An IoT/IoMT Security Testbed for Anomaly-based Intrusion Detection Systems

Georgios Zachos
Instituto de Telecomunicacoes
Aveiro, Portugal
Faculty of Engineering and Science
University of Greenwich
Chatham Maritime, UK
g.zachos@av.it.pt

Georgios Mantas
Instituto de Telecomunicacoes
Aveiro, Portugal
Faculty of Engineering and Science
University of Greenwich
Chatham Maritime, UK
gimantas@av.it.pt

Ismael Essop
Faculty of Engineering and Science
University of Greenwich
Chatham Maritime, UK
i.a.essop@greenwich.ac.uk

Kyriakos Porfyrakis
Faculty of Engineering and Science
University of Greenwich
Chatham Maritime, UK
k.porfyrakis@greenwich.ac.uk

Joaquim Manuel C.S. Bastos
Instituto de Telecomunicacoes
Aveiro, Portugal
jbastos@av.it.pt

Jonathan Rodriguez
Instituto de Telecomunicacoes
Aveiro, Portugal
Faculty of Computing, Engineering and Science, University of South Wales
Pontypridd, UK
jonathan@av.it.pt

***Abstract*— Over the past few years, the Internet of Things (IoT) is transforming the healthcare sector through the introduction of the Internet of Medical Things (IoMT) technology whose purpose is the improvement of the patient's quality of life. Nevertheless, IoMT networks are still vulnerable to a wide range of threats because of their heterogeneity and resource-constrained characteristics. Thus, novel security mechanisms such as accurate and efficient intrusion detection systems (IDSs), taking into consideration the inherent limitations of the IoMT networks, are required to be developed before IoMT networks reach their full potential in the market. In our previous works, we presented the system architecture of a novel hybrid anomaly-based IDS (AIDS) for IoMT networks and the implementation of its prototype. The next step is the testing and evaluation of the performance of the proposed AIDS under different types of attacks. However, there is a lack of existing IoT testbeds that can be used to test and evaluate the performance of an AIDS as a whole system running on different IoT devices, networks and platforms, and being under different types of IoT attacks. Therefore, in this paper, we present the development of a functional IoT/IoMT security testbed for testing and evaluating AIDSs. In addition, we intend this work to serve as a guidance for other researchers or engineers who aim to develop specific IoT/IoMT testbeds for evaluating their own AIDSs under different types of IoT attacks.**

Keywords—IoT/IoMT, IoT/IoMT Testbed, Intrusion Detection System (IDS), Eclipse Hono, Eclipse Ditto, Influxdb, Grafana, Suricata

I. INTRODUCTION

The Internet of Things (IoT) paradigm is transforming the healthcare sector with the introduction of the Internet of Medical Things (IoMT) technology which aims to improve the patient's quality of life by enabling personalized e-health services without limitations on time and location [1]–[5]. However, the wide range of different communication technologies (e.g., WLANs, Bluetooth, Zigbee) and types of IoMT devices (e.g., medical sensors, actuators) incorporated in IoMT edge networks are vulnerable to various types of security threats, and this, in turn, raises many security and privacy challenges for such networks, as well as for the healthcare systems relying on these networks [6]–[9]. For instance, an adversary may compromise IoT-based healthcare systems through their IoMT networks in order to manipulate sensing data (e.g., by injecting fake data) and cause malfunctions to the compromised IoT-based healthcare systems that, in turn, will jeopardize the integrity or the availability of the healthcare services provided by these systems [2]. Consequently, security solutions protecting IoMT networks from attackers are essential for the acceptance and wide adoption of such networks in the coming next years.

Nevertheless, the high resource requirements of complex and heavyweight conventional security mechanisms cannot be afforded by (a) the resource-constrained IoMT edge devices with limited processing power, storage capacity, and battery life, and/or (b) the constrained environment in which the IoMT devices are deployed and interconnected using lightweight communication protocols [10]. Therefore, novel security mechanisms are necessary to be developed in order to address the pressing security challenges of IoMT networks in an effective and efficient manner, considering their inherent limitations stemming from their resource-constrained characteristics, before IoMT networks gain the trust of all involved stakeholders and reach their full potential in the market [8], [11].

Toward this direction, the industry and research community currently foresee anomaly-based intrusion detection as a promising security solution that can play a significant role in protecting IoT/IoMT networks, as long as novel lightweight anomaly-based intrusion detection systems (AIDSs) are developed [10], [12]–[15]. In our previous work in [16], we presented the system architecture for a novel hybrid AIDS for IoMT networks, and in [17], we described the details of the implementation process that led to a prototype of the proposed AIDS in [16]. The next step is the testing and evaluation of the performance of the proposed AIDS under different types of attacks. However, the existing IoT testbeds in the literature are focused on either (i) testing the functionality of the deployed IoT devices and/or involved protocols, algorithms and deployed services [18] or (ii) performing security analysis and vulnerability assessments of

This work is supported by the European Regional Development Fund (FEDER), through the Regional Operational Programme of Lisbon (POR LISBOA 2020) and the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework [Project Augmanity with Nr. 046103 (POCI-01-0247-FEDER-046103)].

