Developing a User-centric Security Focused IoT Cyber Range

Jonathan Starkey

2020

MSc Internet of Things
Bournemouth University
Faculty of Science & Technology
Department of Computing and Informatics

Bournemouth University, Department of Computing and Informatics, Masters Dissertation

Abstract

The paradigm of Internet of Things has developed beyond premature architectural design solutions to that of successful application of IoT technologies in systems of high technological sophistication. However, while fundamental elements of IoT systems can and have been extensively studied utilising IoT testbeds, a lacking of cyber security training in IoT has resulted in market disruptions on numerous occasions (Nicolas Falliere and Chien 2010) & (Kolias et al. 2017). To address the apparent skill gap in IoT cyber security (Topham et al. 2016), this dissertation presents the workings in design and development of an IoT Cyber Range (IoT-CR); an IoT testbed designed for research and training in IoT security. The IoT-CR project allows the user to specify and interact with customised virtual and physical IoT networks through the use of web, simulator and hardware technologies. Chapter 2 provides an overview of existing IoT testbeds and IoT-CR systems with analysis coherent within the context of the proposed project. Then in chapter 3, in presenting the architecture and workings of the project, justifications and limitations are addressed. In chapters 4 and 5 the development of the project is recorded. The working project can be utilised in the scope of the dissertation by designing and running various scenario simulations to involve and educate the user before allowing them configuration of their own networks. In chapter 6 the dissertation then demonstrates the use of a scenario via a red/blue team scenario involving a variant of man-in-the-middle attack using IoT devices before discussing preliminary results extracted from the testbed. Finally the dissertation is concluded in chapter 7. This work has been published to the MDPI 'Sensors' peer-reviewed international journal (Starkey et al. 2020) (see appendix), an esteemed journal with an impact factor of 3 (MDPI 2020).

Dissertation Declaration

This Dissertation/Project Report is submitted in partial fulfilment of the requirements for a Masters degree at Bournemouth University. I declare that this Dissertation/ Project Report is my own work and that it does not contravene any academic offence as specified in the University's regulations.

Retention

I agree that, should the University wish to retain it for reference purposes, a copy of my Dissertation/Project Report may be held by Bournemouth University normally for a period of 3 academic years. I understand that my Dissertation/Project Report may be destroyed once the retention period has expired. I am also aware that the University does not guarantee to retain this Dissertation/Project Report for any length of time (if at all) and that I have been advised to retain a copy for my future reference.

Confidentiality

I confirm that this Dissertation/Project Report does not contain information of a commercial or confidential nature or include personal information other than that which would normally be in the public domain unless the relevant permissions have been obtained. In particular any information which identifies a particular individual's religious or political beliefs, information relating to their health, ethnicity, criminal history or personal life has been anonymised unless permission for its publication has been granted from the person to whom it relates.

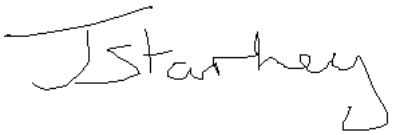
Copyright

The copyright for this dissertation remains with me.

Requests for Information

I agree that this dissertation may be made available as the result of a request for information under the Freedom of Information Act.

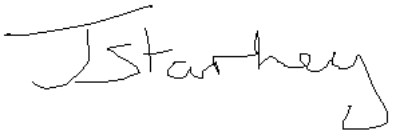
Name: Jonathan Starkey
Date: 22-09-2020
Programme: MSc IoT
Signed:



Original Work Declaration

This dissertation and the project that it is based on are my own work, except where stated, in accordance with University regulations.

Signed: 22-09-2020



Contents

List of Figures

Chapter 1

Introduction

As a major paradigm of modern networks 'Internet of Things' (IoT) connects a multitude of bespoke devices together, adopting a number of protocols for communication. Whilst this paradigm is being highly adopted, the technology must not continue without maintaining the correct safety and security standards. Facilities must therefore exist, for researchers to experiment with IoT environments to improve the security of next generation IoT products. One such facility is the 'IoT Cyber Range' (IoT-CR), an IoT testbed which focuses on cyber security.

1.1 Problem Definition

Cyber ranges provide ample opportunity for research, but the expanding IoT market introduces the cost of acquiring an ever growing range of hardware and software. Configuration, maintenance and capability can prevent researchers for operating in particular domains or with particular protocols. As such, an identified skills gap has occurred in the domain of IoT security (Topham et al. 2016).
To over come these issues IoT-CR can be set up to provide a facility for users to experiment and learn about IoT hardware and software. *The overarching objective of this dissertation is to provide such an IoT-CR, one that can educate novice IoT users, without limiting the capabilities of advanced users.* In order to achieve this, pre-defined scenarios are supplied as templates for users to build an understanding of the project system. At the same time users can upload custom code to configure the system uniquely.
Virtual IoT-CR can be deployed whereby simulators, emulators and/or hypervisors adopt the properties of IoT devices and networks. A further objective of this dissertation is to provide such a virtual IoT-CR allowing users access to a number of virtual IoT devices in order to run simulations and receive results, as if they had run the simulations locally.
In some scenarios, virtual systems are not sufficient to resolve the research requirements and a physical IoT-CR is required. Another objective of this dissertation is to provide a

physical IoT-CR through the configuration of physical IoT hardware capable of running custom code and suitable for research and development.
By providing a range of facilities accompanied by predefined scenarios, the project has a higher chance of appealing to a larger spectrum of user abilities, and a higher chance of application within a federation.

1.2 Problem Objectives

The following describes the project objectives and provides aims to monitor progress. These aims act as milestones and allow for retrospective assessment towards the end of the project, by comparison against the success criteria.

1.2.1 Objectives, Aims and Success Criteria

Objective 1 - Research Cyber Ranges
Aim 1: Gain understanding of current facilities available.
Success Criteria: Demonstrate research of cyber ranges facilities.
Aim 2: Gain understanding of current usage/operation.
Success Criteria: Demonstrate the possible implementation of cyber range.

Objective 2 - Set up the proposed virtual IoT-CR.
Aim 1: Research available virtual IoT technologies and critically analyse.
Success Criteria: Demonstrate research of virtual IoT technologies.
Aim 2: Choose virtual technologies and design IoT-CR.
Success Criteria: Demonstrate designs and justification for the virtual range.
Aim 3: Develop (program) with the chosen virtual technology.
Success Criteria: Demonstrate a working and validated virtual IoT network.
Aim 4: Integrate with our purpose built API (BU Undergrad Project).
Success Criteria: Demonstrate a fully operational virtual IoT-CR.

Objective 3 - Set up the proposed physical IoT-CR.
Aim 1: Research available physical IoT systems and critically analyse.
Success Criteria: Demonstrate research of physical IoT technologies.
Aim 2: Choose physical technologies and design IoT-CR.
Success Criteria: Demonstrate designs and justification for the physical range.
Aim 3: Develop with the chosen physical technology (consuming and repurposing another BU Undergrad Project (Lucas 2019)).
Success Criteria: Demonstrate a working and validated physical IoT network.
Aim 4: Build a job resource management system for physical technology.

Success Criteria: Demonstrate a working job resource management system.

Aim 5: Integrate with our purpose built API.

Success Criteria: Demonstrate a fully operational physical IoT-CR.

Objective 4 - Design scenarios for cyber range.

Aim 1: Research scenarios for IoT-CR.

Success Criteria: Demonstrate research of scenarios for use in IoT-CR systems.

Aim 2: Design scenarios applicable for new users, in the scope of IoT-CR security.

Success Criteria: Demonstrate the designed scenarios for use in IoT-CR systems.

Aim 3: Implement scenario in project for example usage.

Success Criteria: Demonstrate the developed scenarios utilising the IoT-CR built.

1.3 Research Rationale

The growth of IoT is occurring rapidly. However, this has resulted in the evolution of an significant and ever-growing cyber-security skill-gap within the IoT sector (Rebecca Vogel 2016). If unaddressed, discrepancy between required and ascertainable security expertise may become too broad, with many facets of IoT security facing possible neglect, enhancing the possibility of IoT related cyber crime and even cyber warfare. With this considered, amending the skill gap takes two forms:- A reactive approach closing the gap through education on the current IoT systems, and the proactive measures of maintaining the closed gap to achieve security on the newest and most bespoke of IoT devices. In order to reduce this skill gap, both the proactive and reactive approaches must apply concurrently - Novice researchers might learn about security vulnerable commercial IoT systems and how to patch or exploit them, whereas advanced researchers may design new algorithms to overcome current attack vectors before devices are manufactured. By provisioning the IoT-CR with both physical and virtual devices, researchers from limited to broad skill sets are able to work within a comprehensive set of devices for a multitude of tasks.

In order to build a functionally desirable IoT-CR, the system must perform in a manner similar to that of traditional IoT testbeds, whilst offering the exposure to new IoT technologies. There are many IoT testbeds available that can be critically analysed and deconstructed to form components for the proposed Cyber Range. In order to increase the exposure of facilities of an IoT-CR, systems can be federated, so that researchers can operate across numerous cyber ranges. A further objective of this dissertation is to position Bournemouth University as an IoT-CR provider potentially within the 'ECHO project', a federated cyber range with funding from European Union's Horizon 2020 research and

innovation programme (ECHO 2020). Furthermore, by utilising the opportunity to publish an article regarding Bournemouth University's novel Cyber Range, this project will be advertised to such federations. As well, the journal 'Sensors' by MDPI, is highly esteemed, with impact factor above 3, further increasing the notoriety of Bournemouth University and the Cyber Range (MDPI 2020).
By providing scenario templates alongside the system, users are able to operate the Cyber Range system immediately, allowing faster uptake of the IoT-CR. The scenarios can also provide a medium to push new IoT technologies to novice users. By doing so, subsections within the current IoT skills gap can be targeted individually, reducing the overall disparagement. For example, as an IoT-CR, security is the focus. Security based scenarios, such as capture the flag or man-in-the-middle, would therefore make logical accompaniment to the system. However if a federation were to focus on network efficiency, the system could be updated to supply the federation with networking scenarios instead.

1.4 Project Plan

Gnatts charts can be seen in appendix. The first chart details the time estimations for deconstructed objectives in chronological order, created at the beginning of the dissertation. The second chart shows the timing of actualised events that lead to the development of this dissertation, updated constantly throughout for monitoring progression against remaining time.

1.5 Risk Analysis

Table 2 in the appendix highlights potential risks during research and suggested solutions. These risks were abstracted from likely events or previous encounters and was written at the beginning of the dissertation.

Chapter 2

Literature Review

2.1 Purpose of study

The main purpose of this study is to review the literature for concepts constituting an IoT testbed or IoT Cyber Range system. Various aspects of a IoT testbeds will be considered:- Definition, requirements, user interaction and scenarios.

2.2 Definition

A Cyber Range provides an isolated, and safe environment to legally practise security training without the associated costs and risks. Researchers (Kavallieratos et al. 2019) define a cyber range to be an interactive, simulated representations of an organisation's local technical infrastructure connected to a simulated Internet. However the definition does not acknowledge the existence of federated setups, where resources may not be local or belonging to a particular organisation. Furthermore, the internet on which the simulations take place need not its self be simulated (Ficco and Palmieri 2019). Cyber Ranges can consist of physical infrastructure, be completely virtualised, or a hybrid. Others (Yamin et al. 2020) define a cyber range as an environment providing testbeds for research and conducting training within the realm of security and also defining six taxonomies.

2.3 Requirements

Six taxonomies for cyber Ranges and security testbeds:- scenarios, monitoring capabilities, learning, management, teaming and the environment.

Scenarios define the tested execution environment, this would include the hardware and software running on the system and parameters for provision of assets on the testbed.

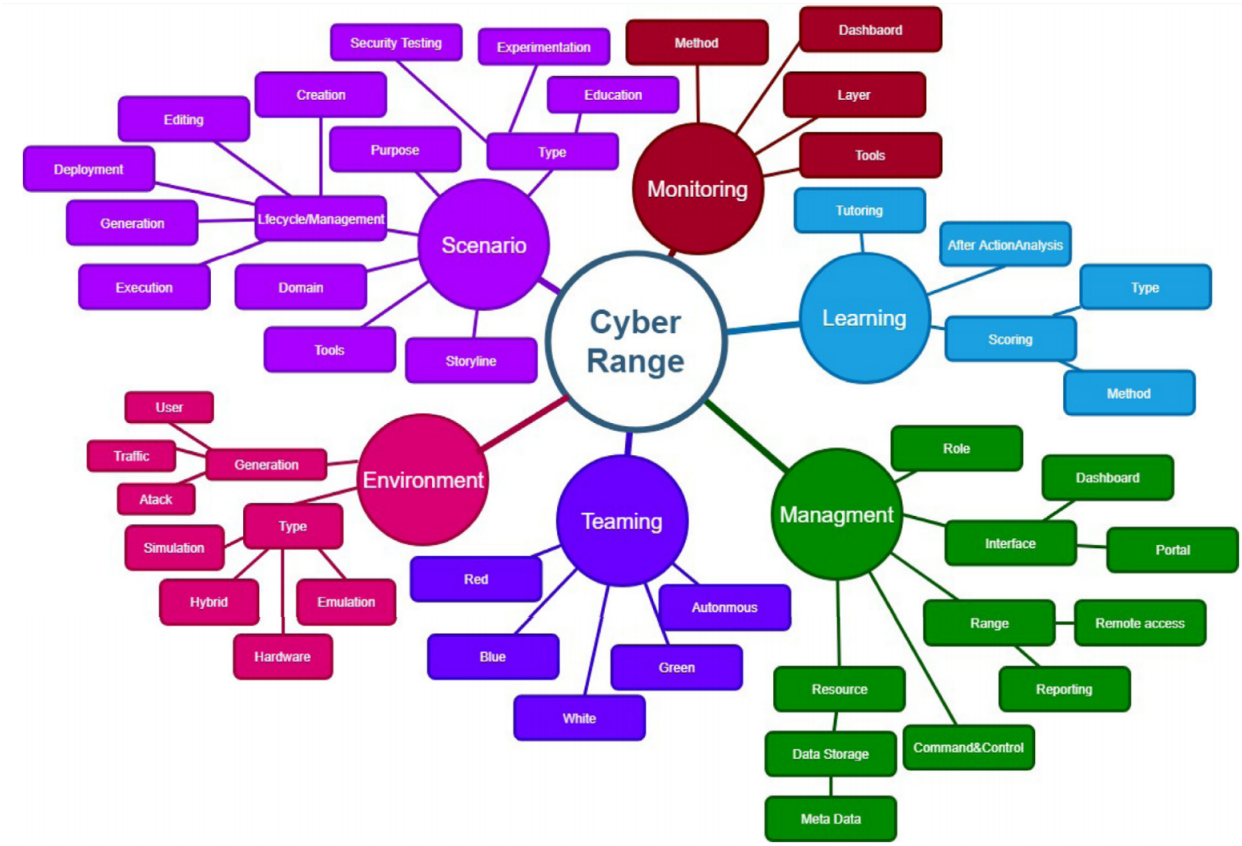


Figure 2.1: Cyber Range Taxonomy Yamin et al. (2020)

Further it would specify the context of use for the system, such as a security or cryptography orientated testbed, and supplying the user with additional background information and stories. Such a story is referred to in this work as the scenario. This supplies the user with the code necessary to run a job on the cyber range without prior knowledge of the system. Monitoring identifies the actions of the user in order to improve the efficiency or learning potential of the system. Such techniques would require an adoption of a ”smart” design, which is outside the scope of the project currently, but could be future work. Learning extends monitoring to the degree of assigning scoring metrics to users to measure the testbed as an effective learning tool. Such a measurement could be introduced in the form of a post-usage review or a game environment in which students achieve scores for completing simulations. Management of the system is includes resource management, allowing the full use of nodes available. Modern operating system software is typically responsible for resource management (Smith 2010), however IoT devices do not run such operating systems. Therefore, management must be a key focus during the development of the testbed to ensure access to resources. This can be decomposed into constituent parts:- a framework for authentication, authorization, accounting, reservation, and experiment scheduling (Gluhak et al. 2011). The teaming aspect refers to the groups and individuals who are involved in the creation and participation of cyber

range scenarios. This includes the red (attacking) team and blue (defending) team. Such teams will be at the designation of the user. The white team is responsible for designing of the scenario, which may be the user depending on their skill level. The environments contain the context to support scenarios. These can be virtualised systems designed as such or designed to emulate real physical systems. Alternatively, truly physical systems or hybrid setups may be implemented (Ficco and Palmieri 2019).

The diagram features six sections. From the diagram it can be shown that any sections can combine to orchestrate a testbed, with additional sections providing one or more features. The diagram denotes a complete cyber range, a federation of such would not combine sections but an entire cyber range with one another. In order to ensure compatibility in the event of federation, the six sections would require either a compatibility layer or reference point specification (Wahle et al. 2009).

Additionally, NIST (National Institute of Standards and Technology) suggests (NIST 2020) that there are specific properties of a cyber range that dictate the standard to which a Cyber Range can be rated. These include:- technical components, realism and fidelity, accessibility and usability, scalability and elasticity, and curriculum and learning outcomes. Technical components describe the technologies used to operate the cyber range. Realism and fidelity overlaps with the scenario taxonomy discussed, supplying the user with the experience to operate and inspect simulations with adequate granularity to make justified changes and receive meaningful results. Therefore, a scenario may benefit from less realistic behaviour in order to allow better comprehension of the underlying technology by the user (NIST 2020). The judgement of accessibility and usability is within the monitoring taxonomy, evaluating the range of users and their speed of system comprehension. Scalability refers to the ability of the cyber range to support the target population of the system. For example this projects aims to support current and future generations of Bournemouth IoT students. Elasticity refers to the time required to increase capacity to support additional users, such as the expansion into a federation of testbeds. Scailability and elasticity are therefore under the technical components and environment taxonomies, providing the software and hardware of the hosted application can support such conditions. Lastly the curriculum and learning outcomes, which determine the effectiveness of the testbed, lie within the learning taxonomy, in order to evaluate the learning potential of the testbed.

2.4 User Interaction

Gluhak et al. (2011) states that user impact is a challenge to be measured and overcome to ensure adoption of IoT.

The work in (Čeleda et al. 2020) presents the KYPO4INDUSTRY cyber range which was designed to address the cyber security skills gap within industrial control systems

(ICS). The training facility is ideal for beginner and intermediate computer science students in a simulated industrial environment. The testbed replicates the physical setup of ICS systems:- PLCs, memory controllers, peripheral devices, isolated networks. Notably, such a sophisticated testbed requires training, designed to “provide an awareness of threats within the ICS domain with practical experience”. The training includes motivation, real attacks, and legal issues, threat modelling, threat creation, and threat deployment.

SecureWSN is a two-parted framework existing of a WSN part and a server part. TinyIPFIX controls data transmission within the WSN and the GUIs CoMaDa and WebMaDa allow user interaction (Schmitt et al. 2017). WebMaDa (Web-based Mobile Access and Data Handling) framework allows monitoring testbeds using mobile devices. WebMaDa highlights the already addressed testbed requirement to provision users and privilege control and the restriction of which measurements are accessible to whom. The research states such controls as paramount for the operation of the testbed in a global (publicly exposed) configuration.

2.5 Testbed Federations

The FIRE (Future Internet Research and Experimentation) provides a federated infrastructure where research and large-scale experimentation testbeds are prevalent (Kalatzis et al. 2018). FIRE seeks to ”create a dynamic, sustainable, large scale European Experimental Facility... to boost European innovation and its competitive role in defining Future Internet concepts.” (Gavras et al. 2007). With a focus on future internet technologies, domains such as:- 5G, Software Defined Networking and Cloud Computing are some areas of operation for residing testbeds. However the federation lacks Cyber Range technologies, testbeds within the focus of security. Futhermore the domains addresses are outside the scope of this testbed project.

GENI (Global Environment for Networking Innovations) is an open source virtual infrastructure for large-scale network experimentation via a federated architecture (Raytheon BBN Technologies 2020) (GENI-NSF 2020). Its focus is on large-scale federated networking, operating across international domains (See Fig. 2.2). Originally GENI’s orchestration of testbeds implemented no unified interface. It is argued that GENI’s varying range of experiment styles, durations, and sizes results in a lack of a unified experiment interface (Berman et al. 2014), however forgoing a unified user interface encourages the development of compatibility tools.

FIESTA-IoT (Federated Interoperability Semantic IoT Testbeds and Applications) provides “experiment-as-a-service” through federated infrastructure of heterogeneous IoT devices (European Union 2020c). Providing access to 10 testbeds across multiple countries, FIESTA-IoT is designed to resolve the issues associated with extracting data from isolated testbeds across different industry sectors. With focus on scalability and inter-

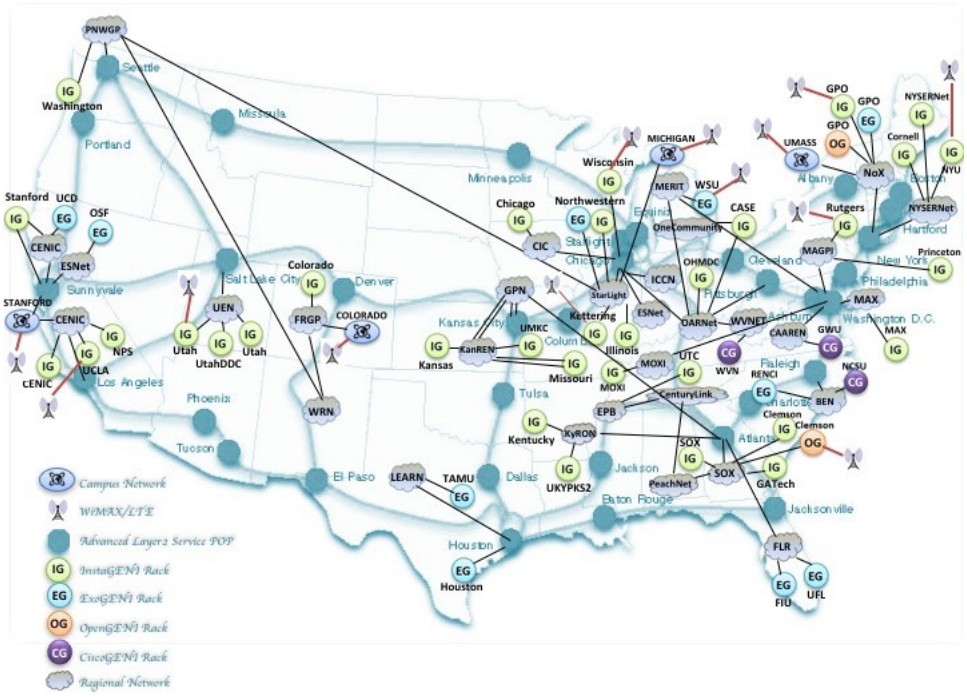


Figure 2.2: Map of the GENI Testbed (Raytheon BBN Technologies 2020)

operability of IoT devices, the platform allows for the translation of data to a common nomenclature through an API. (Kalatzis et al. 2018) describes the importance of proving a common standard for accessing of data allows for the semantic interoperability of these testbeds, in discovering resources and accessing experiment data streams. This work demonstrates that a reference point specification is not required for the initial development of a testbed, as future modifications or translations will enable interoperability between federations. However, the performance overhead as a result of such implementations is not clear in the research.

The Pan-European laboratory (Panlab) is further evidence that a reference point specification is not required during the preliminary lifetime of testbed. Panlab is built upon the federation of originally distributed testbeds that become interconnected. The technical infrastructure underlying the federation is a web service through which available resources can be queried and requested (Wahle et al. 2009). The available resources are stored in a repository, whereby a processing engine is responsible for provisioning the requested infrastructure. Each testbed maintains an independent domain and communicated via an implemented reference point specification, held on a gateway, located at the testbed's network edge. Such implementations also highlight the commonality of addressing testbeds through Application Programming Interfaces (APIs).

European Union funding shows that there is an identified need to close the gap and

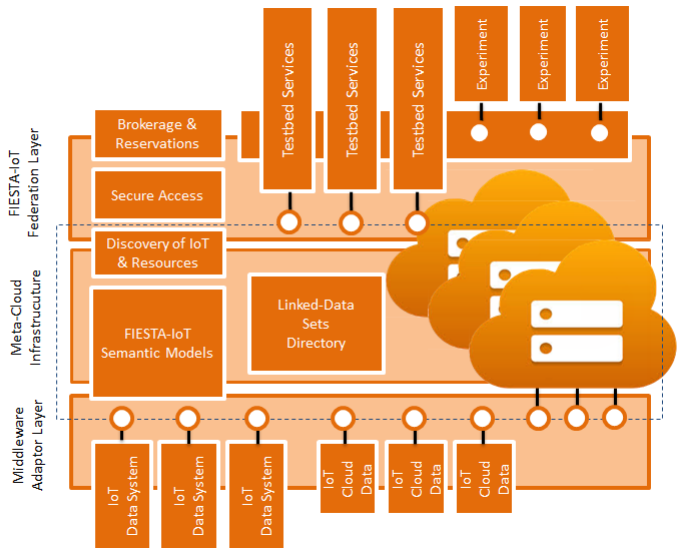


Figure 2.3: The FIESTA-IoT Architecture (European Union 2020c)

build the skill sets required for future technologies, as projects are initiated to improve IoT comprehension across Europe. Horizon 2020 is one such funded project with four state-of-the-art and compatible cyber security projects: (European Union 2020d)

- The ECHO project (ECHO 2020) is described as a system of federated cyber ranges designed to increase the competency of cyber security within the European Union.
- CONCORDIA project is a cybersecurity competence network providing an ecosystem to lead research, technology, and industrial and public competencies (European Union 2020a).
- SPARTA is another cyber security competence network aimed to coordinate research, innovation, and training within the European Union (ANSII 2020).
- CyberSec4Europe is a research project focused on the implementation of potential government structures in order to create a European Cybersecurity Competence Network with an emphasis for best practise examples (European Union 2020b).

2.6 Scenarios

Building the computing infrastructure is only the first step towards the successful execution of the cyber exercises. The design, validation, and deployment of scenarios are costly and error-prone activities that may require specialized personnel for weeks or even months. Furthermore, a misconfiguration in the resulting scenario can spoil the entire cyber exercise (Russo et al. 2018). For this reason, like other design and development

solutions, a definition language can be used. Scenario Definition Languages (SDLs) allow for infrastructure, topology, network and simulation control through a declared markup language that cloud provides and other third party's can understand (Jafer et al. 2016). However the use of an SDL can be problematic, as the complexity of IoT devices and technology requires the SDL to maintain versatility which can result in over complication and over complexity of development. In addition to this requirement, it is quite difficult to port IoT applications to different platforms whilst maintaining full feature set support Ahrens et al. (2009).

