ConCERTS: An IoT Cybersecurity Research Range for Education, Experimentation, and Security Research

Dave McKay[a], Matthew Bush[b], Marko Kovacevic[c] and Atefeh Mashatan[d]
Cybersecurity Research Lab, Ted Rogers School of Information Technology Management, Toronto Metropolitan University, 350 Victoria St, Toronto, Canada
{*dave.mckay, matthew.bush, marko.kovacevic, amashatan*}*@torontomu.ca*

Keywords: Cybersecurity Range, IoT Security, Simulated IoT Devices, Scripted IoT, Reproducible IoT Datasets.

Abstract: Internet-of-Things (IoT) connected devices in enterprise and residential environments are weak points in network security. Security testing over a variety of IoT network configurations is hampered by device, infrastructure, and architectural heterogeneity that is often associated with IoT deployments. Though IoT testing environments have been developed, they often lack the flexibility, control, and visibility required for varied and repeatable education, research, and testing scenarios. This paper aims to address this gap by both exploring the shortcomings of existing IoT test environments and proposing ConCERTS - a Configurable Cybersecurity Education Range and Testing Stack as a novel architectural approach for IoT testbeds and cybersecurity research ranges.

1 INTRODUCTION

The Internet of Things (IoT) is rapidly changing our lives and its influence will only continue to grow. The number of IoT enabled devices is predicted to reach 24.1 billion by 2030 (Xenofontos et al., 2022). Though these devices vary immensely in computing power and purpose, they are all capable of receiving and transmitting data and are therefore vulnerable to attacks. IoT devices also have access to the physical world through a combination of sensors, actuators, and controllers. The ubiquity of IoT and its access to the physical world makes these devices particularly important to secure. Unfortunately, increased IoT connectivity is creating a number of security concerns and potential attack surfaces (Modarresi and Symons, 2020). Potential attacks and vulnerabilities that could compromise a number of devices have already been discovered (Khoury et al., 2021). Managing the security of these devices is made even more difficult by the sheer number of device types and network configurations that exist. IoT devices are small, ubiquitous, low-powered, forgettable, and prime targets for malware. Most consumers never update the firmware on the smart light bulb or camera in their home (Tawalbeh et al., 2020). The combination of internet accessible IoT devices and unpatched firmware greatly increases the attack surface of IoT networks. The Mirai Botnet is an example of malware that took advantage of lax IoT security to take over a huge number of devices worldwide which were subsequently used in distributed denial of service (DDoS) attacks (De Donno et al., 2018). To improve education and advance IoT security, researchers often turn to creating various IoT testbeds and ranges. While many testbeds and ranges are accessible to researchers, they have shortfalls in control, flexibility, reproducibility, or visibility. This can be remedied through creating a custom test environment or manually reconfiguring an existing one, these solutions often require a prohibitive degree of time and technical expertise. To address the challenges present in traditional IoT research and education, a novel architectural approach, named Configurable Cybersecurity Education Range and Testing Stack (ConCERTS), is presented here along with two demonstration scenarios.

This paper explores several existing IoT ranges and testbeds that have applications for both security research and education. A literature review has been undertaken to identify gaps in the design considerations undertaken in developing them. The missing considerations are identified as requirements for the proposed ConCERTS architecture. The architecture

[a] https://orcid.org/0009-0007-4450-8614
[b] https://orcid.org/0000-0002-1894-3424
[c] https://orcid.org/0000-0001-5918-2337
[d] https://orcid.org/0000-0001-9448-9123

McKay, D., Bush, M., Kovacevic, M. and Mashatan, A.
ConCERTS: An IoT Cybersecurity Research Range for Education, Experimentation, and Security Research.
DOI: 10.5220/0013370900003899

In *Proceedings of the 11th International Conference on Information Systems Security and Privacy (ICISSP 2025) - Volume 1*, pages 71-82
ISBN: 978-989-758-735-1; ISSN: 2184-4356

enables an open environment in which researchers have complete visibility, control, reproducibility, and flexibility over all core building blocks of IoT - from the control server, down to the physical sensors and actuators. ConCERTS enables researchers to create fully localized and extensible test scenarios, capture activity data and play it back. A basic device, such as a Raspberry Pi, can be connected to a variety of physical sensors and can be have its behaviour remotely configured to allow one device to simulate several very different IoT devices. Controllers for these simulated IoT devices run locally and can be programmed to enable many use cases (e.g., access control, inventory, etc.). Device activity scripting allows researchers to perform reproducible experiments and generate comprehensive datasets that are not limited by the heterogeneity of the network or human and environmental interactions. The viability of the proposed architecture has been tested by implementing two scenarios, a door lock card reader and a Heating Ventilation and Air Conditioning (HVAC) system.

The structure of this paper is as follows. Section 2 details the background and reports on the literature review of existing approaches for testbeds and ranges. Section 3 outlines the key requirements identified as necessary for IoT ranges and testbeds for research and training. Section 4 proposes an architecture for ConCERTS and details the implementation of the demonstrations. Section 5 provides a discussion on relevant outcomes of the approach taken. Section 6 outlines the next steps and potential future work related to this project. The conclusions of this work are drawn in Section 7.