IoT devices [19], [20]. Consequently, there is a lack of IoT security testbeds that can be used to test and evaluate the performance of an AIDS as a whole system running on different IoT devices, networks and platforms, and being under different types of IoT attacks.

Therefore, in this paper, the main objective is the design and development of a functional IoT/IoMT security testbed for evaluating AIDSs such as the one we proposed in [16] and [17]. Additionally, we intend this work to serve as a guidance for other researchers or engineers who aim to develop specific IoT/IoMT security testbeds for evaluating their own AIDSs under different types of IoT attacks.

Following the introduction, this paper is organized as follows. Section II presents the details of our developed IoT/IoMT security testbed as well as of its different components. Section III describes the two types of tests that we performed to verify that the different components of our testbed are operating normally. Finally, Section IV concludes this paper and provides hints for future work.

II. Developed IoT/IoMT Security Testbed

The architecture of the developed IoT/IoMT security testbed consists of three different major components (i.e., IoT/IoMT Server, IoT/IoMT Gateway, and IoT/IoMT devices) as shown in Fig. 1.

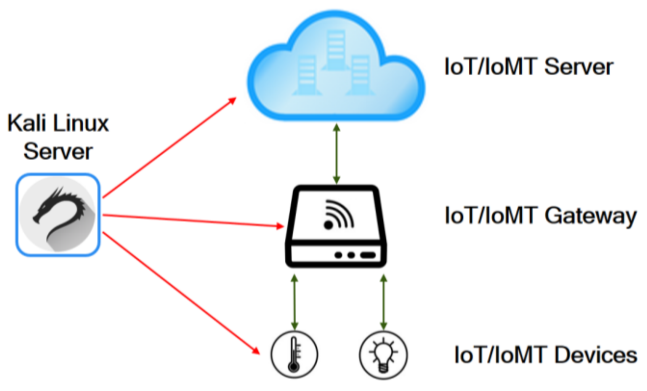


Fig. 1. Overview of the developed IoT/IoMT security testbed.

A. *IoT/IoMT Devices*

In general, the IoT/IoMT devices possess limited resources in regard to computation and storage. On the one hand, an IoT/IoMT device (e.g., an IoMT sensor device) may constitute a source of sensing data that can be related to patient's environment or wellbeing. On the other hand, an IoT/IoMT device (e.g., an IoMT actuator device such as an insulin pump) may operate as a receiver of data. In this case, the received data may take the form of commands that the IoT/IoMT device is required to execute in order to realize an action (e.g., inject insulin into the body of a patient).

In our testbed, we use a Raspberry Pi 4 Model B device as an IoT/IoMT device. In addition, we use Raspbian [21] as the operating system (OS) in the Raspberry Pi 4 IoT/IoMT device that simulates the functionality of a sensor. This is achieved by creating and running a python script that periodically sends randomly generated values (i.e., temperature and humidity) to the IoT/IoMT Gateway.

In particular, the python script starts a daemon process that generates a separate thread (i.e., "send" thread) that periodically sends randomly generated sensing data to the IoT/IoMT Gateway. After creating the "send thread", the main thread of the daemon process keeps monitoring a temporary "ctrlfile" file. When the stored value of the "ctrlfile" file changes from "1" to "0", then the daemon process along with its threads terminate their operation. The pseudocode of our python script that randomly generates and sends sensing data in the form of a report is presented below.

Algorithm: IoT/IoMT Device
Input: devID, period, port, gateway IP, gateway port
Action: *Sending random sensing data to IoT/IoMT Gateway*

```
1. store input parameters and initialize extra parameters
2. daemonize the current process
3. create a tmp ctrlfile to control the daemon process
4. operating ← 1
5. create and start a separate thread (send thread) that handles
   sending random data to the Gateway
6.     payload ← empty dictionary
7.     while operating == 1:
8.         payload[“temperature”] ← random(20,30)
9.         payload[“humidity”] ← random(5,15)
10.        connect to Gateway using port, gateway IP and
           gateway port
11.        create report including devID and payload
12.        send created report to Gateway
13.        close connection to Gateway
14.        sleep for period secs
15. while operating == 1:
16.    sleep for 1 second
17.    open tmp ctrlfile
18.    read value from tmp ctrlfile
19.    store the read value to the operating variable
20.    close tmp ctrlfile
21. deallocate resources and exit
```

B. *IoT/IoMT Gateway*

The main role of the IoT/IoMT Gateway is to act as a relay node. On the one hand, the IoT/IoMT Gateway receives the sensing data of the connected IoT/IoMT devices and in turn, sends the sensing data to the IoT/IoMT Server. On the other hand, the IoT/IoMT Gateway receives IoT/IoMT device commands from the IoT/IoMT Server and in turn, sends these commands to the connected IoT/IoMT devices to be executed.

