Casting a Wide Net: An Internet of Things Testbed for Cybersecurity Education and Research

Jay Thom, Tapadhir Das, Bibek Shrestha, Shamik Sengupta, Engin Arslan
Department of Computer Science and Engineering, University of Nevada, Reno, USA
1664 N. Virginia street m/s 0171 Reno, NV 89557 775-784-6905
Email: jthom@unr.edu, tapadhird@nevada.unr.edu, bibek.shrestha@nevada.unr.edu,
ssengupta@unr.edu, earslan@unr.edu

***Abstract*—The last few years have seen an explosive growth in the Internet of Things (IoT) as billions of new low-powered devices are connected to the network in every sector, from household items to sensors, healthcare devices, and industrial controls. Devices often rely on *vertical* architectures, with each type of device requiring a separate platform for control, data sharing, and storage. New *horizontal* architectures are needed to allow sharing of resources between devices to optimize hardware efficiency. In addition, new networking paradigms such as Software Defined Networking (SDN) are required to efficiently manage increased delay-sensitive network demand. This broad range of new technologies and skills make entry difficult for students and new researchers, and innovative practical testbeds for IoT systems and SDN will be required for training and research. We propose an IoT testbed with multiple networking layers and heterogeneous devices to simultaneously support networking research, anomaly detection, and security principles applied specifically to IoT for education and research, providing a complex yet practical hands-on environment made entirely of open-source tools and Commercial Off-The-Shelf (COTS) materials that can be replicated for use by others seeking to build such a system.**

***Index Terms*—Iot, IIot, Iot Testbeds, Software Defined Networks, IoT Security, Cybersecurity.**

I. Introduction

Around the turn of the 21st century, the idea of an *Internet of Things* (IoT) began to emerge as a concept; a vision of billions of devices such as low-powered sensors, cameras, watches, household devices, cars and even airplanes all connected simultaneously to the Internet and able to communicate and share data with one another. Since that early vision IoT has continued to expand and is considered, by some, to be the next industrial revolution. Today IoT devices are everywhere, effectively surrounding us in every aspect of our lives. It has become apparent that this trend is here to stay, with reports indicating that there are as many as 22 billion IoT devices operating worldwide, and we can expect as many as 38.6 billion by the end of 2025 and over 50 billion by 2030 [1]. As pervasive as this technology has become, it is still under development and is changing the way we view many aspects of technology, including networking and communications, data processing and sharing, and power consumption.

One of the areas IoT is finding wide acceptance is in household devices. The concept of a *smart home* has become commonplace, seamlessly integrating multiple household systems. Features such as temperature controls, security, access controls, lighting, entertainment, and appliances are network connected and can be remotely managed. These heterogeneous systems are linked to their respective control centers using wireless technologies, but still utilize a largely disjoint architecture. This means each set of systems utilize their own connections to control and data storage servers rather than working together. Future growth will require more *horizontal* coordination, with systems sharing control plane resources to improve resource efficiency and facilitate data sharing.

Other areas of growth include the Industrial Internet of Things (IIoT) as manufacturing, healthcare, and military applications are becoming more distributed. In addition, the concept of *Smart Cities* has gained traction, with increased coordination in public safety systems, transportation, and traffic controls in an effort to accommodate a growing population. A *smart power grid* has also been largely implemented, synchronizing power production and utilization for more effective delivery. The growth of these technologies raise many questions about IoT devices, their implementation, networking, data storage, control, and system security. In addition, rapidly evolving multi-faceted technologies make it difficult for students and new researchers to gain necessary experience and engage in research and development.

Many of today's IoT systems rely on specialized platforms and application domains, and could greatly benefit from a more integrated architecture. To address these concerns, IoT testbeds must be built to allow hands-on experience and the development of expertise in fields such as programming, networking, circuits, and micro-controllers [5]. Although there exists IoT testbeds to address some of these requirements, they are often very expensive to build and aim to address a subset of desired functionalities, demanding low-cost environments that provide multi-use capabilities combining research, development, testing, and education in a single system are still needed. This work aims to address this critical gap by developing a low-cost, flexible, and realistic IoT testbed that will significantly lower the learning curve for students and new researchers to participate in IoT research.