2 BACKGROUND AND RELATED WORK

IoT devices are popular for DDoS attacks because the randomness of where the devices reside makes it difficult to detect abnormal traffic patterns for anti-DDoS systems (De Donno et al., 2018). Security researchers have even developed a ransomware that locks and controls your smart thermostat (Franceschi-Bicchierai, 2016). Many IoT devices on Shodan often include vulnerable firmware or default usernames and passwords (Albataineh and Alsmadi, 2019). These devices are prime targets for attackers and are an entry point into networks. Heterogeneity is another significant concern for IoT networks with many devices using different communication protocols and control interfaces (Hamad et al., 2019). IoT devices do not always possess robust or easily accessible monitoring capabilities, nor do they natively possess the ability to aggregate and analyze homogeneous data centrally within the network. Many of these devices also need to be configured and connected independently. Not only does this complicate security for existing networks, but it makes provisioning test networks for research and education much more difficult. Many IoT devices may even store personal or enumerable data in plaintext (Lodge, 2016). While some IoT devices cryptographically sign updates to prevent exploits, these updates can be compromised during the transport or deployment phases, or in the situations where an attacker has physical access to the device (Maxxz, 2016). Kolias et al. (Kolias et al., 2016) highlight security and privacy threats to commercial IoT use cases, including leakage of sensitive user information, including personally identifiable information, as well as unauthorized function execution. Numerous architectural vulnerabilities were discovered, including insecure communications, protocols, and network services. Additionally, the use of Cloud services meant placing trust and control in the hands of a third party. The design, development, and testing of an IoT product or system requires a target environment that closely aligns with real-world use cases. Aligning with IoT architecture, IoT simulators and ranges need to support scenarios consisting of heterogeneous elements, be computationally efficient, and support custom requirements (Sharif and Sadeghi-Niaraki, 2017). Traditional IoT research focuses on large-scale systems (Sánchez et al., 2013) (Teranishi et al., 2015) or security testing of individual commercial IoT devices (Hale et al., 2018). Most ranges are too complex for the individual researcher to quickly configure and run their own educational or research scenarios. Additionally, the size and scope of most IoT ranges make granular control and visibility difficult.

2.1 Review of IoT Testbeds and Ranges

The following is a survey of existing IoT testbeds and ranges with a description of their architecture, purpose, and an indication of missing features that would provide control, flexibility, reproducibility, or visibility.

FIT IoT-Lab: The Very Large Scale IoT Testbed is one of the most robust open source IoT testbeds providing access to the range via the internet. Despite being deployed at six different sites across France, these testbeds are all interconnected and available at the same web portal. Additionally, all testbeds possess common REST interfaces and Command Line Interface tools. The distinct benefit this testbed provides is a platform to test and understand networking

and IoT specific architecture. In contrast to the range, FIT is a cloud-based platform as a service (PaaS) that uses specialized IoT hardware, more complex WSNs, operating systems and does not have a security focus (Adjih et al., 2015).

OpenTestBed (Muñoz et al., 2019), an open-source IoT testbed built with Raspberry Pi and OpenMote B shields, uses a group of 80 motes (IoT Nodes) that emulate traditional IoT devices. Using Wi-Fi and WSNs, these devices simulate real-world deployments through minimal setup. What is particularly alluring about OpenTestBed is that it provides the opportunity to load one's own firmware onto devices and observe behaviour when that firmware is running. When considering infrastructure, set up time, and support servers, traditional testbeds can be quite costly. OpenTestBed acts as a low-cost alternative (Muñoz et al., 2019). OpenTestBed devices communicate over WiFi to a single controller, are self-contained, and require only a power connection for installation. It can be set up within a lab environment and uses Raspberry Pi for each node. OpenTestBed focuses on IoT development using Wireless Sensor Networks and custom firmware. In addition, it does not provide any specialized logging concerning device usage or security.

Jones et al. presented a testbed that incorporates a number of low-powered sensor tags in a mesh network topology (Jones et al., 2018). The network can operate with Wi-Fi and 6LoWPAN. The sensor tags communicate sensor data and intrusion detection system (IDS) alerts to an AWS cloud environment via message queue telemetry transport (MQTT) for processing, storing, and displaying data. The primary use case of the range is testing an edge-based IDS. The authors validated the effectiveness of the testbed by running several denial-of-service attacks and exploring the collected data. While the IDS being used in this case is edge-based, the authors do not discuss exploring the security of cloud or edge-based applications themselves. The data being logged in this case does not include application states and is limited to things like RAM usage, and CPU usage.

Siboni et al. describe their efforts to create a testbed for testing individual IoT devices in (Siboni et al., 2019). The testbed can run multiple security tests on a given device and automatically record the results for researchers to use. The testbed consists of a number of modular components for data collection, repeatable testing, and management. The physical testbed contains an orchestrating machine that communicates with an analysis machine and a control and communication machine located inside a shielded room. The testbed is notably capable of context-based testing where it generates various environmental stimuli such as lighting or audio changes. This testbed however is limited to device testing only and so cannot be used for full network tests or application-level testing.

SecLab is another physical testbed that has a particularly interesting architecture (Schwaiger et al., 2022) . Interchangeable embedded nodes are controlled directly by a USB connection to an agent machine. The agent machine is capable of loading programs onto the embedded nodes and orchestrating when the embedded nodes are active. The researcher playing the role of the attacker in this system can communicate with embedded nodes via REST API calls. This enables a control network that allows much greater control, flexibility, and reproducibility for security tests. This makes simple network level testing much easier but does not account for some components of a complete IoT environment such as interfacing with cloud applications or more robust edge computation.

CyberIoT is a web-based security training platform (Sharifi et al., 2021). CyberIoT allows the creation of scenarios that operate within cloud-based sandboxes. To do this, the platform has a user-friendly web presentation, applications for scenarios and scripting, data collection and monitoring, and infrastructure and sandbox provisioning. The platform allows communication with a public API that sends commands to a cloud controller. Nodes can instantiate new virtual machines on demand and the network controllers are responsible for configuring and allocating VLANs and IP addresses. While this is cost-effective for training purposes, it lacks the ability to test networks with physical IoT devices. Additionally, there are no details about running a scenario as a test of the system, so it is unclear what sort of specific capabilities the range has in practice.