In our testbed, we use another Raspberry Pi 4 Model B device as the IoT/IoMT Gateway. We use Ubuntu 20.04 [22] as the operating system (OS) in the Raspberry Pi 4 IoT/IoMT Gateway device that simulates the functionality of a Gateway. This functionality has been achieved by creating and running a python script that receives the data sent periodically from the connected IoT/IoMT devices. Afterward, the script formats the received data and sends them properly to the IoT/IoMT Server.

In particular, the python script starts a daemon process that generates a separate thread (i.e., "acc" thread) that accepts connections from the IoT/IoMT devices. For each accepted connection, the "acc thread" creates separate thread (i.e., "rcv thread") that handles the receiving of data from the connected IoT/IoMT device and then the received data are formatted properly into JSON and sent to the IoT/IoMT Server. After creating the "acc" thread, the main thread of the daemon process keeps monitoring a temporary "ctrlfile" file. When the stored value of the "ctrlfile" file changes from "1" to "0", then the daemon process along with its threads terminate their operation. The pseudocode of our python script is presented below.

```
Algorithm: IoT/IoMT Gateway
Input: port, server IP, server port
Action: Receiving and sending IoT/IoMT device data to the IoT/IoMT Server
1. store input parameters and initialize extra parameters
2. daemonize the current process
3. create a tmp ctrlfile to control the daemon process
4. operating ← 1
5. create a separate thread (acc thread) that handles accepting connections from IoT/IoMT devices
6.     while operating == 1:
7.         (clientsock, ip addr) ← accept()
8.         create and start a separate thread (rcv thread) that receives the data from connected IoT/IoMT device, formats them and sends them to IoT/IoMT Server
9.             data ← rcvdata(clientsock)
10.            create a json payload based on received data
11.            send the json payload to the IoT/IoMT paltform
12. while operating == 1:
13.     sleep for 1 second
14.     open tmp ctrlfile
15.     read value from tmp ctrlfile
16.     store the read value to the operating variable
17.     close tmp ctrlfile
18. deallocate resources and exit
```

C. IoT/IoMT Server

The main roles of the IoT/IoMT Server are to (i) receive the data from the IoT/IoMT devices through the Gateway, (ii) visualize the received data, and (iii) send appropriate commands back to the IoT/IoMT devices through the Gateway. In the proposed testbed, the IoT/IoMT Server consists of multiple different components as shown in Fig. 2 and explained below.

1) Eclipse Hono

Initially, the Eclipse Hono [23] component is deployed in Kubernetes [24] in the IoT/IoMT Server to provide the IoT messaging layer. It enables several IoT/IoMT devices to connect to the IoT/IoMT server using various protocols (e.g., MQTT). Besides, Eclipse Hono allows communicating, through the Gateway, with the connected IoT devices in a uniform way regardless of the device communication protocol. The advantages of Eclipse Hono are the following [23]:

- **IoT Protocols support**: IoT devices can communicate with Eclipse Hono and vice-versa using different common IoT protocols such as HTTP, MQTT, AMQP and CoAP which are supported by Eclipse Hono out of the box. Moreover, through a simple mechanism available in Eclipse Hono, new custom protocol adapters can be added in order to connect and interact with devices that employ different communication protocols.
- **Scalability**: Eclipse Hono is designed so that several IoT devices can be connected to it. The scalability is achieved using a micro service-based architecture and a reactive programming model.
- **Uniform API**: Regardless of the device protocol, Eclipse Hono utilizes a simple and uniform API in