Cyber security scenarios typically feature the appearance of teams. The teaming allocates groups from individuals who are involved in the creation and participation of cyber range scenarios. The red team carries out the attacking, as an adversary to the typical agenda of the system. The blue team is responsible for defending attacks executed by the red team. The white team is responsible for the designing of scenario, ensuring that the simulation is corrected executed according to the intended design. The green team carries out monitoring and maintenance of scenario infrastructure, ensuring the continuation of the simulation. Other teams may be in effect, to interact with the attacking and defending sides for positive or negative effects (Østby et al. 2019).

Chapter 3

Background Study

3.1 The Stack

3.1.1 Hardware

Software based testbed solutions can ”suffer from several imperfections as they make artificial assumptions on radio propagation, traffic, failure patterns, and topologies” (Egea-Lùpez et al. 2005). A hardware testbed could support any range of marketed IoT device configurations. For this project the scope of applicable devices is reduced by factors of:- costs of procurement and maintenance, availability of stock and geographical restrictions, speed of procurement and potential for future reuse. For these reasons the Raspberry Pi device was researched.
Raspberry Pi 4 offers ARM (Cortex-A72 ARM v8) chipset capable of running IoT operating systems found in table 3.1. The initial networking capabilities offer 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, Bluetooth Low Energy and Gigabit Ethernet (Raspberry Pi Foundation 2020b). These wireless protocols are frequently used in IoT settings. The device also offers 40 GPIO header which can be used to extend the functionality of the device to include sensors (Raspberry Pi Foundation 2020a). The device requires a minimum power supply of 2.5 amps.
Zolertia RE-Mote likewise has support for 2.4-GHz making it compatible with typical domestic Wi-Fi. The RE-Mote also supports lower frequency ranges of 863-950MHz allowing compatibility with numerous other protocol standards that are adopted for IoT:- IEEE 802.15.4, 6LoWPAN and ZigBee. The Re-Mote contains 2 bi-directional radios, capable of receiving and transmitting up to 20 kilometres. Whilst the physical testbed configuration is both indoors and permanent, a future revision of this testbed for wireless capabilities would benefit from the increased range of Re-Motes antennas. As with the Pi 4, the hardware supports peripheral devices through IO pins (Zolertia 2016) for future expansion of the device or network. In line with the power resource constraints of IoT devices (Sabri et al. 2017), the RE-Mote features ’sleep’ power states capable of power

	Contiki-NG	RIOT OS	TinyOS
Language	C	C or C++	nesC
Simulator	Yes	No	No
RAM/ROM	2KB/30KB	1.5KB/5KB	10KB,100KB
Multi-threading	Protothreads	Yes	No
Real-Time	Yes	Yes	No
License	BSD	GNU	BSD
Native Port	Yes	Yes	No
Network Protocol Support	802.15.4, BLE, 6LoWPAN, RPL, CoAP, MQTT	802.15.4, BLE, Zigbee, LPWAN, 6LoWPAN, RPL, CoAP, MQTT	802.15.4, Zigbee, 6LoWPAN, RPL, CoAP, MQTT.
Source	(Contiki 2020)	(Nohlgård 2017)	(Decker 2013)

Table 3.1: Comparison of Zolertia ReMote compatible IoT Operating Systems.

consumption rates of 0.4μA, enabling the testbed to consume minimal power when not in operation (Zolertia 2020b).
Given these advantages, within the parameters discussed, the RE-Motes are the more practical choice for the development of an IoT testbed.

3.1.2 Operating System

There are many technologies available to supply an operating environment. Virtualisation offers a solution for the entire software stack:- from the firmware to the application software. Depending on the adoption of the Pi 4 or Zolertia hardware there are a number of choice operating systems available.

The critical drawback to the TinyOS operating system is the operation on the nesC programming language. This requirement would be inherited within the user experience, negatively impacting novice system users with overcoming the learning curve of a lesser known language. Designed as an IoT friendly OS (Nohlgård 2017), RIOT OS provides the richest network protocol support of the compared operating systems, whilst requiring the lowest storage requirements. Limited within the scope of the parameters on table 3.1, RIOT OS would be the choice operating system.

3.1.3 Application Software

In order to run IoT like simulations virtually, emulation software must be run. OpenWSN is a wireless sensor network framework for development on IoT device firmware (Claeys 2020). The framework offers emulation for development, through a python virtual en-

vironment. Alternative software applications have been built encompassing a graphical interface alongside the emulation.
Such a software is Cooja, capable of compiling and running IoT device code on the hosts x86 computer. It is popular with the IoT research community (Hendrawan and Arsa 2017) (Naik and Joshi 2017) (Mahmud et al. 2019) (Oliveira and Vazão 2018). It can be used to emulate sensor nodes for virtual Wireless Sensor Network (WSN) simulations and is capable of custom node definition via the implementation of a Java class. This software could be used to allow definitions of devices by advanced users of a virtual testbed, to enable testing without the associated costs of developing new physical hardware. However, (Gluhak et al. 2011) observes that an IoT testbed requires more than WSN capabilities.
Research by (Munoz et al. 2019) shows the operation of heterogeneous physical testbed named *OpenTestBed*. An open-source testbed, designed to facilitate cheap setup costs with off-the-shelf commodity hardware and maintain reliability and repeatability through the adoption of open-source software. The software uses logging information through serial connections to facilitate a variety of simulations. It's physical architecture consists of a Raspberry Pi and 4 OpenMote B motes. The Raspberry Pi acts as a central server running a python MQTT broker program to send and receive job metadata. This configuration is termed an "OtBox", a logical testbed unit that allows non federated grouping for data aggregation via the broker. There is also support for integrating OpenTestBed into OpenWSN allowing for application of the serial data by external tools. The focused of the OpenTestBed is to serve as a proof-of-concept that dedicated testbeds and cyber ranges can be developed cheaply compared to traditional institutionally dedicated testbeds.

3.2 Chosen Solution

The following section highlights the research and subsequent solutions that will be development in the course of this project.

3.2.1 Architecture and Scope

Research, (Gluhak et al. 2011) surveyed multiple IoT testbeds to define the scope and architecture options for current testbeds. Figure 3.1 shows the resulting work.

They further (fig 3.2) define the two or three tier architectures available to form a network for the testbed.

Figure 3.3 shows the proposed approach this work aims to undertake, in contrast figure 3.1.
The testbed will be developed with an emphasis on domain of security, in order to qualify as an IoT-CR. This can have both benefits and drawbacks, as domain specificity can reduce the applicable target audience, it enables specific skill sets within the domain to

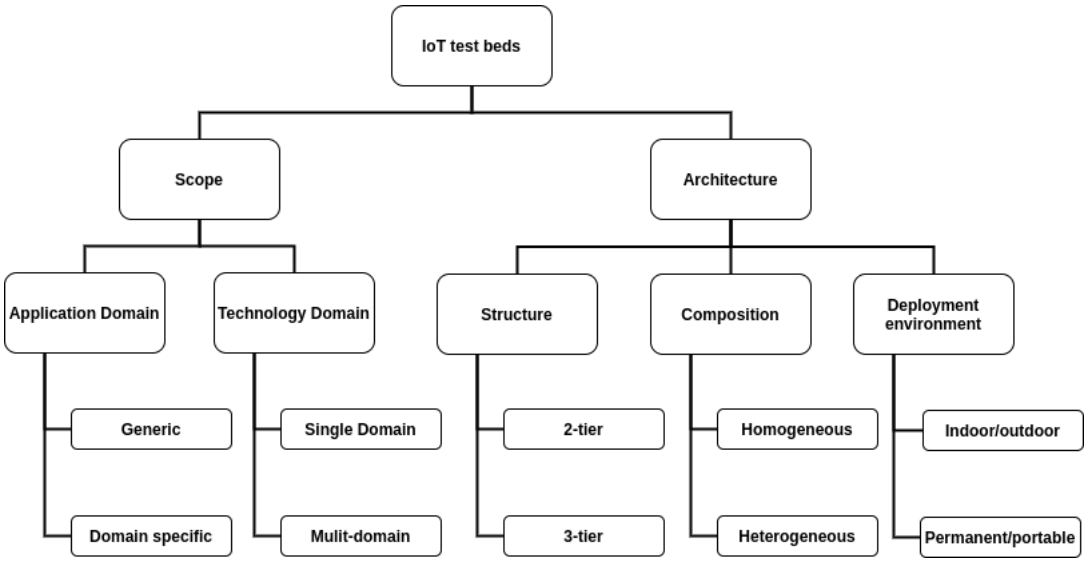


Figure 3.1: (Gluhak et al. 2011) (recreated) Scope and architecture of testbeds.

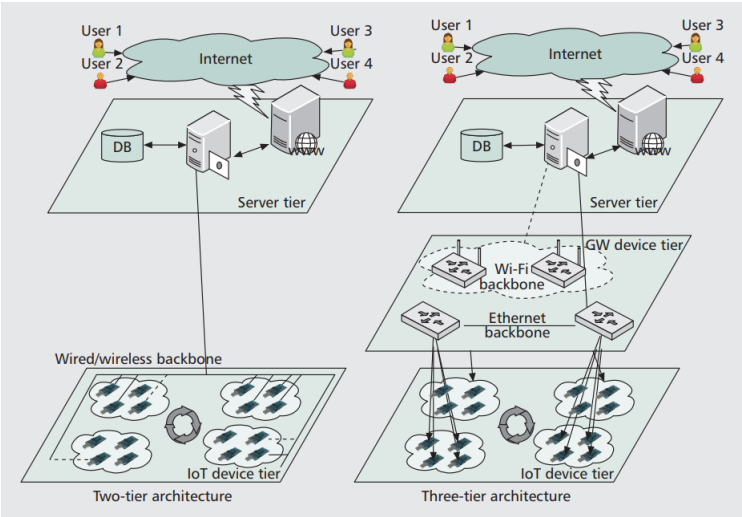


Figure 3.2: Two and three tier architectures. (Gluhak et al. 2011)

be targeted. ”A multidomain testbed is one that combines different IoT technologies into a common experimental facility”. The testbed will feature a combination of the virtual and physical components, making the testbed multi-domain. A technological mutli-domain testbed increases the coverage of accessible IoT use cases.

The overall architecture of the physical and virtual testbed does not feature any hierarchical network responsibilities, such as a gateway router bridging to the traditional internet, therefore the testbed is classified as 2 tier. The difference can be seen in figure 3.2. As the system features support for a number of different devices, across both the virtual and physical space, the network is heterogeneous. The Deployment environment is both persistent, as the motes will be wired into the server, preventing movement and therefore likewise, indoors. With the intention of initiating an in-house project for continuous

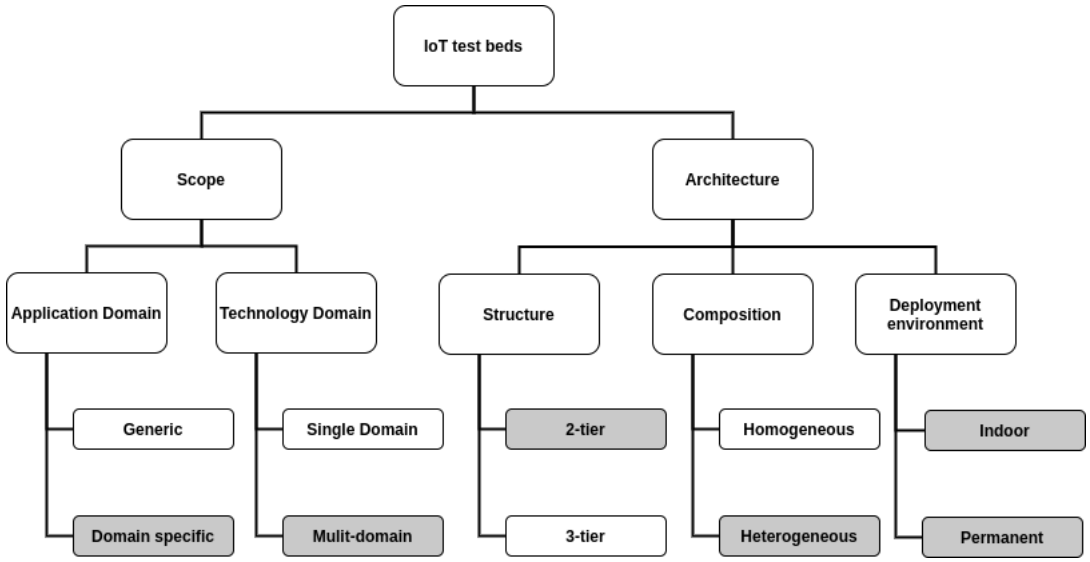


Figure 3.3: The planned architecture.

development, a custom solution is proposed, whereby the any testbed expansion can be documented and undertaken by IoT students, enabling continued evolution during the research lifetime of IoT testbeds at Bournemouth University.

3.2.2 Functional Considerations

Unlike other sections of this project, the chosen solution is not departmentalised into virtual and physical. Subsequently the delivered solution seeks to find a common ground between both, in order to create a solution that is cross architecture.

Contiki-NG provides the operating system that underlines this project. Hosting an emulated operating system through file structure and build tools, contiki-NG enables us to emulate multiple devices to build software accordingly. Written in C, this open source operating system can be modified in it's through the modification of C source files. Additionally users can run simulations on the native Contiki-NG platform which can compile and execute supplied C code. Notably, when built from source code, it can be compiled and configured to operate with Cooja simulator.

Cooja provides the a software framework for running contiki supported scripts, within a simulation setting, providing a graphical user environment for control and analysis of the simulation in real time. Cooja also allows for headless operation for debug. This can be used for the virtual testbed.

The Zolertia RE-Mote was developed to target the hardware development platform market. It was designed by universities and industrial partners as a joint program in the frame of the European research project 'REliable, Resilient and secUre IoT for sMart city applications' (RERUM) (Lignan 2016). Designed because of a gap in existing IoT platforms lacking an industrial grade design, the device incorporates IoT features such as an ultra-

low power consuming ARM Cortex-M3 system on chip and a 600 milliamp hour battery. By utilising the Zolertia RE-Mote, the compatibility with Contiki-ng continues, allowing the operating system to address and run code on the devices using built in compatible packages. Contiki-NG is compatible with the ARMv7-M Architecture found in Zolertia motes (Zolertia 2020a), which its self is fully compatible with it's predecessor the Zolertia Z1 mote. The Z1 was released more then 5 years ago, adopted in more than 23 countries and featured in more than 40 scientific publications, it has been part of the academic and research community since its launch (Lignan 2016). Therefore there is a likelihood of further popularity with the successor model. Therefore the testbed will support projects featuring these devices.

If the acquisition of Re-Motes is not practical, for any reason given within the risk analysis table "Changes to hardware", appendix table 2, Contiki-NG can also provide a native platform to enable code to run on emulated hardware.
There are multiple programming languages that have compatibility with different elements of the project. Cooja is written in java, Contiki-NG uses C, however python is the language chosen to build the system. Firstly it the language to which I am most familiar with, secondly the supporting projects (Lucas 2019) are written in python, therefore it is already a requirement that the language be installed on the system. Adopting the language will allow me to share a common virtual environment (venv) and development environment with during the development of the API, or any subsequent revisions.

3.2.3 Non Functional considerations

The diagram 3.4 uses the System Modelling Language (SYSML) in order to deconstruct the requirements of the system into logical units of completion (OMGSYSML 2020).

Figure 3.4 shows a high level depiction of the requirements of the project. The diagram initialises the exposure of elements such as the "Database" and "Motes". The interaction between these blocks are described in figure 5.1.
There are three major requirements to the system:- Virtual, Physical and UI make up the core requirements to the "Run IoT Simulations on a provisioned testbed" requirement. Essentially these lower level requirements must all be met in order for the higher level requirement to be met. In the next section these requirements are arranged within the development methodology of the project, in order to ensure these requirements are addressed.

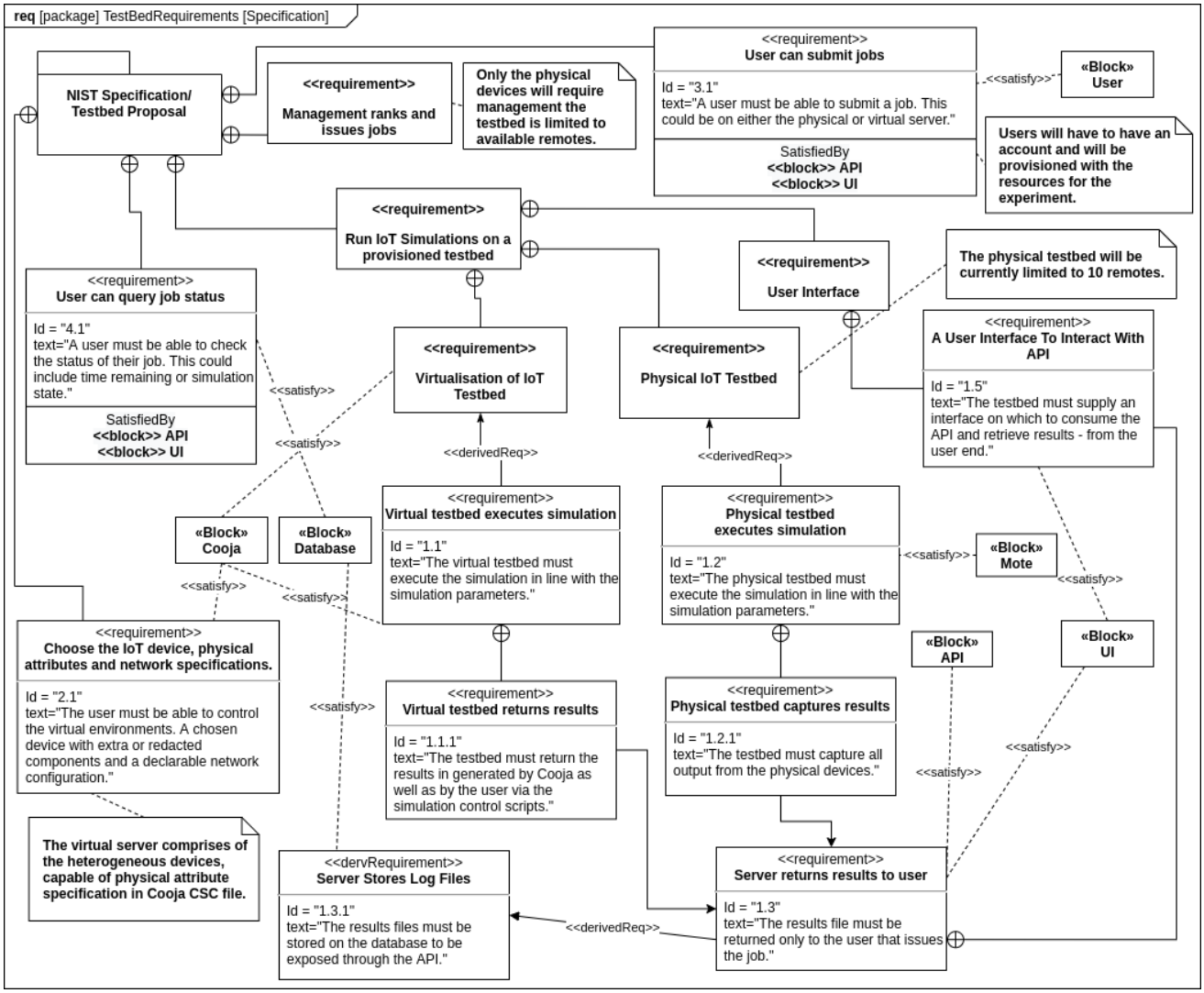


Figure 3.4: Processing Job Requests.

Chapter 4

Methodology

4.1 Development Methodology

4.1.1 Available Methodologies

Many software design methodologies exist that dictate the interaction between developers, users, stakeholders, contractors and the software its self. With multiple deadlines and concurrent research and development, not all of the above entities are in scope, instead the chosen methodology focuses on the aims set forward by the project rather than communication with an end client or contractor.
Linear methodologies, such as waterfall, may be suited to this project. The deadlines are clearly known in advance, which allows for a time plan to utilise the available time and deliver a product by the deadline. However the requirements are less defined, given that the system has not been specifically contracted. While the design of the system can occur upfront, the restricted time schedule of the first deadline means the physical testbed and subsequent functionality of the system will not have been fully designed until after the first deadline. This is contradictory to the waterfall methodology where design of the system occurs ahead of implementation. Likewise there are multiple separate implementation occurrences, for the virtual and physical testbed as well as the graphical user interface. Lastly the waterfall methodology fosters maintenance periods for post implementation, however given the time frame of the project this may be out of scope.
An agile methodology, Dynamic Systems Development Methodology (DSDM) allows for the completion of tasks based on a priority, such as the MoSCoW method. Whilst such a priority system could be implemented, it would not suit the non-granular objectives set out in the project. As an IoT-CR or testbed is free to offer services or features it desires, so long as no federation permits otherwise, there are no priorities implied upon the project by any such contractors or stakeholders, rendering the MoSCoW method redundant.
As this is an individual project, other agile methodologies that require greater than one person are not an option (such as eXtreme Programming or Joint Application Develop-

ment) are not suitable either.
Rapid Application Development (RAD) is another agile software development methodology. RAD undertakes an iterative approach to building a software solution. By specifying the deconstructed objectives upfront with clear time frames, (Prashanth Kumarl and Prashanth, 2014) developers are encouraged to start development promptly. Such a solution is ideal for this project, having already defined the deconstructed objectives in the form of aims (see section 1.2.1). However such a methodology risks poorly defined objectives impacting the whole project. This risk is mitigated due to the isolation of objectives into logical groupings and continuous lookup against the time sheets - appendix 1 2.

4.1.2 Chosen Methodology

RAD has been chosen for this project due to the prompt startup (of individual) development and the ability to adapt time frames to suit the two deadlines, allowing for possible further or updated deadlines. Using the objectives and aims already defined, focus can be applied to specific areas of work, allowing for adequate level of research and development within the time frames defined.
This project will undergo two agile 'sprints' whereby the following will occur:

Sprint	Deadline	Deliverable(s)
1	Paper Draft Due Date	virtual IoT-CR + user interface + scenario + paper
2	Dissertation Due Date	physical IoT-CR + management system + dissertation

Sprint 1 will commence immediately, lasting until the agreed deadline of the paper. This print will see the development and delivery of a functioning virtual Cyber Range. The IoT-CR will utilise the purpose written API (reference oliver here) in order to accept calls to run operations on virtual machine or virtual software running instances of IoT operating systems. This will require the study of the database model. This will also require the study of the API, which will be incorporated into a Graphical User Interface (GUI) also developed and delivered within this sprint. The virtual IoT-CR system and GUI will be tested with the supplied scenario, a pre-designed testbed simulation that is to be provided with the user to demonstrate the system and at work. Therefore the scenario must be researched, designed and programmed to work with the virtual CR. Lastly, this sprint will be marked as completed only once the virtual IoT-CR system is approved and the accompanying paper has been completed and submitted.
Sprint 2 will be conducted following sprint 1. Sprint 2 will see the addition of the hardware IoT-CR to operate alongside the virtual counterpart. This sprint will last for the duration of the project until the deadline of the dissertation. The hardware IoT-CR will require

the acquisition of IoT hardware, setup of the testbed and development of a management system to control the concurrent running of jobs across motes, as concurrency is a requirement of a testbed (Gluhak et al. 2011). Another undergraduate project (Lucas 2019) has setup a physical testbed and accompanying code valid for limited flashing capabilities of the devices, that can be repurposed for the CR. The second sprint will see the development of a resource management system to handle resource allocation at runtime for to enable job concurrency across the physical Cyber Range. This is a requirement not present in the virtual IoT-CR, given the operating systems ability for resource allocation and concurrency for virtualisation of simulations. It is imperative that the given scenario can run on the physical testbed, therefore additional scenario development time must be allocated to the second sprint. The sprint can be marked as completed when the physical IoT-CR system is configured and capable of running jobs.

Time before, during and after the creation of both testbeds must be put aside for writing both accompanying papers. This is recorded within the time management tables - appendix figures 1 and 2. Resources for the two sprints will be stored on a privately hosted version control server.

Chapter 5

Design and Development

5.1 Design

The design is focused around the end-user and is based on the four design principles for Cyber Range development defined by Schwab and Kline (Schwab and Kline 2019). These principles are ambiguous, with no direction of application or restriction to it's practise. This development addresses the principles with the proposed design solutions.

1. ”Provide tools and capabilities that reduce the cognitive burden on experimenters wherever possible.”
2. ”Allow experimenters to encode their goals and constraints and leverage this information to help guide experiment construction.”
3. ”Provide flexibility in design. A good architecture evolves with both its users and technology, and newly developed capabilities.”
4. ”Provide multifaceted guidance to help experimenter produce high-quality experiments.”

Principle 1 Utilising tools to transfer knowledge between entities reduces the cognitive burden on users allowing concentration on significant objectives. This project provides the user interface program designed to prevent the unnecessary manual insertion of the API curl commands, in order to aid the user.
Principle 2 states that goals and constraints must be inherently defined within the simulation parameters to aid the construction alongside the requirements. This will be adopted into the design of the system, ensuring the user has control over the definitions of the simulation, through the use of the SDL.
Principle 3 states that as the IoT landscape evolves, new capabilities should be designed with allowances for flexibility to meet these changing requirements. This project abides by this principle, with the aim of the ongoing development of the testbed by subsequent

students, to ensure the solution remains state-of-the-art.
Principle 4 states that user guidance is supplied to guarantee high quality simulations. This project aims to supply the tools available to guide the user, including the provisioning of a scenario simulation guiding the user start-to-finish with experience of the system before developing custom simulation.

5.1.1 System

Working with Oliver Knock the API endpoints where specified, including the requirements of the 3 file types, and the expected user interaction. The diagram 5.1 has been created using SYSML, in line with the requirements diagram, in order to visualise how the overall system is to function and be interacted with.