Thom et al. detail building an IoT testbed for both security research and education (Thom et al., 2021). They employ a combination of physical and virtual devices as well as a software defined networking (SDN) segment to create a robust testing environment. They have a miniature smart city component complete with electric trains, street lighting, and realistic traffic signals. There are also several honeypot devices that act as decoys which aid both in developing security skills and conducting research. The SDN segment allows for testing the development of security techniques in an SDN environment. Traffic monitoring and data collection are performed at multiple points in the network. A disadvantage to this system is it cannot be deployed to capture and replay real world data as the network structure is largely in-

flexible. There is also no mention of IoT applications interacting with cloud or edge services.

Nock et al. aim to address the difficulty of cybersecurity training by creating an IoT range with a modular front and back-end (Nock et al., 2020). Their approach supports both virtual devices and a physical network of Zolertia RE-MOTE IoT devices in a wireless mesh network. These devices can be configured to work with a number of sensors, but the authors place no great emphasis on this feature. These two components are coupled by a RESTful API which allows users to ignore the complexity and focus on security training scenarios without limiting the additions that can be made to support new IoT technologies in the back end. They emulate an IoT network using Cooja which users can then build scenarios on top of using Contiki-NG. Users can run red/blue team scenarios and simulate cyberattacks to promote security education. While running experiments is possible, the range is far more oriented toward education and training.

Abu Waraga et al. detail their efforts to create an IoT-based security testbed for running repeatable tests on devices (Abu Waraga et al., 2020). The testbed captures data on a central administrative machine running the Kali Linux operating system that oversees all network traffic. The authors also test devices with mobile applications but data that is collected is limited by the access that closed-source commercial applications provide. The testbed captures network data on application packets, but it cannot be compared with application-level logs in this case. The researchers also provide two sample analyses to validate their testbed using a smart lightbulb and wireless camera. After an extensive manual review of the devices, automated testing is performed. A report is generated, with "vulnerable or "not vulnerable" declarations for each case. While this testbed identifies vulnerabilities within commercial devices, it does not enable the researcher to create their own vulnerabilities or device scenarios. Additionally, the closed-source nature of many commercial applications makes it difficult to explore cloud and application interactions in this context.

IoT Cyber Range (IoT CR) enables the user to outline and work on customizable IoT networks. Virtualized, scalable, and modular - IoT CR can run multiple scenarios at once (Nock et al., 2020). IoT CR uses Contiki-NG, an embedded OS for IoT networked devices, and Cooja, a full network stack emulator designed for Contiki-NG. Contiki-NG enables ease of transfer of OS from the virtualized environment to physical devices. Because the devices are simulated rather than emulated, many virtualized devices can run within a virtualization server because of the low power requirements needed (Nock et al., 2020). However, the virtual nature of the devices eliminates common attacks through wireless protocols. Additionally, the focus is on PaaS without the possibility of a do-it-yourself (DIY) implementation, setup, and experimentation. IoT-CR's focus is a virtual range where security professionals can simulate and teach attack and defense strategies.

Leaf (Ficco and Palmieri, 2019) is an open-source cybersecurity training platform that focuses on IoT edge computing scenarios that closely aligns with real-world production deployments. Tasks are assigned to users or the scenario by organizers. These tasks must be completed so that points can be assigned to users or teams. The data tracked for the scenario includes timelines for events and task completion which allows detailed scenario reviews. While the authors' platform provides tools for provisioning and running scenarios mimicking an enterprise setting, it does not aid in configuring the infrastructure. This approach grants great flexibility, but it likely makes the initial setup of scenarios very cumbersome.

Other full stack simulators, which simulate complete real-world deployments, include DPWSim (Han et al., 2014) (Device Profiles for Web Services), and iFogSim [10] (Fog Computing). Big Data Processing Simulators like IOTSim (Zeng et al., 2017) and SIMIoT (Sotiriadis et al., 2014) focus on cloud computing and data center mechanics. Network simulators including Cooja (Osterlind et al., 2006), used in Nock's Cyber Range (Nock et al., 2020) and CupCarbon (Mehdi et al., 2014), a smart city IoT simulator focus on protocol research within IoT, typically with wireless sensor networks. Many of these are far too complex for individual researchers to completely control their test environment.

3 REQUIREMENTS

This section presents a set of requirements for an IoT range to be developed for the purpose of education, testing, and research from the perspective of the stated goals of control, flexibility, reproducibility, and visibility. The Configurable Cybersecurity Education Range and Testing Stack (ConCERTS) project has requirements that have been distilled from existing range and testbed best-practices and gaps in the review. Table 1 provides a comparison between the features of some existing IoT testbeds and ranges and those identified as requirements for the ConCERTS architecture.

The goal of user control should involve the re-

Table 1: Existing Testbeds/Ranges and Comparison with ConCERTS.

Testbed/Range Requirements	Abu Waraga	fit-iot-lab	FICCO	Jones	Open Testbed	IoT CR	SecLab	Cyber IoT	Siboni	Thom	Con CERTS
Open Source	✓		✓		✓		✓	✓		✓	✓
Composable		✓	✓			✓	✓	✓		✓	✓
Operable	✓	✓	✓	✓	✓	✓	✓		✓		✓
Virtual		✓	✓			✓		✓		✓	✓
Simulated				✓	✓	✓	✓				✓
Commercial IoT	✓								✓	✓	✓
Logging	✓	✓				✓	✓		✓	✓	✓
Capture from commerical devices	✓								✓	✓	✓
Scripting	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
Replay		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Convert logs to scripts											✓

searcher or student in all aspects of the range. As both a security testing tool and educational device, ConCERTS should be accessible, composable, and operable, prompting requirements for open source development and user controlled operations. The user should be given full access to the source code, they should be able to combine devices in any mix they require, and they should be able to control the activity, actions, logging, scripting, and replay for any device and the whole network of devices.

