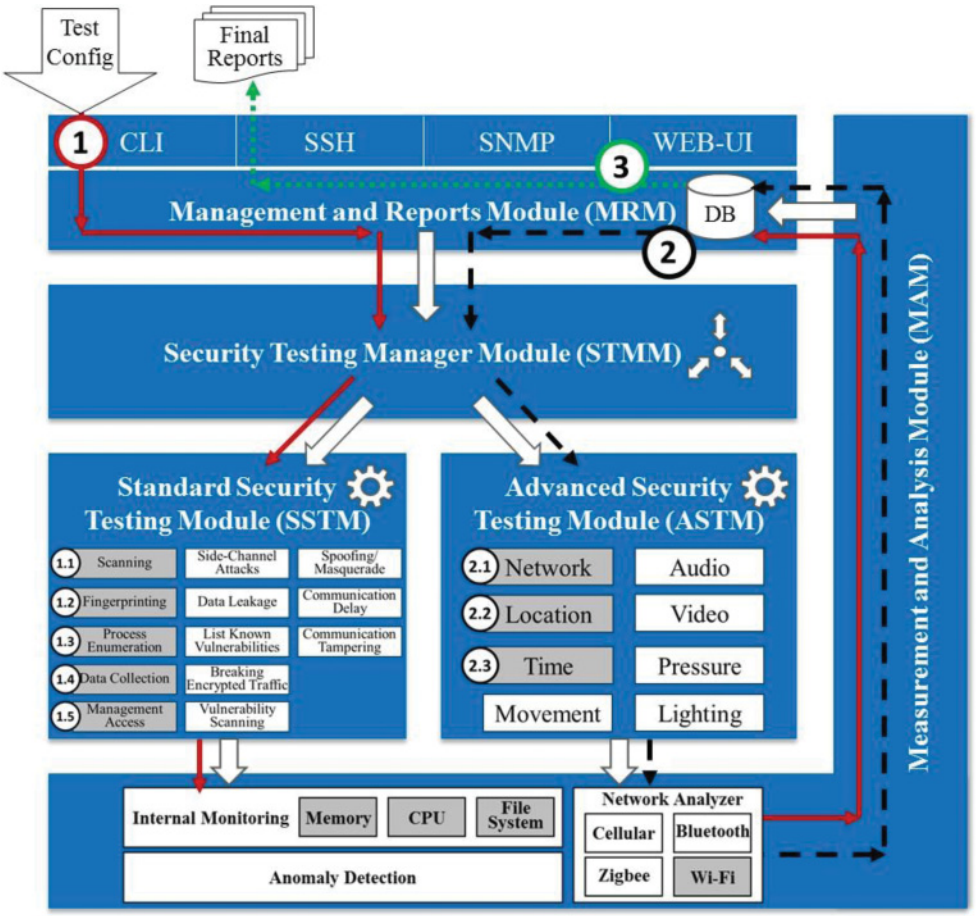


Fig. 1. Advanced security testbed framework – abstract functional architecture model. The testing process starts by loading a config file in the testbed via the MRM. Based on the configuration loaded, a standard security testing phase (Phase 1 in the red line) is conducted using the SSTM component. Then using the results obtained in Phase 1, a context-based security testing phase is conducted (Phase 2 in the black dashed line) using the ASTM component. These testing phases are controlled by the STMM. Finally, a forensic analysis is performed by the MRM based on the results obtained from both phases, and the final results are sent to the user (Phase 3 in green dashed line). Note that during our proof-of-concept (described in Section 4) we used several functional modules (marked in gray in the figure), in order to illustrate our proposed security testbed framework's operation.

User Intervention and Automation. The testbed should be able to support user intervention and automation capabilities during all phases of the test sequence.

Testbed Enhancement Capability. All of the components in the testbed should be implemented as a plugin framework to support future operational capabilities.

3.2. Functional Architecture Model

The functional architecture model of the advanced security testbed, as illustrated in Figure 1, is designed based on the list of requirements and security tests defined above. The suggested model is a layer-based platform model with a modular structure. This means that any type of wearable device can be tested in the proposed security testbed framework, and also that any relevant simulator and/or measurement and analysis tool

can be deployed in the testbed. An in-depth description of the modules that comprise the functional model and the interactions between these modules as a complete security testing system are provided.

Management and Reports Module (MRM). This module is responsible for a set of management and control actions, including starting the test, enrolling new devices, simulators, tests, measurement and analysis tools and communication channels, and generating the final reports upon completion of the test. The testbed operator (the user) interfaces with the testbed through this module using one of the communication interfaces (CLI\SSH\SNMP\WEB-UI) in order to initiate the test, as well as to receive the final reports. Accordingly, the module interacts with the Security Testing Manager Module and the Measurements and Analysis Module, respectively. The MRM holds 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 stores information regarding the test itself (including config files, system snapshots, and test results).

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 operational testing modules (the Standard and Advanced Security Testing Modules) in order to execute the required set of tests, in the right order, based on predefined configurations provided by the user (via the MRM).

Standard Security Testing Module (SSTM). This module performs standard security testing based on vulnerability assessment and penetration test methodology, in order to assess the security level of the WIoT-DUT. See Table I for a list of supported tests. SSTM is an operational module which executes a set of security tests as plugins, each of which performs a specific task in the testing process. The module interacts with the Measurements and Analysis Module in order to monitor and analyze the test performed.

Advanced Security Testing Module (ASTM). This module generates various environmental stimuli for each sensor/device under test. It is an operational module which simulates different environmental triggers, in order to identify and detect context-based attacks that may be launched by the WIoT-DUT. 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 [Tippenhauer et al. 2011]), 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 in order to monitor and analyze the test performed.

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. The analysis components process the collected data and evaluate the results according to a predefined success criterion (for example, binary pass/fail or a scale from 1 [pass] to 5 [fail]). The following is an example of a predefined success criterion: "If an SSH service is open on the tested device, and it is possible to access the device using a dictionary attack, then the test result is fail; if otherwise, the result is pass." We believe that most of the predefined success criteria are not generic and are defined for a specific tested

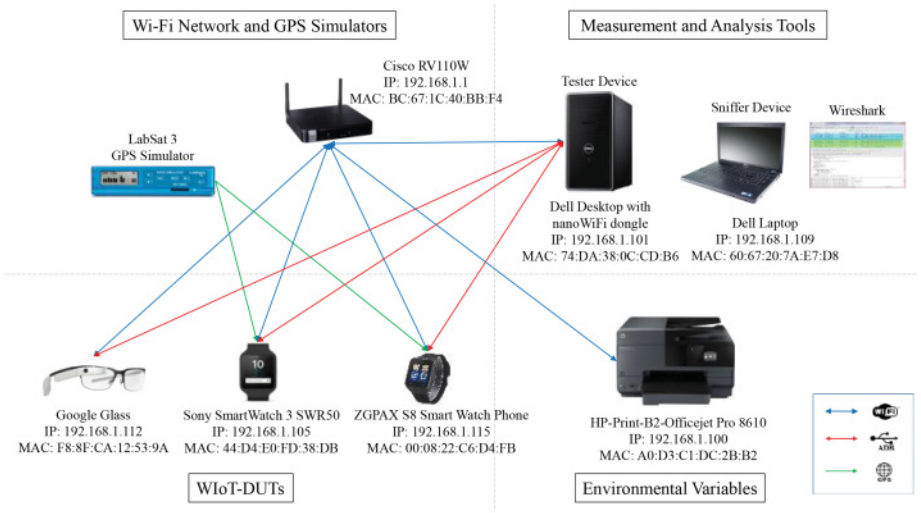
Fig. 2. The shielded room built especially for the "IoT Security Testbed" project.