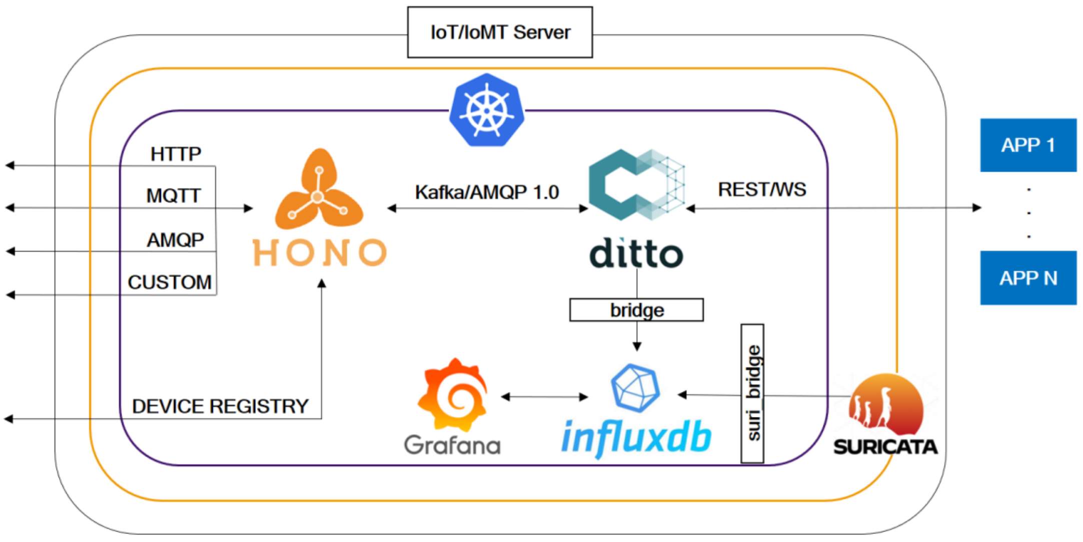


Fig. 2. Internal components of the deployed IoT/IoMT Server of the proposed IoT/IoMT security testbed.

order to (i) allow the connection of different types of devices to a (cloud) back end, and (ii) communicate with all the connected devices.

- **Secure by default**: Eclipse Hono is designed for security by default. Both common authentication mechanisms such as username/password and X.509 client certificates are supported in order that the identity of a device can be verified. In addition, transport layer security (TLS) is used by Eclipse Hono when it communicates with devices.
- **Messaging Patterns**: APIs for important IoT communication patterns are provided. On the one hand, sensor readings can be reported through Telemetry and Event messages. On the other hand, operations on devices can be invoked by applications through the Command & Control messages.

2) Eclipse Ditto

To complement the Eclipse Hono component, we added the Eclipse Ditto [25] component by deploying it in Kubernetes [24] in the IoT/IoMT Server. Eclipse Ditto implements a software pattern called “digital twin”. A “digital twin” is defined as a virtual representation of its real world counterpart (e.g., real world device).

More specifically, in our case, Eclipse Ditto is deployed in order to receive the data coming from Eclipse Hono and the IoT/IoMT devices via the AMQP 1.0 API of Eclipse Hono. Based on the received data from Eclipse Hono, Ditto updates the digital twins of the IoT/IoMT devices. In addition, Eclipse Ditto provides fully-fledged, authorization aware APIs (i.e., REST, WebSocket) so that consumer applications can easily interact with the physical IoT/IoMT devices and all aspects around them via their digital twins in Eclipse Ditto. The advantages of Eclipse Ditto are the following [25]:

- **Device as a Service**: Ditto allows to handle a “thing” as another web service via its digital twin. In essence, an IoT device is abstracted into a digital twin and the interaction with the physical IoT device is performed through the digital twin API.
- **Access control enforcement:** It is possible to authorize each API call on a digital twin. Thus, an appropriate resource-based access check can be applied and this, in turn, can ensure that specific users are able to only see/modify specific parts of a digital twin.

3) InfluxDB

After setting up Hono and Ditto, we added the InfluxDB component to the IoT/IoMT server in order to store the information of the digital twins in Ditto. InfluxDB [26] is a component implementing a time-series database and an InfluxDB instance is deployed in Kubernetes [24] in the IoT/IoMT Server. By utilizing Eclispe Ditto’s API, a custom container called “bridge” is deployed to collect the data related to the digital twins from Eclipse Ditto and store them properly in the InfluxDB instance.

4) Grafana

The next step was to add Grafana [27] to the IoT/IoMT Server. Grafana is a multi-platform open source analytics and interactive visualization web application and is capable of enhancing the IoT/IoMT Server with visualization and dashboarding capabilities. The deployed Grafana instance was configured so that data of the InfluxDB instance can be visualized.

D. Kali Linux Server

Kali Linux is a penetration testing tool that is available from the website in [28] and is considered as a standard tool used in the industry for penetration testing purposes. In particular, Kali Linux is an open-source, Debian-based Linux distribution aimed at advanced Penetration Testing and Security Auditing, and it contains several hundred tools targeted towards various information security tasks, such as Penetration Testing, Security Research, Computer Forensics and Reverse Engineering, Vulnerability Management and Red Team Testing [29]. Besides, Kali Linux is a solution capable of targeting multiple different platforms, and is accessible and freely available to information security professionals and hobbyists [29].