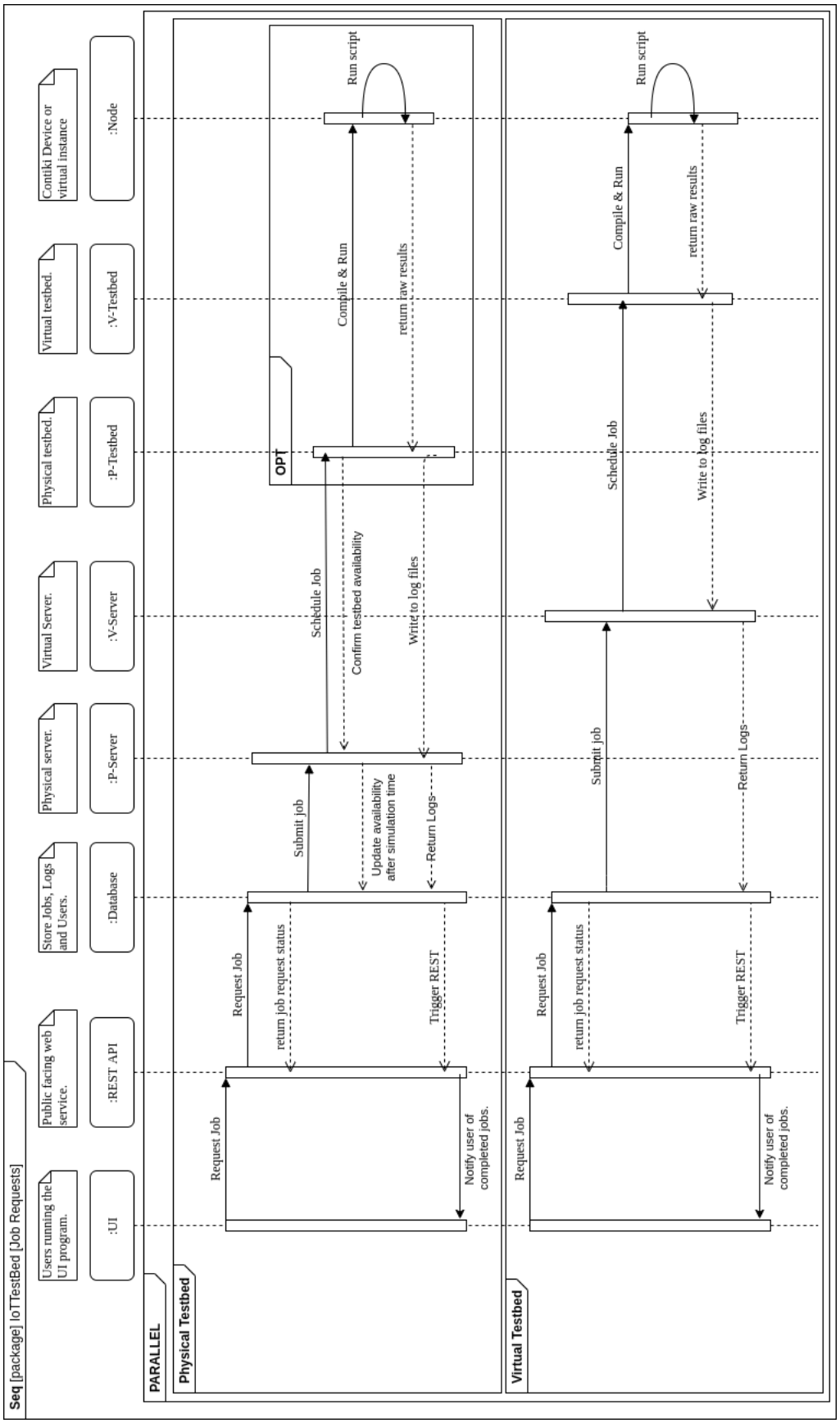


Figure 5.1: Sequence Diagram: Processing Job Requests.

The notation shows the systems working in parallel in order to provide operation for both the physical and virtual testbeds. However in order to allow the initial and independent development of the virtual sever in line with the deadlines of the paper, see appendix 2, the two components are capable of operating independently. This independent architecture can enable future migration of the virtual services to a cloud provider, making use of full or hybrid cloud configuration options. Additionally this separation can allow manipulation for conformity with regulations of a federated testbed. Research shows that both cloud virtualisation and federations provided the means to scale to the requirements of an organisation (Gluhak et al. 2011).
The physical server relies on resource management in order to fully utilise the available motes connected to the server. In this way, simulations consisting of a low number of motes have a higher chance of running concurrently, making use of the resources optimally. This design also encourages optimal usage by the users, by only requesting as many nodes as required for the task, more jobs can be executed within any given time period.

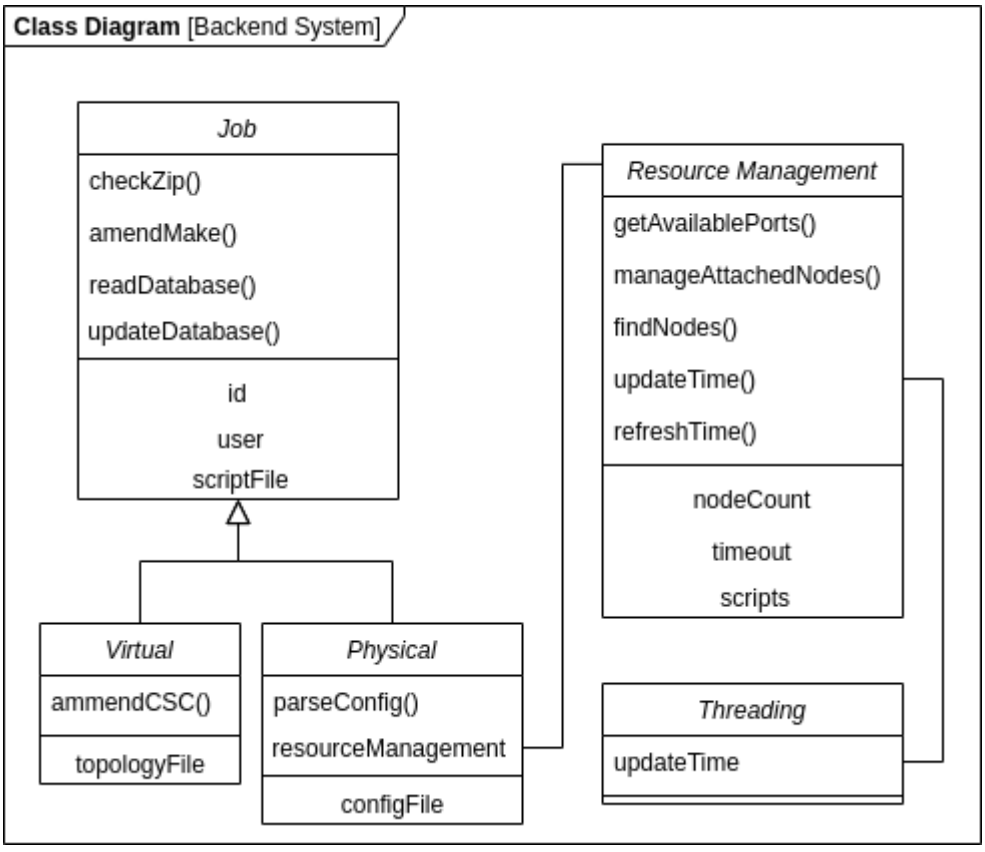


Figure 5.2: Class Diagram: Processing Job Requests.

Figure 5.2 declares the format of the code to be written for operation of the backend server systems. The *Job* class is responsible for the database interaction and file

manipulation, as file references are stored as paths on the database. The *virtual* class manipulated the CSC file in relation to the current job. The *physical* class likewise will handle the current job configuration file and call upon the resource manager when executing the job. The *resource Management* class handles the interaction with the physical hardware:- USB ports, available devices, timeouts. It is also expected that a *threading* class be required in order to independently maintain a time record of currently running simulations.

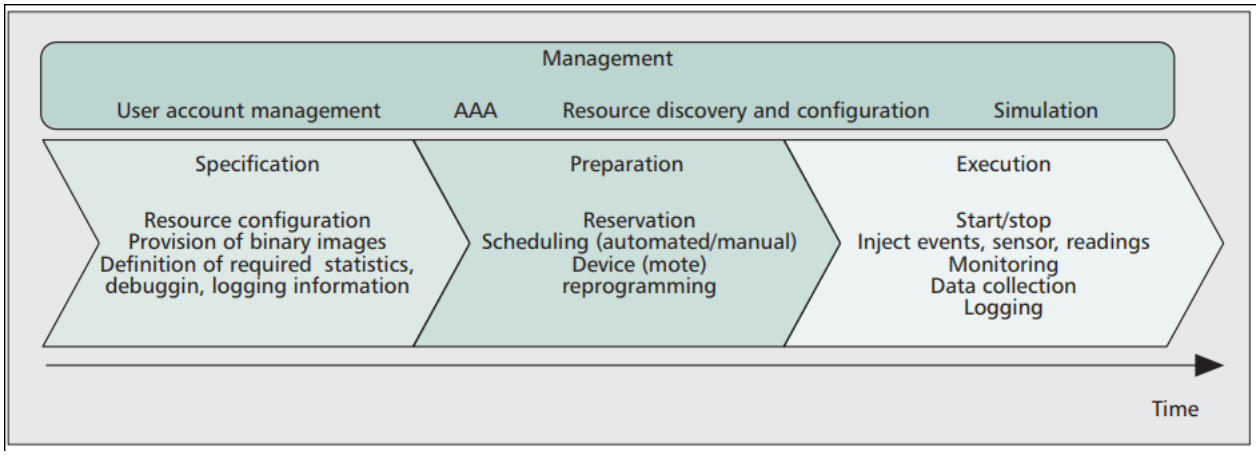


Figure 5.3: Workflow of a typical testbed - (Gluhak et al. 2011).

Figure 5.3 shows the typical workflow of a testbed according to (Gluhak et al. 2011). As such the depicted workflow exemplifies how the system will function. The *User account Management* refers to the ability to permit and provision users, such as 'credit' schemes seen in other facilities:- (FIT 2020) and (CENSIS 2020). The testbed has no plans to enforce no such limitations. The *Resource discovery and configuration* comprises of the reservation and scheduling, activities which are handled by the resource management, as explained in the class diagram. *Simulation* is the monitoring and control of the simulation, including logging. These processes are separated across in both the IoT devices themselves and the software that manages them. Cooja can control the simulation code execution through a virtual CPU, whereas the zolertia RE-Motes will use onboard ARM chips. *AAA* is undefined in the paper. These elements, which have been satisfied in the design, can combine to form the management component of a testbed's workflow.

5.2 Development

5.2.1 API Python Wrapper

In the advent of targeting advanced users, a curl compatible API is provided for users of command line interfaces. However, to to aid novice users in consuming the API, which would otherwise require the understanding of lengthy curl requests, a lightweight python3 client wrapper is implemented and provided. This can be found in the artefact. The wrapper allows the user to sign up and login to the Cyber Range by operating on the API endpoints through a menu driven interface. The wrapper holds the JTW token associated with the logged in account, allowing users to query their files, jobs and fetch log files. Figure 5.4 shows a screenshot of the wrapper in use, with options for creating a job from a scenario, running the job and collecting the log files etc.

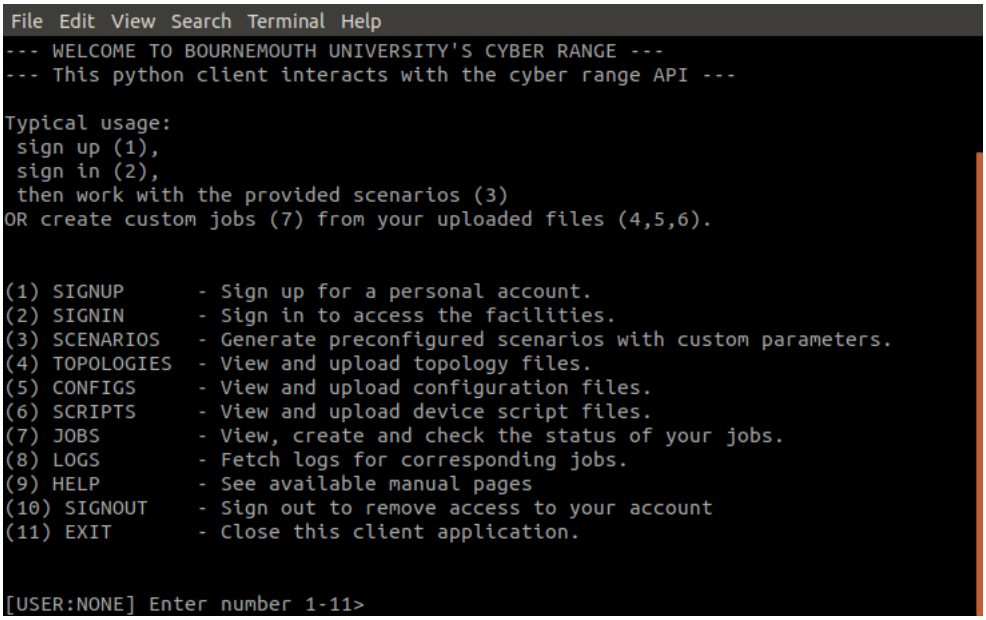


Figure 5.4: The home screen of the wrapper.

5.2.2 VIRTUAL IoT-CR Back-end Operation

In order to satisfy the various needs of IoT researchers, the cyber range is virtualised to offer a range of devices that would not otherwise be accessible in physical form. The proposed Cyber Range operates using Cooja, a network simulator written in Java and designed specifically for Wireless Sensor Networks. Cooja is a tool provided within the Contiki-NG Operating System, which itself is focused on ”dependable (secure and reliable) low-power communication and standard protocols, such as IPv6/6LoWPAN, 6TiSCH, RPL, and CoAP” (Contiki 2020). Contiki-NG is also popular with the WSN research communities.

Cooja is able to operate using the scripts and libraries available within Contiki-NG, providing a graphical framework for users to assign custom scripts to virtualised devices, control these devices within a topology map and test networking scenarios with the provision of tools such as viewing real-time device standard output or adding breakpoints within the simulation for key events.

Cooja also has the ability to run in headless mode (i.e. with no graphical user interface) thanks to an inbuilt debug mode. Cooja allows simulations to be saved in '.CSC' format, saving all the parameters for the simulation in XML. Whilst this can have the typical usage of saving and reloading a simulation, the CSC files can also be passed to Cooja via the command line to force a non-graphical Cooja process. Furthermore the 'Simulation Script Editor' tool provided within Cooja allows for users to have scripted control over the simulation via Javascript code - seen in Fig. 5.5.

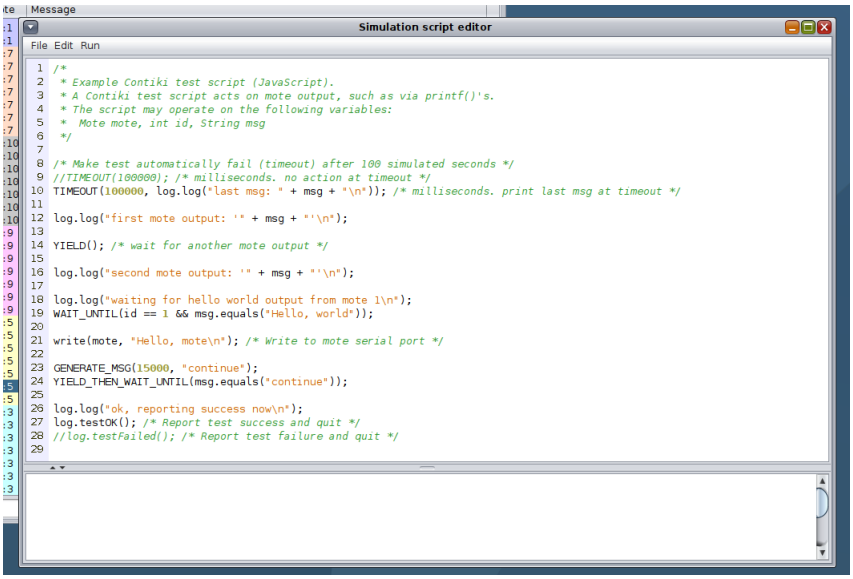


Figure 5.5: Simulation Script Editor w/example Javascript code

The Simulation Script Editor allows the user to save the simulation CSC file amended with the Javascript code as another XML element, allowing the simulation to run automatically, without the need for manual intervention. Users can write their own code or use a number of preconfigured scripts. This allows for a logical detachment from Cooja, steering the Cyber Range away from simply providing Cooja as an Application-as-a-Service to utilising Cooja's simulation abilities to provide users with a Cyber Range experience similar to other IoT testbeds. As the Simulation Script Editor is fully integrated within Cooja, and therefore Contiki-NG, all system defined events (YIELD, PROCESS, YIELD_THEN_WAIT_EVENT_UNTIL, PROCESS_WAIT_EVENT_UNTIL, etc.) (Contiki-NG 2020) can be detected and triggered within the Javascript code, allowing programmatic simulation control.

Given the simulated nature, rather than emulated, the system can run on servers capable

of handling multiple jobs, as not much computing power is required to emulate IoT devices that are typically resource constrained. Furthermore, virtual systems offer scalability beyond physical systems due to the hardware independent properties, but can restrict the experience in such cases as interacting with physical buttons or sensors. Cooja uses the hosting Operating System to maintain the resource allocation required for simulations, so future development allows us to operate on hosted server-grade hardware for improved performance with costs dictated by system usage rather then maintenance of physical devices.
The automated handling of Cooja, dictating the control and running of jobs, is written in python3. This script polls the database periodically for newly submitted jobs. When new jobs are run in Cooja the logs, standard output and standard error are captured by default, as well as any logs generated by the user within their simulation script code (the Javascript code). These logs are then exposed over the API for users to consume.
In order to enable initial user comprehension and to enable quicker job development, the system can provide scenarios that offer the complete Contiki-NG project build folder and Cooja simulation (CSC) file required for any job, shown in 5.3.

5.2.3 PHYSICAL IoT-CR Back-end Operation

First the server must identify the devices connected to the system. This is done by importing the code from ”testbed_src” - the work of (Lucas 2019). The server maintains a list of devices connected in contrast to a list of devices in use in order provide resource management, a server controlled by the operating system in the virtual testbed. As such the system is designed to scale, supporting the maximum USB devices the server host hardware allows. Figure 5.6 shows the initial testing and development, a Proof-of-Concept (PoC) of flashing multiple motes.

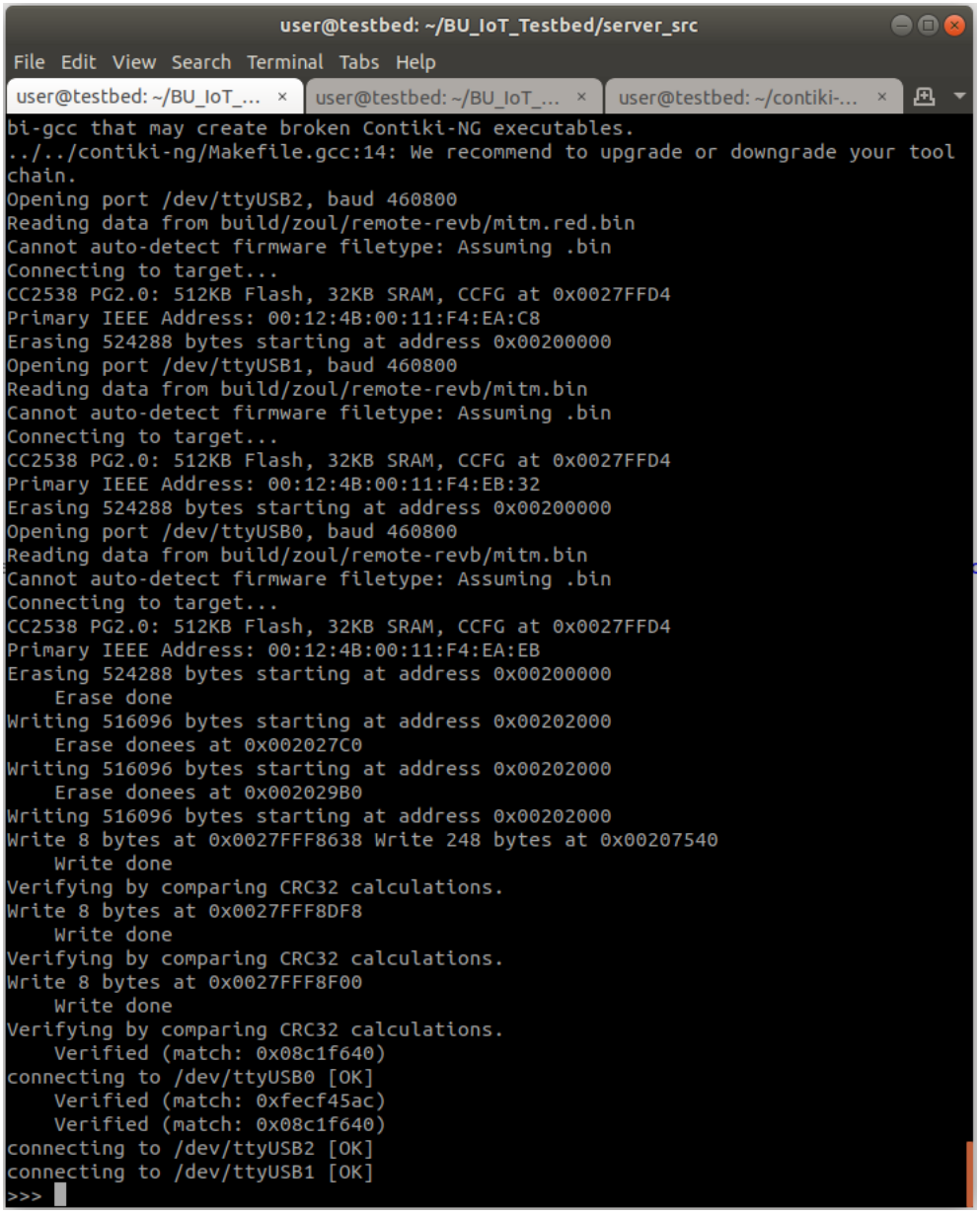


Figure 5.6: 3 motes flashed concurrently.

Similarly to the virtual server, the physical server will poll the database for jobs registered through the API. Replacing the Cooja CSC file, the user defined 'config' XML file controls the simulation on hardware devices - appendix listing 1. This file acts as a custom defined SDL file (Jafer et al. 2016). The file is again parsed and simulation parameters are established in order to request availability on the testbed. If the testbed has no vacancy, the job is simply dropped pending another database poll. Where there is vacancy, the simulation is executed within the simulation parameters, notably the time duration. The timeout duration replaces the Javascript simulation control code, given that the motes execute processes on dedicates CPUs, remote processing of system calls is not possible. Individual motes are flashed with their assigned scripts, through threads

that allow enable synced flashing to ensure coherent start times amongst the nodes. When the timeout occurs the system marks the motes as vacant and returns the log files through the API.

5.3 Cyber-Range Scenarios

Generating these scenarios, users will be able to understand the format of the simulation code in use, providing the opportunity to change the parameters of a given scenario such as: wireless protocols, quantity of nodes, density of network, percentage of nodes running particular code (accounting for sink nodes or different team sizes). In section 6.1, the example user generates the code for the scenario ”Pass the token”. This scenario demonstrates typical blue team/red team cyber security events. The scenario plays out as two opposing teams trying to share a token value; whilst the blue team attempts to increment and share the token between themselves, the opposing red team attempts to intercept and decrement the token before passing it onto the blue team, resembling a simplistic man-in-the-middle attack. Blue team tries to increment the value resembling a defence response. No node may send more than a single token without receiving a new token value from an inbound packet, this effectively locks out other packets to enable the effect from the man-in-the-middle attack. The blue teams token value must reach an upper bound, whilst the red team tries to achieve a lower bound. When a node reaches either of the thresholds, it declares a state of win or defeat and terminates the decentralised program, posting the state to logs. Figure 5.7 gives a diagrammatic view of the scenario.

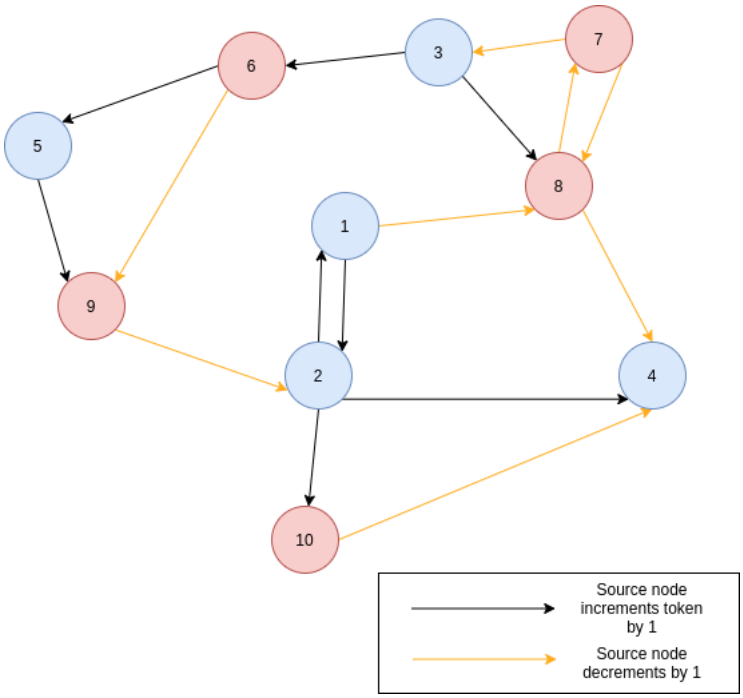


Figure 5.7: 10-node network depicting the token modification and exchange.

In this scenario example, with a starting value of 10, a walk across the network encompassing nodes {1,8,7,3,6,5,9,2,4} results to a token of value 12. However, as all nodes that start the simulation are able to send one packet before locking, multiple walks across the network can occur at a given time. Walks can be cut short by other walks, due to the node single packet lock, varying the simulation on each run, and at the point that each node reaches the upper or lower threshold, causing the eventual end of the simulation. It is worth noting that the scenario can be executed with a variety of underlying networking protocols and technologies, including both non-IP (e.g. NullNet) and IPv6 (e.g. RPL and 6LoWPAN) networking stacks. This allows the trainees to repeat simulations with varying numbers of nodes, using different network technologies, record results and compare different aspects of IoT networking. They gain a working understanding of a cyber security team structure and exposure to an high-level programming of IoT devices.

Chapter 6

Results

6.1 Demonstration

On first arrival at the system, users are required to make an account (fig. 6.1). The username and password are then used for login and the email notification for job completion. Users are given the change to generate the required files for a job, in the form of a scenario. These files are added to the users set of files so that custom variations of the scenario can be created. Alternatively, users can upload files, as seen in Fig. 6.4. Figures 6.5 and 6.6 show the creation and subsequent scheduling of the scenario job. In Fig. 6.5, the user picks from each file category to build a job, which is saved to the user account. Only when the user wishes to run the job do they utilise the job scheduling (figure 6.6). Once the job is completed, users will get an email informing them (Fig. 6.8) and access to the logs is given in the client wrapper (Fig. 6.7).