IoT device and/or tested scenario. For example, the success criterion of a data leakage test may be defined differently within the scope of private or enterprise usage scenarios of an IoT. Alternatively, in some cases, a success criterion cannot be clearly defined, and therefore the analysis components extract useful insights that can be investigated and interpreted by the system operator. As an example, we can apply a network-based anomaly detection component that processes the recorded network traffic of the tested WIoT and detects anomalous events in the system. In this case, the pass/fail decision will be based on the number of detected anomalies and a predefined threshold provided by the system operator in advance. The detected anomalies should then be investigated and interpreted by the system operator using a dedicated exploration tool which is part of the user interface.

4. SECURITY TESTBED OPERATION – A PROOF-OF-CONCEPT

In this section we present a proof-of-concept for our proposed security testbed operation on selected wearable devices (Google Glass and smartwatch devices). The proof-of-concept was deployed within a shielded room (depicted in Figure 2) which served as our IoT security testbed environment. The shielded room allowed us to conduct various tests, such as simulating GPS locations, in a neutral environment with minimal external disruptions. As can be seen in Figure 2, the testbed was equipped with an internal IP camera that documented and recorded the course of a test, as well as an external workstation that was used to control the testbed's operation, including defining, executing, and analyzing tests.

The proof-of-concept demonstration was conducted in two phases. First, as part of the standard security testing phase (using the SSTM component), a preliminary security analysis is conducted for all of the wearable IoT devices under test (WIoT-DUTs). Then, based on the information collected and analyzed, a context-based security testing process is executed (by the ASTM). This is done by using a GPS simulator that simulates different locations and times of day as the triggers for context-based attacks that are carried out by malicious applications installed on the smartwatch devices. Finally, forensic analysis is performed in order to detect the context-based attacks in the testbed. Note that we implement two malicious applications specifically for the



(a)

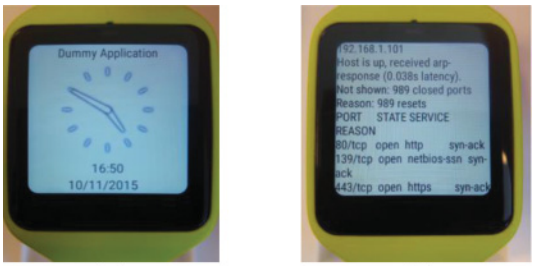
(b)

Fig. 3. Testbed configuration: (a) The components we used during the examination, including: Wi-Fi and GPS simulators, Google Glass and smartwatch devices (the WIoT-DUTs), measurement and analysis tools, and a Wi-Fi printer which served as an environmental variable component in the network; (b) The actual testbed configuration employed during the proof-of-concept.

proof-of-concept in order to illustrate the operation of the testbed, as is discussed next. The practical testbed configuration and the full testing procedure are presented below.

4.1. Testbed Configuration

The scenarios discussed below refer to Bring-Your-Own-Device (BYOD) use cases, where more and more employees bring their personal mobile devices, including wearable IoT devices (and more specifically, smartwatch devices), to the workplace. We built an isolated Wi-Fi network in our lab in order to simulate a small organizational environment, as is illustrated in Figure 3. During our examination, a network simulator



(a) (b)

Fig. 4. (a) A "malicious" application running on the Sony smartwatch device; (b) The network mapping attack is executed once the location for the attack is identified (Wi-Fi is already enabled on the device).

(using a Wi-Fi router) and a GPS simulator (LabSat 3 device) were used as part of the simulator array in the testbed framework. In addition, different measurement and analysis tools were used, including a sniffer device based on the Wireshark network protocol analyzer tool [Combs 2007] that monitored the communication traffic during the test. Moreover, we also employed a tester device that ran dedicated scripts which recorded the internal status data of the WIoT-DUTs during the test (via ADB connectivity). The tester device was also used for executing the standard security tests. The wearable devices tested (the WIoT-DUTs) during the testing process are the Google Glass device and two different smartwatch devices, the Sony Smartwatch 3 SWR50 and the ZGPAX S8 Smart Watch Phone. Moreover, a Wi-Fi printer was used as an environmental component in the testbed.

4.2. Pre-Testing Setup

In order to illustrate a full testing process using our security testbed proof-of-concept implementation, including both standard and context-based security testing, we developed two "malicious" applications, one for each smartwatch device. These "malicious" applications read the time from the watch and the GPS raw data directly from the internal built-in GPS sensor of the smartwatch devices. This information is then used for time-based and location-based attacks, respectively, meaning the time and location are the contextual triggers for the attacks we implemented. Note that these applications can be any legitimate applications that currently exist for smartwatch devices (such as a fitness application that uses the GPS connectivity), but once the conditions are met (time of day, location identification, and/or Wi-Fi connectivity) the applications can covertly execute a context-based attack. For example, the application we implemented and installed on the Sony smartwatch device actually performs a network mapping attack once the location is identified by the application and the Wi-Fi connectivity is enabled in the device. We implemented this network mapping attack by utilizing the Nmap tool that was modified to run in an Android Wear environment (this was done by adjusting and enhancing an open source code Holden [2015]). In this case, we used Nmap as an attack tool that requires only a standard mode of operation, without the need for a rooted device. The information received during the attack includes all IP addresses and open TCP/UDP ports for each IP address, for all hosts (wireless-based) connected to our Wi-Fi network, as illustrated in Figure 4.

The second "malicious" application we implemented was installed on the ZGPAX S8 Smart Watch Phone device. This application executes a fake access point attack based on the time of day as the trigger. For this purpose, we adjusted our previous code attack that simulates a Wi-Fi printer service (as presented in iTrust [2015]) to operate under the Android OS. In this case, the ZGPAX device poses as a legitimate AP in the network in order to "silently" collect sensitive information from the organization. That

is, first, the smartwatch phone device is connected directly to the Wi-Fi's organizational network as a legitimate client. Then, once the specific hour of the day the attacker set in advance is identified, the application opens a malicious AP with the same SSID name of another legitimate AP in the network, in this case a Wi-Fi printer. This is an important scenario wherein sensitive data can be collected ("silently") from the enterprise network without anyone noticing (the IT department would be unaware of this attack scenario, since there are no tools to monitor such cases).