Flexibility in the system is required so that the range can be used for a variety of research and educational purposes over a variety of scenarios, with a variety of devices. Again, having an open source system allows for the flexibility of modifying the code or extending the system architecture. The system should work in virtual environments, physical simulations, physical simulations with real-world sensors, IoT devices under control from the simulation, independent IoT devices, or any combination of these. This can provide flexibility in the range to be used for such varied use cases as a testbed for a new device, for creating data sets, anomaly detection, and learning IoT networking skills. Each device, collection of devices, or controller should have the ability to be configured on how it reacts to events. The ability to easily reconfigure virtual or simulated devices with or without sensors allows users to avoid the costly acquisition of many devices and instead have a combination of virtual, inexpensive reusable simulated devices or commercial IoT devices based on the what the researcher or student has on hand or in their budget.

An important requirement for experimentation and to test learned skills is to ensure reproducible results with the system. The goal of reproducibility entails having logging of events and activity, scripting to simulate events and activity, and the ability to turn logged events and activity into scripts. The user should have the option to write scripts for desired network activity or to capture real-world generated activity to play back. A combination should also be supported. Allowing the system to convert log files from actual activity allows for a physical way to generate scripts that match real-world events instead of just constructed scenarios.

To observe and learn from the range activity, all of the actions must be visible to the researcher or student. The system requires the ability to log all events that trigger system activity and to log the activity as well. The logs need to be accessible and preferably in a common format so that they can be combined for analysis. The system should provide a facility to capture the activity from commercial IoT devices as those devices may not provide access to logs or provide full transparency of events and activities in their logs.

4 PROPOSED ARCHITECTURE

The following architecture has been proposed with the goals of improving control, flexibility, reproducibility, and visibility over the existing systems from Table 1. The intent is to provide an IoT range that can be used for training, experimentation, and research. The requirements for hardware and software indicate that the system needs to intermingle virtual, simulated, and commercial devices running IoT applications and to provide a system to manage the scenarios to be set up and operated in the range. These requirements are best served by the architecture by considering them as independent layers. The scenario, application, and hardware layers as shown in Figure 1.

The scenario layer provides the control over the configuration of the IoT network, the scripting, and gathering logs of the application activity. The design of this layer supports reproducibility using scripting and converting logs to scripts. The flexibility requirement is supported by allowing the user to configure the devices, network, and application scenario. The control is gained through delivering configurations and scripts to devices and starting and stopping scripted scenarios. And the visibility is handled in this layer by gathering the common format logs from each device for analysis as well as having the facility to query each device for its state during script execution.

The scenario layer communicates with the ConCERTS devices in the application layer through the use of a common REST API. There is a basic set of API calls to configure the device, read the configuration, read logs, load a script, run, and stop the script, and query the current state of the device, sensors, and actuators. The scenario layer can store all of this information in a database so that the scenario can easily be run multiple times. This scenario data can be edited to reconfigure the scenario with new or modified devices, and the events can be edited to change the activity that is triggered.

The application layer is where the virtual, simulated, and commercial IoT device's applications operate. Each device has support for generating logs, running scripts, responding to events, and generating activity. Simulated devices can be virtual or simulated on hardware with real-world sensors. For a pure virtual device the device is run entirely in software and may be a virtual machine, container, or just run on any type of device that will support the code base and connectivity requirements. Virtual devices can only be used for scripted scenarios. The scripts can be custom crafted to provide a simulation of real-world activity, or they can be used to re-run scripts that have been generated from log files of real-world activity. The simulated devices are intended to be run on single-board computers (SBC) like a Raspberry Pi. The value of a simulated device on an SBC is the ability to connect the device to sensors, actuators, or dev kits. The sensors can be electronic components that measure conditions like temperature, light, or motion. The actuators are electronic components that activate devices like door locks, electrical motors, or lights. Commercial devices can be included in the ConCERTS devices by wrapping the device in a simulated device proxy using a man-in-the-middle (MITM) scenario where the simulated device captures all activity from the commercial device, logs it, and allows for scripting, replays, and modifications of behaviour. In this manner, real-world activity can be captured at the application layer to be used for scripts at the scenario layer. The application layer is where the device controllers are located. The controllers provide decision making support to the devices. These may be used as bridges between sensors and actuators. For example, if the temperature in a room goes below a predetermined level, the furnace is activated, and if it goes above a certain threshold, the air conditioning is activated. The controllers can be configured from the scenario layer. In this example the predetermined thresholds for heating and air conditioning and what devices to send the command to are part of the configuration for the controller.

The hardware layer is composed of commercial devices, commercial devices where the firmware or software has been modified to support ConCERTS, simulated devices that use a dev kit to customize advanced communication behaviour, and simulated devices that read from sensors or activate actuators. Allowing for hardware devices increases the flexibility of the ConCERTS architecture. ConCERTS can be used for capturing real-world activity from devices situated in a network with genuine interactions with the physical environment and people, or it can be setup to be triggered manually in a scripted scenario where people trigger the sensors in a composed or improvised manner. A commercial device can be encapsulated by ConCERTS devices to provide a simulation that acts as a device proxy for that one device so that to the device the proxy appears as the real-world environment it expects to communicate with. The firmware on physical devices can be extracted from a UART interface or by pulling the firmware chip. The firmware can be modified or extended to support ConCERTS REST API calls or to support a MITM device proxy set up where a simulated ConCERTS device can capture all of the activity and allow the device to participate in ConCERTS scenarios.