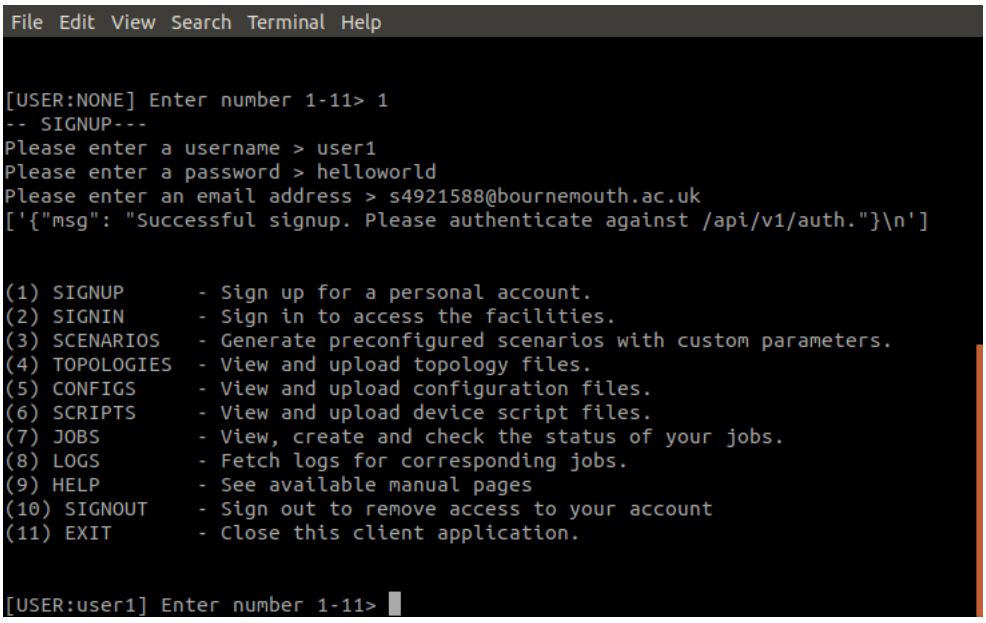


Figure 6.1: The signing up process. Username now appears in prompt.

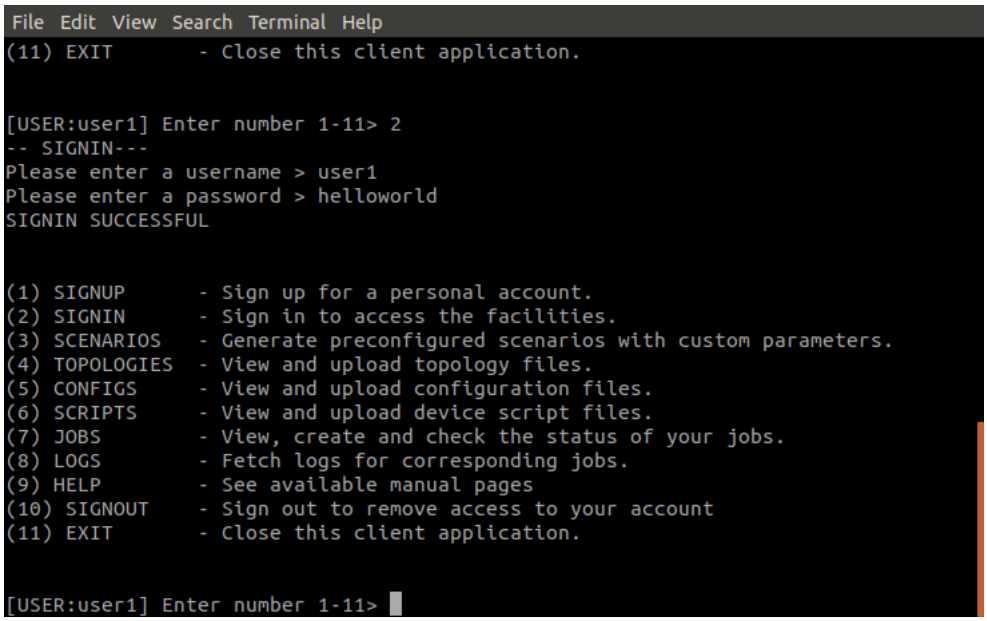


Figure 6.2: The signing in process.

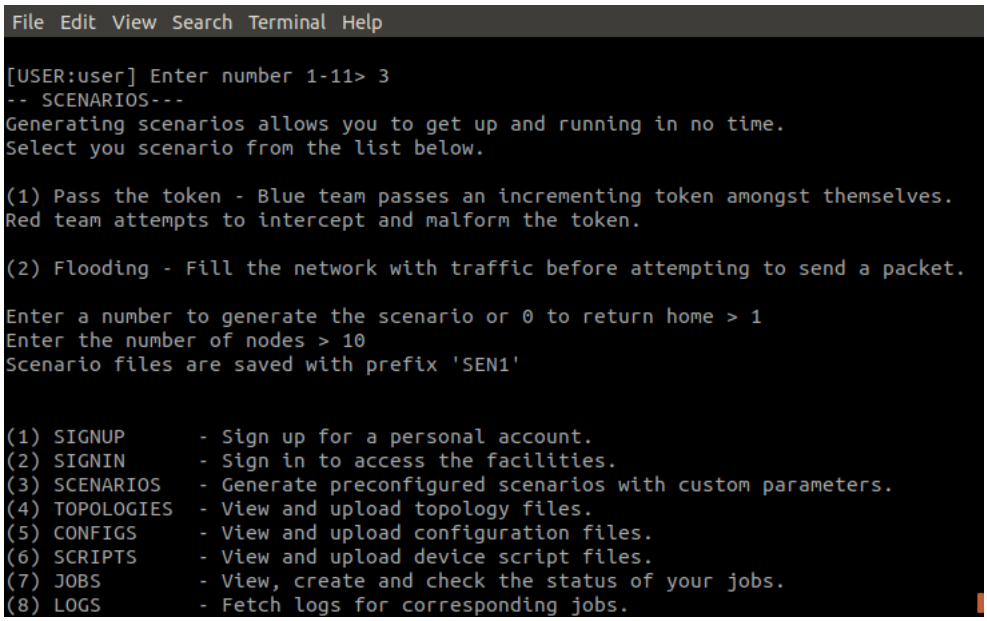


Figure 6.3: The creation of a scenario with parameters.

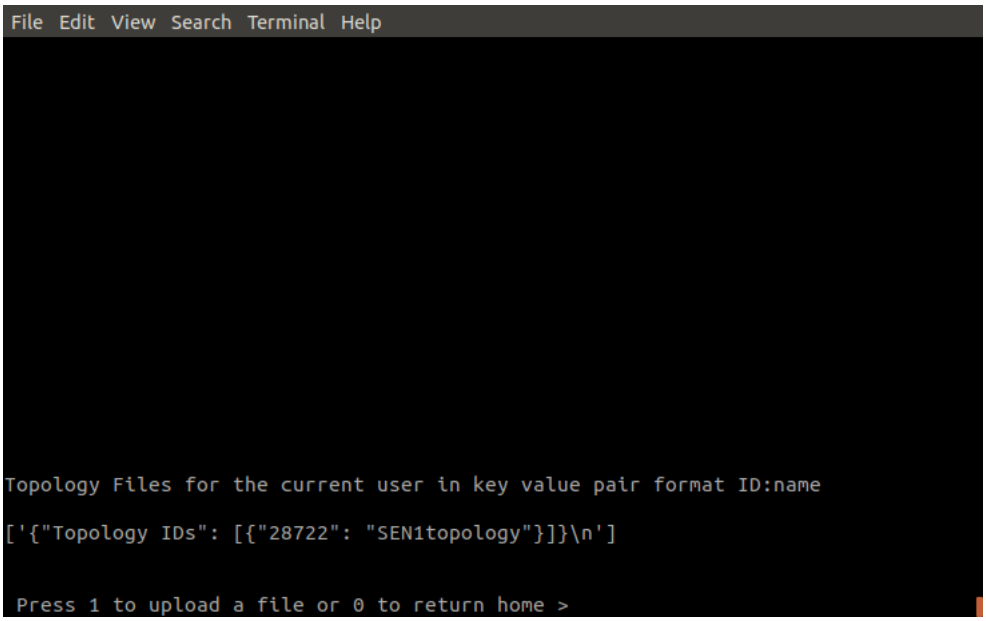


Figure 6.4: Topology page. The scenario topology file already present.

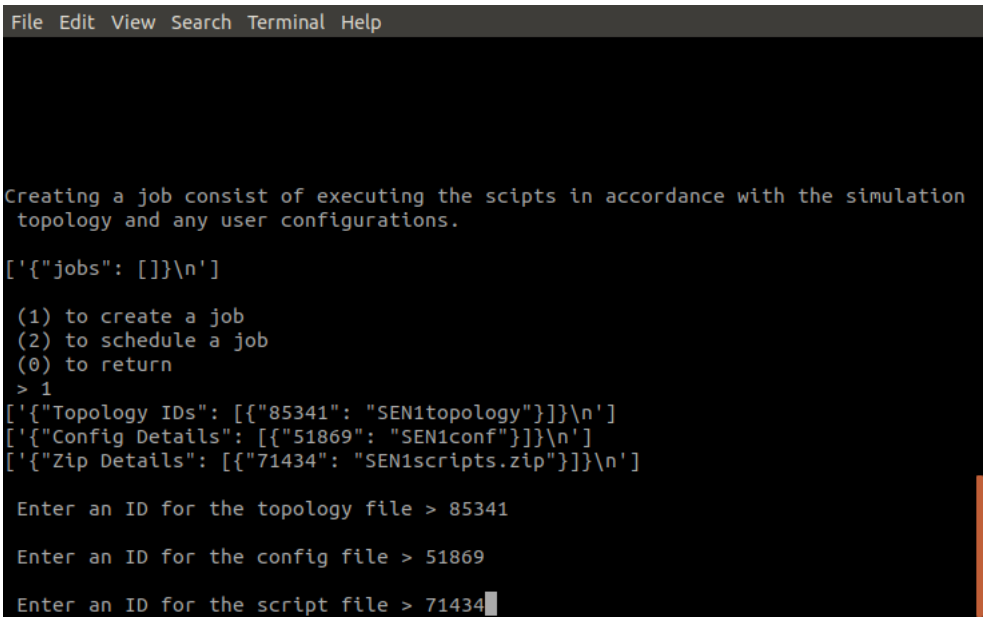


Figure 6.5: Job creation. The user creates the job from the already uploaded files.

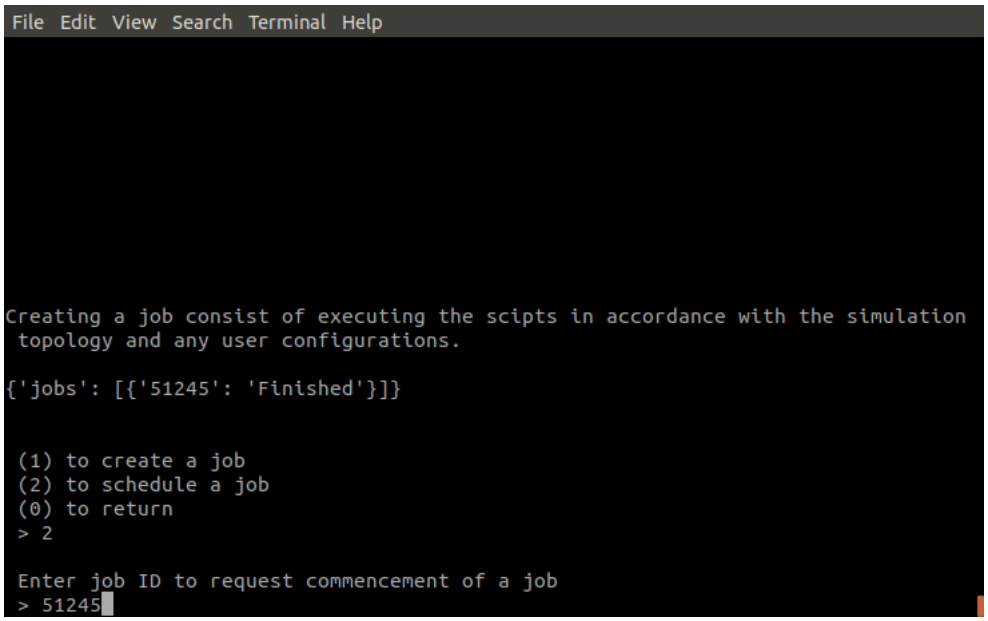


Figure 6.6: Job schedule. The user can enable the job to run, in doing so it's state changes to finished.

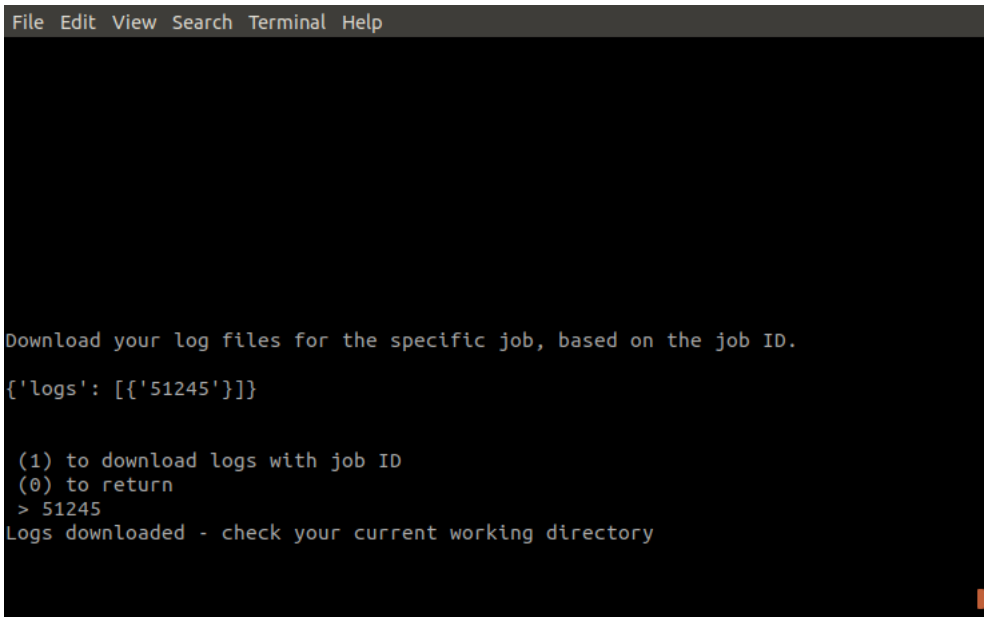


Figure 6.7: Downloading the logs. A user can download all logs for each job.

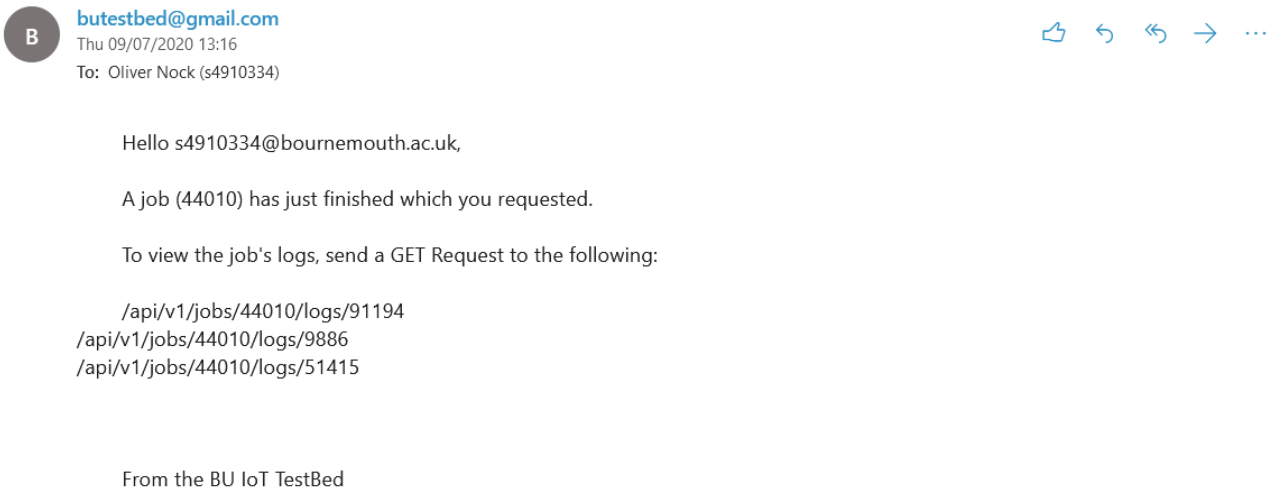


Figure 6.8: Example of communication with the user via email informing of log availability.

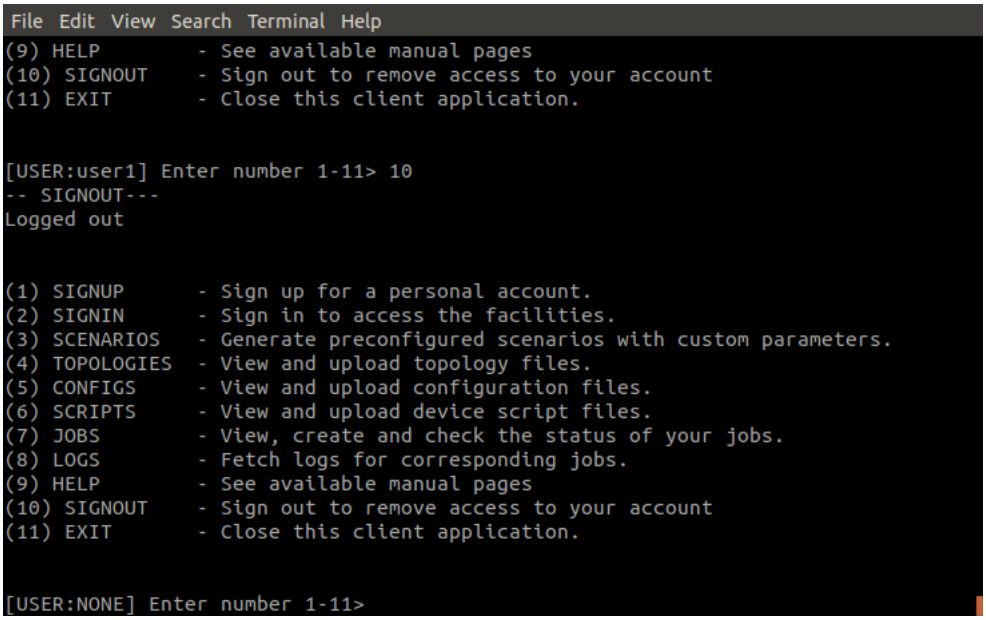


Figure 6.9: The signing out process. You can see the username changes to NONE.

The server is in constant relay with the UI, parsing the JWT token and registering the job in the database. Once in the database either the physical or virtual server can accept the job. On successful run of the job the servers can update the database and email the user. Section 6.2 highlights the differences between the virtual and physical server operation.

6.2 System In Use

6.2.1 Virtual Testbed

The virtual system was used to create the screenshots seen in section 6.1. As documented, the system observes jobs registered through the API, using a polling mechanism, which can be modified if the frequency of jobs increases/decreases. The server extracts the file paths stored on the database and parses the associating Cooja CSC file. The file is an example of Cooja operating a scenario definition language, using eXtensible Markup Language (XML). An example of this file can be found in appendix listing 2.
The code modifies the submitted files in order to format a valid configuration file in regards to the system configuration PATH variables and scope of Cooja operating headless. Other elements parsed included the simulation JavaScript code to control the simulation without subsequent access to the running simulation. This JavaScript code is therefore required to include a terminate function to end the simulation within the parameters of the experiment.
When the code runs the job the Cooja output logs, as well as the OS standard output, are written to the server and a database entry updating the job status. The API project handling the database triggers the email subsequently.
The virtual system generates two logs which are returned by the virtual server:- ”COOJA.log” shows the output of the JavaScript simulation control code allowing the user to see the state of simulation termination, ”nohup.out” shows the standard output and standard error that would be observed by the user when running such a simulation in front of their own personal computer.
The virtual system can operate functionally, however a critical error within the java framework will occasionally arise causing fatal exception to the execution of Cooja while running in headless mode using the script engine to control the simulation, see figure 6.10 highlighting the content of a ”COOJA.log” file on such an occasion.

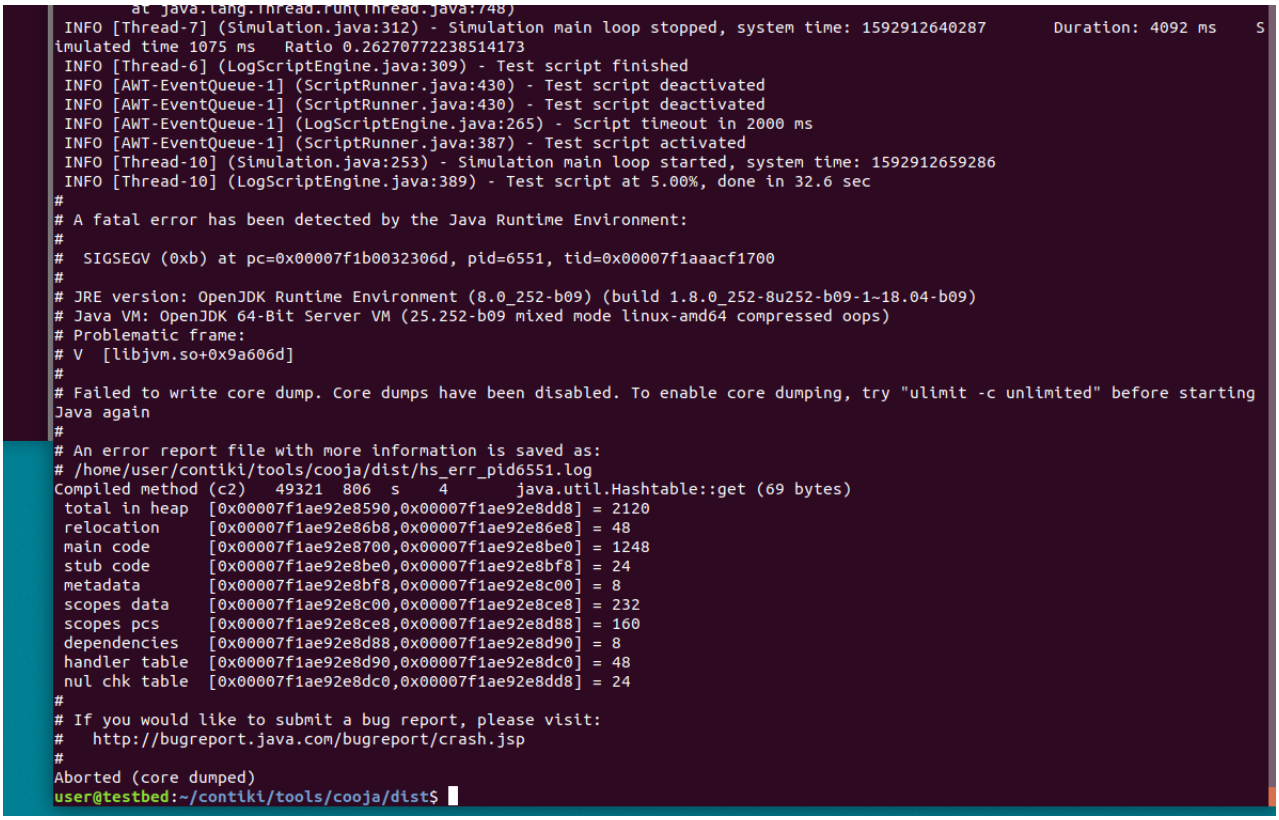


Figure 6.10: Cooja JRE error.

This error occurred frequently, preventing the use of the scenario script to generate reliable results. This is elaborated upon in subsection 6.3.1.

6.2.2 Physical Testbed

Motes are connected to USB hubs, operating over USB PCIE card passed through to a QEMU virtual machine running a Debian based Linux distribution. By addressing the motes over serial emulated USB connections, the connection has the capability of accessing the onboard flash storage and serial console output.

Figure 6.11: Features of Zolertia RE-Mote - (Zolertia 2016).

The server code can address and run scripts within the database and flash the motes to execute the programs, keeping track of running nodes and allowing for concurrent simulations. Figure 6.12 shows the physical testbed in full operation. This set-up is a temporary measure, given the restrictions on the campus, until the testbed can be configured within Bournemouth University's Innovation Lab.

Figure 6.12: Physical Testbed - Running live jobs.

The hardware testbed maintained reliability under continuous operation of development and testing, allowing preliminary results to be generated, demonstrating the potential of the system. Running the configured hardware and utilising the aforementioned scenario, the testbed was operated under the following parameters:

- Network size: 10 Zolertia Motes
- Network protocol: Nullnet - an IPv6-less MAC networking protocol
- Mobility: Zero mobility across all nodes.
- All nodes commence execution at the same time.
- Simulation is terminated after 600 seconds (10 minutes).

On receipt of the log files, a user can easily generate figures for analysis of the simulation, using parsing tools such as 'GREP'.

The results table 6.1, shows the metrics extracted from execution of the discussed scenario. Here 10 motes form 2 teams of varying size disparity. Sending packets via

Blue vs Red	Avg. Packets Sent	Avg. Packets Received	Duration (secs)	Winning Team
5 vs 5	256112	262634	600	Timeout
6 vs 4	52208	55442	420	Blue
7 vs 3	42028	45511	356	Blue
4 vs 6	52101	55621	422	Red
3 vs 7	41986	42420	321	Red

Table 6.1: Results following a 10 mote Blue vs. Red simulation using the provided scenario.

broadcast, every node records both the sent and received packets until the node reaches the defined threshold, where one team is declared the winner, or until the simulation times out.
These preliminary results serve to exhibit the utility of the testbed. In running the scenario discussed across several network sizes, it is shown that a novice user can generate quantifiable results. The flexibility of the testbed can be shown not only through the variable parameters, but also through the creative configuration by the user - to add a third team or increase the team disparity for example. The testbed use encourages users to design future scenarios to address and combat the security challenges of IoT, closing the identified skills gap (Topham et al. 2016).
To ensure validity of the testbed the simulation was run manually across the same network size. Across 3 separated experiments, and allowing for the delay required to trigger all nodes at once, the results for the manual simulation were within +/- 2% of those recorded over the operation on the testbed. The results therefore show, that control of the simulation is without interference or bias.