4.3. Testbed Operation – Security Testing Process

The security testing process conducted by our proposed security testbed framework is illustrated in Figure 1. First, we executed a preliminary security analysis for Google Glass, the Sony smartwatch, and the ZGPAX smartwatch devices. In order to demonstrate the feasibility of the implementation of the proposed testbed, as part of the Standard Security Testing Module (SSTM), we implemented the following subset of security tests from Table I: Scanning, Fingerprinting, Process Enumeration, Data Collection, and Management Access (illustrated as Phase 1 in Figure 1 by the red line). For this purpose, we utilized different security testing tools available online, such as the Nmap security scanner tool [Lyon 2009], the Kali Linux penetration testing environment [Offensive Security 2016], and several scripting tools that we implemented for this testing phase. Using these generic tools we investigated the WIoT-DUTs from different aspects, including: the operating system, communication channels, firmware and hardware (sensor point of view), and the applications installed in the device, as discussed next. Note that all of the above tests were conducted using the tester device shown in Figure 3 and using the Kali Linux platform.

Scanning. First, in order to identify the WIoT-DUTs in the testbed and for analyzing the information exposed by the WIoT-DUTs via different communication channels, we implemented the scanning test (sub-process 1.1 in Figure 1). During this test we managed to detect all WIoT-DUTs in the testbed, both via scanning the wireless communication channel (by scanning the Wi-Fi network using an "arping" command), and via scanning wired connectivity (by scanning all USB ports of the tester device and detecting the ADB connectivity of all of the devices under test. For that, we enabled developer mode in all of the devices). This means that the WIoT-DUTs are accessible in the testbed for further analysis, both via wireless and wired communication channels, as defined in Table I.

Fingerprinting. Next, the fingerprinting test (sub-process 1.2 in Figure 1) was run for each WIoT-DUT separately, in order to identify the properties of the device, such as its type (type of wearable device), the OS installed, software versions, open ports (TCP/UDP), communication channels supported by the device and their configurations, device memory, a list of features and device capabilities (such as a set of sensors supported by the device), etc. We used the Nmap tool (for wireless communication) and also ran a dedicated script (that executes different commands, such as "getprop," "pm list features," etc.) via the ADB connectivity, in order to collect the information needed for the fingerprinting test.

Process Enumeration. For the process enumeration test (sub-process 1.3 in Figure 1), we executed a script (that uses the "top" and "pm list packages" commands via ADB shell) in order to list all of the processes running and application packages installed on the WIoT-DUTs, as well as to monitor their CPU and memory consumptions. Using the above information, we further analyzed the devices' activities.

Data Collection. During this security test (sub-process 1.4 in Figure 1), we employed the internal device monitoring tools (via MAM component in Figure 1). Using

these tools, we are able to investigate the WIoT-DUTs' activities (from memory, CPU, file system perspectives, and more) while running the applications installed on these WIoT-DUTs. We also extracted further information for each application, such as its permissions (using "dumpsys package" command), etc. Based on this examination, we tagged selected applications as suspicious applications (e.g., applications that use the GPS while running) that were later tested by the ASTM component.

Management Access. In this security test (sub-process 1.5 in Figure 1) both telnet and SSH connections were tested, in order to examine whether these services were opened unexpectedly and/or accessible. In this test we tried to connect to the WIoT-DUTs via these connections using a dictionary attack methodology. We used common usernames (such as "root," "admin," etc.) and a common password list (by utilizing the well-known password list "rockyou.txt" database) for this test. In all cases, both telnet and SSH connections were found to be closed. Note that although we could examine the list of all of the WIoT-DUTs' open ports, we decided to actively perform the security connectivity test and try to connect to the WIoT-DUTs via these connections (telnet and SSH) in order to illustrate the testbed capabilities.

All of the results obtained during this phase, including the list of all suspicious applications mentioned above, were stored in the system database as shown in Figure 1. Note that in our case, the list of suspicious applications includes the malicious applications that we implemented specifically for the proof-of-concept. This list was then used as input for the context-based security testing phase in the testing process. The above list of suspicious applications can be generated by either the testing process itself (by identifying abnormal behavior during the test, e.g., GPS activated unexpectedly, etc.) or by examining each application installed in the device against whitelisted and blacklisted application databases available online (e.g., based on application ratings, etc.)

Context-Based Security Test. In order to determine whether the devices under test are compromised by malicious applications, a context-based security testing phase is executed next. In this phase (illustrated as Phase 2 in Figure 1 by the black dashed line), both of the smartwatch devices were further tested by examining the suspected application (from the list generated in the previous phase) that was installed on it. For this matter, we used a GPS simulator device (LabSat 3) in order to realistically simulate the environmental conditions (i.e., locations and times) that would trigger the internal sensor activities of the tested smartwatch devices, and would accordingly trigger the attacks discussed above. Therefore, prior to the test performed in our lab, we recorded a predetermined path around our campus that was later replayed during the testing process, in order to illustrate changes in space and time for the attacks. Note, the overall testing time ($\sim$10 minutes) of the advanced/dynamic security testing phase performed in the proof-of-concept is defined based on the recorded path, shown in Figure 5.

Regarding the test performed in Phase 2 (Figure 1), first, as mentioned, we established an isolated Wi-Fi network in our lab and connected the WIoT-DUTs (the Sony and ZGPAX smartwatch devices) to the tester device (as shown in Figure 2). For each WIoT-DUT, an ADB connection was opened in the tester computer, in order to monitor and track the device's internal activities during the testing process. For that matter, we employed several measurement and analysis tools using scripts that read and locally store (on the tester device) the internal state of the WIoT-DUTs at the beginning, during, and at the end of the test. The recording includes the memory and CPU consumption, the file system configuration, the space usage (used for temporal file tracking), a list of active processes in the WIoT-DUT, a list of SSIDs available in the network, and the time and location received by the GPS simulator (which will be used

Fig. 5. The recorded path around our campus as shown in U-Blox application (supplied with the GPS).

in the forensic analysis procedure to identify the time and locations of the attacks). Note that we record these parameters, since we expect to see changes in the internal state of the WIoT-DUTs at the time of the attacks. In addition, we use a Sniffer device from the measurement and analysis tools (as shown in Figure 2), in order to monitor and track communication changes during the testing process. Here as well, we expect to see changes in the communication at the time of the attacks.

At the beginning of the test, we started our "malicious" applications (the suspicious applications) and ran the scripts from the tester device for each of the smartwatch devices under test (via the ADB connections) in order to record their internal state data during the testing process. Once the initial data collection is complete, both the GPS simulator and the Wireshark application were started simultaneously. This was done in order to synchronize the time of the recorded path and the network traffic monitoring. This point is defined as the starting point, T0, of the test. In this phase, the recorded path is replayed in the testbed by the GPS simulator in order to illustrate changes in space (locations) and time. Accordingly, once the locations and times for the context-based attacks defined above were identified by the WIoT-DUTs, the attacks were executed in the testbed. Moreover, we also injected controlled false alarms during this period (by executing a port scan with the laptop on one of the devices which is not one of the WIoT-DUTs). This was done to illustrate the forensic analysis for these events, such that, the testbed should be able to handle this type of situation as a comprehensive security testing system. Finally, once the replay of the recorded path by the GPS simulator was finished (after $\sim$10 minutes) and defined as the ending point, Tn, of the test, we stopped the test (stopped the traffic monitoring and automatic scripting) and stored the test results (communication monitoring and WIoT-DUT internal state data that were recorded during the test), in the testbed system database, and the overall test was complete.