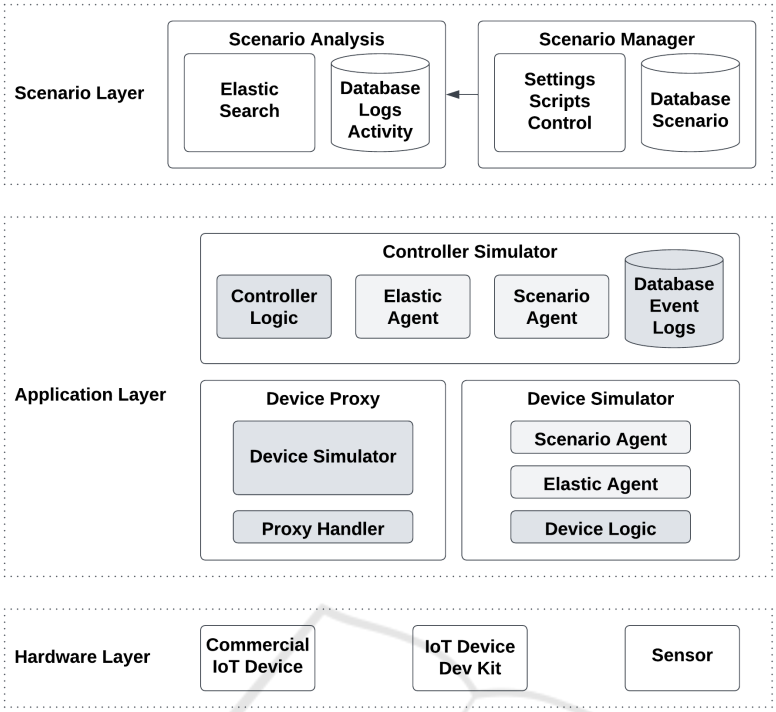

Figure 1: An architectural diagram of the ConCERTS components.

4.1 Architecture Components

This section identifies the various architecture components that exist within the layers as shown in Figure 1. The source code is published on GitHub (McKay and Singh, 2024) https://github.com/CRLTeam/ConCERTS-device.

4.1.1 Scenario Manager

The scenario manager operates at the scenario layer and is responsible for simplifying configuration, scripting, and communication for scenarios. Figure 1 identifies the Settings and Scripts Control as well as the Scenario Database. It is responsible for sending and receiving REST API calls to and from various controllers and devices on the network. These REST API calls serve a variety of purposes.

Device and controller configuration and management are initially handled by the Settings Control of the Scenario Manager. This component provides device and service controllers with the specific information they need to set up nodes on the network. This information includes application code configuration, sensor configuration, logging instructions, and other modules executed by controllers.

The event Scripts Controller is responsible for the unique level of consistency and reproducibility that ConCERTS offers. It communicates with the various devices and controllers in the network and orchestrates events according to its predetermined scripts. It loads the configuration and scripting information from the scenario database allowing scenario designers to easily provision consistent scenarios.

4.1.2 Scenario Analysis

The Elastic Search server is responsible for configuring Elastic agent policies, querying device logs, and aggregating data. Elastic can also be configured to perform intrusion detection and other SIEM functions beyond simple data collection. The datastore is a PostgreSQL database that integrates log and activity data collected from Elastic agents, configuration data and scripts from the scenario manager, and other miscellaneous data the user wishes to collect outside of Elastic.

4.1.3 Controller Simulator

The controller simulator facilitates the development of simple controllers that can support the basic logic to respond to events from devices, make a decision on that event, and send out events to other devices. In one of the demonstration scenarios in this paper, the controller simulates a building's HVAC system. It takes inputs from a thermostat and turns on or off fan, furnace, and AC unit devices.

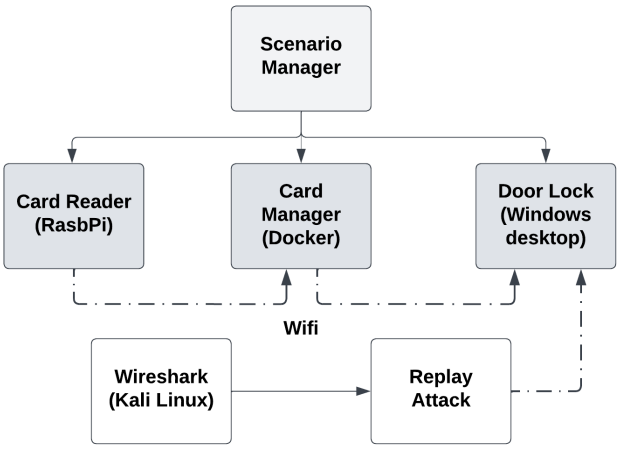

Figure 2: Replay attack device configuration.

4.1.4 Device Simulator

The device simulator is a software application that simulates a simple IoT device in a very small amount of plain code. It can respond to events from a REST call, an event generated in a simulation script, or from a sensor. This makes the device simulator a very simple application to understand and to modify. The device contains the scenario agent that holds the REST calls for the scenario manager to call it, and it has an Elastic agent so that the Elastic server can contact it and retrieve the event logs.

4.1.5 Device Proxy

A commercial device can be encapsulated by a device proxy so that it can participate with the application and scenario layers of ConCERTS. The proxy catches the outgoing signals and can redirect them through the device just like activity from a sensor. Depending on the abilities of a device, it can be passive or it can be directed to trigger events from the scripting agent.

4.2 Architecture Demonstration: Replay Attack

A replay attack on an IoT device is executed by capturing commands that were sent to an IoT device and playing them back maliciously. Using this type of attack provides a simple demonstration of running a security breach within an Enterprise IoT range environment. The demonstration setup uses several ConCERTS components as depicted in Figure 2. The Scenario Manager provides the configuration of the Card Reader and Door Lock devices and the Card Manager controller. It defines the behaviour of each device and identifies them to the controller. The devices each run the Device Simulator. The Card Reader was run on a Raspberry Pi and the Door Lock on a Windows desktop to demonstrate flexibility in the options for device platforms. The Card Manager controller was run in a Docker container on a Linux server as a further demonstration of options. The Wireshark tool was used to capture the network activity and the replay attack was executed using a curl command.