6.3 Summary

6.3.1 Results

This project has presented an architecture for a functioning testbed that sets a model for a custom yet configurable environment comprising of both virtual and physical components. The work then shows the applied use of an IoT testbed within the context of a network simulation, demonstrating the successful operation. The work gives potential to an IoT Cyber Range through the exposure to users within a security context.
The virtual testbed operates to provide users with the majority of the available heterogeneous multi-domain environments frequently sort for IoT simulations. However the complexity and or range of the devices must be reduced to prevent error, or improvements must be undertaken to improve the reliability of the underlying related Java packages, which is outside the scope of the project.

The physical testbed servers its function as an IoT testbed - capable of managing its resources to accept, execute and returning log files for registered jobs. The user(s) can generate results, replicating those acquired by way of physical interaction with IoT equipment.

6.3.2 Artefact

This artefact is a built artefact, falling under the categorisation of a 'Network realisation/simulation', comprising of both parts. The artefact develops a testbed based on state-of-the-art research into the domain of IoT. It presents, in the form of a virtual machine, the complete testbed solution code and virtual system. The code written and developed solely under the effort of this project is presented for marking under the format of a ZIP containing multiple repositories:- screen_src, server_src and simulation_src. These repositories consist of code in various languages, supplementing areas of development and operation for both the physical and virtual testbeds:
screen_src - Python3 code for the terminal GUI program.
server_src - Python3 code for the virtual and physical server operation. Named respectively, the files run the virtual and physical components of the server.
simulation_src - C code for the scenario simulation.

The testbed is also supplied to the department of Computing and Informatics in the form of a virtual machine image. Allowing the running of the testbed as a native testbed (physical testbed) or within a cloud context (the virtual testbed). The code and virtual machine allow for continued use of the system for subsequent IoT projects.

Chapter 7

Conclusion

As IoT continues to develop both as a paradigm and physical expression of such a paradigm, the identified skill deficiency in the IoT security industry must be addresses. This highlights the requirement for IoT testing facilities that enable logical detachment of users and resources - to enable a safety zone for novice security researchers. This work presented an IoT testbed providing:- a user-friendly UI, IoT testbed designed to host multiple users and the execution of multiple training scenarios across numerous device architectures. The work demonstrated its potential via results from a man-in-the-middle like attack scenario with red/blue teams competing across an IPv6-less physical IoT network.

7.1 Evaluation

The following table 7.1 records the completion of all the project objectives, declared in table 1.2.1.

The RAD methodology has worked well in this project due to the time frame that was given. RAD helped to develop simulations that were thorough but not excessive, keeping me within the time frames set out at the start of the project. Time management was monitored using spreadsheet software, the original plan (appendix 1) set out weekly constraints for objectives which, by using an live updated copy, allowed tracking of overall project progress (appendix 2). The numerous deadline adjustments were easily adapted into the schedule of either sprint allowing for research alongside development. Although knowledge of extended schedules in advanced could have aided in the planning of increased development complexity. The updated plan shows the delivery of the project within the time scales.

Research, (Berman et al. 2014) states that the concepts of “sliceability” (resources provisioning and isolation for separate experiments), and deep systems level programmability expands the potential for experimental networks. Both these are provided in the scope of

Objective	Aim	Chapter	Criteria Satisfaction
1	1	Chapter 2	Research of present virtual cyber range facilities.
1	1	Chapter 3	Research of possible implementations.
2	1	Chapter 2	Research of present virtual testbed facilities.
2	2	Chapter 3	Chosen solution for the virtual testbed.
2	3	Chapter 5	The development of the virtual testbed.
2	4	Chapter 6	The working testbed implemented with the API.
3	1	Chapter 2	Research of physical cyber range facilities.
3	2	Chapter 3	The chosen solution for the physical testbed.
3	3	Chapter 5	The development of the physical testbed.
3	4	Chapter 5	The development of the physical testbed.
3	5	Chapter 6	The working physical testbed implemented with the API.
4	1	Chapter 2	Research of IoT testbed scenarios.
4	2	Chapter 5	The development of the testbed scenario.
4	3	Chapter 6	Results generated from the scenario on the testbed.

Table 7.1: Table of objectives evaluated.

the system - sliceability in resource management provisioning and cooja simulations and programmability through the contiki-ng operating systems and operation with compatible devices.
The testbed however requires further development to update the testbed with the changing requirements of federations and the IoT device market. Further improvements to the underlying technologies will also improve the reliability of the testbed software.

Sociopolitical factors have effected the delivery and initial requirement of this project. In terms of development, acquisition of hardware was delayed and access to testbed setup environment postponed. The use of this IoT testbed for evaluation by subsequent IoT students within a research environment is currently on hold.
The successful publication of the MDPI journal paper (Starkey et al. 2020) demonstrates the success of the project. The acceptance of the publication shows the relevance of the work undertaken to the current trends of IoT. As well, it acknowledges the effort to address the trends within the provided time frame and quality of the solution thereof.

7.2 Future Work

Internally, the project can be expanded within any subdomain of IoT. With the ability for future students or researches to develop scenarios that can be executed upon the testbed. The system could be included as a university facility for remote work, given the current

social climate and requirements for remote interactive sessions with students.
Externally, on going work could include the extension of the testbed in line with the H2020 ECHO project (ECHO 2020). The testbed would operate in cyber-security training scenarios addressing issues of modern IoT networks.
The virtual testbed has been developed to enable a cloud hosted solution, bringing the benefits of cloud services to form Experiment-as-a-service (EaaS). Further considerations in the development of the physical testbed have ensured a solution capable of long range wireless topologies. As such test physical testbed can be spread up to 20 kilometres, perhaps campus wide to enable IoT usage within the monitoring and data analytic domains. This would require a wireless resource management system, which could be developed using MQTT broker software, such as RabbitMQ (RabbitMQ 2020). Further technologies in API systems, such as GraphQL (FACEBOOK 2020) could improve the API to allow querying of log data, to the granularity of individual variables, to allow better user automation of simulations. Furthermore exposing logs as such could enable the development of a dashboard for graphing and reports.

WORD COUNT: 10,357

Bibliography

Ahrens, K., Eveslage, I., Fischer, J., Kühnlenz, F. and Weber, D., 2009. The challenges of using sdl for the development of wireless sensor networks. R. Reed, A. Bilgic and R. Gotzhein, eds., *SDL 2009: Design for Motes and Mobiles*, Berlin, Heidelberg: Springer Berlin Heidelberg, 200–221.

ANSII, 2020. Sparta – a cybersecurity competence network to coordinate research, innovation and training within the european union. Available from: `https://www.ssi.gouv.fr/uploads/2019/02/press-release-sparta.pdf` [Accessed 21 June 2020].

Berman, M., Chase, J. S., Landweber, L., Nakao, A., Ott, M., Raychaudhuri, D., Ricci, R. and Seskar, I., 2014. Geni: A federated testbed for innovative network experiments. *Computer Networks*, 61, 5–23.

Čeleda, P., Vykopal, J., Švábenskỳ, V. and Slavíček, K., 2020. Kypo4industry: A testbed for teaching cybersecurity of industrial control systems. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 1026–1032.

CENSIS, 2020. *IoT Testbed*. Available from: `https://censis.org.uk/access-our-services/services-and-facilities/iot-testbed/` [Accessed 26 July 2020].

Claeys, T., 2020. *OpenWSN firmware*. Available from: `https://github.com/openwsn-berkeley/openwsn-fw/` [Accessed 16 July 2020].

Contiki, 2020. *Contiki-NG*. Available from: `https://github.com/contiki-ng/contiki-ng` [Accessed 11 June 2020].

Contiki-NG, 2020. *Documentation: Processes and events*. Available from: `https://github.com/contiki-ng/contiki-ng/wiki/Documentation:-Processes-and-events` [Accessed 12 June 2020].

Decker, E., 2013. *TinyOS HOME*. Available from: `https://github.com/tinyos/tinyos-main/wiki` [Accessed 17 June 2020].

ECHO, 2020. *ECHO Project Summary*. Available from: `https://echonetwork.eu/project-summary` [Accessed 13 June 2020].

Egea-Lùpez, E., Vales-Alonso, J., Martínez-Sala, A. S., Pavón-Mariño, P. and García-Haro, J., 2005. *Simulation Tools for Wireless Sensor Networks*. Available from: https://pdfs.semanticscholar.org/35e1/df79f7648f2d596e30aff6d6874c3e77cdd8.pdf [Accessed 18 June 2020].

European Union, 2020a. Concordia. Available from: https://www.concordia-h2020.eu/ [Accessed 21 June 2020].

European Union, 2020b. Cyber security for europe. Available from: https://cybersec4europe.eu/about/ [Accessed 22 June 2020].

European Union, 2020c. Federated interoperable semantic iot testbeds and applications. Available from: http://fiesta-iot.eu/index.php/iot-experiments-as-a-service/ [Accessed 21 June 2020].

European Union, 2020d. Horizon 2020. Available from: https://ec.europa.eu/programmes/horizon2020/en [Accessed 20 June 2020].

FACEBOOK, 2020. *GraphQL*. Available from: https://github.com/graphql/graphql-spec [Accessed 23 August 2020].

Ficco, M. and Palmieri, F., 2019. Leaf: an open-source cybersecurity training platform for realistic edge-iot scenarios. *Journal of Systems Architecture*, 97, 107–129.

FIT, 2020. *The Very Large Scale IoT Testbed*. Available from: https://www.iot-lab.info/ [Accessed 26 July 2020].

Gavras, A., Karila, A., Fdida, S., May, M. and Potts, M., 2007. Future internet research and experimentation: The fire initiative. 37 (3), 89–92. URL https://doi.org/10.1145/1273445.1273460.

GENI-NSF, 2020. Geni: Global environment for network innovations. Available from: https://github.com/GENI-NSF [Accessed 20 June 2020].

Gluhak, A., Krco, S., Nati, M., Pfisterer, D., Mitton, N. and Razafindralambo, T., 2011. A survey on facilities for experimental internet of things research. *IEEE Communications Magazine*, 49 (11), 58–67.

Gluhak, A., Krco, S., Nati, M., Pfisterer, D., Mitton, N. and Razafindralambo, T., 2011. A survey on facilities for experimental internet of things research. *IEEE Communications Magazine*, 49 (11), 58–67.

Hendrawan, I. N. R. and Arsa, I. G. N. W., 2017. Zolertia z1 energy usage simulation with cooja simulator. *2017 1st International Conference on Informatics and Computational Sciences (ICICoS)*, 147–152.

Jafer, S., Chhaya, B., Durak, U. and Gerlach, T., 2016. *Formal Scenario Definition Language for Aviation: Aircraft Landing Case Study*. URL https://arc.aiaa.org/doi/abs/10.2514/6.2016-3521.

Kalatzis, N., Routis, G., Roussaki, I. and Papavassiliou, S., 2018. Enabling data interoperability for federated iot experimentation infrastructures. *2018 Global Internet of Things Summit (GIoTS)*, IEEE, 1–6.

Kavallieratos, G., Katsikas, S. K. and Gkioulos, V., 2019. Towards a cyber-physical range. *Proceedings of the 5th on Cyber-Physical System Security Workshop*, 25–34.

Kolias, C., Kambourakis, G., Stavrou, A. and Voas, J., 2017. Ddos in the iot: Mirai and other botnets. *Computer*, 50 (7), 80–84.

Lignan, A., 2016. *Zolertia RE-Mote platform*. Available from: https://github.com/Zolertia/Resources/wiki/RE-Mote [Accessed 20 June 2020].

Lucas, J., 2019. *The Design and Development of an IoT Testbed for Experimental Security Research*. Available from: https://brightspace.bournemouth.ac.uk/d2l/le/content/22666/viewContent/368949/View [Accessed 14 June 2020].

Mahmud, A., Hossain, F., Juhin, F. and Choity, T. A., 2019. Merging the communication protocols 6lowpan-coap and rpl-coap : Simulation and performance analysis using cooja simulator. *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, 1–6.

MDPI, 2020. *Sensors — Open Access Journal*. Available from: https://www.mdpi.com/journal/sensors [Accessed 15 June 2020].

Munoz, J., Rincon, F., Chang, T., Vilajosana, X., Vermeulen, B., Walcarius, T., Van de Meerssche, W. and Watteyne, T., 2019. Opentestbed: Poor man's iot testbed. *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 467–471.

Naik, K. P. and Joshi, U. R., 2017. Performance analysis of constrained application protocol using cooja simulator in contiki os. *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, 547–550.

Nicolas Falliere, L. O. M. and Chien, E., 2010. W32.stuxnet dossier. Security Response. Available from: https://www.wired.com/images_blogs/threatlevel/2010/10/w32_stuxnet_dossier.pdf [Accessed 17/04/2020].

NIST, 2020. The cyber range: A guide. Available from: https://www.nist.gov/document/cyber-range-guide [Accessed 22 June 2020].

Nohlgård, J., 2017. *RIOT Introduction*. Available from: https://github.com/RIOT-OS/RIOT/wiki/Introduction [Accessed 18 June 2020].

Oliveira, A. and Vazão, T., 2018. Connecting ns-3 with cooja. *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 1–6.

OMGSYSML, 2020. *What is SysML?* Available from: https://www.omgsysml.org/what-is-sysml.htm [Accessed 3 July 2020].

RabbitMQ, 2020. *RabbitMQ*. Available from: https://github.com/rabbitmq [Accessed 19 August 2020].

Raspberry Pi Foundation, 2020a. *Getting started with the Internet of Things*. Available from: https://www.raspberrypi.org/blog/getting-started-with-iot/ [Accessed 12 July 2020].

Raspberry Pi Foundation, 2020b. *Raspberry Pi 4 Tech Specs*. Available from: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/ [Accessed 8 July 2020].

Raytheon BBN Technologies, 2020. What is geni. Available from: https://www.geni.net/about-geni/what-is-geni/ [Accessed 18 June 2020].

Rebecca Vogel, 2016. *Closing the cybersecurity skills gap*. Available from: https://www.academia.edu/25380112/CLOSING_THE_CYBERSECURITY_SKILLS_GAP [Accessed 14 June 2020].

Russo, E., Costa, G. and Armando, A., 2018. Scenario design and validation for next generation cyber ranges. *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, 1–4.

Sabri, C., Kriaa, L. and Azzouz, S. L., 2017. Comparison of iot constrained devices operating systems: A survey. *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, 369–375.

Schmitt, C., Anliker, C. and Stiller, B., 2017. Efficient and secure pull requests for emergency cases using a mobile access framework. Q. Z. Sheng, Y. Qin, L. Yao and B. Benatallah, eds., *Managing the Web of Things*, Boston: Morgan Kaufmann, 229 – 247. URL http://www.sciencedirect.com/science/article/pii/B978012809764900010X.

Schwab, S. and Kline, E., 2019. Cybersecurity experimentation at program scale: Guidelines and principles for future testbeds. *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, IEEE, 94–102.

Smith, B., 2010. Operating system resource management. *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS)*, 1–1.

Starkey, J., Nock, O. and Angelopoulos, C. M., 2020. *Addressing the Security Gap in IoT: Towards an IoT Cyber Range*. Available from: https://www.mdpi.com/1424-8220/20/18/5439 [Accessed 22 September 2020].

Topham, L., Kifayat, K., A Younis, Y., Shi, Q. and Askwith, B., 2016. Cyber security teaching and learning laboratories: A survey. *Information & Security: An International Journal*, 35.

Wahle, S., Magedanz, T., Gavras, A., Hrasnica, H. and Denazis, S., 2009. Technical infrastructure for a pan-european federation of testbeds. *2009 5th International Conference on Testbeds and Research Infrastructures for the Development of Networks Communities and Workshops*, 1–8.

Yamin, M. M., Katt, B. and Gkioulos, V., 2020. Cyber ranges and security testbeds: Scenarios, functions, tools and architecture. *Computers & Security*, 88, 101636.

Zolertia, 2016. *Zolertia RE-Mote Revision B*. Available from: https://github.com/Zolertia/Resources/blob/master/RE-Mote/Hardware/Revision%20B/Datasheets/ZOL-RM0x-B%20-%20RE-Mote%20revision%20B%20Datasheet\%20v.1.0.0.pdf [Accessed 24 June 2020].

Zolertia, 2020a. *Internet of things hardware solutions, start your 6lowpan project*. Available from: https://zolertia.io/ [Accessed 20 June 2020].

Zolertia, 2020b. *RE-MOTE*. Available from: https://zolertia.io/product/re-mote/ [Accessed 20 June 2020].

Østby, G., Lovell, K. N. and Katt, B., 2019. Excon teams in cyber security training. *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, 14–19.

Appendices

Risks	Solutions
Simulations are outside of the capability of the software.	Provide the means to operate the simulations through the federated system.
Changes to design and or software adopted.	Allow time throughout the project to research the most appropriate/compatible software. Adjust IoT-CR features according to practical abilities.
Changes/access to hardware required.	Allow time throughout the project to research the most appropriate/compatible hardware. If cost, location, size or functionality of hardware prevents adoption, ensure fallback hardware or software is present.
All aims can not be achieved in the required time frame.	Set a priority list to establish precedent to allow revoking of less important aims within a given objective.
Unforeseen Circumstances (eg natural disasters, power outages, corruption of data etc.).	Loss or corruption of data, either via accidental damage or a naturally occur event, can be mitigated by keeping a backup of all digital data (from data collect by simulations to the code that runs the simulations themselves). This includes the report documentation.
Affected by illness.	Set small margin of error in time planning to account for this possibility.
Artefact is infeasible	Complete the artefact with only the feasible features.

Table 2: Risk Analysis

Appendix code listing 1 contains an example user configuration file.

Listing 1: User Configuration Code

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
<user>user</user>
<description> This is a job description.</description>
<durationSeconds>300</durationSeconds>
<numberOfNodes>4</numberOfNodes>
<nodes>
<node><id>0</id><script>mitm.c</script></node>
<node><id>1</id><script>mitm.c</script></node>
<node><id>3</id><script>mitm.red.c</script></node>
<node><id>4</id><script>mitm.red.c</script></node>
</nodes>
</config>
```

Appendix code listing 2 contains an example Cooja CSC configuration file.

Listing 2: User Configuration Code

```
<?xml version="1.0" encoding="UTF-8"?>
<simconf>
  <project EXPORT="discard">[APPS_DIR]/mrm</project>
  <project EXPORT="discard">[APPS_DIR]/mspsim</project>
  <project EXPORT="discard">[APPS_DIR]/avrora</project>
  <project EXPORT="discard">[APPS_DIR]/serial_socket</project>
  <project EXPORT="discard">[APPS_DIR]/collect-view</project>
  <project EXPORT="discard">[APPS_DIR]/powertracker</project>
  <simulation>
    <title>My simulation</title>
    <randomseed>123456</randomseed>
    <motedelay_us>1000000</motedelay_us>
    <radiomedium>
      org.contikios.cooja.radiomediums.UDGM
      <transmitting_range>50.0</transmitting_range>
      <interference_range>100.0</interference_range>
      <success_ratio_tx>1.0</success_ratio_tx>
      <success_ratio_rx>1.0</success_ratio_rx>
    </radiomedium>
    <events>
      <logoutput>40000</logoutput>
    </events>
    <motetype>
      org.contikios.cooja.contikimote.ContikiMoteType
      <identifier>mtype965</identifier>
      <description>Cooja Mote Type #1</description>
      <source>[CONTIKI_DIR]/examples/ipv6/rpl-border-router/
        ↪ border-router.c</source>
      <commands>make border-router.cooja TARGET=cooja</commands>
      <moteinterface>org.contikios.cooja.interfaces.Position</
        ↪ moteinterface>
      <moteinterface>org.contikios.cooja.interfaces.Battery</
        ↪ moteinterface>
      <moteinterface>org.contikios.cooja.contikimote.interfaces.
        ↪ ContikiVib</moteinterface>
      <moteinterface>org.contikios.cooja.contikimote.interfaces.
```

```
      ↪ ContikiMoteID</moteinterface>
    <moteinterface>org.contikios.cooja.contikimote.interfaces.
      ↪ ContikiRS232</moteinterface>
    <moteinterface>org.contikios.cooja.contikimote.interfaces.
      ↪ ContikiBeeper</moteinterface>
    <moteinterface>org.contikios.cooja.interfaces.RimeAddress<
      ↪ /moteinterface>
    <moteinterface>org.contikios.cooja.contikimote.interfaces.
      ↪ ContikiIPAddress</moteinterface>
    <moteinterface>org.contikios.cooja.contikimote.interfaces.
      ↪ ContikiRadio</moteinterface>
    <moteinterface>org.contikios.cooja.contikimote.interfaces.
      ↪ ContikiButton</moteinterface>
    <moteinterface>org.contikios.cooja.contikimote.interfaces.
      ↪ ContikiPIR</moteinterface>
    <moteinterface>org.contikios.cooja.contikimote.interfaces.
      ↪ ContikiClock</moteinterface>
    <moteinterface>org.contikios.cooja.contikimote.interfaces.
      ↪ ContikiLED</moteinterface>
    <moteinterface>org.contikios.cooja.contikimote.interfaces.
      ↪ ContikiCFS</moteinterface>
    <moteinterface>org.contikios.cooja.contikimote.interfaces.
      ↪ ContikiEEPROM</moteinterface>
    <moteinterface>org.contikios.cooja.interfaces.
      ↪ Mote2MoteRelations</moteinterface>
    <moteinterface>org.contikios.cooja.interfaces.
      ↪ MoteAttributes</moteinterface>
    <symbols>false</symbols>
  </motetype>
  <mote>
    <interface_config>
      org.contikios.cooja.interfaces.Position
      <x>36.55243979065026</x>
      <y>20.210298896800527</y>
      <z>0.0</z>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiMoteID
      <id>1</id>
```

```
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiRadio
      <bitrate>250.0</bitrate>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiEEPROM
      <eeprom></eeprom>
    </interface_config>
    <motetype_identifier>mtype965</motetype_identifier>
  </mote>
  <mote>
    <interface_config>
      org.contikios.cooja.interfaces.Position
      <x>33.216310693297515</x>
      <y>16.250870041135677</y>
      <z>0.0</z>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiMoteID
      <id>2</id>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiRadio
      <bitrate>250.0</bitrate>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiEEPROM
      <eeprom></eeprom>
    </interface_config>
    <motetype_identifier>mtype965</motetype_identifier>
  </mote>
  <mote>
    <interface_config>
      org.contikios.cooja.interfaces.Position
      <x>59.1769353843821</x>
      <y>64.57403229896832</y>
      <z>0.0</z>
    </interface_config>
```

```
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiMoteID
      <id>3</id>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiRadio
      <bitrate>250.0</bitrate>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiEEPROM
      <eeprom></eeprom>
    </interface_config>
    <motetype_identifier>mtype965</motetype_identifier>
  </mote>
  <mote>
    <interface_config>
      org.contikios.cooja.interfaces.Position
      <x>67.7647075623329</x>
      <y>42.13346404589221</y>
      <z>0.0</z>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiMoteID
      <id>4</id>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiRadio
      <bitrate>250.0</bitrate>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiEEPROM
      <eeprom></eeprom>
    </interface_config>
    <motetype_identifier>mtype965</motetype_identifier>
  </mote>
  <mote>
    <interface_config>
      org.contikios.cooja.interfaces.Position
      <x>94.42661166869458</x>
```

```
      <y>1.918390166445394</y>
      <z>0.0</z>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiMoteID
      <id>5</id>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiRadio
      <bitrate>250.0</bitrate>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiEEPROM
      <eeprom></eeprom>
    </interface_config>
    <motetype_identifier>mtype965</motetype_identifier>
  </mote>
  <mote>
    <interface_config>
      org.contikios.cooja.interfaces.Position
      <x>2.657243973986345</x>
      <y>20.176785796346742</y>
      <z>0.0</z>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiMoteID
      <id>6</id>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiRadio
      <bitrate>250.0</bitrate>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiEEPROM
      <eeprom></eeprom>
    </interface_config>
    <motetype_identifier>mtype965</motetype_identifier>
  </mote>
  <mote>
```

```
    <interface_config>
      org.contikios.cooja.interfaces.Position
      <x>7.2119270814658964</x>
      <y>84.22049441976216</y>
      <z>0.0</z>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiMoteID
      <id>7</id>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiRadio
      <bitrate>250.0</bitrate>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiEEPROM
      <eeprom></eeprom>
    </interface_config>
    <motetype_identifier>mtype965</motetype_identifier>
  </mote>
  <mote>
    <interface_config>
      org.contikios.cooja.interfaces.Position
      <x>48.00327272744834</x>
      <y>65.41560244760497</y>
      <z>0.0</z>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiMoteID
      <id>8</id>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiRadio
      <bitrate>250.0</bitrate>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiEEPROM
      <eeprom></eeprom>
    </interface_config>
```

```
    <motetype_identifier>mtype965</motetype_identifier>
  </mote>
  <mote>
    <interface_config>
      org.contikios.cooja.interfaces.Position
      <x>46.609766957461886</x>
      <y>56.475072251290705</y>
      <z>0.0</z>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiMoteID
      <id>9</id>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiRadio
      <bitrate>250.0</bitrate>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiEEPROM
      <eeprom></eeprom>
    </interface_config>
    <motetype_identifier>mtype965</motetype_identifier>
  </mote>
  <mote>
    <interface_config>
      org.contikios.cooja.interfaces.Position
      <x>69.70018728828595</x>
      <y>16.571466568926173</y>
      <z>0.0</z>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiMoteID
      <id>10</id>
    </interface_config>
    <interface_config>
      org.contikios.cooja.contikimote.interfaces.ContikiRadio
      <bitrate>250.0</bitrate>
    </interface_config>
    <interface_config>
```

```
        org.contikios.cooja.contikimote.interfaces.ContikiEEPROM
        <eeprom></eeprom>
      </interface_config>
      <motetype_identifier>mtype965</motetype_identifier>
    </mote>
</simulation>
<plugin>
    org.contikios.cooja.plugins.SimControl
    <width>280</width>
    <z>5</z>
    <height>160</height>
    <location_x>400</location_x>
    <location_y>0</location_y>
</plugin>
<plugin>
    org.contikios.cooja.plugins.Visualizer
    <plugin_config>
      <moterelations>true</moterelations>
      <skin>org.contikios.cooja.plugins.skins.IDVisualizerSkin</
          ↪ skin>
      <skin>org.contikios.cooja.plugins.skins.GridVisualizerSkin
          ↪ </skin>
      <skin>org.contikios.cooja.plugins.skins.
          ↪ TrafficVisualizerSkin</skin>
      <skin>org.contikios.cooja.plugins.skins.UDGMVisualizerSkin
          ↪ </skin>
      <skin>org.contikios.cooja.plugins.skins.LogVisualizerSkin<
          ↪ /skin>
      <viewport>3.8218397621683935 0.0 0.0 3.8218397621683935
          ↪ 8.480530120092867 8.395492909798913</viewport>
    </plugin_config>
    <width>400</width>
    <z>1</z>
    <height>400</height>
    <location_x>1</location_x>
    <location_y>1</location_y>
</plugin>
<plugin>
    org.contikios.cooja.plugins.LogListener
```

