IEEE International Conference on Information Networking (ICOIN), Barcelona, Spain, 2020

A Monitoring and Control Gateway for IoT Edge Devices in Smart Home

Fadi Aloul, Imran Zualkernan, Shams Shapsough, Mohammed Towheed
Department of Computer Science and Engineering
American University of Sharjah
Sharjah, United Arab Emirates (UAE)
{faloul, izualkernan, b00046058, b00060653}@aus.edu

***Abstract*—Security aspects of Internet of Things (IoT) systems are not well understood. Therefore, the rapid adoption of systems using IoT technologies is poised to create a large number of exposed systems with new security exploits and vulnerabilities. This is especially critical for systems deployed in a smart home environment. Any vulnerabilities or security issues in such a system can compromise the physical security of residents and the exchange of data by unsophisticated edge devices can have severe ramifications if not addressed properly. Edge-devices contribute significantly to security risks for these IoT systems. Edge-devices are based on resource-constrained, wireless-enabled microcontrollers typically running primitive operating systems. The resource-constrained nature of edge devices, in tandem with IoT network protocols, creates many unique security challenges. This work presents a generic IoT Monitoring and Control Gateway (MCG) providing edge security testing and control measures. These measures allow homeowners to enforce levels of security to the edge layer consisting of home appliances and gadgets. The MCG found many vulnerabilities in various home appliances like cameras, routers and music streaming systems.**

Keywords— IoT Edge Devices, Security, Control, Smart Home

I. Introduction

The popularity of Internet of Things (IoT) has enabled rapid development of systems and applications. These systems utilize smart sensors and heterogeneous networks in a variety of domains such as healthcare, industrial automation or smart spaces. The data exchanged in applications, such as home remote monitoring using cameras, is highly sensitive and personal and should not be accessible by unauthorized parties. Security of IoT devices has a variety of dimensions that can be addressed using a multitude of techniques [1]. For example, measures must be taken to ensure sensitive data is kept confidential and secure. One proposed approach is to implement security at the network layer [2]. Edge nodes are small devices that collect information from the surrounding environments. For Example, such devices may monitor a resident's personal room or collect climate data like temperature or humidity of the home, or the presence or absence of residents. An example of exploiting these nodes was highlighted when the Mirai botnets penetrated edge nodes in a smart home and carried out massive Distributed Denial of Service (DDoS) attacks against well-known websites like Netflix and Github [3]. A typical edge device collects data using sensors and transmits this data to the IoT network. Edge devices need to optimize power consumption because they are often remotely located and rely on small batteries for power. While some work has addressed the security of edge devices [4], many security holes in edge nodes of IoT systems are still not well understood. This lack of understanding is reflected in a recent increase of cyberattacks that compromised and exploited edge devices in several IoT systems. IoT research has gone into the development and deployment of novel and experimental IoT systems, with less focus on securing them [5].

This paper presents a generic IoT Monitoring and Control Gateway (MCG) that provides edge security testing and control measures. These measures allow home owners to enforce levels of security to the edge layer consisting of appliances and other home gadgets. The MCG can be integrated with different back-end systems using a robust Application Programming Interface (API) that allows configuration, monitoring, and error checking. The availability of such assessment tool and the enforcement of security requirement designed from a business context provide a protection and monitoring capability to the most vulnerable parts of IoT systems and smart homes.

The main contributions of this paper are:

- A security API which is lightweight and resource friendly API in terms of CPU and power usage.
- The API is independent of hardware and software technologies and hence can run on any embedded device such as a Raspberry Pi or a machine that runs Kali Linux.
- The API utilizes open source technologies such as the Kali tools that come pre-installed with every Kali distribution and is budget friendly.
- In terms of utility, the API provides a preliminary security check and insures a standard basic level of security control for the devices in the smart home.

The rest of the paper is organized as follows. A summary of related work is discussed in Section II. This is followed by a description of the design approach and the proposed architecture in Sections III and IV, respectively. Experimental results are shown in Section V. The paper ends with a conclusion in Section VI.