Furthermore, Kali Linux provides support for several wireless devices, and is capable of running properly on different hardware. In addition, to achieve its purposes, Kali Linux utilizes a custom kernel that is always kept updated for injection vulnerabilities. In our testbed, Kali Linux is set up on the Kali Linux Server, as shown in Fig. 1, whose purpose is to launch specific attacks (i.e., scanning attack and Denial of Service attack) on the different components of the IoT/IoMT security testbed (i.e., IoT/IoMT Server, Gateway, and devices). Based on the launched attacks, it will be possible to evaluate the performance of AIDSs deployed in the future.

III. Demonstrations of the IoT/IoMT Security Testbed

After setting up of the proposed testbed, we performed two tests in order to verify: a) the proper operation of the different components of the testbed related to the sensor data flow, as well as b) the proper operation of the Kali Linux server by implementing two types of attacks (i.e., scanning attack and Denial of Service attack) against the IoT/IoMT server. The implemented attacks will also allow the evaluation of future proposed AIDSs when they are deployed on the testbed.

A. Demonstration of the Sensor Data Flow

The first test relates to the verification of the normal operation of the testbed components (i.e., IoT/IoMT Server, IoT/IoMT Gateway, and IoT/IoMT devices). A demo IoT/IoMT Sensor was registered in Eclipse Hono and a corresponding digital twin was setup in Eclipse Ditto. Furthermore, we executed our python scripts in the IoT/IoMT device (i.e., Raspberry Pi 4 device) and the IoT/IoMT Gateway (i.e., another Raspberry Pi 4 device). Thus, telemetry data regarding two measurements (i.e., temperature, humidity) were randomly generated in the IoT/IoMT device every 1 second for 10 minutes, and through the IoT/IoMT Gateway, these data were sent to Eclipse Hono’s HTTP adapter. The generated temperature values were set to be in the [20, 30] range, while the generated humidity values were set to be in the [5, 15] range. Afterwards, the sensor data were collected from the digital twin in Eclipse Ditto, stored in the InfluxDB instance and then visualized in Grafana as shown in Fig. 3.

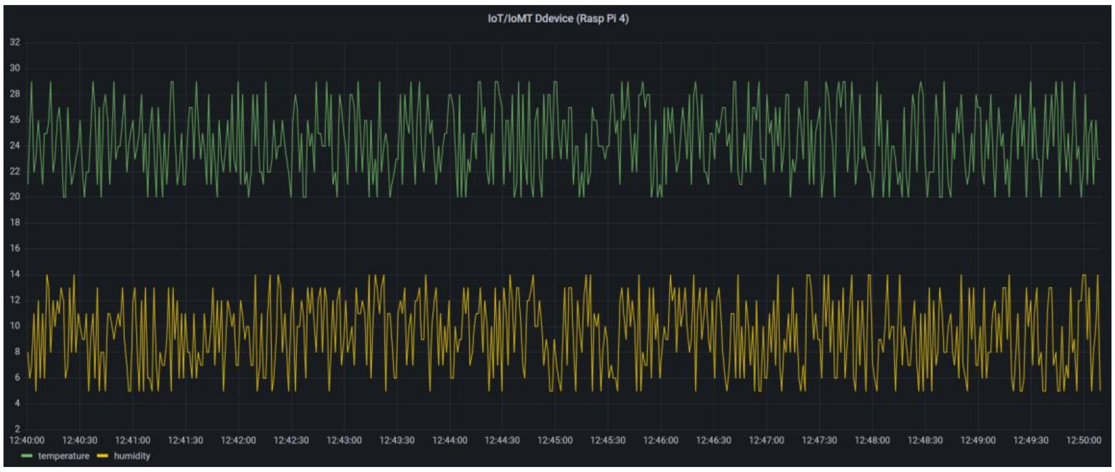


Fig. 3. Randomly generated temperature and humidity values from IoT/IoMT device visualized in Grafana.

B. *Demonstration of Attacks on the IoT/IoMT Server*

The second test relates to the verification of the proper operation of the Kali Linux server [28] to launch attacks against the IoT/IoMT server. This was achieved by i) implementing a Scanning attack and a Denial of Service (DoS) attack against the IoT/IoMT server, as shown in Fig. 4, with the use of the Kali Linux server, and ii) detecting both attacks with the use of the Suricata IDS [30] that was set up on the IoT/IoMT server.

The Suricata IDS is an open source and publicly available signature-based IDS that was installed and configured to monitor the API endpoints of Hono and Ditto. In addition, we used a custom container called “suri_bridge” that was deployed to collect the alert logs of Suricata IDS and store them properly in the InfluxDB instance. By storing the alert logs in the InfluxDB instance, we were able to use Grafana in order to visualize the alert logs.