```
    <plugin_config>
      <filter />
      <formatted_time />
      <coloring />
    </plugin_config>
    <width>737</width>
    <z>4</z>
    <height>240</height>
    <location_x>400</location_x>
    <location_y>160</location_y>
  </plugin>
  <plugin>
    org.contikios.cooja.plugins.TimeLine
    <plugin_config>
      <mote>0</mote>
      <mote>1</mote>
      <mote>2</mote>
      <mote>3</mote>
      <mote>4</mote>
      <mote>5</mote>
      <mote>6</mote>
      <mote>7</mote>
      <mote>8</mote>
      <mote>9</mote>
      <showRadioRXTX />
      <showRadioHW />
      <showLEDs />
      <zoomfactor>500.0</zoomfactor>
    </plugin_config>
    <width>1137</width>
    <z>3</z>
    <height>166</height>
    <location_x>0</location_x>
    <location_y>702</location_y>
  </plugin>
  <plugin>
    org.contikios.cooja.plugins.Notes
    <plugin_config>
      <notes>Enter notes here</notes>
```

```
      <decorations>true</decorations>
    </plugin_config>
    <width>457</width>
    <z>2</z>
    <height>160</height>
    <location_x>680</location_x>
    <location_y>0</location_y>
  </plugin>
  <plugin>
    org.contikios.cooja.plugins.ScriptRunner
    <plugin_config>
      <script>TIMEOUT(300000, log.log("Performance Calculation"
        ↪ + "\n"));

packetsReceived= new Array();
packetsSent = new Array();
timeReceived = new Array();
timeSent = new Array();
count = new Array();

nodeCount = 27;
data_length = 23;
senderID = 0;
receiverID = 0;
PDR=0;
e_count = 0;

for(i = 3; i &lt;= nodeCount; i++)
{
packetsReceived[i] = 0;
packetsSent[i] = 0;
timeSent[i] = 0;
timeReceived[i] = 0;
}

while(1) {
YIELD();
msgArray = msg.split(' ');
if(msgArray[0].equals("Sending"))
```

```
{
if(msgArray.length == 5)
{
// sent packet
senderID = parseInt(msgArray[4]);
packetsSent[senderID]++;
timeSent[senderID] = time;
}
}
if(msgArray[0].equals("Got"))
{
if(msgArray.length == 6)
{
receiverID = parseInt(msgArray[5]);
packetsReceived[receiverID]++;
timeReceived[receiverID] = time;

log.log("receiverID " + receiverID + " PacketReceived= " +
    ↪ packetsReceived[receiverID] + "\n");

if (timeReceived[receiverID] &gt; 0)
{
count[receiverID]++
}

totalReceived = totalSent = 0;
totalclient=0;

for(i = 3; i &lt;= nodeCount; i++)
{
totalclient++;
totalReceived += packetsReceived[i];
totalSent += packetsSent[i];
log.log("MoteID= " + i + " ReceivedPackets= " + packetsReceived[
    ↪ i] + " SendingPackets= " + packetsSent[i] + "\n");
}

log.log(" Generated Packets " + totalSent + "\n");
log.log(" ReceivedPackets " + totalReceived + "\n");
```