The demonstration makes good use of the Scenario and Application layers. This simulates the activity that may occur at a typical office front door. When a card is read at the Card Reader device, the card's identity number is passed on to the Card Manager controller. The controller checks its database to see if the identity number is allowed to unlock the door. If it is, the Card Manager sends a command to the Door Lock device to unlock the door for a set period of time to allow someone to enter. The card reading events are written as a script that holds the card reading event information. Each event has the card identity number and an amount of time to wait until the running the next event. This can be used to simulate the cards being presented to the card reader over the course of time. The script triggers the card reader to send the card number to the controller that can then command the door lock to open. Only the card reader device needs to run a script in this scenario as the events trigger the controller and the lock device.

In this scenario, the command packets are unencrypted. A separate attacker node was configured on the network using a device operating with a Kali Linux distribution and Wireshark installed. As a training scenario, this is where the trainee would begin the attack. The attacker uses Wireshark to capture network traffic activity. The attacker can look for network packets that contain an HTTP POST. This

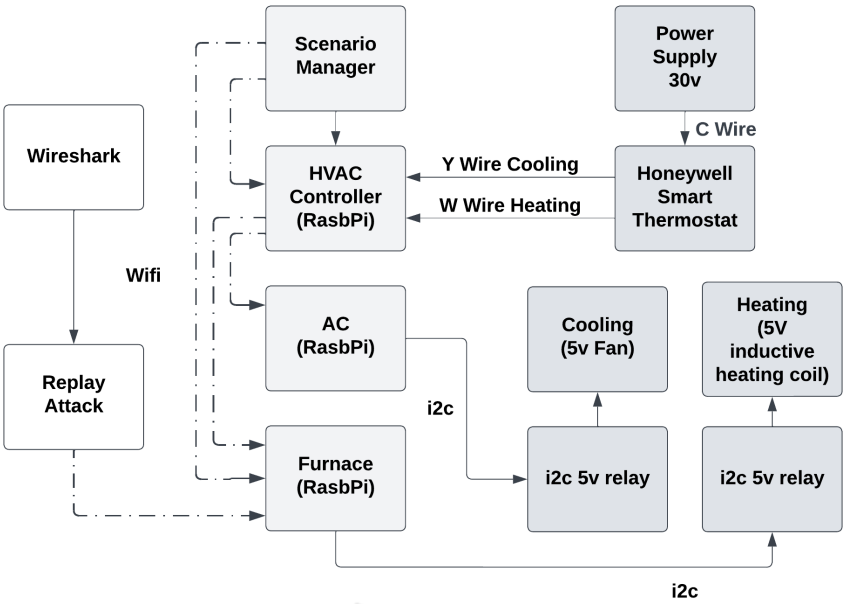

Figure 3: Virtual, simulated with sensors and actuators, and real-world device demonstration.

is a way to spot REST API calls used to communicate with and perform commands related to relevant IoT devices. If the attacker is unsure how the command will present itself, they can simply filter by source or destination device. The attacker then opens a detailed view of the individual packet and can view the full request Uniform Resource Identifier (URI). The attacker now has the URI, including the port required. Finally, the attacker can view the data and variables portion of the command and reconstruct the command. The attacker can then construct and forward the data to the relevant device using a curl command. This command allows the attacker to bypass the card reader device and card manager controller by sending a reconstructed open command directly to the door lock device. All events are logged locally on each device and controller, and upon scenario completion they are aggregated in a central scenario database where they can be reviewed and analyzed.

This demonstration was set up in four steps. In the first step, the card reader, door lock, and card card manager were all installed. The card reader and door lock required installing node.js and cloning the Github repository onto the Raspberry PI and the desktop computer, then they were ready to run. The card manager was installed by running an image of a docker container of the Github repository on a virtual machine running Linux under the ProxMox virtual environment. For the second step, each device was configured by making REST calls to the device's IP address using curl commands. The configuration information contained settings for the card reader to access the card manager, the card manager to access the door lock, and the valid card numbers that will unlock the door. The third step was to create a basic script. For this scenario, the script is a JSON data structure that has an array of card numbers, and how long to wait until the next card swipe. An example would be:

```
{
   script:  [
      {card:1234, wait:120},
      {card:2227, wait:500}
   ]
}
```

This would scan card 1234 then wait 2 minutes and scan card 2227. The script is sent to the card reader as JSON data with a REST call using a curl command. The final step was to use another REST API curl command to start the script. Now the researcher can execute the script as many times as they like to gather the data that they need. This scenario can be set up and run in less than an hour by a student that has basic knowledge in the Linux CLI, and running applications. It is simpler if all of the virtual devices are run in the same way - either all on one computer or on multiple Raspberry Pi devices.

4.3 Architecture Demonstration: Hybrid Architecture

The second demonstration was developed to explore the versatility of the ConCERTS hardware layer by connecting real world sensors and actuators to the simulated devices, and running them in combination with a commercial IoT device that is providing the inputs to a controller. This scenario models a Heat-

ing, Ventilation, and Air Conditioning (HVAC) system that would commonly be found in an office building as depicted in Figure 3. In this demonstration scenario, as the temperature rises on the thermostat above a set value, the thermostat triggers the controller to turn off the heating, and when the temperature rises even higher, turn on the AC. When the temperature drops below a certain value, the controller tells the AC to turn off and as it goes lower it triggers the heater to turn on. To build this demonstration scenario a commercial thermostat IoT device from Honeywell was used to connect to a simulated HVAC controller. The simulated HVAC controller was configured to trigger events for heating and cooling by sending commands to the AC and heating devices. The heating device uses a physical inductive heating coil to heat a piece of metal positioned near the thermostat. The air conditioning device uses a physical fan to blow room temperature air across the piece of metal to cool it down. The trigger events are stored in the logs of each device and the logs are pulled by the scenario manager. The scenario manager can convert the logs to scripts to allow the system to replay the exact events as they happened with or without the hardware connected. This example demonstrates ConCERTS devices using physical actuators and complementing commercial IoT hardware to capture real-world event data from devices and using that data to replicate the exact scenario as many times as required using ConCERTS scripting simulations.