II. Related Work

A number of testbeds have been offered recently to facilitate IoT research and development. They are presented here by category:

A. Home IoT Testbeds

In [2], Yamin et. al. introduce *build it, break it, fix it* philosophy, where students are expected to design and construct an automated home IoT environment with an emphasis on secure design principles. An exercise is held during a two-day boot camp where students are separated into two groups, each constructing its own secure environment. The groups are tasked with attacking the opposing teams' environment to identify security weaknesses. Several constraints for the proposed environment are given, such as mandatory door locks, a simple user interface, cloud data storage, security alerts, etc. Pre- and post-event surveys are given to reinforce the principles learned. This type of approach is beneficial in that students play dual roles of *maker* and *breaker*, which is not normally provided by cybersecurity exercises.

Another approach to teaching home IoT systems is presented in [3], where a *smart home* environment consisting of a security system, air conditioning system, entertainment system, lighting system, and smart appliances is provided. The environment is constructed using a miniature doll house fitted with devices and sensors connected through a home controller which in turn communicates with a central management system via web sockets. The smart home server communicates in the same way with other clients such as a smart phone, a smart home assistant, etc. simulating a realistic environment for students to interact with.

The development of a *multi-dimensional* IoT testbed and associated challenges are described in [4]. The construction of a realistic and controlled environment is detailed, which supports multiple communication protocols, data processing, gateway operations, cloud integration, node deployment, and security concerns. This platform allows for interactive teaching of IoT concepts and practice-based education using Commercial Off-The-Shelf (COTS) components.

A free and remotely-accessible platform *AssIUT* is described in [5]. Unlike other testbeds that involve hundreds to thousands of IoT devices and servers to conduct advanced research experiments, AssIUT is built to accommodate students and novice researchers. A system of *layers* is defined, separating IoT concepts into groups for more effective learning. Layers are defined as *Things layer (Layer 0)* which contains IoT devices to be connected; *Sensor layer (Layer 1)* which involves sensors, actuators, etc.; *Nodes layer (Layer 2)* that includes processing nodes; *Communication layer(Layer 3)* that contains communication modules; and *Cloud layer(Layer 4)* that encapsulates large cloud platforms. Users can remotely log into a control portal and write/compile their own software solutions and upload binaries to an Arduino micro controller on reserved IoT nodes in a Software Defined network. A guide including experimental examples is included to facilitate the process. The AssIUT framework depends on custom hardware devices connected via wireless services (WiFi, LoRa, Zigbee, and 3G/4G Cellular networks) to form a software defined network, which is in turn dependent on cloud services. While very useful, it does not incorporate physical (wired) or virtual Ethernet networks, and lacks the tangible element provided by the testbed proposed in this paper. In addition, our testbed also emphasizes IoT and network security, a feature not present in AssIUT.

B. Security-Specific Testbeds

The design and implementation of an automated IoT security testbed is developed in [6]. The system automatically tests devices for vulnerabilities based on device type. Two basic device-types are described; those that host a website for interaction as in a web camera, and those that advertise their state to a remote server and receive instructions back to make changes, as in a smart light bulb. This categorization is then used to implement an automated device testing and vulnerability detection framework by analyzing communication patterns of devices. The system exposes devices to the Internet, and automatically disable network access when an anomalous communication behavior is detected.

ISAAC [7] is a realistic cyber-physical testbed system designed to facilitate learning and research specifically for IIoT. Recognizing that both complexity and real time interactions in many cyber-physical systems cannot be reflected by simulations alone, ISAAC provides a controlled environment for testing device resiliency to attack. The system is adaptive and re-configurable, and provides both evaluation as well as teaching opportunities.

Some of the problems associated with the heterogeneous nature of IoT are addressed in [8]. The authors develop a security testbed framework capable of evaluating devices of different types by incorporating machine learning techniques to perform standard and advanced security testings, effectively detecting compromised IoT devices.

C. Software Defined Networking (SDN) Testbeds

Software Defined Networking (SDN) is an important technology for IoT ecosystem in that it enables the management of network nodes through programming rather than through traditional methods involving node autonomy and system administration. This potentially provides for greater bandwidth flexibility and management of large IoT systems. Several methods have been explored to integrate SDN for IoT communication. In [9], authors propose SDN as a solution to consolidate disjoint IoT platforms wherein multiple service providers can use the same platform to supply services and share information. This allows for more rapid development of technology and optimizes utilization of resources. Their basic motivation for developing this IoT architecture is to promote the reuse of various resources and to allow the rapid introduction and deployment of new IoT services and applications. Their design principles emphasize layered architecture, openness and programmability, data provisioning and sharing at different levels, and interoperability.