4.4. Detecting Context-based Attacks

In order to identify the context-based attacks in our testbed, a forensic analysis is performed for each smartwatch device tested, based on the recorded information (communication and internal status of the DUTs) obtained during the testing process discussed above. As will be shown next, this was done by examining both the suspicious behavior of the tested devices, from memory consumption and CPU utilization points of view, and the communication transmissions recorded during the test. Note that the locations and times for the attacks discussed above were randomly selected from the recorded path, meaning in each new execution of the dummy applications, different locations and times will be selected for the context-based attacks. Accordingly, forensic

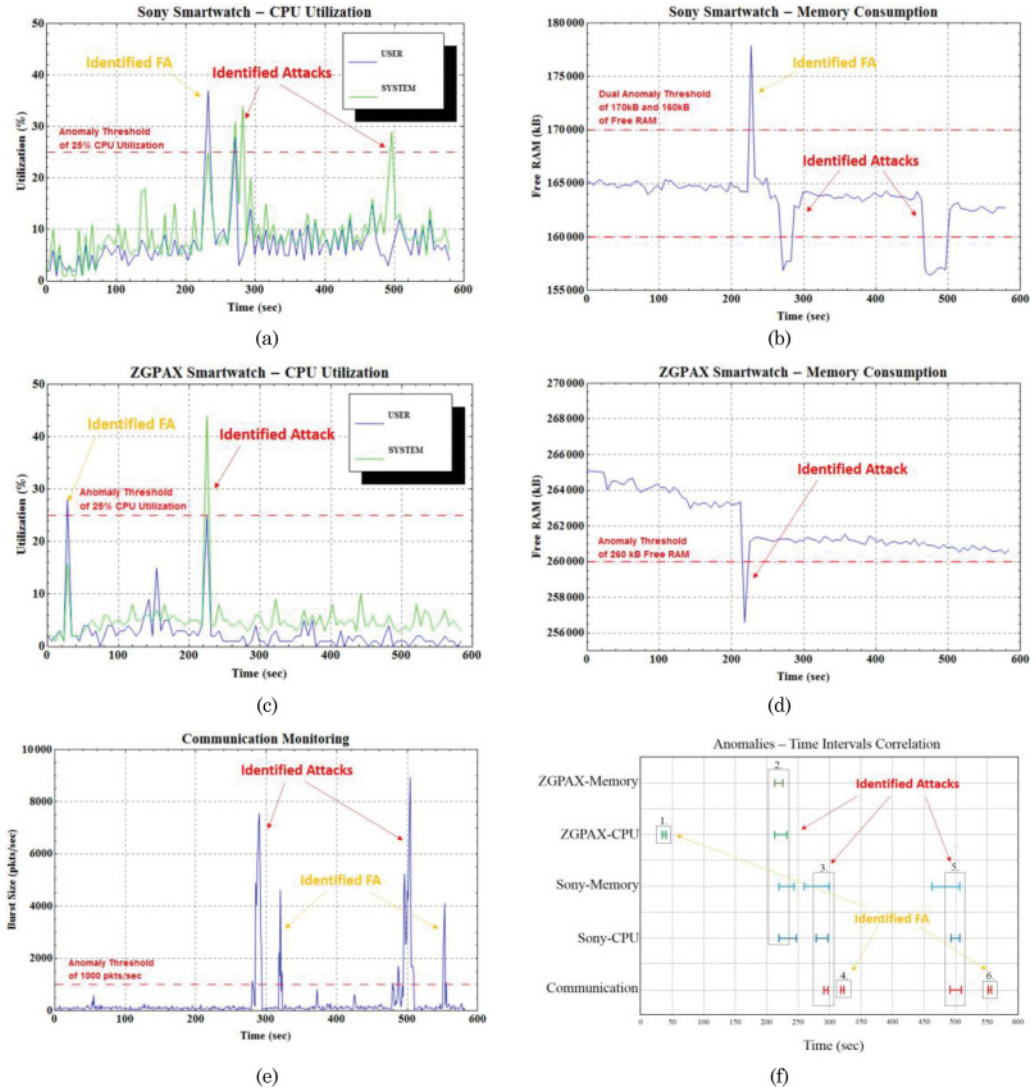


Fig. 6. The results obtained from the testing process, including the internal status of the WIoT-DUTs, based on CPU utilization (user and system perspectives in percentages) and memory consumption (in kB, from the free RAM point of view): (a) Sony smartwatch CPU utilization; (b) Sony smartwatch memory consumption; (c) ZGPAX smartwatch-phone CPU utilization; (d) ZGPAX smartwatch-phone memory consumption; (e) Communication monitoring recorded during the testing process; and (f) Correlation in the time dimension between the communication and WIoT-DUTs anomalies.

analysis is performed manually and individually on findings obtained for each new test executed in the testbed, utilizing the testing methodology presented below.

Figure 6 shows the internal status of the Sony smartwatch and the ZGPAX smartwatch devices, from both CPU utilization (user and system perspectives in percentages) and memory consumption (from the free RAM point of view in kB) that were recorded during the test, with respect to the testing time (in seconds). Regarding the free RAM parameter, it should be noted that when the application starts to run, the system memory decreases, as is shown in the respective graphs. Moreover, the communication

activity obtained during the test performed, in terms of packets per second (axis Y), with respect to the testing time in seconds (axis X) is presented in Figure 6(e). The graph was generated using the information extracted from the IO Graph tool (Wireshark). Finally, Figure 6(f) presents the correlation in the time dimension between all of the events/anomalies that occurred during the testing process.

Note that the emphasis here is on anomalies and not on the actual attacks that were executed. Recall the following points: we do not know when the attacks occurred, and the testbed should be able to deal with false alarm events. After the post-mortem procedure, we will be able to declare which of the anomalies were attacks and which were false alarm events. Therefore, as shown next, we manually analyzed each anomaly in order to understand its origin (the source of the deviation in the graph), and to find any correlation between the anomalies that occurred during the test.

Moreover, during the entire forensic analysis process presented here, we focus on the major deviations that show significant changes in the graphs. For the CPU analysis, an anomaly is defined as high CPU utilization, and from the memory perspective, an anomaly is defined as high memory consumption/releasing. For the communication analysis, an anomaly is defined as a burst of transmissions with high traffic volume. An anomaly threshold that was determined based on the results obtained is employed, in order to define the anomalies for each parameter (communication, CPU, and memory).