Fig. 5 and Fig. 6 depict the command used in Kali Linux to perform the Scanning attack and the corresponding alert logs of Suricata in Grafana, respectively. Moreover, Fig. 7 and Fig. 8 depict the command used in Kali Linux to perform the DoS attack and the corresponding alert logs of Suricata in Grafana, respectively.

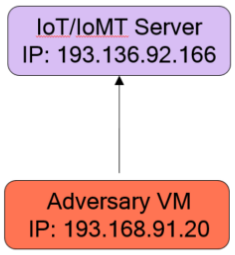
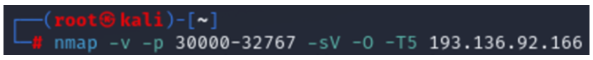
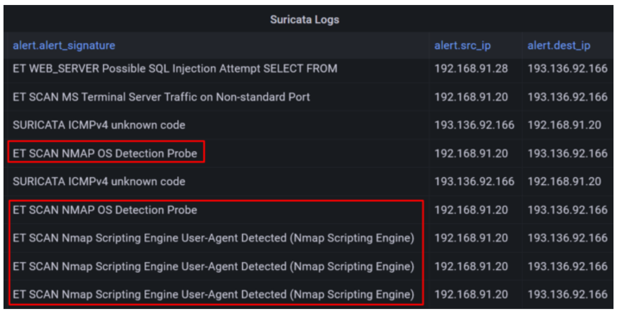


Fig. 4. Usage of Kali Linux on the Adversary VM to launch attacks against the IoT/IoMT Server.



```
┌──(root㉿kali)-[~]
└─# nmap -v -p 30000-32767 -sV -O -T5 193.136.92.166
```

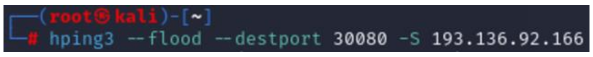
Fig. 5. Kali command to perform the scanning attack on the IoT/IoMT Server.



Suricata Logs

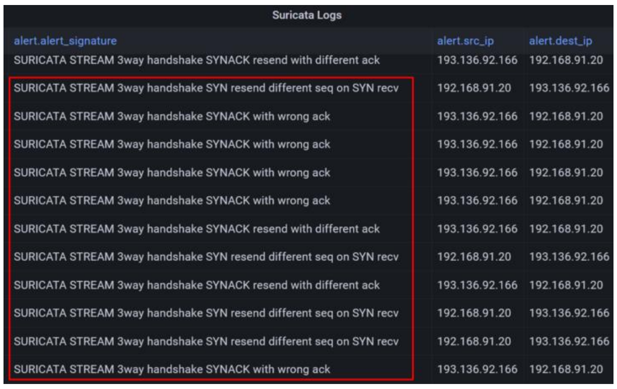
alert.alert_signature	alert.src_ip	alert.dest_ip
ET WEB_SERVER Possible SQL Injection Attempt SELECT FROM	192.168.91.28	193.136.92.166
ET SCAN MS Terminal Server Traffic on Non-standard Port	192.168.91.20	193.136.92.166
SURICATA ICMPv4 unknown code	193.136.92.166	192.168.91.20
ET SCAN NMAP OS Detection Probe	192.168.91.20	193.136.92.166
SURICATA ICMPv4 unknown code	193.136.92.166	192.168.91.20
ET SCAN NMAP OS Detection Probe	192.168.91.20	193.136.92.166
ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	192.168.91.20	193.136.92.166
ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	192.168.91.20	193.136.92.166
ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	192.168.91.20	193.136.92.166

Fig. 6. Suricata alert logs visualized in Grafana during the Scanning attack on the IoT/IoMT Server.



```
┌──(root㉿kali)-[~]
└─# hping3 --flood --destport 30080 -S 193.136.92.166
```

Fig. 7. Kali command to perform the DoS attack on the IoT/IoMT Server.



Suricata Logs

alert.alert_signature	alert.src_ip	alert.dest_ip
SURICATA STREAM 3way handshake SYNACK resend with different ack	193.136.92.166	192.168.91.20
SURICATA STREAM 3way handshake SYN resend different seq on SYN recv	192.168.91.20	193.136.92.166
SURICATA STREAM 3way handshake SYNACK with wrong ack	193.136.92.166	192.168.91.20
SURICATA STREAM 3way handshake SYNACK with wrong ack	193.136.92.166	192.168.91.20
SURICATA STREAM 3way handshake SYNACK with wrong ack	193.136.92.166	192.168.91.20
SURICATA STREAM 3way handshake SYNACK with wrong ack	193.136.92.166	192.168.91.20
SURICATA STREAM 3way handshake SYNACK resend with different ack	193.136.92.166	192.168.91.20
SURICATA STREAM 3way handshake SYN resend different seq on SYN recv	192.168.91.20	193.136.92.166
SURICATA STREAM 3way handshake SYNACK resend with different ack	193.136.92.166	192.168.91.20
SURICATA STREAM 3way handshake SYN resend different seq on SYN recv	192.168.91.20	193.136.92.166
SURICATA STREAM 3way handshake SYN resend different seq on SYN recv	192.168.91.20	193.136.92.166
SURICATA STREAM 3way handshake SYNACK with wrong ack	193.136.92.166	192.168.91.20

Fig. 8. Suricata alert logs visualized in Grafana during the DoS attack on the IoT/IoMT Server.