Guo et. al. explain in [10] how SDN introduces a vehicle for raising the level of abstraction for network configuration, enabling the network control plane logic to be decoupled from the network forwarding hardware thus moving the control logic and state to a programmable software component; the controller. However, as networks become large, a single controller becomes a bottleneck as it is unable to manage all network elements. They propose a system in which controllers are layered vertically, with a master controller managing lower level controllers, and allowing a SDN network to become large, opening the door for better services to IoT devices.

D. Education-Oriented Testbeds

Numerous IoT testbeds focus specifically on education. [11] is an extension of aforementioned AssIUT testbed focusing on providing a platform to conduct student competitions. The competitions consist of registration, tutorial, missions tackling, evaluation, and scoring phases. After the announcement of the competitions, a duration of two weeks is given to teams to register in an online form. Tutorial sessions were conducted remotely in the form of video-conference online meetings to help the teams know more about testbed architecture and usage. The tutorial sessions included presentations, program demos, and question sessions from the teams.

Guo et. al. developed an IIoT testbed to allow students and researchers gain experience in security and smart manufacturing related topics with a focus on securing networked industrial systems and collecting, analyzing, and visualizing the machinery data [12]. Its stated purpose is research and education, forcing an emphasis on flexibility and ease of use. The platform is designed to be *friendly* toward the study of IIoT devices and security, and is made up of low-cost off-the-shelf components. However, while it addresses fog computing, it does not provide access to a programmable SDN controller.

As IIoT deals specifically with industrial control systems, security is crucial for safe operation of these infrastructures. Since there is a global shortage of qualified personnel with relevant skills, Celeda et. al. presented KYPO4INDUSTRY, a testbed that is customized specifically for training and education in IIoT systems for beginning and intermediate level computer science students to learn cybersecurity in a simulated industrial environment [13]. The system is constructed using open sources software and re-configurable modules to facilitate hands-on projects in a flipped classroom format.

LICSTER [16] is also a testbed built specifically for training in industrial control systems. Sauer et. al. describe three approaches to building effective IIoT testbeds; virtualized, real-world, and hybrid. Likewise, there are different tasks for which a testbed can be used. For instance, for security scenarios and attacks on Industrial Control Systems (ICS), a real-world testbed which utilizes physical processes is preferred, as it helps students to more fully understand the impact of different attack vectors on a production environment. This type of environment can be very expensive to build as it requires proprietary devices. LICSTER addresses this problem by providing a low-cost simulated environment using open source software and commercial off-the-shelf equipment.

E. Other Testbeds

Other frameworks developed for IoT testbeds include OpenTestBed [14], an open source and open hardware testbed that can be reproduced by anyone wishing to develop their own testbed environment. The authors attempt to share enough detail to allow interested parties to replicate their work, which includes commercially available state-of-the-art devices, and can emulate an environment which is representative of the users specific use case. The system is capable of loading arbitrary binary images on any device, and can send and receive serial bytes between them.

Finally, Raglin et. al. present a conceptual framework they call Smart CCR IoT, an IoT testbed [15]. Their goal is to design a scalable testbed consisting of IoT-based technologies, infrastructure, processes, data gathering, and location-specific information that can emulate a real-time understanding of a physical environment. The testbed is based on Arizona State University's *Blue Light Pole* system, a network of 700 public safety devices spread across the Tempe, Phoenix Downtown, West, and Polytechnic Campuses. The poles provide an emulated IoT network spanning campus, city, and regional scales capable of disseminating data from sensors, cameras, and other smart devices. It provides an environment for experimentation in optimal networking, computing, and storage technologies, along with techniques for edge computing, software defined networks, publish/subscribe data models, and emerging wireless technologies.