5 DISCUSSION

The driving goals of improved control, flexibility, reproducibility, and visibility has produced a highly flexible tool that can simulate or compliment an IoT cybersecurity research range. The review of existing approaches revealed that many existing ranges and testbeds required technical expertise to use in a meaningful capacity. ConCERTS aims to remedy this with its unique architecture and software stack that relies on high-level programming languages and uniform communication formatting. It should be noted that ConCERTS does not presently intend to perform hardware or firmware level testing and that these low-level details are hidden to enhance usability. Instead, ConCERTS focuses on application, network, and service-level testing and education. The goal for ConCERTS is to engage the IoT cybersecurity community to add more simulated devices, to create data sets along with the scripts to reproduce them, and to create workshops around setting up and running ConCERTS scenarios. To keep this open, ConCERTS has been released as an open source project under the permissive MIT license.

In opting for open source software and configurable hardware, ConCERTS avoids the problem of proprietary applications obfuscating valuable test data and reducing user control. Proprietary software may communicate with cloud services or require several interactions with infrastructure that is difficult to recreate without a tool like the ConCERTS Device Proxy. Scripted events provide a degree of abstraction that allows ConCERTS to simulate complex scenarios without necessarily recreating the entire environment being tested. In general, event-based simulation makes it much easier to consistently recreate scenarios for specific areas of research and education.

In working with students, we have anecdotally shown that in two hours they have the ability to add new device types to the code base and set up hybrid devices that use sensors. The demonstration the students worked on was a motion detector connected to a lighting controller that drove a light. The motion detector was a simple proximity sensor and the light was just an LED with both connected to Raspberry Pis, but with ConCERTS this can be used to capture real-world activity, and turn those logs into scripts. By just waving their hands in front of the proximity detector, the students were able to turn the activity logs into scripts, allow for adjustments of those scripts, and replay the scripts to watch the commands turn the LED light on and off just like when it was manually activated.

Consistent communication methods between devices and controllers and event-based scripting enables a high level of visibility into the scenarios. The controller passes on events to the application-level logic with logs being collected at both levels. This architecture allows a user to aggregate network, application, and event-level data. This means ConCERTS can reliably associate specific events and states that are normally decoupled in a traditional testing environment. For example, cloud applications are usually hidden from the user and cannot be monitored as part of a holistic testing environment. With this ability to view events on the service controller that simulates cloud applications, ConCERTS can gain a more comprehensive view of IoT security and see how actions at various levels affect the entire system.

The demonstration scenarios show how this architecture can be easily used to set up, configure, and run cybersecurity training scenarios. The replay attack scenario only required configuring two devices with simple applications for reading cards and unlocking doors and providing access to an operating system preloaded with security tools. Configuration

was as simple as sending predefined REST API calls to the relevant devices and following instructions for how to perform the attack. It is easy to imagine that further development of IoT applications and services, coupled with a UI, would allow users to simply provision a number of educational scenarios.

6 FUTURE WORK

Future work on ConCERTS will consist of engaging the cybersecurity community to develop more devices, controllers, and features. Development efforts will look to further develop integrated logging capabilities that support analysis. Associating both encrypted and unencrypted network packets with event-level logs would provide the ability to compare application-level behaviour with network behaviour. Additionally, to enhance ease of use, ConCERTS will be looking to utilize Ansible scripts to configure settings that cannot be abstracted at the controller level. This would allow for the installation of additional software outside of the device simulator, perform remote updates, and configure network settings for several devices at the same time. Another future addition would be to support the extraction and modification of firmware from a device to be run on an emulator such as QEMU as part of the device simulation software. This would allow for commercial devices to participate in the ConCERTS network while allowing for testing low-level vulnerabilities. Finally, the intention is to expand the set of example demonstrations to show the full capacity that ConCERTS brings to educational, research, and experimental applications.

7 CONCLUSION

The architectural requirements of ConCERTS was driven by a review of existing approaches and the desire to improve control, flexibility, reproducibility, and visibility. The architecture removes the complexity of studying IoT networks, applications, and services by abstracting away much of the lower-level configuration. As both a physical range and a software stack, it is hoped that ConCERTS can not only be used to further research goals for the authors of this paper, but that others will be able to provision their own networks for testing and educational purposes. The range has the unique advantage of having a low entry cost, a focus on usability, and a novel degree of control over the whole network and scenario. In simulating real-world applications and use cases, the range allows for the necessary applications, datastores, and sensors to represent real devices such as security card door locks, HVAC systems and more. Event-based scripting allows for effortless reproducibility and a unique ability to associate and aggregate logging data. ConCERTS offers researchers the opportunity to make, hack, and break devices and IoT systems with full control over configuration and data. As IoT security research presses onward, it is crucial that tools like ConCERTS provide researchers the right tools for the job.

ACKNOWLEDGEMENTS

The authors express their sincere gratitude to the anonymous reviewers for their insightful and detailed feedback. This research was supported in part by the Natural Sciences and Engineering Research Council (NSERC RGPIN-2019-06150] of Canada, the Canada Research Chairs program (CRC-2021-09-01), the Ontario Ministry of Colleges and Universities Small Infrastructure Fund (SIF 40704), and the Canada Foundation for Innovation's John R. Evans Leaders Fund (CFI JELF 40704).