II. Related Work

In [2], a system is proposed for IoT Security as an alternative to solutions that are embedded into a device. They proposed a three-party architecture in which a specialist provider offers security-as-a-service. The susceptibility of IoT devices to botnet attacks was demonstrated in [3] where Mirai botnets were used to take over IoT devices and carry out Distributed Denial of Service (DDoS) attacks against important websites like Netflix and Github. Inter-device communication lies at the core of IoT

and, therefore should be secure. This issue was explored in [6] where communication packets between IoT devices over WIFI and Bluetooth were captured using Wireshark, Kismet and Ubertooth. These packets were analyzed and reported. The researchers in [7] approached IoT security in a 'hands-on' manner where three use cases were constructed to show (1) leakage of personally identifiable information (PII), (2) leakage of sensitive user information, and (3) unauthorized execution of functions. The results were analyzed to show common vulnerabilities with the IoT. The Sablo distributed security platform for IoT was discussed in [8]. An additional security testing phase was proposed in addition to the regular execution in the IoT system to analyze traffic real-time. The researchers in [9] provided a frame-of-reference for those beginning with IoT device vulnerabilities. This research used the Kali tool Nessus for common vulnerability assessment of a wide range of IoT devices like Phillips Hue Bulb and Amazon Echo. Machine learning has great potential in IoT security, and an approach was presented in [10] where a threat model was developed which could recognize potential attacks against various layers in a layered IoT framework. Researchers in [11] developed a security testbed for wearable IoT devices which could also report on the communication between wearables and their application counterparts like Fitbit Studio installed on user's smartphones. The researchers in [12] presented a comprehensive taxonomy of IoT vulnerabilities and in [13] the authors explained what a penetration testbed for IoT devices should contain. The researchers in [14] conducted a holistic security analysis of IoT systems starting with standard penetration testing of individual devices to data and context-based tests. Lastly, in [15], the researchers demonstrated the working of an IoT security testbed which is an isolated system and requires dedicated hardware to run. The primary utility of this system is to test if IoT devices meet minimum security requirements.

III. Design Approach

The primary design approach for research presented here consists of leveraging custom-made programs and penetration software that target specific vulnerabilities. There are a number of testbeds aimed at testing the security of embedded systems [16-21]. In general, these programs are designed to investigate Wireless Sensor Networks (WSNs) and embedded systems that utilize WIFI, ZigBee and Bluetooth protocols. The purpose of each of these evaluations is to assess different aspects of general-purpose embedded systems like performance, security and resource utilization. These programs usually check for generic vulnerabilities such as weak and trivial passwords, Denial of Service (DoS) attack opportunities, unprotected data, etc. Open-source and proprietary tools [16-26] have also been used for testing small wireless sensor devices and heterogeneous communication devices. Such tools can enable access and control into various edge nodes as well as provide monitoring capabilities. Moreover, these tools are complemented with user-defined scripts and tests.

IV. Proposed Architecture

The aim of this work was to develop a Monitoring and Control Gateway (MCG) to check and verify that connected IoT devices are properly assessed for vulnerabilities and meet a set of security requirements. MCG has the following tasks:

- Detect and identify connected edge devices
- Get and store device information
- Preform basic black-box penetration testing
- Share security report, notifications and messages with admin

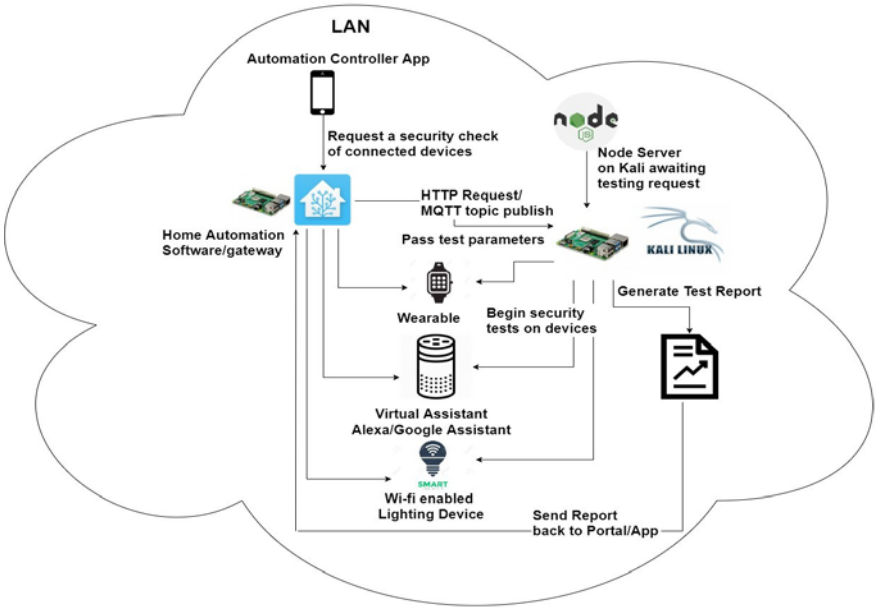

Fig. 1. Monitoring and Control Gateway (MCG) API Architecture

TABLE I. TESTS CONDUCTED BY THE MONITORING AND CONTROL GATEWAY (MCG).