```
PDR=(totalReceived / totalSent) * 100
log.log("Packet Delivery Ratio" + PDR + "\n");
}
}
}</script>
        <active>false</active>
      </plugin_config>
      <width>600</width>
      <z>0</z>
      <height>700</height>
      <location_x>331</location_x>
      <location_y>65</location_y>
    </plugin>
</simconf>
```

Below is the soon to be published paper that describes the testbed incentive and operation.

sensors

MDPI

Article

Addressing the Security Gap in IoT: Towards an IoT Cyber Range

Oliver Nock, Jonathan Starkey and Constantinos Marios Angelopoulos *

Faculty of Science and Technology, Department of Computing and Informatic, Bournemouth University, Poole, Dorset BH12 5BB, UK; s4910334@bournemouth.ac.uk (O.N.); s4921588@bournemouth.ac.uk (J.S.)

* Correspondence: mangelopoulos@bournemouth.ac.uk

Received: 21 July 2020; Accepted: 17 September 2020; Published: 22 September 2020

Abstract: The paradigm of Internet of Things has now reached a maturity level where the pertinent research goal is the successful application of IoT technologies in systems of high technological readiness level. However, while basic aspects of IoT connectivity and networking have been well studied and adequately addressed, this has not been the case for cyber security aspects of IoT. This is nicely demonstrated by the number of IoT testbeds focusing on networking aspects and the lack of IoT testbeds focusing on security aspects. Towards addressing the existing and growing skills-shortage in IoT cyber security, we present an IoT Cyber Range (IoT-CR); an IoT testbed designed for research and training in IoT security. The IoT-CR allows the user to specify and work on customisable IoT networks, both virtual and physical, and supports the concurrent execution of multiple scenarios in a scalable way following a modular architecture. We first provide an overview of existing, state of the art IoT testbeds and cyber security related initiatives. We then present the design and architecture of the IoT Cyber Range, also detailing the corresponding RESTful APIs that help de-associate the IoT-CR tiers and obfuscate underlying complexities. The design is focused around the end-user and is based on the four design principles for Cyber Range development discussed in the introduction. Finally, we demonstrate the use of the facility via a red/blue team scenario involving a variant of man-in-the-middle attack using IoT devices. Future work includes the use of the IoT-CR by cohorts of trainees in order to evaluate the effectiveness of specific scenarios in acquiring IoT-related cyber-security knowledge and skills, as well as the IoT-CR integration with a pan-European cyber-security competence network.

Keywords: cyber-range; IoT; testbed; cyber-security

1. Introduction

Following more than two decades of active research, the technological paradigm of Internet of Things (IoT) has now reached a high maturity level at which the pertaining question is the application of IoT technologies in systems of high technological readiness that are either close to the market or already commercialised. In this context, while the basic networking and interoperability aspects of IoT have been adequately addressed or solved, this is not the case for the cyber-security aspects.

This is nicely demonstrated by recent cyber-security incidents in IoT systems that attracted much attention. The Mirai botnet is one such example. The Mirai malware's source code was based off the Bashlite malware and infected IoT devices, such as internet-connected security cameras, to create a botnet [1]. This botnet is interfaced by Command and Control (C&C) servers by its operators. Scanners (dedicated servers) look for vulnerable devices, and loaders (also dedicated servers) load the malware onto the vulnerable device as a payload. Malware servers host resources, such as binaries and other executables (e.g., scripts, etc.) that will be utilized by the botnet during an attack. The Mirai botnet is an example of an exploitation of the LAN Mistrust problem domain. The botnet was used in October

2016 to conduct a Distributed Denial of Service (DDoS) attack against people's routers across the world and caused many of the most popular websites at the time to be rendered unavailable [2]. This attack served as a proof of concept that IoT devices could be utilized in wide-scale networking attacks and derivatives of the Mirai malware have been found since. Additionally, in 2016, researchers were able to remotely access a Tesla Model S from a distance of 12 miles away [3]. They were able to interfere with functionalities ranging from car door locks, to the car's breaks and dashboard computer system due to these and other functionalities being electronically controlled, and stemmed from (according to Tesla) the car's web browser being used whilst connected to a malicious Wi-Fi hotspot. This is another example of exploitation of the LAN Mistrust problem domain.

The aforementioned examples highlight the significant gap that currently exists in terms of cyber-security competencies in IoT [4]. If the discrepancy between this skills-shortage and the required security expertise becomes too large, this may result in a decrease of security as-a-whole in IoT environments, with a subsequent increase in cybercrime. The exponential growth of IoT has led to the commissioning of millions of new devices, which are collecting and transmitting information but with many vendors not conforming to security best practises. This means there are millions of potentially vulnerable devices which could be exploited, thus revealing new vectors of attack. There is a dire need for dedicated infrastructure to be used for evaluating and training cyber-security competencies in IoT.

A respected method of training an individual and increasing their competency is via practising scenarios on a cyber range. This allows for trainees to experiment and hone their abilities to deal with situations that make occur in a safe, and isolated manner. Cyber ranges also provide suitable infrastructure as a test-bed allowing experimentation. It is noted that there are many cyber ranges which are useful for the study of cyber security, and that there are many test-beds which allow for the testing of IoT devices and networks. Paradoxically however, there is very little literature regarding IoT cyber ranges and security.

Our contribution. We address the existing and ever increasing problem of skills-shortage and lack of research infrastructure focused on cyber-security in IoT. We present an IoT Cyber Range; i.e., an IoT test-bed that is designed to support research and training on cyber security aspects of IoT systems and networks. The design is focused around the end-user and is based on the four design principles for Cyber Range development defined by Schwab and Kline [5] . The architecture is modular, consisting of a front-end and a back-end that are loosely coupled via a RESTful API. This obfuscates the underlying complexity of the back-end from the end-user, while at the same time isolating the front-end from future extensions in the supported IoT technologies and system architectures at the back-end. The architecture is scalable, allowing for multiple users and sessions running concurrently. This is achieved by leveraging upon Cooja - a state of the art emulator of IoT networks - that is run in headless mode. Each user is able to specify their scenario (network topology, configuration and IoT application developed in Contiki-NG) and submit it for execution; then, the system provides them with log files detailing the emulation. We demonstrate the use of the facility via a red/blue team scenario involving a variant of man-in-the-middle attack using IoT devices

The rest of this paper is articulated as follows. Section 2 reviews the existing related literature including existing IoT experimental facilities, federations of such facilities and cyber ranges. Section 3 explores the IoT Cyber Range architecture and available official security guidance on development of cyber ranges. Section 4 describes the technical implementation of the front-end engine and user interface. Section 5 describes the technical implementation of the IoT Cyber Range engine in the back-end. Section 6 discusses scenarios for cyber security training and demonstrates a proof-of-concept. Section 7 concludes this work by summarising our contribution and providing insights of our future work.

2. Related Work

2.1. Cyber Ranges

There are varying definitions of what constitutes a cyber range. Yamin et al. in [6] define a cyber range as an environment providing testbeds for research and conducting training through programs. Alternatively, Kavallieratos et al. in [7] describe a cyber range as an interactive, simulated representations of an organization's local technical infrastructure connected to a simulated Internet. They provide an isolated, and safe environment to legally practise security training without the risk of consequence.

According to Ficco and Palmieri [8], cyber ranges can consist of physical infrastructure, be completely virtualised, or a hybrid between the two. The suitability of each option depends on the wider context on which the cyber range is being built. Yamin et al. [6] also argue that there are 6 aspects that are needed for a cyber range to be considered fully functioning and effective. These are monitoring capabilities, learning, management, teaming, the environment, and scenarios. Monitoring capabilities allow for the observation of participants for effective learning, and that the cyber range is performing to an acceptable standard. This involves designated observers using methods, tools, and layers of which monitoring is being performed. Learning involves the scoring of users which means one can determine whether the cyber range is an effective learning tool. Management involves the "assignment of roles and duties to individuals and teams", which includes role management, resource management, and range management. The teaming aspect refers to the groups and individuals who are involved in the creation and participation of cyber range scenarios. This includes the red team (attacking), blue team (defending), white team (designing of scenario), green team (monitoring and maintenance of scenario infrastructure), amongst others. Environments contain the services which are used to support scenarios. These can be virtualised, physical, or a hybrid of the two [8]. Scenarios define the execution environment, context, and additional background information and stories behind exercises which are used to test individuals. These combined aspects define the characteristics of a contemporary cyber range.

2.2. IoT Testbeds and Cyber-Ranges

In [9] authors survey multiple IoT testbeds and experimenting facilities to identify mutual requirements. Authors found that most testbeds need to scale appropriately to facilitate their requirements. As IoT becomes truly global, a global-sized infrastructure will be needed to allow for full scale experimentation. This could be offered through federation or virtualisation of nodes. The heterogeneity of IoT devices in various contexts needs to be replicated in experiment infrastructure to provide a realistic and credible scenario. There is also a need for repeatability of experiments to validate results. This means that experimentation parameters need to be recorded in a way that can be shared with others. Concurrency of multiple users and experiments is necessary to make a testbed economically viable and allow for further investment and research. Increased robustness of experimental infrastructure allows for moving of cyber ranges towards a more credible and realistic experimental environment.

In [10], authors describe OpenTestBed; an open-source testbed, with comparatively cheap setup costs, replicable with its open-source and off-the-shelf nature of its components. It is simple in its nature by generically logging information through serial connections to best facilitate a variety of tests. It's physical architecture consists of a Raspberry Pi, 4 OpenMote B motes, a screen for output, and a QR code. The Raspberry Pi simply acts as a central server running a single python program. These components are grouped together to form an "OtBox". It is not a federated testbed but multiple "OtBoxes" can be grouped together allowing for data aggregation via an MQTT broker. There is also support for integrating OpenTestBed into OpenWSN allowing for application of the serial data by external tools. The focused aspect of this testbed is to serve as a proof-of-concept that dedicated testbeds and cyber ranges can be

developed cheaply compared to traditional institutionally dedicated testbeds. However, the user interface is limited and generated data likely homogeneously.

The authors in [11] present the KYPO4INDUSTRY cyber range which was designed to address the cyber security skills gap within industrial control systems (ICS). The training facility is ideal for beginner and intermediate computer science students in a simulated industrial environment. The testbed consists of a physical setup of ICS hardware nodes, such as PLCs, memory, and peripheral devices, interconnected by an isolated network. Authors in [11] further propose a course alongside the testbed which is designed to "provide an awareness of threats within the ICS domain with practical experience". This course is the equivalent of 13 weeks of dedicated study and involves content ranging from motivation, real attacks, and legal issues, to threat modelling, creating, and deploying an ICS Capture-the-Flag (CTF) game. This course serves as an evaluation of the testbed's effectiveness in teaching, but the paper does not state the results from course participants, denying any potential scrutiny.

2.3. Federated Testbeds and Cyber-Ranges

GENI (Global Environment for Networking Innovations) is a scalable infrastructure and provides services such as a virtual laboratory for conducting large-scale network experimentation via a federated architecture [12]. It is open source and is maintained by a community of stakeholders [13]. Some research argues that the concepts of "sliceability" (virtualisation and simultaneous sharing of resources while maintaining a degree of isolation for separate experiments), and deep programmability (the ability to influence low-level behaviours and interactions for the purpose of modifying to an experiments context) expands the potential for experimental networks [14]. Its focus on large-scale, federated networking solutions is unique amongst other testbeds, existing on an international domain. However, these solutions don't relate to any dedicated IoT paradigms. It is argued that due to GENI's varying range of experiment styles, durations, and sizes there is no single experiment interface. To solve this, GENI allows for interfacing through multiple APIs. The research continues to argue that forgoing the benefits of a single user interface (standardization, concentrated support, etc.), is a purposeful strategy which encourages the development of interoperating developer tools [14]. Though it should be stated that while initial results seem positive, it is too early to draw any conclusions.

FIESTA-IoT (Federated Interoperability Semantic IoT Testbeds and Applications) provides a federated infrastructure to allow for the experimentation of heterogeneous IoT technologies through an "experiment-as-a-service" solution [15]. The platform provides access to 10 testbeds across multiple countries with the capability of further scalability. FIESTA-IoT is designed to solve issues pertaining to isolated data from testbeds in different industry sectors. [16] describes how the platform allows for the translation of data to a common FIESTA-IoT ontology via a common API. Effectively the platform works as a proxy for its federated testbeds to proving a common standard for access of data allowing for the semantic interoperability of these testbeds, access of corresponding data streams and plug-ins and discovering of resources. The focus of the testbed is on the scalability and interoperability of IoT devices on a large scale and aims to confirm or deny the functionality of these devices and infrastructures, with the architecture following a modular approach. A limitation to this federated cyber range is that there are no dedicated security testbeds. This is significant regarding the large security hole that IoT devices and systems currently entail. Within this testbed, there is a lack of security related services. Experiments are deployed utilizing the system's testbed services. There is no mention as to whether these are scheduled or queued.

The FIRE (Future Internet Research and Experimentation) and its successor (FIRE+) provides a federated infrastructure where multiple testbeds and platforms allow for the connection between research and large-scale experimentation [16]. This testbed allows for research and experimentation into the field known as Future Internet where one studies the internet's prospects and emerging related technologies. FIRE(+) focuses on the management of diverse resources and facilitates the experimental life-cycle in areas such as sensor networks, IoT, 5G, SDN (Software Defined Networking),

and Cloud Computing etc. As part of this federated solution, there is no dedicated IoT testbed for security-specific training and experimentation. The testbed's focus is similar to GENI, but based in Europe. During the design process, careful attention was given in order to align to the GENI testbed in the US, allowing for interoperability between the two testbeds. The 4 key areas of discussion were identified as "resource discovery, reservation, and provision", "monitoring and measurement", "experiment control", and "SLA management and reputation services" [17]. There is no mention of security-focused or security-specific evaluation of technologies.

There are multiple, entirely virtual cyber ranges that allow for the practical training of cyber security through specific scenarios. The emergence of the cloud paradigm has allowed for the industrial scale access to virtual machines across the internet allowing for users to train against specific scenarios on-demand. The on-demand nature and scale of these solutions allows for a variation of scenarios covering differing topics from IoT best practices for security, to secure coding practices, configuring cloud solutions like AWS securely, differing aspects of web security, penetration testing, malware analysis, open-source intelligence, etc. The architecture consists of a cloud back-end, with a central portal for access. ImmersiveLabs [18] offers a dedicated virtual cyber range solution to educational institutions with a competitive aspect in terms of intra- and inter-university leader boards. HackTheBox [19] encourages users to hack its front page in order to gain an invite code to sign-up, with similar competitive leader boards. TryHackMe [20] allows for multiple users to form teams in formal cyber security competitions (such as the HackBack CTF). HackTheBox and TryHackMe both require access to a VPN via OpenVPN clients in order to safely run the scenarios on an isolated network. Neither of these virtual solutions offer a dedicated IoT-related security testbed as part of a federated solution.

There also exist four flagship projects, funded by the European Union as part of the Horizon 2020 program [21]. These projects are designed to work simultaneously and to complement each other with closely related objectives and goals. The ECHO project [22] is described as a system of federated cyber ranges designed to increase the competency of cyber security within the European Union. The CONCORDIA project is a cybersecurity competence network providing an ecosystem to lead research, technology, and industrial and public competencies [23]. Similar to Concordia [23], SPARTA is another cyber security competence network aimed to coordinate research, innovation, and training within the European Union [24]. CyberSec4Europe is a research project focused on the implementation of potential government structures in order to create a European Cybersecurity Competence Network with an emphasis for best practise examples [25].

2.4. Existing Guidelines on Cyber Range Development

There is little official guidance from the NCSC (National Cyber Security Centre) or ENISA (European Union Agency for Cybersecurity) regarding the design and development of cyber ranges. ENISA have stated that cyber ranges for the development of competencies is an agenda item [26], so there is the possibility that guidelines could be developed in the future.

In [5], authors specify four principles for building cyber ranges which are focused on cyber security. These principles were developed with years of experience in developing cyber security testbeds. They are defined as follows:

1. Provide tools and capabilities that reduce the cognitive burden on experimenters wherever possible.
2. Allow experimenters to encode their goals and constraints and leverage this information to help guide experiment construction.
3. Provide flexibility in design. A good architecture evolves with both its users and technology, and newly developed capabilities.
4. Provide multifaceted guidance to help experimenter produce high-quality experiments.

Principle 1 states that using tools to allow the transfer of knowledge between people reduces the cognitive burden on experimenters allowing them to concentrate on their tasks at hand. Principle 2 states that goals and constraints should be included in the experiments' design requirements to help focus the construction to meet these requirements. Principle 3 states that architectures evolve with their users, and newly evolved capabilities should be designed with leeway in mind to provide flexibility to meet these changing requirements. Principle 4 states that guidance should be available from all angles to help stream-line design and implementation of good-quality experiments and scenarios. This includes automated or human-driven guidance. These principles offer some level of detail but are ambiguous in their application with no guidance from the authors on how to apply these rules in practise. Having this guidance available would allow for novice experimenters transition into well-experienced experimenters with the necessary skills and knowledge to create high-quality experiments.

Additionally, NIST (National Institute of Standards and Technology) suggests in [27] that there are specific properties of a cyber range that should exist in order for the cyber range to be considered of a high standard. These include technical components (learning management system, target infrastructure, virtualisation layer, etc.), accessibility and usability (i.e., a cyber range should be both useable and accessible to its target audience), scalability, etc. Scalability allows for potential growth to enable new scenarios and provide additional flexibility and agility to rising threats and new scenarios.

3. IoT Cyber Range Architecture

This section elaborates on the high-level architecture of the system. Section 3.1 covers existing official security guidance, or lack thereof, for developing cyber ranges and testbeds. Section 3.2 explores sequence diagrams to help visualise user interactions with the system for example tasks. Section 3.2 details a high level overview of the architecture of the system including both the front-end and the back-end. The selected approach for the development of this artefact is IBM's RADM approach. IBM [28] recommend using RESTful API Design Model (RADM) for RESTful API model-driven development which allows for one to describe an API, its contents, and the technical structure of the model. The models are UML-based, and support the four stages of RESTful API development life-cycle. In an iterative fashion, one elicits API requirements, builds the API models, generates API specification and pseudo-code, and then implements and manages the API. This approach allows for the quick development of meaningful functionality and provides value to projects in a fast turnaround environment. Its individual steps of the iterative cycle are simple, and it allows for an adaptive development process to potentially changing requirements. The models created as a result of the RADM approach will be shown in this section via sequence diagrams to show use-case examples of API interactions.

3.1. User Interface

To enable the elicitation of requirements, this section explores actions which need to be conducted in order for the web interface to function. Below sequence diagrams are presented which explore the interaction between the user and the system, and the sequence of steps and interactions when performing a key task. From these diagrams one can explore required functionalities which allows for functional requirements, and supportive requirements.

3.1.1. Resource Provisioning

The below sequence diagram depicted in Figure 1 shows the steps taken and interaction between the user and different layers of the system for queuing a job onto a resource node. This involves authentication (where the user credentials are verified) and authorization (where use privileges are verified) via interaction with the user interface and database, and the selecting of jobs and forwarding to the IoT Cyber Range. The job status is then updated and returned to user as confirmation of a successful job run. Indexing shows when a sub-action is part of an action.

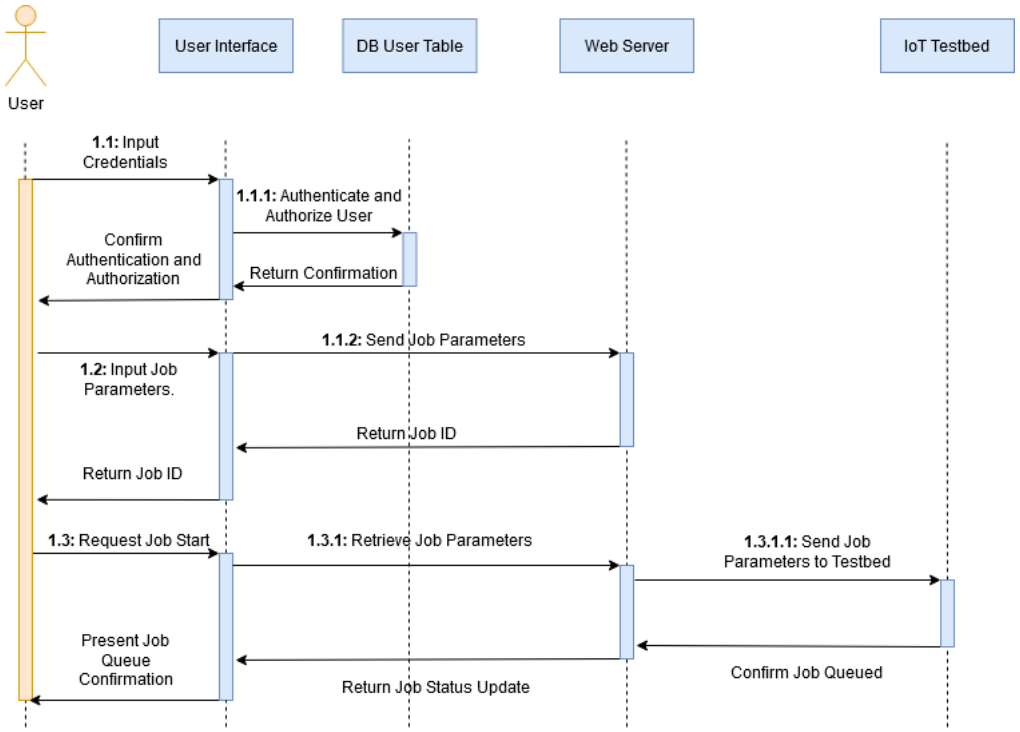


Figure 1. Job Queueing Sequence Diagram.

3.1.2. Retrieving Experiment Logs

The sequence diagram depicted in Figure 2 describes the interactions between the user and the system when requesting logs pertaining to a job. The user enters their login credentials which are queried against the user table in the database to return a confirmation that the user is authorized to proceed. The user can then request the retrieval of logs which is forwarded from the interface to the Logs table with a search parameter specifying the log ID. The logs are displayed to the user. Indexing shows when a sub-action is part of an action.

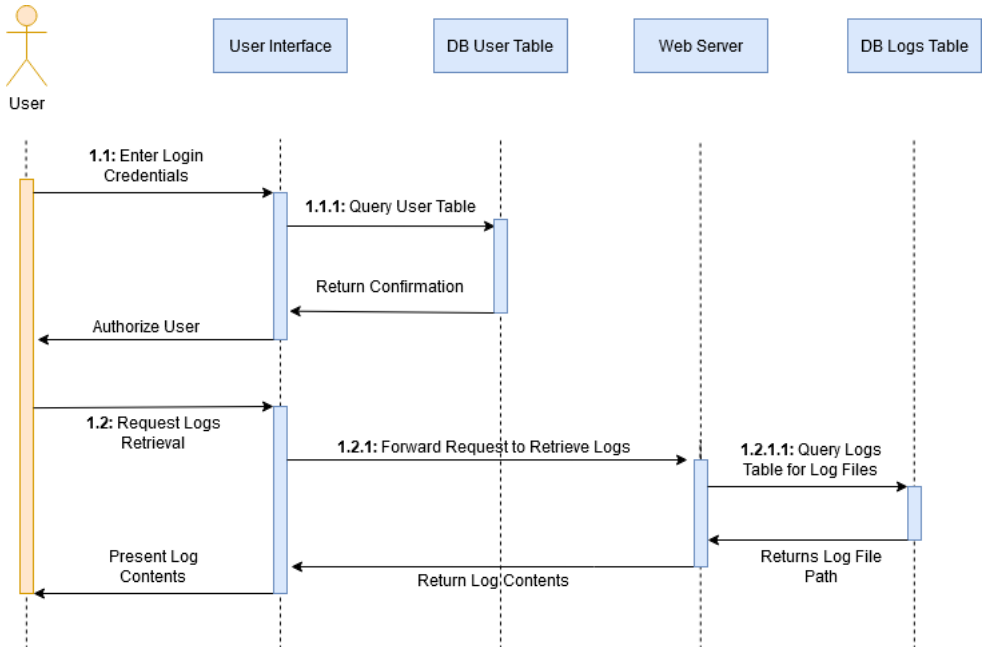


Figure 2. Log Retrieval Sequence Diagram.

3.1.3. Inquiring Resource Availability

The below sequence diagram (Figure 3) describes the actions and interactions between the user and the system when requesting node statuses and their availability. This includes the authentication of a user and the forwarding of the user's subsequent request to the testbed. The user enters their login credentials which are then queried against the user table in the database. The confirmation is returned, and the user is authorized.

The user can then request the node states. The interface queries the back-end, then queries the nodes table returning the number of available nodes both physical and virtual. Indexing shows when a sub-action is part of an action.

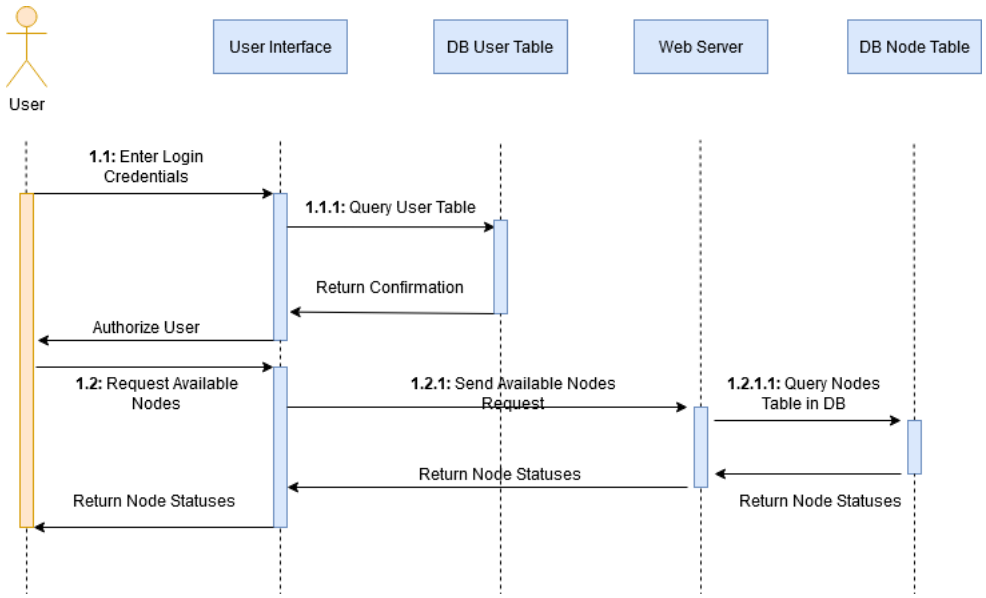


Figure 3. Retrieving Available Nodes Sequence Diagram.

3.2. Architecture Block Diagram

The below diagrams (Figures 4 and 5) depict the high-level interaction between different modules of the system. At the north end exists the user interface tier which consists of a user API and graphical templates allowing for interaction between the testbed and its users. These two interfaces are maintained by the front-end engine which handles customer requests, and authentication. The Resource API exists at the south end of the front-end engine in order to allow for communication from the IoT testbed tier when updating node statuses or sending back logs. On the opposite end of this communication line exists the IoT testbed API which allows for the receiving of job parameters from the Resource API. This API provides an additional layer of abstraction from the physical and virtual testbed resources in the IoT testbed tier. At the lower tier, the facility comprises both physical and virtual IoT resources. In particular, the physical component consists of twenty RE-MOTE IoT devices provided by Zolertia [29]. The devices form a wireless peer-to-peer mesh network over IEEE802.15.4 and are connected to the IoT-CR server over a USB-tree cabled topology for management purposes. The virtual component is powered by Cooja; a network emulator using the Contiki-NG embedded OS.