From the perspective of the Sony smartwatch device shown in Figure 6, it can be seen that there were three major anomalies that occurred in both the CPU utilization (Figure 6(a)) and the memory consumption (Figure 6(b)), which were defined by the selected thresholds. Note that, in the memory graph, we used a dual threshold for the analysis. We then define the time intervals for all the anomalies obtained. From the CPU point of view, the time intervals are as follows: the first anomaly is between 217 and 248, the second is 260-299, and the third anomaly is between 485 and 507 (seconds). From the free RAM point of view, the time intervals of the anomalies are: 217-243, 260-300, and 463-507 (seconds).

From the ZGPAX smartwatch-phone device analysis perspective, shown in Figure 6, it can be seen that there are two major anomalies in the CPU utilization graph (Figure 6(c)), and only one anomaly in the memory consumption graph (Figure 6(d)), both of which were defined by the selected thresholds. The time intervals for the anomalies in the CPU graphs are between 34 and 40 seconds and 212 to 231 seconds, and the anomaly shown in the memory graph is in the time interval of 212-225 seconds.

Next, we also examined the communication monitoring that was recorded during the testing process for one of the tests performed in our lab. For that matter, we manually analyzed the pcap file (generated by Wireshark) and the list of all available SSIDs in the network (recorded using the scripts we developed). As can be seen in Figure 6(e), four anomalies/deviations are shown in the graph with respect to a threshold of 1000 packets per second, as indicated by the red dashed line in the graph (Figure 6(e)). Using this threshold, we defined the time intervals in which the anomalies occurred. In this case, the first anomaly is defined as the time interval between 280 and 294 (seconds), the second is 318-323, the third is 479-510, and the fourth anomaly is defined as the time interval between 551 and 556 (seconds).

After defining the time intervals for all anomalies that occurred during the test (based on the analyses shown above), we can now try to find a correlation between these anomalies in order to identify and detect the context-based attacks executed in the testbed. Figure 6(f) presents the correlation in the time dimension between all the anomalies that occurred (denoted by points 1 to 6 in the graph). Regarding points 3 and 5 in Figure 6(f), as can be seen, there is an indication for correlation between the anomalies that occurred in the CPU and memory parameters of the Sony smartwatch

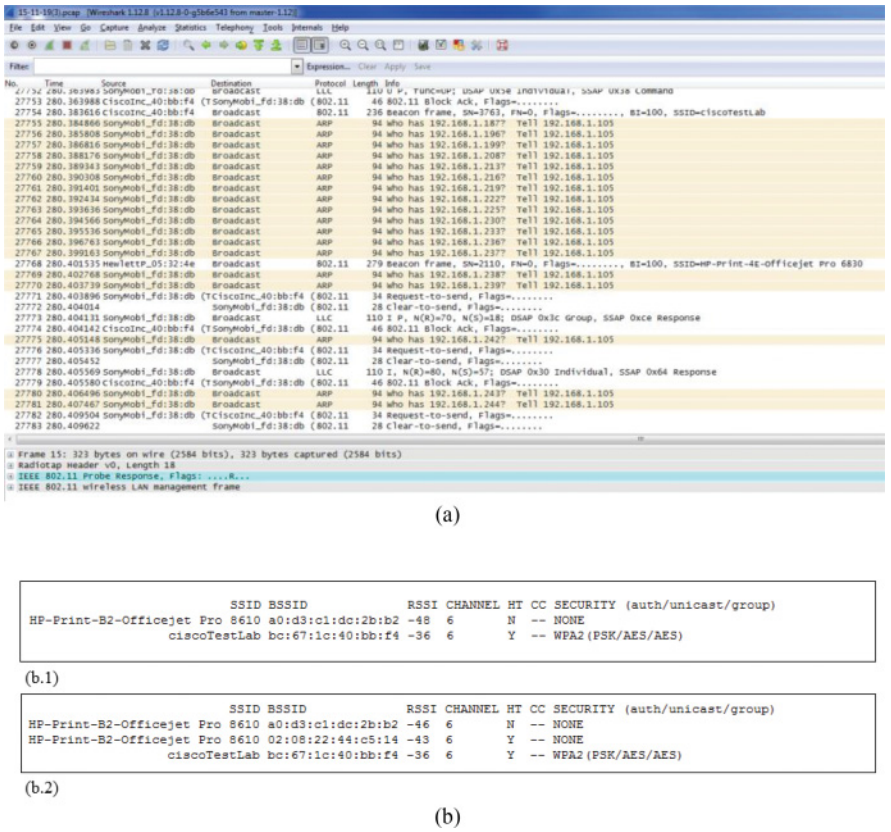


Fig. 7. (a) The network traces shown in the pcap file for one of the anomalies defined during the forensic analysis for the Sony device. As can be seen, a sort of network scanning is executed by the device; (b) Illustration of the fake access point attack in the testbed environment: (b.1) The SSID list before the attack was executed, (b.2) At the point of the attack, a new AP with the same SSID name as the Wi-Fi printer is added to the network (with the BSSID name of the ZGPAX smartwatch-phone device).

device to the anomalies that occurred in the communication space at these points of the test. Namely, at that time in the test the Sony device performs some activity which influences the network. Therefore, we further investigated these time intervals and analyzed the network traces from the pcap file at these indications of anomaly. From the analysis, it was found that the Sony smartwatch device executed some sort of network scanning at that time in the test, as illustrated in Figure 7(a). Therefore, we can actually declare that we managed to detect two network mapping attacks that were executed by the Sony smartwatch device at these points of the test.

Now, recall that as part of the information collected during the test (using the dedicated scripts), we also record the locations and times (the replayed information) that the GPS simulator transmits in the testbed. Accordingly, from the above analysis we can now obtain the specific locations and times that the network mapping attacks were executed by the Sony smartwatch device in the testbed (with respect to geo-fencing and time frame parameters of 50 meters and 2 minutes, respectively) as follows: the first attack occurred on 08-11-2015, 12:17:06 (this is the actual date and time of the path we recorded prior to the test) at location: latitude=31.26445, longitude=34.8128716, and the second attack occurred on 08-11-2015, 12:20:22 at location: latitude=31.26309, longitude=34.8116266.

Regarding point 2 in Figure 6(f), as can be seen, there is no anomaly in the communication space at that point of the test. However, there is correlation between the anomalies caused by the Sony device and those that were caused by the ZGPAX device (from both the CPU and memory parameters) at that point. This means that one of the devices, either the Sony or ZGPAX device, performs some activity in the testbed that may affect the other device (such that the anomalies of the affected device can be explained as internal memory management and CPU processing of that device due to this activity). To understand the origin of these anomalies, we further examined and analyzed the SSID list (the list of all Wi-Fi networks around our lab) that was recorded during the test. This examination indicated that at that point of the test (point 2 in Figure 6(f)) another access point was added to the list with the same SSID name as one that already existed in the network. The new/fake SSID name added is HP-Print-B2-Officejet Pro 8610, with a different BSSID name (MAC address 02:08:22:44:C5:14) than the actual printer, as shown in Figure 7(b). This BSSID name is related to the ZGPAX smartwatch-phone device (belonging to the fake AP that opened in the smartwatch device due to the attack). Accordingly, this indicates that we actually managed to detect the fake access point attack (fake Wi-Fi printer attack) that was executed by the ZGPAX smartwatch-phone device during the test. As before, now we can determine the specific location and time of the fake access point attack (again, with respect to the geo-fencing and time frame parameters), as 08-11-2015, 12:15:46 at location: latitude=31.2644366, longitude=34.8119433.