Test	Tool	Description
Port scanning & Basic Reconnaissance	Nmap	Gather and collect information on the device (if applicable) like operating system, IP and MAC addresses as well as open ports by utilizing TCP SYN stealth scans as well as UDP scans on ports 80 (HTTP), 443 (HTTPS/SSL), 22 (SSH), 25 (SMTP), 110 (remote mail server), 445 (SMB).
Common Vulnerabilities Check	Vulscan and nmap-vulners	Uses well known Common Vulnerability Exploit (CVE) databases to report CVEs found on the ports scanned and for the device itself
Active DoS Pen-test	Xerxes	Perform a controlled Sync flood and Ping of Death (PoD) attacks against a specific device and observing its response.
Additional CVE information and related exploits	Metasploit and searchsploit	Find more information on detected CVEs for device or for any port
Service alive/Web server running	Nikto	Find out if any web server is running (HTTP) or Server Message Block (SMB) is alive or remote desktop service is enabled (RDP)
Hidden links/Missing headers	Dirb	Analyze communication packets from http, https, ssl for missing headers or hidden links to resources
Service credential dictionary attack	Hydra	Conducts dictionary attacks against open ports running services like ssh, smb, or rdp

Fig. 1 shows the overall architecture of the MCG. The MCG API is written in Node.js and is running on a Kali Linux based Raspberry Pi [22] waiting for HTTP or MQTT requests. A user of a home automation platform (e.g. Home Assistant) which is also running independently on another Raspberry Pi or a similar home automation platform, can request for a security check of connected devices on the home network. This HTTP or MQTT request reaches the MCG, and based on request parameters such as the host addresses to check or the tests to carry out, the MCG begins security tests on the smart home devices and gadgets. This is done asynchronously and once the tests are completed, a report is generated which is sent back to the platform or the application via MQTT or HTTP.

The setup shown in Fig. 1 is inside a local area network, and the API are packaged and installed locally within the network of anyone who wishes to test their smart home system and devices. The system is flexible enough and can handle more complex hardware other than the edge devices. For demonstration purposes, a Raspberry Pi [21] was used as the base hardware for the MCG. The MCG receives commands from a specific backend and execute them as well as store device information. That information alongside the security reports is stored in a Redis database [16]. Metaspolit [23], Hydra [24] and other tools were tested on the RPi create a professional Pen-test unit that is both effective and accessible. In regard to the security testing functionality, MCG currently performs the tasks/tests as shown in Table I.

The MCG after conducting the above tests, generates a report and notifications that can be reviewed by an administrator in order to commission and decommission devices from the system and network.

The MCG's penetration testing functionality is intended to extract preliminary information on the overall security of the home system. However, MCG also needs to take into account that health and operation of the system as well. The pen-test unit should not inadvertently take down or interrupt system operations. For that purpose, system health/status monitors are used to periodically check the system's current status and report back [25].

A. API

The MCG services are exposed as an asynchronous API which allows any IoT based system to run its monitoring and error checking operations on its edge devices. For example, a home automation platform makes API calls to the MCG to configure and schedule a set of tests, monitor the currently connected devices and network traffic, and lastly check up on the system by analyzing generated reports and notifications for each device. These calls are done via HTTP or MQTT requests. The tests are conducted at the gateway between the MCG and the connected IoT devices. The Security API constantly checks for new vulnerabilities in known IoT devices and updates tests. Moreover, self-testing functions are of utmost importance as the system needs to assess its own integrity and provide a layer of trust to users. The API is implemented in an asynchronous pattern for both HTTP and MQTT requests, which allows the testing system to start tests and be notified when a change is detected without having multiple REST calls and acknowledgments. An asynchronous RESTful route is implemented for an HTTP request while MQTT's asynchronous nature is utilized by default as the calling platform will subscribe to the result topics and can continue its usual tasks without waiting for the report.