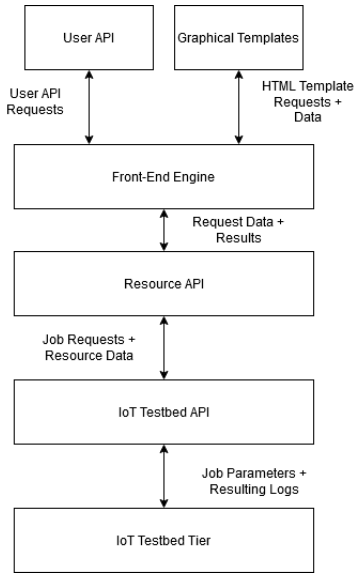


Figure 4. System UML Block Diagram.

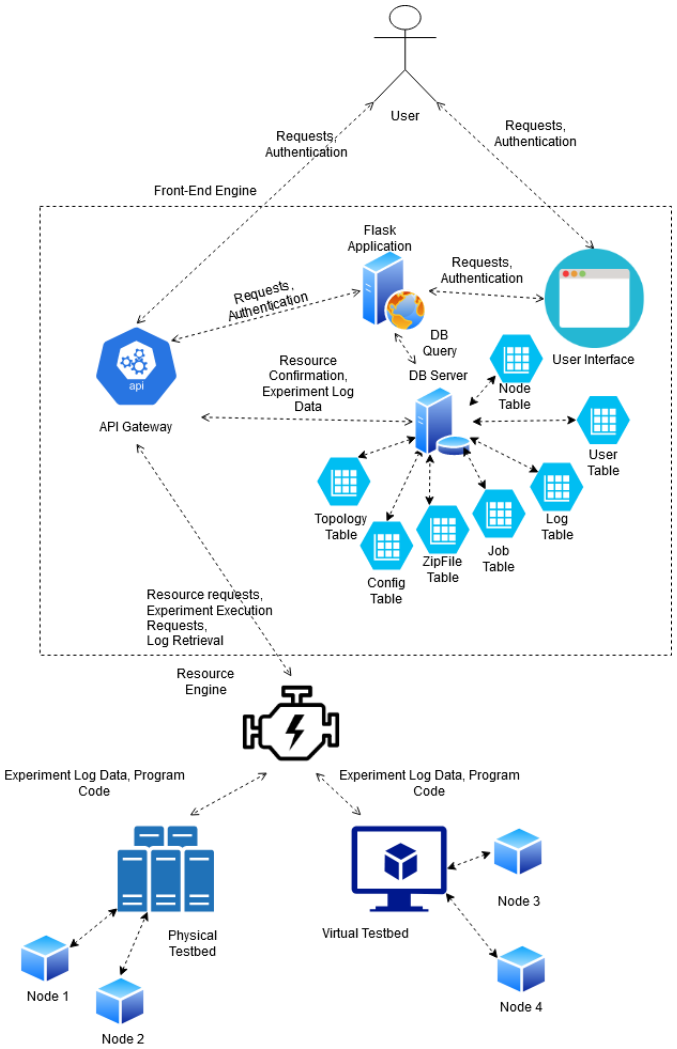


Figure 5. System Architecture of the IoT Cyber Range.

4. Front-End User Interface

This section explores the low-level design, implementation, and testing of the system application's front-end. This includes tools and libraries used in development, the full layout of the system architecture, Object Orientated Programming (OOP) class diagrams, the schema of the API structure, web interface designs, highlighted code snippets, and testing including unit testing and user testing.

4.1. Selected Tools for Development

The Web Interface and API to the testbed is written in Python 3.7 [30]. The IoT testbed tier is being written in Python. Using Python to construct the Web Interface allows for easier integration of services between the front-end engine and the IoT testbed tier. Further to this, the Python syntax is easy to read and quicker to type allowing for a faster development time. Speed of execution is not a priority for this project so this is not a bottleneck for implementation. Python facilitates importing of other codebases. In particular a codebase called Flask [31] which is a lightweight web framework allowing for the rendering of API structures, HTML templates, and back-end functionality. It also supports interfacing to databases via the flask-sqlalchemy dependency. HTML5 [32] templates and CSS3 [33] were used for the creation of graphical interfaces with bootstrap (Otto et al., 2020) also being used to provide an adaptive interface layout to different resolutions. This would allow a user to view the graphical interface on a mobile device if required. Further to this, Curl [34] is used to test API calls. Git [35] is used to allow for version control of the software code, as well as providing a remote backup for the project code as per risk analysis. SQLite3 [36] is used as a lightweight means of persistent data storage. SQLite3 does not require a Database Management System (DBMS) to function as it works as a stand-alone file. This provides efficiency for system resources in terms of storage space and resource memory.

4.2. Database Design

The ERD depicted in Figure 6 consists of seven tables: User, ZipFile, Config, Topology, Job, Log, Node. The User table holds user information such as username, the user type, etc., and is utilized during authentication and authorization. The ZipFile table handles information pertaining to ZipFile details. Config pertains to configuration information, and Topology pertains to topology information. These 3 tables link to the Job table which combines the 3 files. From a Job, one or more logs can be created. The Node table is unrelated to the other items and pertains to nodes which are currently available to the users.

4.3. Api Structure

The base URL of the API will be "/api/v1/" to allow for an API version control framework. It uses a JSON schema. The API URL structure is as follows:

Authentication-'/api/v1/auth' The user must be able to authenticate. This endpoint takes a POST request with JSON data requiring a username, and password field. A successful POST request will allow for the creation of a JSON Web Token. A 'username' and 'password' key will be required as data.

Sign Up-'/api/v1/signup' This endpoint allows a user to create an account by sending a post request containing the following keys: 'username', a potential 'password', and an 'email' address. This endpoint returns a message stating a successful sign-up, or a failed sign-up if the username is already taken. A user by default determined as a "customer" type. Currently in order for a user to be deemed an admin, or a system user type, they must contact the admins of the testbed.

Available Nodes-'/api/v1/nodes' A customer may send a GET request to this endpoint in order to retrieve the number of physical and virtual nodes available. This use case returns a message stating the number of each. If the user is a system user then they have the ability to send a POST request to add a node. This needs to include a name of the new node, and its 'node_type', either "virtual"

or "physical". A DELETE request with a 'node_id' key allows for the deletion of nodes rendering them no longer available.

Topologies-'/api/v1/topologies' The user must be able to run an experiment, and the topologies file is part of this. The topologies endpoint allows a user to view their uploaded topology files showing their filename and their associated ID. Sending a POST request allows for the uploading of another topology file with a 'file' key. A POST request would return a message stating the file has been uploaded with an associated ID, or that the file failed to upload as it already exists.

Configs-'/api/v1/configs' The 'configs' endpoint allows a user to view their uploaded configuration files showing their filename and their associated ID. Sending a POST request allows for the uploading of another configuration file with a 'file' key. A POST request would return a message stating the file has been uploaded with an associated ID, or that the file failed to upload.

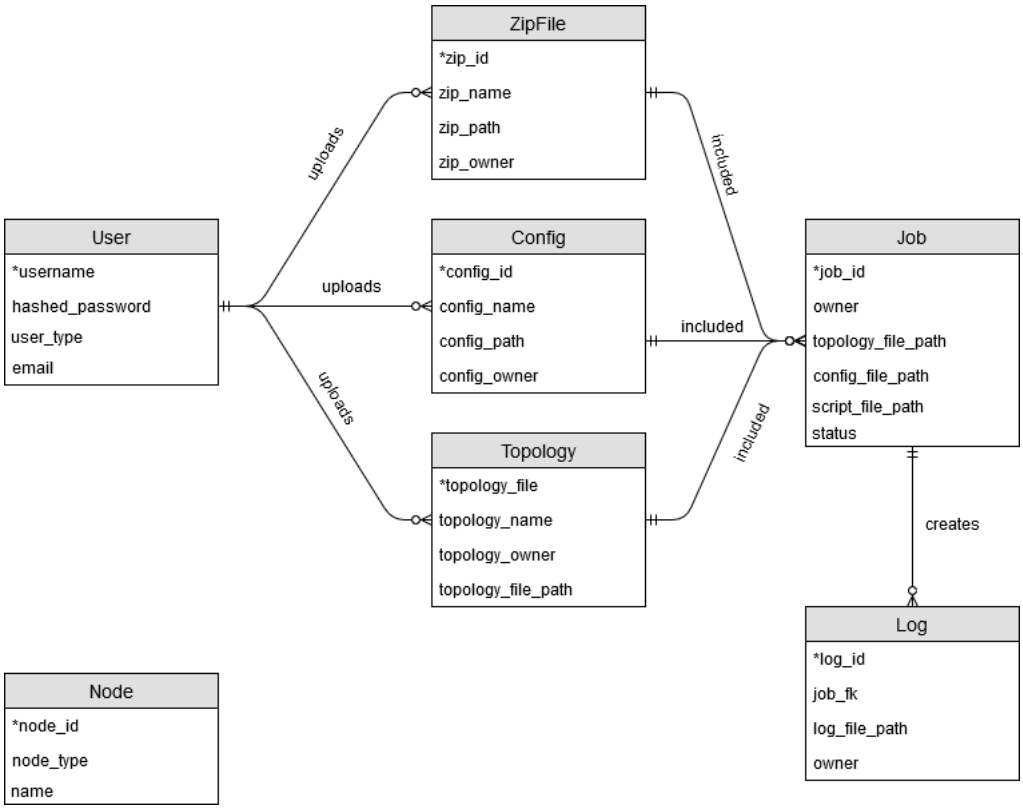


Figure 6. Log Retrieval Sequence Diagram.

Scripts-'/api/v1/scripts' The user should be able to upload scripts to facilitate an experiment. Scripts need to be uploaded as .zip files. The scripts endpoint allows a user to view their uploaded zip files showing their filename and their associated ID. Sending a POST request allows for the uploading of another zip file with a 'file' key. A POST request would return a message stating the file has been uploaded with an associated ID, or that the file failed to upload.

Jobs-'/api/v1/jobs' The jobs endpoint allows for a user to view their created jobs and their ID's by sending a GET request. Sending a POST request with an uploaded 'topology_id', an uploaded 'config_id', and an uploaded 'zip_id' will result in the creation of a new job and an ID will be returned. A GET request to see a specific job file can be done by sending the request to '/api/v1/jobs/<job_id>' where <job_id> is the ID of the job. A job can be run with a GET request sent to '/api/v1/job/<job_id>/run' where <job_id> is the ID of the job to be run.

Logs-'/api/v1/jobs/<job_id>/logs' One can see log IDs associated with a job by viewing the endpoint above with a GET request. A system user can send a POST request to add an additional log which includes the logfile ('file') as a key and the job_id its associated with. The file needs to

be a zip in case multiple logs are returned. A user can see log contents be sending a Get request to '/api/v1/jobs/<job_id>/'. A successful POST request will allow for an email to be sent to the user who requested the job to be run, letting them know that logs are ready to be observed.

Help-'/api/v1/help' A user can send a GET request to view helpful information from this endpoint. Further help is also available by some additional endpoints detailed within the help information.

4.4. API Python Wrapper

To aid users in consuming the API, which would otherwise require lengthy curl requests, a lightweight python3 client wrapper is implemented Figure 7. This wrapper allows the user to sign up and login to the Cyber Range by operating on the API endpoints presented in Section 4.3 through a menu driven interface. The wrapper holds the JTW token associated with the logged in account, allowing users to query their files, jobs and fetch log files. Below are screenshots of the wrapper in use, creating a job from a scenario, running the job and collecting the log files.

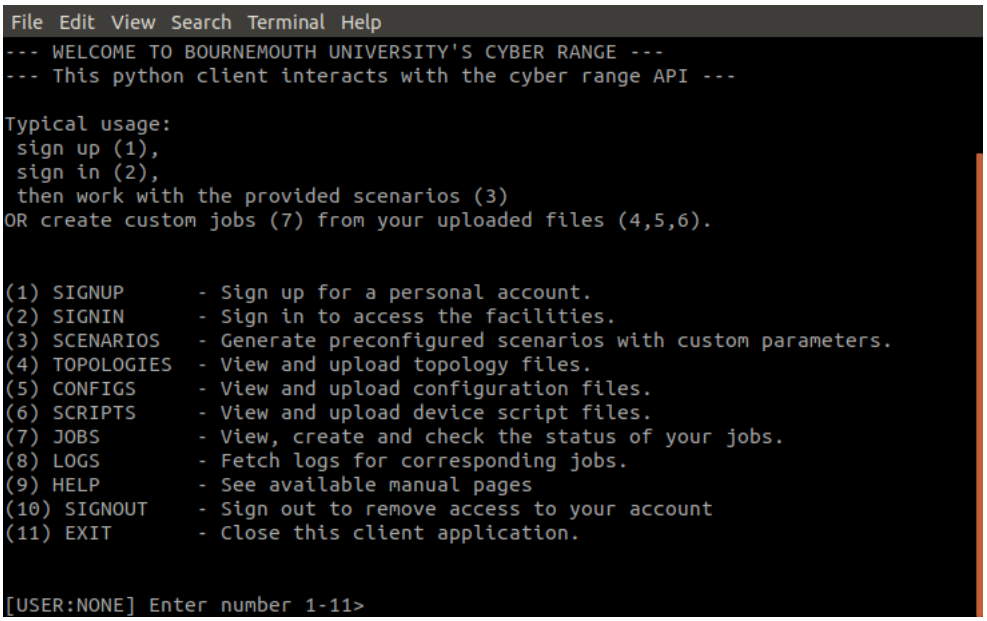


Figure 7. The home screen of the wrapper.

5. IoT-CR Back-End Operation

In order to satisfy the various needs of IoT researchers, the cyber range is virtualised to offer a range of devices that would not otherwise be accessible in physical form. The presented Cyber Range is powered by Contiki-NG-probably the most commonly used embedded OS for IoT networked systems- and Cooja; a network simulator for Contiki-NG written in Java and designed specifically for Wireless Sensor Networks. Cooja is a tool provided within the Contiki-NG Operating System, which itself focusses on "dependable (secure and reliable) low-power communication and standard protocols for IoT, such as IPv6/6LoWPAN, 6TiSCH, RPL, and CoAP" [37]. Cooja provides a detailed emulation of the entire network stack (from the link layer to the application layer) using as input the same Contiki-NG source code that would be deployed in actual physical IoT systems. This enables the researcher to focus either on individual devices or on the networking aspects of a system. Furthermore, it provides great agility to the IoT-CR as scenarios developed to be run virtually can be easily ported to physical IoT systems.

Cooja is able to operate using the scripts and libraries available within Contiki-NG, providing a graphical framework for users to assign custom scripts to virtualised devices, control these devices

within a topology map and test networking scenarios with the provision of tools such as viewing real-time device standard output or adding breakpoints within the simulation for key events.

Cooja also has the ability to run in headless mode (i.e., with no graphical user interface) thanks to an inbuilt debug mode. Cooja allows simulations to be saved in '.CSC' format, saving all the parameters for the simulation in XML. Whilst this can have the typical usage of saving and reloading a simulation, the CSC files can also be passed to Cooja via the command line to force a non-graphical Cooja process. Furthermore the 'Simulation Script Editor' tool provided within Cooja allows for users to have scripted control over the simulation via JavaScript code-seen in Figure 8.

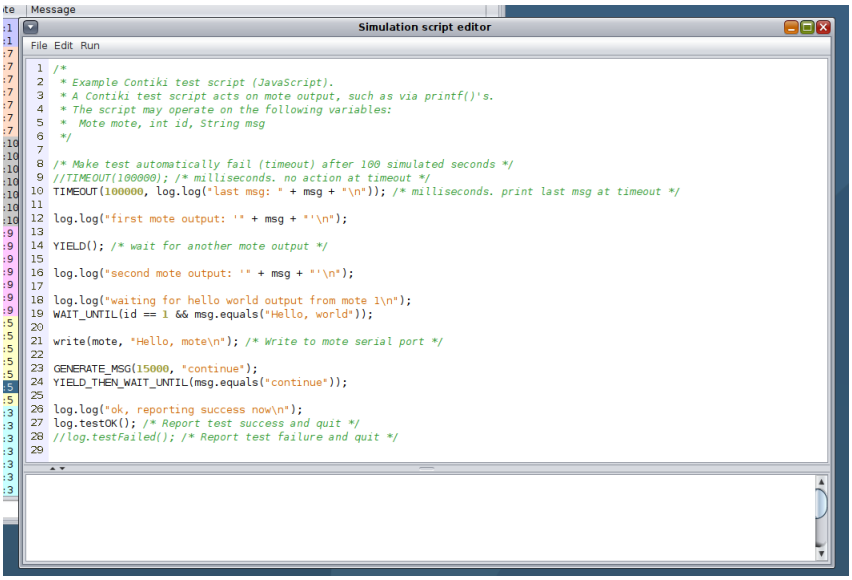


Figure 8. Simulation Script Editor with example JavaScript Code.

The Simulation Script Editor allows the user to save the simulation CSC file amended with the JavaScript code as another XML element, allowing the simulation to run automated, without the need for manual intervention. Users can create their own code or use a number of preconfigured scripts. This allows for a logical detachment from Cooja, steering the Cyber Range away from simply providing Cooja as an Application-as-a-Service to utilising Cooja's simulation abilities to provide users with a Cyber Range experience similar to other IoT testbeds. As the Simulation Script Editor is fully integrated within Cooja, and therefore Contiki-NG, all system defined events (YIELD, PROCESS, YIELD_THEN_WAIT_EVENT_UNTIL, PROCESS_WAIT_EVENT_UNTIL, etc.) [38] can be detected and triggered within the JavaScript code.

Given the simulated nature, rather than emulated, the system can run on servers capable of handling multiple jobs, as not much computing power is required to emulate IoT devices that are typically resource constrained. Furthermore, virtual systems offer scalability beyond physical systems due to the hardware independent properties, but can restrict the experience in such cases as interacting with physical buttons or sensors. Cooja uses the hosting Operating System to maintain the resource allocation required for simulations, so future development allows us to operate on hosted server-grade hardware for improved performance with costs dictated by system usage rather then maintenance of physical devices.

The automated handling of Cooja, dictating the control and running of jobs, is written in Python 3. This script polls the database periodically for newly submitted jobs. When new jobs are run in Cooja the logs, standard output and standard error are captured by default, as well as any logs generated by the user within their simulation script code (the JavaScript code). These logs are then exposed over the API for users to consume.

6. Demonstration of a Cyber-Range Scenario

In order to enable initial user comprehension and to enable quicker job development, the system can provide scenarios that offer the complete Contiki-NG project build folder and Cooja simulation (CSC) file required for any job. Generating these scenarios, users will be able to understand the format of the simulation code in use, providing the opportunity to change the parameters of a given scenario such as: wireless protocols, quantity of nodes, density of network, percentage of nodes running particular code (accounting for sink nodes or different team sizes). In Section 6.1, the example user generates the code for the scenario "Pass the token". This scenario demonstrates typical blue team/red team cyber security events. The scenario plays out as two opposing teams trying to share a token value; whilst the blue team attempts to increment and share the token between themselves, the opposing red team attempts to intercept and decrement the token before passing it onto the blue team, resembling a simplistic man-in-the-middle attack. Blue team tries to increment the value resembling a defence response. No node may send more than a single token without receiving a new token value from an inbound packet, this effectively locks out other packets to enable the effect from the man-in-the-middle attack. The blue teams token value must reach an upper bound, whilst the red team tries to achieve a lower bound. When a node reaches either of the thresholds, it declares a state of win or defeat and terminates the decentralised program, posting the state to logs. Figure 9 gives a diagrammatic view of the scenario.

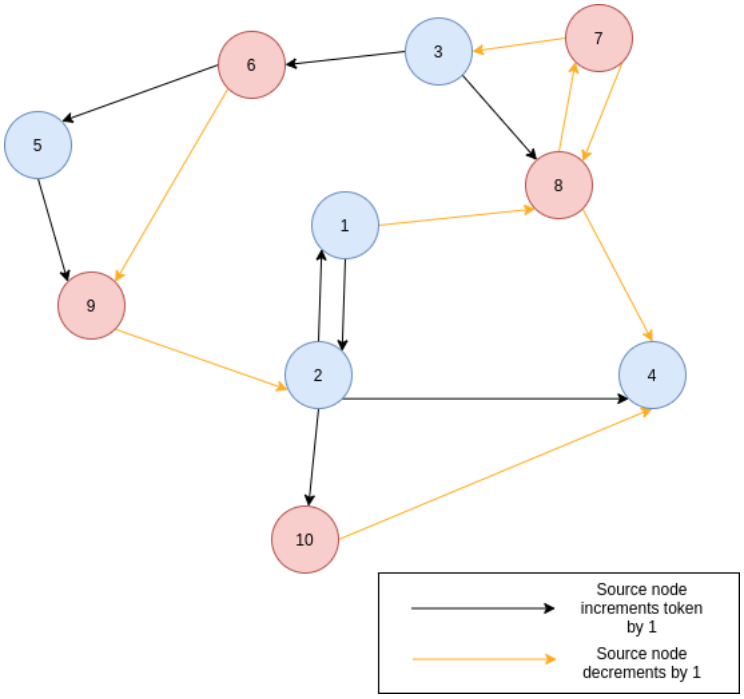


Figure 9. Ten-node network depicting the token modification and exchange.

In this scenario example, with a starting value of 10, a walk across the network encompassing nodes {1,8,7,3,6,5,9,2,4} results to a token of value 12. However, as all nodes that start the simulation are able to send one packet before locking, multiple walks across the network can occur at a given time. Walks can be cut short by other walks, due to the node single packet lock, varying the simulation on each run, and at the point that each node reaches the upper or lower threshold, causing the eventual end of the simulation. It is worth noting that the scenario can be executed with a variety of underlying networking protocols and technologies, including both non-IP (e.g., NullNet) and IPv6 (e.g., RPL and 6LoWPAN) networking stacks. This allows the trainees to repeat simulations with varying numbers of nodes, using different network technologies, record results and compare different aspects of IoT

networking. They gain a working understanding of a cyber security team structure and exposure to the high-level C programming of IoT devices.

6.1. Demonstration

On first arrival at the system, users are required to make an account (Figure 10a). The username and password are then used for login and the email notification for job completion. Users are given the chance to generate the required files for a job, in the form of a scenario. These files are added to the users set of files so that custom variations of the scenario can be created. Such variations can include any parameters defined within the Cooja CSC file:- number of nodes, simulation duration time, assignment of particular nodes to scripts. Essentially the user is given the same level of control over the testbed as any adversary would hold over an actualized physical IoT Cyber Range. Alternatively, users can upload files, as seen in Figure 10d. Figure 10e,f show the creation and subsequent scheduling of the scenario job. In Figure 10e, the user picks from each file category to build a job, which is saved to the user account. Only when the user wishes to run the job do they utilise the job scheduling (Figure 10f). Once the job is completed, users will get an email informing them (Figure 11) and access to the logs is given in the client wrapper (Figure 10g).

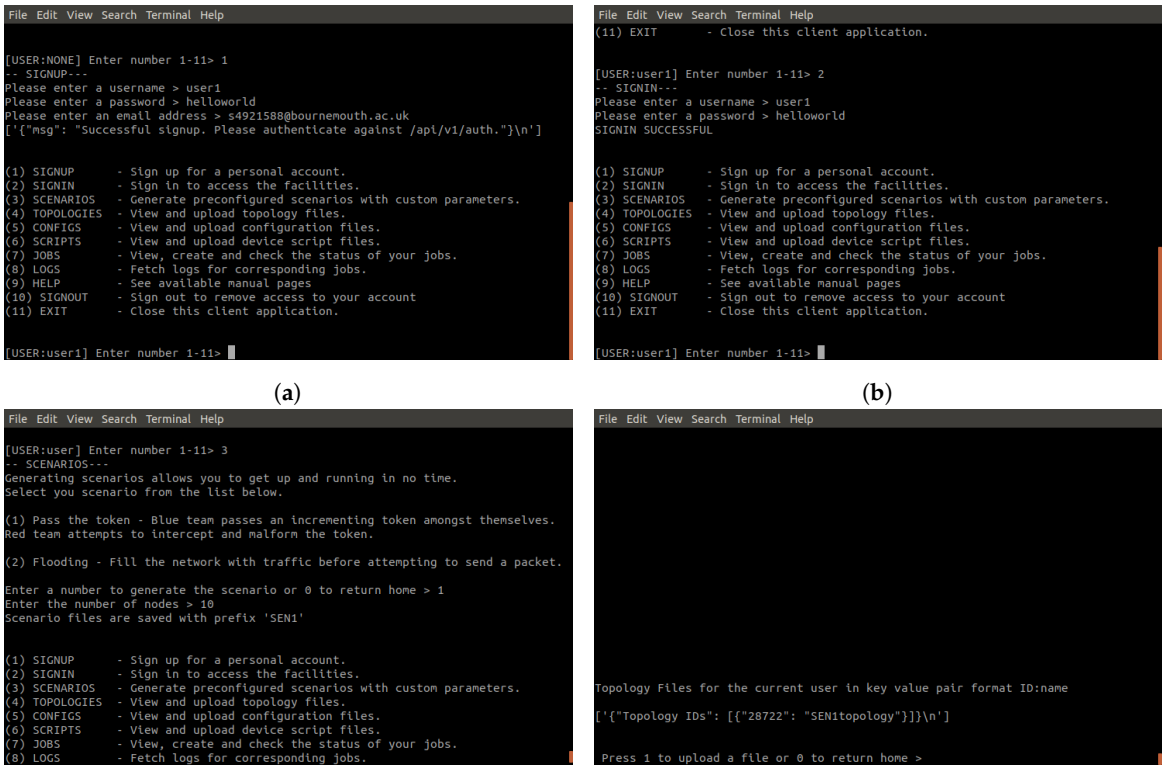


Figure 10. *Cont.*

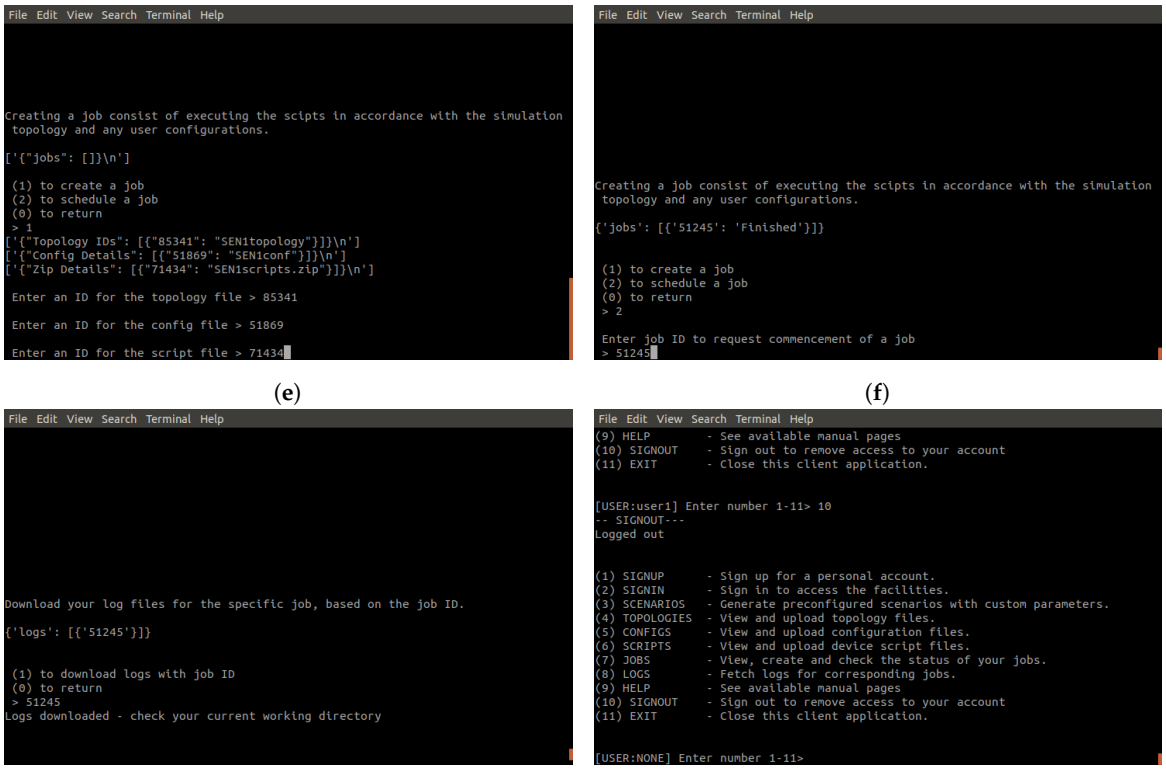


Figure 10. Python Wrapper Screenshots. (**a**) The signing up process. (**b**) The signing in process. (**c**) The creation of a scenario with parameters. (**d**) Topology page. You can see the scenario topology file already present. (**e**) Job creation. The user creates the job from the already uploaded files. (**f**) Job schedule. The user can enable the job to run, in doing so it's state changes to finished. (**g**) Downloading the logs. A user can download all logs for each job. (**h**) The signing out process. You can see the username changes to NONE.

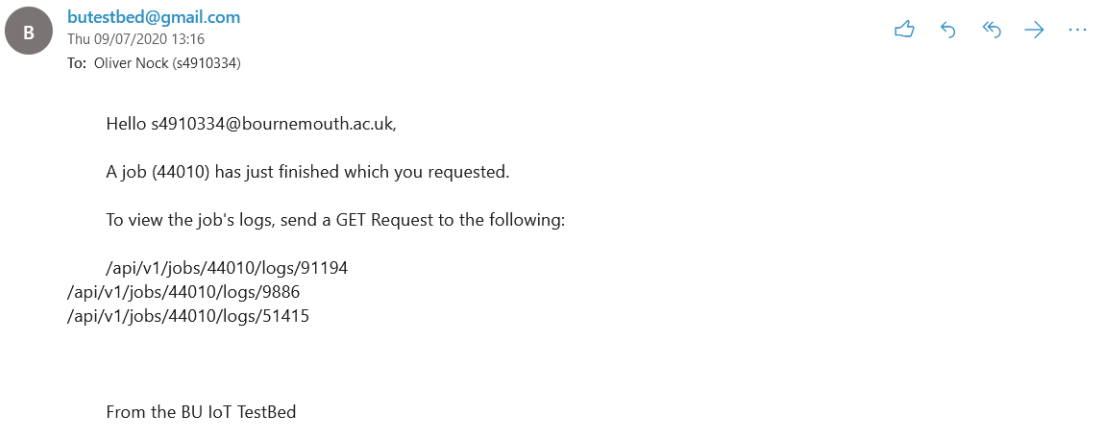


Figure 11. Example of communication with the user via email informing of log availability.

6.2. Results

Utilising the demonstration, the testbed was operated under the following parameters:

- Network size: 10 Zolertia Motes
- Network protocol: Nullnet-an IPv6-less MAC networking protocol
- Mobility: Zero mobility across all nodes.

- All nodes commence execution at the same time.
- Simulation is terminated after 600 s (10 min).

The results Table 1, shows the metrics extracted from execution of the discussed scenario. Here 10 motes form 2 teams of varying size disparity. Sending packets via broadcast, every node records both the sent and received packets until the node reaches the defined threshold, where one team is declared the winner, or until the simulation times out.

Table 1. Results following a 10 mote Blue vs. Red simulation using the provided scenario.

Number of Nodes	Blue/Red Division	Avg. Packets Sent	Avg. Packets Received	Duration (Secs)	Winning Team
10	5/5	256,112	262,634	600	-
10	6/4	52,208	55,442	420	Blue
10	7/3	42,028	45,511	356	Blue
10	4/6	52,101	55,621	422	Red
10	3/7	41,986	42,420	321	Red

These preliminary results serve to exhibit the utility of the testbed. In running the scenario discussed across several network sizes, it is shown that a novice user can generate quantifiable results. The flexibility of the testbed can be shown not only through the variable parameters, but also through the creative configuration by the user - to add a third team or increase the team disparity for example. The testbed encourages users to design future scenarios to address and combat the security challenges of IoT, closing the identified skills gap.

7. Conclusions and Future Work

As the technological paradigm of Internet of Things matures towards higher readiness levels, the gap in efficiently addressing corresponding cyber security aspects of IoT systems and the shortage in related IoT security skills become of increasing importance. This highlights the need for IoT experimenting and training facilities that are focused on security. In this work we presented an IoT Cyber Range; a user-focused IoT testbed designed to host multiple users and the execution of multiple training scenarios concurrently. We demonstrated its use via a red/blue team scenario involving a variant of man-in-the-middle attack using IoT devices.

On going work includes the extension of the IoT-CR towards its federation with the European network of cybersecurity centres of the H2020 ECHO project https://echonetwork.eu/, thus contributing to the build up of regional cybersecurity competence and capacity in Europe. The facility will be accompanied by readily available cyber-security training scenarios addressing nominal security issues of modern IoT networked systems. The efficiency of the scenarios in helping the trainees acquire new knowledge and skills will be evaluated by having diverse cohorts use the facility—ranging from low and moderate (e.g., undergraduate and postgraduate students) to high (e.g., IoT and cyber security professionals) expertise.

Author Contributions: All authors contributed equally to this work. O.N. worked on the background literature, as well as the front-end development of the IoT-CR. J.S. worked on the design and development of the back-end and the development of the cyber security scenario. C.M.A. supervised and provided guidance, both regarding the text and the technical implementation. All authors contributed to text writing. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by ECHO https://echonetwork.eu/ under the European Union's Horizon 2020 research and innovation programme; grant agreement no 830943.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Marzano, A.; Alexander, D.; Fonseca, O.; Fazzion, E.; Hoepers, C.; Steding-Jessen, K.; Chaves, M.H.; Cunha, Í.; Guedes, D.; Meira, W. The evolution of bashlite and mirai iot botnets. In Proceedings of the 2018 IEEE Symposium on Computers and Communications (ISCC), Natal, Brazil, 25–28 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 00813–00818.
2. BBC. Mirai Botnet: Three Admit Creating and Running Attack Tool. 2017. Available online: https://www.bbc.co.uk/news/technology-42342221 (accessed on 15 April 2020).
3. Solon, O. Team of Hackers Take Remote Control of Tesla Model S from 12 Miles Away. 2016. Available online: https://www.theguardian.com/technology/2016/sep/20/tesla-model-s-chinese-hack-remote-control-brakes (accessed on 12 May 2020).
4. Vogel, R. *Closing the Cybersecurity Skills Gap*; Charles Sturt University: Bathurst, Australia, 2016. Available online: https://www.academia.edu/25380112/CLOSING_THE_CYBERSECURITY_SKILLS_GAP (accessed on 12 May 2020).
5. Schwab, S.; Kline, E. Cybersecurity Experimentation at Program Scale: Guidelines and Principles for Future Testbeds. In Proceedings of the 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Stockholm, Sweden, 17–19 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 94–102.
6. Yamin, M.M.; Katt, B.; Gkioulos, V. Cyber ranges and security testbeds: Scenarios, functions, tools and architecture. *Comput. Secur.* **2020**, *88*, 101636.
7. Kavallieratos, G.; Katsikas, S.K.; Gkioulos, V. Towards a cyber-physical range. In Proceedings of the 5th on Cyber-Physical System Security Workshop, Auckland, New Zealand, 8 July 2019; pp. 25–34.
8. Ficco, M.; Palmieri, F. Leaf: An open-source cybersecurity training platform for realistic edge-IoT scenarios. *J. Syst. Archit.* **2019**, *97*, 107–129.
9. Gluhak, A.; Krco, S.; Nati, M.; Pfisterer, D.; Mitton, N.; Razafindralambo, T. A survey on facilities for experimental internet of things research. *IEEE Commun. Mag.* **2011**, *49*, 58–67.
10. Munoz, J.; Rincon, F.; Chang, T.; Vilajosana, X.; Vermeulen, B.; Walcarius, T.; Van de Meerssche, W.; Watteyne, T. OpenTestBed: Poor Man's IoT Testbed. In Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 29 April–2 May 2019; IEEE: Piscataway, NJ, USA, 2019, pp. 467–471.
11. Čeleda, P.; Vykopal, J.; Švábenskỳ, V.; Slavíček, K. KYPO4INDUSTRY: A Testbed for Teaching Cybersecurity of Industrial Control Systems. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education, Portland, OR, USA, 11–14 March 2020; pp. 1026–1032.
12. Technologies, R.B. What Is GENI. 2020. Available online: https://www.geni.net/about-geni/what-is-geni/ (accessed on 24 February 2020).
13. GENI-NSF. GENI: Global Environment for Network Innovations. 2020. Available online: https://github.com/GENI-NSF (accessed on 26 February 2020).
14. Berman, M.; Chase, J.S.; Landweber, L.; Nakao, A.; Ott, M.; Raychaudhuri, D.; Ricci, R.; Seskar, I. GENI: A federated testbed for innovative network experiments. *Comput. Netw.* **2014**, *61*, 5–23.
15. Union, E. Federated Interoperable Semantic IoT Testbeds and Applications. 2020. Available online: http://fiesta-iot.eu/index.php/iot-experiments-as-a-service/ (accessed on 24 February 2020).
16. Kalatzis, N.; Routis, G.; Roussaki, I.; Papavassiliou, S. Enabling data interoperability for federated IoT experimentation infrastructures. In Proceedings of the 2018 Global Internet of Things Summit (GIoTS), Bilbao, Spain, 4–7 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
17. Wauters, T.; Vermeulen, B.; Vandenberghe, W.; Demeester, P.; Taylor, S.; Baron, L.; Smirnov, M.; Al-Hazmi, Y.; Willner, A.; Sawyer, M.; et al. Federation of Internet experimentation facilities: architecture and implementation. In Proceedings of the IEEE European Conference on Networks and Communications 2014 (EuCNC'2014), Bologna, Italy, 23–26 June 2014.
18. Labs, I. Immersive Labs. 2020. Available online: https://www.immersivelabs.com/ (accessed on 20 May 2020).
19. Box, H.T. Hack The Box—Penetration Testing Labs. 2020. Available online: https://www.hackthebox.eu/ (accessed on 20 May 2020).
20. TryHackMe. TryHackMe | Hacking Training. 2020. Available online: https://tryhackme.com/ (accessed on 20 May 2020).

21. Union, E. Horizon 2020. 2020. Available online: https://ec.europa.eu/programmes/horizon2020/en (accessed on 6 July 2020).
22. Network, E. ECHO Project Summary. 2020. Available online: https://echonetwork.eu/project-summary/ (accessed on 6 July 2020).
23. Union, E. Concordia. 2020. Available online: https://www.concordia-h2020.eu/ (accessed on 6 July 2020).
24. ANSII. SPARTA—A Cybersecurity Competence Network to Coordinate Research, Innovation and Training within the European Union. 2020. Available online: https://www.ssi.gouv.fr/en/actualite/sparta-a-cybersecurity-competence-network-to-coordinate-research-innovation-and-training-within-the-european-union/ (accessed on 6 July 2020).
25. Union, E. Cyber Security for Europe. 2020. Available online: https://cybersec4europe.eu/about/ (accessed on 6 July 2020).
26. European Union Agency For Network and Information Security. *Priorities for EU Research*; Network and Security: Columbia, MD, USA, 2017.
27. NIST. The Cyber Range: A Guide. 2020. Available online: https://www.nist.gov/system/files/documents/2020/06/25/TheCyberRange-AGuide(NIST-NICE)(Draft)-062420_1315.pdf (accessed on 25 August 2020).
28. IBM. RESTful API Design Methodology. 2020. Available online: https://www.ibm.com/support/knowledgecenter/SSRASJ_8.8.0/com.ibm.ima.ug_soa/soa/InfoSphere/restful/restful_intro.html (accessed on 11 March 2020).
29. Lignan, A. *Zolertia RE-Mote Platform*; Github: San Francisco, CA, USA, 2016. Available online: https://github.com/Zolertia/Resources/wiki/RE-Mote (accessed on 5 May 2020).
30. Foundation, P.S. Welcome to Python.org. 2020. Available online: https://www.python.org/ (accessed on 4 May 2020).
31. Pallets. Flask Project. 2020. Available online: https://palletsprojects.com/p/flask/ (accessed on 4 May 2020).
32. Group, W.H.A.T.W. Web Hypertext Application Technology Working Group. 2020 Available online: https://whatwg.org/ (accessed on 4 May 2020).
33. Cosortium, W.W.W. Cascading Style Sheets. 2020. Available online: https://www.w3.org/Style/CSS/ (accessed on 4 May 2020).
34. Stenberg, D. Curl. 2020. Available online: https://curl.haxx.se/ (accessed on 4 May 2020).
35. Torvalds, L. Git. 2020. Available online: https://git-scm.com/ (accessed on 4 May 2020).
36. Consortium, S. SQLIte Home Page. 2020. Available online: https://sqlite.org/index.html/ (accessed on 4 May 2020).
37. Dunkels, A.; Gronvall, B.; Voigt, T. Contiki-a lightweight and flexible operating system for tiny networked sensors. In Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks, Tampa, FL, USA, 16–18 November 2004; IEEE: Piscataway, NJ, USA, 2004; pp. 455–462.
38. Noman, U.A.; Negash, B.; Rahmani, A.M.; Liljeberg, P.; Tenhunen, H. From threads to events: Adapting a lightweight middleware for Contiki OS. In Proceedings of the 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 8–11 January 2017; IEEE: Piscataway, NJ, USA, 2017, pp. 486–491.

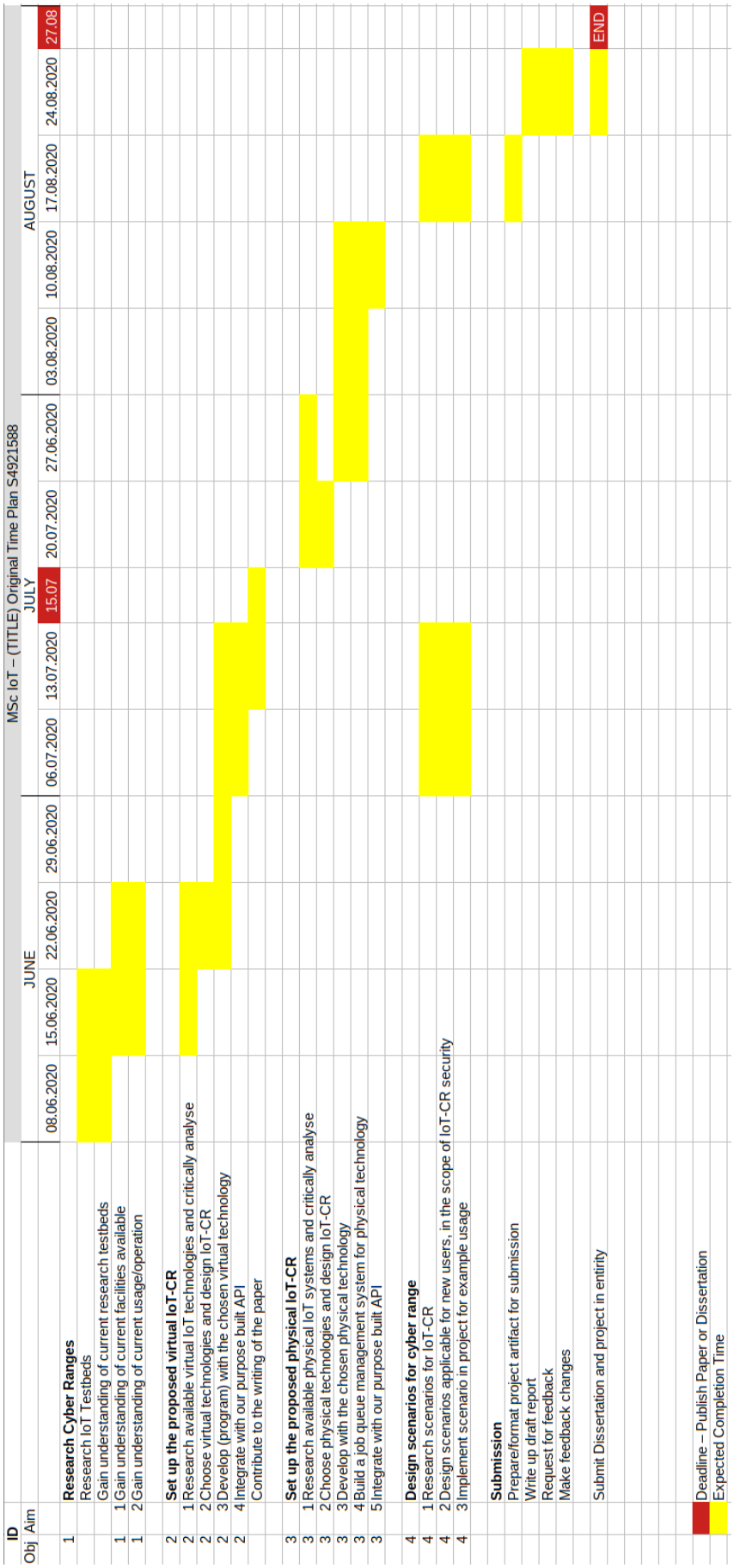


Figure 1: Original plan time sheet created 08.06.2020

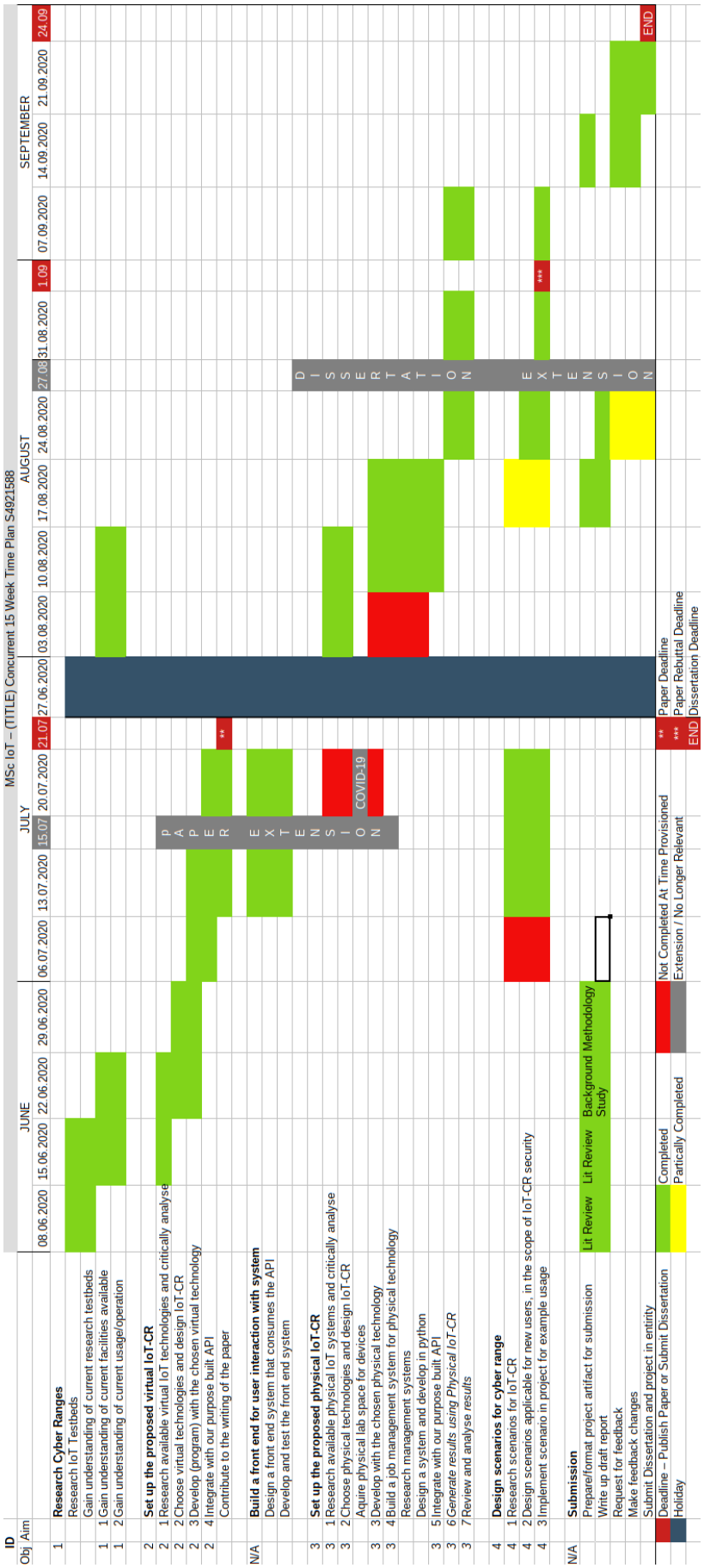


Figure 2: Updated time sheet completed 21.09.2020