IV. CONCLUSION

In this paper, we presented the development process of a functional IoT/IoMT security testbed for evaluating AIDSs such as the one we proposed in [16] and [17]. In particular, we described the different components of the developed testbed as well as the role and functionality of each component. Besides, we intend this work to serve as a guidance for other researchers or engineers who aim to develop specific IoT/IoMT security testbeds for evaluating their own AIDSs under different types of IoT attacks. As future work, we plan to use the developed IoT/IoMT security testbed in order to test and evaluate our proposed AIDS in [16]. In addition, we aim to integrate more resources-constrained IoT devices, such as Tmote Sky devices [31], on the testbed in order to evaluate the energy efficiency of our proposed AIDS [16].

REFERENCES

[1] J. J. P. C. Rodrigues *et al.*, "Enabling Technologies for the Internet of Health Things," *IEEE Access*, vol. 6, pp. 13129–13141, 2018, doi: 10.1109/ACCESS.2017.2789329.

[2] M. Papaioannou *et al.*, "A Survey on Security Threats and Countermeasures in Internet of Medical Things (IoMT)," *Trans. Emerg. Telecommun. Technol.*, p. e4049, 2020, doi: 10.1002/ett.4049.

[3] S. M. R. Islam, D. Kwak, M. H. Kabir, M. Hossain, and K. S. Kwak, "The internet of things for health care: A comprehensive survey," *IEEE Access*, vol. 3, pp. 678–708, 2015, doi: 10.1109/ACCESS.2015.2437951.

[4] E. Karavatselou, M.-A. Fengou, G. Mantas, and D. Lymberopoulos, "Profile Management System in Ubiquitous Healthcare Cloud Computing Environment," in *Broadband Communications, Networks, and Systems*, 2019, pp. 105–114.

[5] M.-A. Fengou, G. Mantas, D. Lymberopoulos, N. Komninos, S. Fengos, and N. Lazarou, "A New Framework Architecture for Next Generation e-Health Services," *IEEE J. Biomed. Heal. Informatics*, vol. 17, no. 1, pp. 9–18, 2013, doi: 10.1109/TITB.2012.2224876.

[6] F. Pelekoudas-Oikonomou *et al.*, "Blockchain-Based Security Mechanisms for IoMT Edge Networks in IoMT-Based Healthcare Monitoring Systems," *Sensors 2022, Vol. 22, Page 2449*, vol. 22, no. 7, p. 2449, Mar. 2022, doi: 10.3390/S22072449.

[7] I. Makhdoom, M. Abolhasan, J. Lipman, R. P. Liu, and W. Ni, "Anatomy of Threats to the Internet of Things," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 2, pp. 1636–1675, 2019, doi: 10.1109/COMST.2018.2874978.

[8] M. Zhang, A. Raghunathan, and N. K. Jha, "Trustworthiness of medical devices and body area networks," *Proc. IEEE*, vol. 102, no. 8, pp. 1174–1188, 2014, doi: 10.1109/JPROC.2014.2322103.

[9] M. Karageorgou, G. Mantas, I. Essop, J. Rodriguez, and D. Lymberopoulos, "Cybersecurity attacks on medical IoT devices for smart city healthcare services," in *IoT Technologies in Smart Cities: From sensors to big data, security and trust*, Institution of Engineering and Technology, 2020, pp. 171–187.

[10] I. Essop, J. C. Ribeiro, M. Papaioannou, G. Zachos, G. Mantas, and J. Rodriguez, "Generating datasets for anomaly-based intrusion detection systems in iot and industrial iot networks," *Sensors*, vol. 21, no. 4, pp. 1–31, 2021, doi: 10.3390/s21041528.

[11] F. Alsubaei, A. Abuhussein, and S. Shiva, "Security and Privacy in the Internet of Medical Things: Taxonomy and Risk Assessment," in *Proceedings - 2017 IEEE 42nd Conference on Local Computer Networks Workshops, LCN Workshops 2017*, 2017, pp. 112–120, doi: 10.1109/LCN.Workshops.2017.72.