When a report is generated by the MCG, the subscribing platform receives this response. The advantage of having asynchronous API calls over synchronous calls depends on the application. In the case of the Security API, performing actions like a DoS Attack will start a coordinated DoS attack against the target and will stop after the specified amount of time which could be a few minutes to a few hours. It is not feasible to block the entire program while the attack finishes which is what a synchronous API would do. Another

TABLE II. SECURITY HOLES OF TYPICAL HOME DEVICES FOUND USING THE MONITORING AND CONTROL GATEWAY (MCG).

Device	Scan Results
Home Linksys Wireless Router	Device and OS details acquired, TCP/IP Fingerprint found, 12 Open ports and services running on them, no common vulnerabilities (CVE), 1 hidden web object found on port 80 (web server), 3 missing headers found in GET requests, 1 brute force login credential found
D-Link Camera (DCS-8000LH)	Device and OS Details acquired, TCP/IP Fingerprint found, 4 open ports and services running on them, 2 Common Vulnerabilities (CVE) found, no web content or web servers, found SSL vulnerability and related issues
Google Chromecast	Device and OS details acquired, TCP/IP fingerprint found, 5 open ports and services running on them, no exploits or CVEs, found one web content URL on port 8008, 3 missing headers found in GET requests
Popcorn Hour Music Player	Device and OS Details found, TCP/IP Fingerprint found, 11 TCP, 4 UDP open ports and services running on them, 1 Common Vulnerability found – Slowloris Denial of Service Attack, 3 web content found on ports 2020, 8008 and 8883, total 11 important headers missing on ports 2020, 8008 and 8883, web server running on device, can be exploited
Yi Antscam Camera	Device details but no host details found, no TCP open ports, TCP/IP fingerprint found, 42 Open UDP ports and services running on them, no Exploits or CVEs, no web content or web servers found on device.
Amazon Echo (Alexa)	Device details found, 2 TCP open ports, 192 UDP open/filtered ports and services running on them, TCP/IP fingerprint found, no exploits or CVEs and no web content or web servers found running on device

example is the information gathering action which runs a tool like *Nmap* to capture open ports of a device and depending on the scope of the scan its run time could vary which would block the entire program for an unspecified amount of time till the method returns. We might have tests scheduled for more than one device to all begin at the same time and blocking the entire program till one API call returns is not ideal in this scenario. What we rather need is for results to come in asynchronously like the sniffer in the Security API, which would capture packets for a specified amount of time, and have them as they are sniffed, rather than call the method every time to capture few packets and then stop. The ideal flow would be to start the sniffer and analyze packets as they come in.

V. RESULTS

The Monitoring and Control Gateway (MCG) was used to test a variety of typical home devices from various vendors. The devices included a low-end home router, home cameras and music streaming devices like Google Chromecast and a streaming music box. As Table II shows, in most cases information like device and OS details was easy to acquire. In addition, many open ports were identified. In some cases, common vulnerabilities (CVEs) were found as well. In addition, Web content URLs were also found on multiple devices. Finally, many devices had missing headers in the GET commands. An example of a final report for one of the devices in Table II is shown in Fig. 2. It highlights the results of port scanning and information gathering followed by vulnerability assessment for discovered CVEs and lastly the discovery of hidden web content and URLs on the device.

VI. CONCLUSION

As the use of Internet of Things (IoT) enabled devices increases, so will attack vectors and the severity of attacks causing new vulnerabilities to come to light and in some cases exploited routinely. The low-cost, low-power nature of many IoT edge-devices offers a challenge for both developers and security researchers. This research provides the first few steps in addressing these challenges and developing a low-cost security control and monitoring measure to add another layer of visibility and security to IoT systems. The results attained from the MCG for the devices in Table II comply with the Open Web Application Security Project's (OWASP) IoT Security Guidance for Manufacturers. As shown in [26], this guidance aims to help manufacturers build more secure products in the IoT space and is at the basic level, giving builders of products a basic set of guidelines to consider from their perspective. As per these guidelines, the results were in line with indices, I1: Insecure Web Interface, I2: Insufficient Authentication/ Authorization, I3: Insecure Network Services, I4: Lack of Transport Encryption and I10: Poor Physical Security. Future work for the MCG includes adding more security tests in compliance with the standards of the time, to enforce self-integrity check and to implement network monitoring.