The other anomalies that occurred during the testing process are denoted by points 1, 4, and 6 in Figure 6(f). Point 1 refers to the anomaly that occurred in the CPU utilization of the ZGPAX device. At that point in time, the test has only begun. Therefore, this anomaly could be explained as internal CPU processing performed due to the synchronization of the ZGPAX smartwatch-phone device with the GPS signal. Note that at the beginning of each new test, the WIoT-DUTs had to resynchronize with the GPS signal transmitted in the lab. Regarding the last two points of anomaly (points 4 and 6 in Figure 6(f)), these referred to anomalies that occurred in the communication space. As can be seen, there is no correlation between these anomalies and the anomalies of the DUTs. Recall that only the Sony smartwatch and the ZGPAX smartwatch-phone devices were actively tested in the testbed during Phase 2 of the testing process. Meaning that besides the GPS simulator device, only these devices were active in the testbed. Therefore, these anomalies can be considered false alarms that were not caused by the WIoT-DUTs. Now, recall that during the execution of the test we actually injected two controlled false alarms by executing a port scan with the laptop on one of the devices in the network which was not a WIoT-DUT. Accordingly, we further examined the network traces in the pcap file during these anomalies' time intervals and found that these were actually port scan events. Hence, these anomalies were identified as the injected false alarms in the final forensic analysis procedure. The final reports for the full testing process presented above are generated by the Management and Reports Module (MRM, as illustrated by Phase 3 in Figure 1 by the green line); then the results are stored in the system database component and sent to the user.

The above examination demonstrates the testbed operation as a complete testing system. Note that since we developed both the malicious applications and the testing platform discussed, one may claim that we are testing what we already know (i.e., we know what to look for in the testbed). Regarding this point, at the time of writing this article, apart from Migicovsky et al. [2014] and Wang et al. [2015], we are not familiar with such context-based attacks executed using smartwatch devices. Accordingly, in order to be able to illustrate the functionality of our proposed security testbed and, as part of the proof-of-concept idea, we had to implement the attacks ourselves. However, we try as much as possible to perform real attack scenarios by using a random process,

whereby the locations and times of the attacks were randomly selected from a predefined route/path (the route is known but the specific locations of the attacks are not) and a predefined time frame, respectively (meaning, we use randomization in the space and time dimensions for the attacks). This means that once the attacks are triggered, we are not aware of the locations or times the attacks are executed and only learn these details after the event has occurred during the evaluation process (during the testing and forensic analysis procedure as discussed in the next subsections). In addition, the proposed security testbed is designed as a generic security testing platform for wearable IoT devices, regardless of the type of device under test, its hardware (sensors, etc.) and software (OS) configurations, and most importantly, the user applications installed on the device. Our comprehensive evaluation demonstrates the robustness of the testbed and its ability to withstand real context-based attacks that may be carried out by compromised wearable IoT devices in the future.

5. RELATED WORK

In this section, we discuss related work, focusing on commercial wearable applications and implementations and emphasizing security and privacy considerations.

A security testbed platform designated for the wearable device environment, called SecuWear, aimed at hardware and software vulnerability assessment, was proposed by Hale et al. [2015]. The proposed platform consists of several open source technologies, including: MetaWear, Apache Cordova, Ubertooth One, and Django. While this security testing platform facilitates assessment of vulnerabilities for the wearable device environment, it is targeted solely at BLE (Bluetooth low energy) security testing and was tested using only basic attack vectors. In addition, it may result in false alarms by identifying unrelated events as security problems. However, it should be noted that the SecuWear platform can be integrated into our proposed security testbed framework as part of the standard security testing module discussed in Section 3, under the vulnerability scan submodule for BLE testing scenarios.

A comprehensive security examination was performed for a common tracker device, the Fitbit Flex fitness device, by Cyr et al. [2014]. The researchers analyzed the device itself, inspected the Bluetooth connectivity between the device and a paired smartphone device, examined the Fitbit Android app installed on the smartphone, and analyzed the communication between that application and the Fitbit web service. They found several security and privacy violations that referred to the type and amount of data collected by the Fitbit device, and to the pairing process, where the smartphone sent the MAC addresses of all nearby Fitbit devices to the Fitbit server (this information could be used to track other Fitbit users). They also found that the BLE credentials can be sent in plaintext from the web service (Fitbit server) to the smartphone, potentially leaving the smartphone vulnerable to MITM attacks.

Migicovsky et al. [2014] introduced a contextual attack, where a group of dishonest students inconspicuously collaborated and cheated on multiple-choice exams (the context) in real-time by utilizing the Pebble smartwatch device. The proposed attack system, called ConTest, included a malicious lightweight application that was installed on the smartwatches and interacted with a cloud-based service that coordinated answers shared by the users during the exam. To the naked eye the watch seemed perfectly innocent where the display of the time and date was concerned, but for the attackers it was a perfect attack surface, since the data (the answers) were encoded on the smartwatch device's screen by inverting a small number of pixels in the date and time display. This simple case scenario illustrates how individuals can maliciously exploit wearable devices for their own advantage, without alerting those around them.

Another contextual attack using the smartwatch device was presented by Wang et al. [2015]. In this article, the authors managed to utilize a smartwatch device in order to

leak sensitive typed information about the user. The proposed attack system, called MoLe (Motion Leaks through Smartwatch Sensors), examined the accelerometer and gyroscope built-in sensors of the smartwatch device while the user (a wearer) typed on a standard keyboard. By combining the results obtained by linking the user's wrist micro-motions while typing with a known list of valid English words, it was possible to identify the typed words with reasonable accuracy. This attack exemplifies how an advanced attacker can exploit a regular smartwatch device within the context of keyboard typing, in order to compromise the privacy of the user.

On the other hand, several other works utilized wearable devices, specifically smartwatch devices, as an automated security appliance for smart environments. For example, Al-Muhtadi et al. [2001] proposed a security solution for active spaces and smart rooms. However, the problem with their solution is the performance degradation of the device in terms of battery life and memory and computational consumption. Sun et al. [2008] suggested an efficient security mechanism using a wearable token system that aimed to provide convenient and simple authentication and session key establishment services. The authors managed to reduce performance overhead, in terms of storage, computation, and communication costs, by implementing efficient key management and communication components, as well as using the transient authentication mechanism. Nonetheless, in general, the main problem with automating security services using mobile devices – and especially wearable devices – is that they are highly vulnerable to theft or loss.

Wearable computing technology is also considered part of the Wireless Body Area Network (WBAN), as discussed in Tufail and Islam [2009]. Similar to cases involving commercial wearable devices, WBAN deployments present security challenges [Lim et al. 2010], since sensitive information about the patient is collected and transferred through the Internet. Aside from an invasion of privacy, more direct threats to the user also exist with eHealth medical wearable devices [Halperin et al. 2008], for example, one could try to attack a patient's pacemaker, potentially causing heart malfunction.