III. Contributions

While IoT technologies are on the rise, there is still a wide gap between state-of-the-art technologies and research/training environments accessible for many universities. Network testbeds can be expensive to build, limited in their flexibility, and difficult to set up. In addition, many of the proposed test-beds in the literature address limited areas of focus; for instance, device testing, network research, or education and training. In addition, none of the previous works address IoT, virtual and physical networking, SDN, and security in a single, inexpensive format that can be used for research, testing, and education simultaneously. It is our aim to provide a complex and multi-level network test-bed environment for IoT wherein students and researchers can interact with both virtual and physical devices in a realistic and easily re-configurable setting. On one hand, network and traffic optimization can be performed on a hybrid virtual/physical software defined network, while concurrently generating and collecting traffic for research in anomaly detection utilizing machine learning techniques. The system incorporates both physical and virtual IoT and IIoT devices to provide real-world data, and also hosts a number of honeypot devices. This provides the ability to conduct both research on the development of honeypot stealth techniques, and also allows for attack and defense training. Students are able to interact with and probe the system through access to dedicated gateway devices located throughout the network. The network provides interaction with both physical and virtual devices; some actual targets and some honeypots.

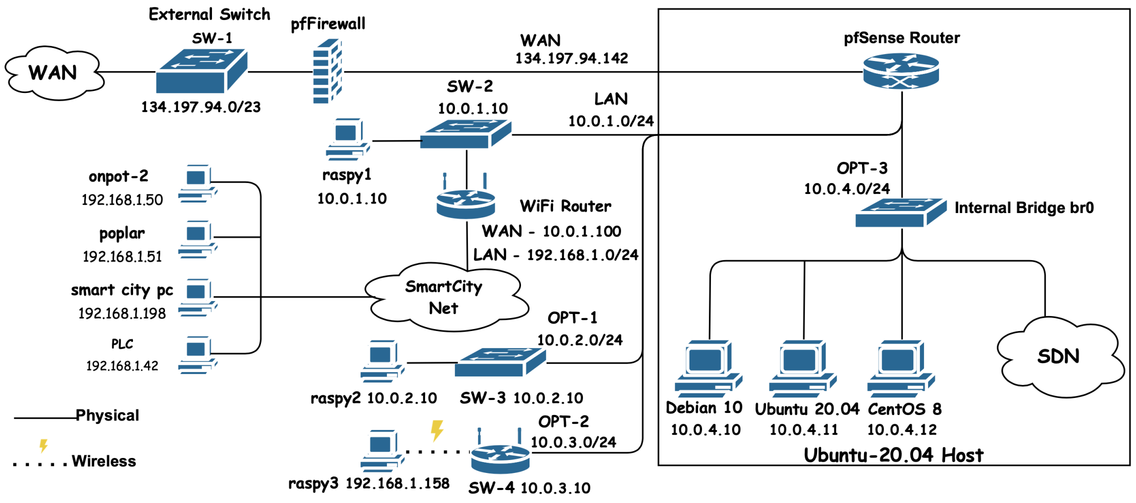


Fig. 1: Hybrid physical/virtual network topology

Students are able to practice security techniques (i.e. scanning, fingerprinting, etc.) across the network, while research on aspects such as honeypot design, IoT communications, and development of SDN networks (i.e. controller design and traffic routing) can be conducted simultaneously.

IV. System Model

To emulate a realistic IoT environment that can simultaneously address research and development, education, and security, a complex environment incorporating multiple network elements is developed. The network contains multiple levels, is re-configurable, and addresses Ethernet, wireless, and software-defined networking architectures. The testbed utilizes both physical and virtual devices on various platforms, and includes a number of open-source tools such as OpenvSwitch, KVM/QEMU, Virt-Manager, Linux Bridge-utils, and several versions of the Linux operating system such as Debian, Ubuntu, CentOS, and Rasbian. A virtual pfSense router is incorporated as a network gateway and firewall, and an OpenDayLight SDN controller using OpenFlow10 manages the software defined network. A *Smart City* model utilizing a Direct Logic programmable logic controller (PLC) is attached the network, allowing students and researchers to interact with mechanical elements via the modbus protocol.

A 24-port managed switch connects the test-bed to a wide area network, which is in turn connected to a server with 5 external Ethernet ports. This host machine supports the virtual router, and utilizes a 4-port Ethernet card with the additional ports passed directly to the router; one for the WAN and three sub nets. In addition, a fourth sub net is connected to a virtual bridge in the host. Within the firewall a *gateway* device is located in each sub net, allowing for internal network access by students to probe the network and gain experience with foot-printing techniques. This also provides students an opportunity to monitor traffic via honeypots, which are distributed throughout the network.