[12] J. Asharf, N. Moustafa, H. Khurshid, E. Debie, W. Haider, and A. Wahab, "A review of intrusion detection systems using machine and deep learning in internet of things: Challenges, solutions and future directions," *Electronics (Switzerland)*, vol. 9, no. 7. MDPI AG, p. 1177, 2020, doi: 10.3390/electronics9071177.

[13] J. Ribeiro, F. B. Saghezchi, G. Mantas, J. Rodriguez, and R. A. Abd-Alhameed, "HIDROID: Prototyping a Behavioral Host-Based Intrusion Detection and Prevention System for Android," *IEEE Access*, vol. 8, pp. 23154–23168, 2020, doi: 10.1109/ACCESS.2020.2969626.

[14] J. Ribeiro, F. B. Saghezchi, G. Mantas, J. Rodriguez, S. J. Shepherd, and R. A. Abd-Alhameed, "Towards an Autonomous Host-based Intrusion Detection System for Android Mobile Devices," in *9th EAI International Conference on Broadband Communications, Networks, and Systems (BROADNETS2018)*, 2018, pp. 139–148.

[15] J. Ribeiro, F. B. Saghezchi, G. Mantas, J. Rodriguez, S. J. Shepherd, and R. A. Abd-Alhameed, "An Autonomous Host-Based Intrusion Detection System for Android Mobile Devices," *Mob. Networks Appl.*, vol. 25, no. 1, pp. 164–172, 2020, doi: 10.1007/s11036-019-01220-y.

[16] G. Zachos, I. Essop, G. Mantas, K. Porfyrakis, J. C. Ribeiro, and J. Rodriguez, "An Anomaly-Based Intrusion Detection System for Internet of Medical Things Networks," *Electron. 2021, Vol. 10, Page 2562*, vol. 10, no. 21, p. 2562, Oct. 2021, doi: 10.3390/ELECTRONICS10212562.

[17] G. Zachos, G. Mantas, I. Essop, K. Porfyrakis, J. C. Ribeiro, and J. Rodriguez, "Prototyping an Anomaly-Based Intrusion Detection System for Internet of Medical Things Networks," in *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD*, 2022, vol. 2022-Novem, pp. 179–183, doi: 10.1109/CAMAD55695.2022.9966912.

[18] M. Chernyshev, Z. Baig, O. Bello, and S. Zeadally, "Internet of things (IoT): Research, simulators, and testbeds," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1637–1647, Jun. 2018, doi: 10.1109/JIOT.2017.2786639.

[19] S. Siboni *et al.*, "Security Testbed for Internet-of-Things Devices," *IEEE Trans. Reliab.*, vol. 68, no. 1, pp. 23–44, Mar. 2019, doi: 10.1109/TR.2018.2864536.

[20] O. Abu Waraga, M. Bettayeb, Q. Nasir, and M. Abu Talib, "Design and implementation of automated IoT security testbed," *Comput. Secur.*, vol. 88, p. 101648, Jan. 2020, doi: 10.1016/J.COSE.2019.101648.

[21] "FrontPage - Raspbian." https://www.raspbian.org/ (accessed Mar. 20, 2023).

[22] "Ubuntu 20.04.5 LTS (Focal Fossa)." https://releases.ubuntu.com/focal/ (accessed Mar. 20, 2023).

[23] "Connect, Command & Control IoT devices :: Eclipse Hono^{TM}." https://www.eclipse.org/hono/ (accessed Dec. 07, 2022).

[24] "Kubernetes Documentation | Kubernetes." https://kubernetes.io/docs/home/ (accessed Dec. 07, 2022).

[25] "Eclipse Ditto^{TM} • open source framework for digital twins in the IoT." https://www.eclipse.org/ditto/ (accessed Dec. 07, 2022).

[26] "Learn about InfluxDB OSS | InfluxDB OSS 1.8 Documentation." https://docs.influxdata.com/influxdb/v1.8/introduction/ (accessed Feb. 27, 2023).

[27] "Grafana documentation | Grafana documentation." https://grafana.com/docs/grafana/latest/ (accessed Feb. 27, 2023).

[28] "Kali Linux | Penetration Testing and Ethical Hacking Linux Distribution." https://www.kali.org/ (accessed Aug. 13, 2021).

[29] "What is Kali Linux? | Kali Linux Documentation." https://www.kali.org/docs/introduction/what-is-kali-linux/ (accessed Mar. 20, 2023).

[30] "Home - Suricata." https://suricata.io/ (accessed May 12, 2022).

[31] "Moteiv Corporation Tmote Sky—Ultra Low Power IEEE 802.15.4 Compliant Wireless Sensor Module." http://www.crew-project.eu/sites/default/files/tmote-sky-datasheet.pdf (accessed Sep. 06, 2021).