Doukas et al. [2012] presented a security system which aggregated health contextual sensor data using a designated IoT gateway. Digital certificates and PKI data encryption is then employed by the gateway in order to securely transfer the information to the cloud service. A security testbed for the eHealth IoT application domain was proposed by Berhanu et al. [2013] as part of the ASSET (Adaptive Security for Smart Internet of Things in eHealth) project. The authors presented architecture consisting of a set of commercial off-the-shelf products and open source software, where low-power sensing objects collected and sent a patient's medical information to an eHealth application in the cloud through a smartphone device (as a WBAN gateway). Their work focused mainly on an energy consumption issue rather than security considerations.

6. DISCUSSION AND FUTURE WORK

Wearable computing is an emerging, ubiquitous technology in the Internet of Things (IoT) ecosystem, where wearable devices, such as activity trackers, smartwatches, smart glasses, and more, define a new Wearable IoT (WIoT) segment as a user-centered environment. However, the extensive benefits and application possibilities provided by wearable computing are accompanied by major potential compromises in data privacy and security, since any smart wearable device becomes a security risk. In addition, analyzing the security of such devices is a complex task due to their heterogeneous nature and the fact that these devices are used in a variety of contexts.

Therefore, in this article, we propose an innovative security testbed framework for wearable IoT devices. The proposed testbed is designed to perform traditional security testing, including discovery, vulnerability scans, and penetration tests, as well as to execute advanced contextual security testing by realistically simulating the

environment in which wearable IoT devices exist (such as location, lighting, movement, etc.), in order to identify and detect context-based attacks that may be carried out by malicious applications installed on such devices.

Security analysis frameworks and standards, such as ITSEC and Common Criteria, have been used by government and military organizations for evaluating security-critical devices. Such frameworks are used to evaluate the adherence of a tested device to a desired level of security and assurance in a white-box approach. Such evaluations are usually time and resource consuming and involve the evaluation of the complete process of specification, implementation, and evaluation of the product/system (and therefore assume access to documentation, design, and development processes and code). The proposed testbed, however, aims for a black-box approach in which we assume that only the final product is available. The proposed framework is also targeted specifically at IoT devices and designed to execute relevant security tests with minimal human intervention. The downside of such an approach is that it cannot provide a mapping of the IoT device (or its functions) to a specific security/assurance level, but rather lists the results of the tests.

In general, detecting context-based attacks requires executing a security test within different contexts. We can assume that simulating all possible contexts in the testbed is not feasible due to the potentially large number of context variables (such as location, time, sound level, motion, etc.) and the infinite number of values for each contextual element. For example, consider the geolocation as a context; although we use SATGEN GPS simulation software,[1] which can be used to create a different user-generated trajectory that can be replayed by the LabSat GPS simulator, it will be impossible to run a context-based test that covers all possible locations. Therefore, we define two types of context-based tests: targeted and sample tests. In a targeted test we assume that a bounded set of contexts to be evaluated by the testbed is provided as an input to the testing process. For example, an IoT device that is going to be deployed in a specific organizational environment will be tested with the organization's specific geographical location, given the execution limits of the testbed. In a sample test, a subset of all possible contexts (those that can be simulated) is evaluated. This subset is selected randomly according to a priori assumptions about contexts of interest (for example, malicious activity is usually executed at night, the device is installed in a home environment).

We demonstrated a proof-of-concept for the testbed operation in Section 4 and showed that our proposed security testbed can serve as a new tool for measuring and analyzing the security of wearable IoT devices in different case scenarios.

In future work, we intend to finalize the implementation of the WIoT security testbed based on the current design and to enhance the testbed's capabilities in order to support all of its features – including the user interface, infrastructure, and security modules. Such operational implementation of a WIoT security testbed will provide us with information about additional requirements for an IoT security testbed, as well as its potential limitations, which is essential for expanding the testbed to other IoT devices used in home and office environments. We also intend to use the enhanced testbed to check "Smart City" IoT devices that are used to collect information from the city environment. Moreover, several open issues related to wearable technology can also be considered in future work, such as developing a lightweight antimalware, or firewall designated specifically for wearable devices, and testing them in our proposed testbed. The idea of detecting whether a wearable device is part of a botnet could also be explored in future work.

[1]http://www.labsat.co.uk/index.php/en/products/satgen-simulator-software.

ACKNOWLEDGMENTS

The authors would like to thank Michael Bohadana and Omer Porzecanski for their technical assistance with the experiments conducted as part of this research.

REFERENCES

John Almasy. 2015. How do wearables fit in your enterprise? Retrieved November 28, 2015 from http://blogs.unisys.com/mobility/how-do-wearables-fit-in-your-enterprise/.

Jalal Al-Muhtadi, Dennis Mickunas, and Roy Campbell. 2001. Wearable security services. In *2001 International Conference on Distributed Computing Systems Workshop*. 266–271. IEEE.

Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The internet of things: A survey. *Computer Networks* 54, 15 (2010), 2787–2805.

Yared Berhanu, Habtamu Abie, and Mohamed Hamdi. 2013. A testbed for adaptive security for IoT in eHealth. In *International Workshop on Adaptive Security* 5. (2013), ACM.

Bitdefender. 2014. Bitdefender research exposes plain-text android wearable devices communication. Video. Retrieved November 28, 2015 from https://www.youtube.com/watch?t=149&V=utVnrq5uCuM.

Brent Blum. 2015a. Are your wearables safe from cyber-security threats? Retrieved November 28, 2015 from https://www.accenture.com/us-en/blogs/blogs-are-your-wearables-safe-from-cyber-security-threats.

Brent Blum. 2015b. How to protect your wearables implementation from cyber-security threats. Retrieved November 28, 2015 from https://www.accenture.com/us-en/blogs/blogs-how-to-protect-your-wearables-implementation-from-cyber-security-threats.

John Brandon. 2014. Wearable devices pose threats to privacy and security. Retrieved November 28, 2015 http://www.foxnews.com/tech/2014/06/18/wearable-devices-pose-threats-to-privacy-and-security.html.

Ken Briodagh. 2015. Wearable security is a matter of establishing standards. Retrieved November 28, 2015 from http://www.iotevolutionworld.com/m2m/articles/401623-wearable-security-a-matter-establishing-standards.htm.

Gerald Combs. 2007. Wireshark–A network protocol analyzer. https://www.wireshark.org/.

Charles Cooper. 2015. Latest security challenges: Wearables. Retrieved November 28, 2015 from http://theartofthehack.com/latest-security-challenge-wearables/.

Britt Cyr, Webb Horn, Daniela Miao, and Michael Specter. 2014. Security analysis of wearable fitness devices (fitbit). Massachusetts Institute of Technology (MIT). Retrieved November 29, 2015 from https://courses.csail.mit.edu/6.857/2014/files/17-cyrbritt-webbhorn-specter-dmiao-hacking-fitbit.pdf.