ACKNOWLEDGMENTS

This project was funded by a research grant from Dubai Electronic Security Center (DESC) in United Arab Emirates (UAE).

TEST REPORT POPCORN HOUR MUSIC PLAYER

TCP PORT SCANNING –

PORT	STATE	SERVICE	REASON	VERSION
22/tcp	closed	ssh	reset ttl 64	
25/tcp	closed	smtp	reset ttl 64	
80/tcp	closed	http	reset ttl 64	
110/tcp	closed	pop3	reset ttl 64	
443/tcp	closed	https	reset ttl 64	
445/tcp	closed	microsoft-ds	reset ttl 64	
23/tcp	open	telnet	syn-ack ttl 64	
2020/tcp	open	http	syn-ack ttl 64	Syabas Popcorn Hour media player http config
4000/tcp	open	remoteanything?	syn-ack ttl 64	
5000/tcp	open	upnp?	syn-ack ttl 64	
6357/tcp	open	upnp	syn-ack ttl 64	
7000/tcp	open	afs3-fileserver?	syn-ack ttl 64	
8008/tcp	open	http	syn-ack ttl 64	
8118/tcp	open	privoxy?	syn-ack ttl 64	
8883/tcp	open	http	syn-ack ttl 64	Syabas Popcorn Hour media player BitTorrent interface
30000/tcp	open	tcpwrapped	syn-ack ttl 64	
39410/tcp	open	upnp	syn-ack ttl 64	

MAC Address: 00:06:DC:8C:E0:31 (Syabas Technology (Amquest))
Not shown: 65524 closed ports
Reason: 65524 resets

UDP PORT SCANNING –

Not shown: 196 closed ports
Reason: 196 port-unreaches

PORT	STATE	SERVICE	REASON	VERSION
137/udp	open	netbios-ns	udp-response ttl 64	Microsoft Windows netbios-ns (workgroup: WORKGROUP)
138/udp	open|filtered	netbios-dgm	no-response	
1900/udp	open|filtered	upnp	no-response	
5353/udp	open|filtered	zeroconf	no-response	

MAC Address: 00:06:DC:8C:E0:31 (Syabas Technology (Amquest))
Service Info: Host: PCH-A500; OS: Windows; CPE: cpe:/o:microsoft:windows

SYSTEM AND DEVICE DETAILS –

MAC Address: 00:06:DC:8C:E0:31 (Syabas Technology (Amquest))
Device type: general purpose
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.10

DISCOVERED VULNERABILITIES (CVE) –

CVE:2007-6750

#	Name	Disclosure Date	Rank	Check	Description
-	----	----------	----	-----	-----------
0	auxiliary/dos/http/slowloris	2009-06-17	normal	No	Slowloris Denial of Service Attack

Uptime guess: 0.022 days (since Sun Sep 29 15:16:06 2019)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=262 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: Device: media device; CPE: cpe:/h:syabas:popcorn_hour

WEB CONTENT AND HIDDEN URLS –

---- Scanning URL: http://192.168.1.153:8008/ ----
+ http://192.168.1.153:8008/deployment (CODE:200|SIZE:0)

---- Scanning URL: http://192.168.1.153:8883/ ----
+ http://192.168.1.153:8883/download.cgi (CODE:200|SIZE:1948)
+ http://192.168.1.153:8883/start.cgi (CODE:200|SIZE:7053)

Fig. 2. Sample Monitoring and Control Gateway (MCG) Test Report.

REFERENCES

[1] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of things (IoT) security: Current status, challenges and prospective measures," in *Proc. of the 10th International Conference for Internet Technology and Secured Transactions* (ICITST), London, pp. 336-341, 2015.

[2] V. Sivaraman, H. Gharakheili, A. Vishwanath, R. Boreli, and O. Mehani, "Network-level security and privacy control for smart-home IoT devices," in *Proc. of the IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications* (WiMob), pp. 163–167, 2015.

[3] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and Other Botnets," in *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[4] S. Shapsough, F. Aloul, and I. Zualkernan, "Securing Low-Resource Edge Devices for IoT Systems," in *Proc. of the IEEE International Symposium on Sensing and Instrumentation in IoT Era* (ISSI), Shanghai, China, 2018.