REFERENCES

Abu Waraga, O., Bettayeb, M., Nasir, Q., and Abu Talib, M. (2020). Design and implementation of automated iot security testbed. *Computers & Security*, 88:101648.

Adjih, C., Baccelli, E., Fleury, E., Harter, G., Mitton, N., Noel, T., Pissard-Gibollet, R., Saint-Marcel, F., Schreiner, G., Vandaele, J., and Watteyne, T. (2015). Fit iot-lab: A large scale open experimental iot testbed. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 459–464.

Albataineh, A. and Alsmadi, I. (2019). Iot and the risk of internet exposure: Risk assessment using shodan queries. In *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, pages 1–5.

De Donno, M., Dragoni, N., Giaretta, A., and Spognardi, A. (2018). Ddos-capable iot malwares: Comparative analysis and mirai investigation. *Security and Communication Networks*, 2018:1–30.

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.

Franceschi-Bicchierai, L. (2016). Hackers make the first-ever ransomware for smart thermostats.

Hale, M., Lotfy, K., Gamble, R., Walter, C., and Lin, J. (2018). Developing a platform to evaluate and assess the security of wearable devices. *Digital Communications and Networks*, 5.

Hamad, S. A., Zhang, W. E., Sheng, Q. Z., and Nepal, S. (2019). Iot device identification via network-flow based fingerprinting and learning.

Han, S. N., Lee, G. M., Crespi, N., Heo, K., Van Luong, N., Brut, M., and Gatellier, P. (2014). Dpwsim: A simulation toolkit for iot applications using devices profile for web services. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 544–547.

Jones, T., Dali, A., Rao, M. R., Biradar, N., Madassery, J., and Liu, K. (2018). Towards a layered and secure internet-of-things testbed via hybrid mesh. In *2018 IEEE International Congress on Internet of Things (ICIOT)*, pages 17–24. IEEE.

Khoury, R., Vignau, B., Hallé, S., Hamou-Lhadj, A., and Razgallah, A. (2021). An analysis of the use of cves by iot malware.

Kolias, C., Stavrou, A., Voas, J., Bojanova, I., and Kuhn, R. (2016). Learning internet-of-things security "hands-on". *IEEE Security Privacy*, 14(1):37–46.

Lodge, D. (2016). Steal your wi-fi key from your doorbell? iot wtf!

Maxxz, J. (2016). Backdooring the frontdoor hacking a "perfectly secure" smart lock.

McKay, D. and Singh, A. (2024). Concerts-device github repository.

Mehdi, K., Lounis, M., Bounceur, A., and Kechadi, T. (2014). Cupcarbon: A multi-agent and discrete event wireless sensor network design and simulation tool.

Modarresi, A. and Symons, J. (2020). Technological heterogeneity and path diversity in smart home resilience: A simulation approach.

Muñoz, 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. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 467–471.

Nock, O., Starkey, J., and Angelopoulos, C. M. (2020). Addressing the security gap in iot: Towards an iot cyber range. *Sensors*, 20(18).

Osterlind, F., Dunkels, A., Eriksson, J., Finne, N., and Voigt, T. (2006). Cross-level sensor network simulation with cooja. In *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, pages 641–648.

Schwaiger, P., Simopoulos, D., and Wolf, A. (2022). Automated iot security testing with seclab. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pages 1–6. IEEE.

Sharif, M. and Sadeghi-Niaraki, A. (2017). Ubiquitous sensor network simulation and emulation environments: A survey. *Journal of Network and Computer Applications*, 93:150–181.

Sharifi, A. Z., Vanijja, V., Pal, D., and Anantasabkit, W. (2021). Cyberiot: An initial conceptualization of a web-based cyber range for iot. In *2021 International Conference on Computational Performance Evaluation (ComPE)*, pages 091–096. IEEE.

Siboni, S., Sachidananda, V., Meidan, Y., Bohadana, M., Mathov, Y., Bhairav, S., Shabtai, A., and Elovici, Y. (2019). Security testbed for internet-of-things devices. *IEEE transactions on reliability*, 68(1):23–44.

Sotiriadis, S., Bessis, N., Asimakopoulou, E., and Mustafee, N. (2014). Towards simulating the internet of things. In *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, pages 444–448.

Sánchez, L., Muñoz, L., Galache, J., Sotres, P., Santana, J., Gutierrez, V., Ramdhany, R., Gluhak, A., Krco, S., Theodoridis, E., and Pfisterer, D. (2013). Smartsantander: Iot experimentation over a smart city testbed. *Computer Networks*.

Tawalbeh, L., Muheidat, F., Tawalbeh, M., and Quwaider, M. (2020). IoT privacy and security: Challenges and solutions. 10(12):4102.

Teranishi, Y., Saito, Y., Murono, S., and Nishinaga, N. (2015). Jose: An open testbed for field trials of large-scale iot services. 62:151–159.

Thom, J., Das, T., Shrestha, B., Sengupta, S., and Arslan, E. (2021). Casting a wide net: An internet of things testbed for cybersecurity education and research. In *2021 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)*, pages 1–8. IEEE.

Xenofontos, C., Zografopoulos, I., Konstantinou, C., Jolfaei, A., Khan, M. K., and Choo, K.-K. R. (2022). Consumer, commercial, and industrial iot (in)security: Attack taxonomy and case studies. *IEEE Internet of Things Journal*, 9(1):199–221.

Zeng, X., Garg, S. K., Strazdins, P., Jayaraman, P. P., Georgakopoulos, D., and Ranjan, R. (2017). Iotsim: A simulator for analysing iot applications. *Journal of Systems Architecture*, 72:93–107. Design Automation for Embedded Ubiquitous Computing Systems.