Brian Donohue. 2014. Same security threats, different devices: Wearables and watchables. Retrieved November 28, 2015 from https://blog.kaspersky.com/same_security_threats_new_devices/6015/.

Charalampos Doukas, Ilias Maglogiannis, Vassiliki Koufi, Flora Malamateniou, and George Vassilacopoulos. 2012. Enabling data protection through PKI encryption in IoT M-Health devices. In *2012 IEEE 12th International Conference on Bioinformatics & Bioengineering (BIBE)*. 25–29. IEEE.

Dan Goodin. 2015. Police body cams found pre-installed with notorious conficker worm. Retrieved November 28, 2015 from http://arstechnica.com/security/2015/11/police-body-cams-found-pre-installed-with-notorious-conficker-worm/.

Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. 2013. Internet of things (IoT): A vision, architectural elements, and future directions. *Fut. Gen. Comput. Syst.* 29, 7 (2013), 1645–1660.

Matthew L. Hale, Dalton Ellis, Rose Gamble, Charles Waler, and Jessica Lin. 2015. SecuWear: An open source, multi-component hardware/software platform for exploring wearable security. In *2015 IEEE International Conference on Mobile Services (MS)*. 97–104. IEEE.

Daniel Halperin, Thomas S. Heydt-Benjamin, Benjamin Ransford, Shane S. Clark, Benessa Defend, Will Morgan, Kevin Fu, Tadayoshi Kohno, and William H. Maisel. 2008. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *IEEE Symposium on Security and Privacy, 2008 (SP 2008)*. 129–142. IEEE, 2008.

Teena Hammond. 2014. The scary truth about data security with wearables. Retrieved November 28, 2015 from http://www.techrepublic.com/article/the-scary-truth-about-data-security-with-wearables/.

Shivayogi Hiremath, Geng Yang, and Kunal Mankodiya. 2014. Wearable internet of things: Concept, architectural components and promises for person-centered healthcare. In *2014 EAI 4th International Conference on Wireless Mobile Communication and Healthcare (Mobihealth)*. IEEE, 2014.

William John Holden. 2015. PIPS – The Pamn IP Scanner: A wrapper for nmap, cross-compiled for ARM android. https://github.com/wjholden/PIPS/tree/master/app/src/main/java/com/wjholden/nmap.

iTrust. 2015. Cyber security patrol (CSP). Retrieved November 29, 2015 from http://itrust.sutd.edu.sg/research/projects/cyber-security-patrol/.

Khyati Jain. 2015. Ransomware attacks threaten wearable devices and internet of things. Retrieved November 28, 2015 from http://thehackernews.com/2015/08/ransomware-android-smartwatch.html.

Kaspersky Labs. 2014. Wear the danger: Kaspersky lab experts warn of security risks facing wearable connected devices. Retrieved November 28, 2015 from http://www.kaspersky.com/au/about/news/virus/2014/wear-the-danger.

Linda Lee, Serge Egelman, Joong Hwa Lee, and David Wagner. 2015. Risk perceptions for wearable devices. *arXiv preprint arXiv*:1504.05694 (2015).

Mike Lennon. 2015. All smartwatches vulnerable to attack: HP study. Retrieved November 28, 2015 from http://www.securityweek.com/all-smartwatches-vulnerable-attack-hp-study.

Shinyoung Lim, Tae Hwan Oh, Young B. Choi, and Tamil Lakshman. 2010. Security issues on wireless body area network for remote healthcare monitoring. In *2010 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC)*. 327–332. IEEE.

John Lindström. 2007. Security challenges for wearable computing-a case study. In *2007 4th International Forum on Applied Wearable Computing (IFAWC)*. 1–8. VDE, 2007.

Clayton Locke. 2014. Top 3 security tips for wearable devices. Retrieved November 28, 2015 from http://www.intelligentenvironments.com/info-centre/blog/top-3-security-tips-for-wearable-devices.

Gordon Lyon. 2009. Nmap-Free security scanner for network exploration and security audits. https://nmap.org/.

Alex Migicovsky, Zakir Durumeric, Jeff Ringenberg, and J. Alex Halderman. 2014. Outsmarting proctors with smartwatches: A case study on wearable computing security. In *Financial Cryptography and Data Security*, 8437, 89–96. Springer Berlin. 2014.

Peter Nguyen. 2014. Wearable tech and personal security breaches: 6 things to know. Retrieved November 28, 2015 from http://blog.hotspotshield.com/2014/12/16/wearable-tech-and-personal-security-breaches/.

Offensive Security. 2016. Kali linux–an advanced penetration testing linux distribution used for penetration testing, ethical hacking and network security assessments. https://www.kali.org/.

Charith Perera, Chi Harold Liu, and Srimal Jayawardena. 2015. The emerging internet of things marketplace from an industrial perspective: A survey. *IEEE Trans. EmergTopics Comput.*

Jenna Puckett. 2014. How to prevent wearable devices from ruining your information security. Retrieved November 28, 2015 from http://www.fiercecio.com/story/how-prevent-wearable-devices-ruining-your-information-security/2014-11-25.

Marc Rogers. 2013. Hacking the internet of things for good. Retrieved November 28, 2015 from https://blog.lookout.com/blog/2013/07/17/hacking-the-internet-of-things-for-good/.

Da-Zhi Sun, Jin-Peng Huai, Ji-Zhou Sun, Jia-Wan Zhang, and Zhi-Yong Feng. 2008. A new design of wearable token system for mobile device security. *IEEE Trans. Consum. Electron.* 54, 4, 1784–1789.

Melanie Swan. 2012. Sensor mania! the internet of things, wearable computing, objective metrics, and the quantified self 2.0. *J. Sens. Actuat. Netw.* 1, 3, 217–253.

Nils Ole Tippenhauer, Christina Pöpper, Kasper Bonne Rasmussen, and Srdjan Capkun. 2011. On the requirements for successful GPS spoofing attacks. In *18th ACM Conference on Computer and Communications Security*. ACM, 2011.

Mellisa Tolentino. 2013. 4 Security Challenges for Fitbit, Google Glass + Other Wearable Devices. Retrieved November 28, 2015 from http://siliconangle.com/blog/2013/05/30/4-security-challenges-for-fitbit-google-glass-other-wearable-devices/.

Farhana Tufail and M. Hassan Islam. 2009. Wearable wireless body area networks. In *International Conference on Information Management and Engineering, 2009 (ICIME'09)*. IEEE, 656–660.

David Upton. 2014. 5 essential wearable tech security tips. Retrieved November 28, 2015 from http://betanews.com/2014/12/09/5-essential-wearable-tech-security-tips/.

He Wang, Ted Tsung-Te Lai, and Romit Roy Choudhury. 2015. MoLe: Motion leaks through smartwatch sensors. In *21st Annual International Conference on Mobile Computing and Networking*. ACM, New York, 155–166.

Received December 2015; revised June 2016; accepted July 2016