[5] C. (Defta) Costinela-Luminiţa and C. (Iacob) Nicoleta-Magdalena, "E-learning Security Vulnerabilities," in *Procedia - Social and Behavioral Sciences*, vol. 46, pp. 2297–2301, 2012.

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

[7] C. Kolias, A. Stavrou, J. Voas, I. Bojanova, and R. Kuhn, "Learning Internet-of-Things Security 'Hands-On,'" in *IEEE Security & Privacy*, vol. 14, no. 1, pp. 37–46, 2016.

[8] C. Săndescu, O. Grigorescu, R. Rughiniș, R. Deaconescu, and M. Calin, "Why IoT security is failing. The Need of a Test Driven Security Approach," in *Proc. of the 7th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, pp. 1–6, 2018.

[9] R. Williams, E. McMahon, S. Samtani, M. Patton, and H. Chen, "Identifying vulnerabilities of consumer Internet of Things (IoT) devices: A scalable approach," in *Proc. of the IEEE International Conference on Intelligence and Security Informatics* (ISI), pp. 179–181, 2017.

[10] J. Pacheco and S. Hariri, "IoT Security Framework for Smart Cyber Infrastructures," in *Proc. of the IEEE 1st International Workshop on Foundations and Applications of Self* Systems* (FAS*W), pp. 242–247, 2016.

[11] S. Siboni, A. Shabtai, N. Tippenhauer, J. Lee, and Y. Elovici, "Advanced Security Testbed Framework for Wearable IoT Devices," in *ACM Transactions on Internet Technology*, vol. 16, no. 4, pp. 26:1–26:25, December 2016.

[12] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations," in *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2702–2733, third-quarter 2019.

[13] C. Chen, Z. Zhang, S. Lee, and S. Shieh, "Penetration Testing in the IoT Age," in *Computer*, vol. 51, no. 4, pp. 82-85, April 2018.

[14] 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. of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security*, New York, NY, USA, pp. 3–10, 2017.

[15] O. Abu Waraga, M. Bettayeb, Q. Nasir, and M. Abu Talib, "Design and implementation of automated IoT security testbed," in *Computers & Security*, vol. 88, January 2020.

[16] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo, "A survey on facilities for experimental internet of things research," in *IEEE Communication Magazine*, vol. 49, no. 11, pp. 58–67, November 2011.

[17] G. Werner-Allen, P. Swieskowski, and M. Welsh, "MoteLab: A Wireless Sensor Network Testbed," in *Proc. of the 4th International Symposium on Information Processing in Sensor Networks*, Boise, ID, USA, April 2005.

[18] E. Ertin *et al.*, "Kansei: A Testbed for Sensing at Scale," in *Proc. of the 5th International Conference on Information Processing in Sensor Networks*, Nashville, TN, USA, pp. 399–406, 2006.

[19] J. Bers, A. Gosain, I. Rose, and M. Welsh, "Citysense: The design and performance of an urban wireless sensor network testbed," in *Proc. of the IEEE International Conference on Technologies for Homeland Security*, 2008.

[20] J. M. Hernández-Muñoz, J. Vercher, L. Muñoz, J. Galache, M. Presser, L. Gómez, and J. Pettersson, "Smart Cities at the Forefront of the Future Internet," in *The Future Internet*, *Lecture Notes in Computer Science*, vol 6656. Springer, Berlin, Heidelberg, pp. 447–462, 2011.

[21] "FIT/IoT-LAB Very large scale open wireless sensor network testbed." [Online]. Available: https://www.iot-lab.info/.

[22] "Buy a Raspberry Pi 3 Model B – Raspberry Pi." [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[23] "Metasploit | Penetration Testing Software, Pen Testing Security," Metasploit. [Online]. Available: https://www.metasploit.com/.

[24] "Kali Linux Penetration Testing Tools." [Online]. Available: https://tools.kali.org/.

[25] "SANS Institute: Reading Room - Risk Management." [Online]. Available:https://www.sans.org/readingroom/whitepapers/riskmanagement/paper/37452.

[26] "OWASP - IoT Security Guidance." [Online]. Available: https://www.owasp.org/index.php/IoT_Security_Guidance.