The three physical ports are connected to external network elements through managed switches and wireless routers, while the internal bridge is connected to numerous virtual machines. The internal bridge supports the virtual SDN. A diagram of the SDN can be seen in Fig. 2. The physical network connects several Raspberry Pi hosts with various operating systems which emulate desktop computers, IoT devices, and honeypots. In addition, a *Smart City* model with a Programmable Logic Controller (PLC) is connected on its own sub-net, see Fig. 3. A diagram of the network topology can be seen in Fig. 1.

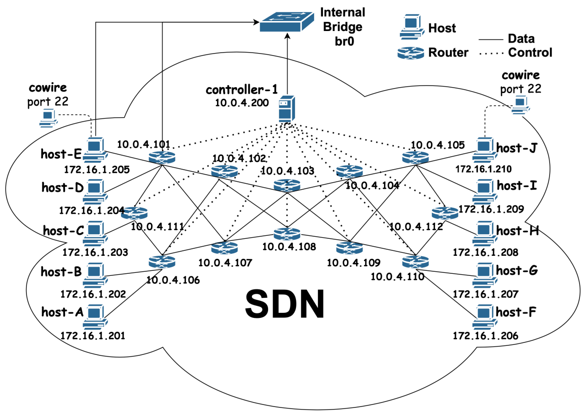


Fig. 2: Topology of the virtual Software Defined Network.

A. Smart City Controller

To emulate realistic industrial controls, the *Smart City* component of the test-bed relies on a Direct Logic PLC which communicates over the network using the *modbus* protocol. The PLC is made up of four modules; networking, digital input, digital output, and an analog I/O module. To offer a visual/tangible aspect the PLC controls an electric train, crossing guards, street lighting, realistic traffic signals, and a simulated nuclear power plant. The power plant has a heating unit and temperature set-points managed by a command and control server, and utilizes a cooling fan and smoke generator. All functions are accessible through the network, giving students an opportunity to view functionality remotely, and to test their skills at both hacking and defending the infrastructure by sending modbus commands directly to the smart city and altering its behavior. A graphic user interface and programming tools are hosted on the command and control server. The sub net hosting the Smart City also contains decoy honeypots (Conpot) emulating similar protocols for both masking the city, and for collecting attack data from the network.

B. Honeypot Devices

One of the goals of the IoT network is to facilitate research and education in cybersecurity. For this reason numerous honeypots of varying types are run in the network to act as decoy devices, and to collect data on attacker activity. Students are given access to both the honeypots and the gateway device in the network, and can develop their skills both at scanning and attempting to gain access to devices, as well as learning to monitor the activities of potential attackers. Devices are hosted both virtually on physical and virtual devices in the network, and also as stand-alone devices hosted on Raspberry Pis. Cowrie, Conpot, Dionaea, and custom honeypots are operated to gather SSH attempts, FTP logins, traffic tunneling attempts, attacker behavior, etc. The goal is to teach students both red-team and blue-team skills and to expose them to various security technologies, as well as to conduct active research on honeypot fingerprinting, device masking, and attack analysis.

C. Software Defined Networks

Traditional networks are typically composed of autonomous fixed-function network devices. These devices have hard-wired functionality that doesn't provide enough flexibility for satisfying the requirements of modern networks [19]. These conventional networks have both control and data plane built into the same device. Software Defined Networking (SDN) on the other hand takes a different approach wherein the control and the data planes are decoupled. Networking devices such as switches and routers implement the data plane which is controlled by the centralized control plane software typically known as a controller. Hence, the software based control plane makes decisions on how the network packets are forwarded and the networking devices execute the policy set by the control plane [20]. This approach improves network management, scalability, programmability, agility, and overall performance of modern SDN networks.

With flexibility provided by a SDN, the collection of network information is greatly simplified. Having a central view of the network helps better understand network status and activities. This information can be used to improve the algorithms designed to detect attacks [21]. With agility and fine-grained control provided by the SDN, responding to detected attacks becomes a simpler task. For example, if a botnet command-and-control (C&C) communication is discovered by the detection algorithm, the control plane can install policies in the data plane that can drop packets related to that specific C&C, effectively terminating the existing communication and eliminating the threat. Hence, SDN can provide an active security layer in the network that was not easily realizable in the traditional networking paradigms.

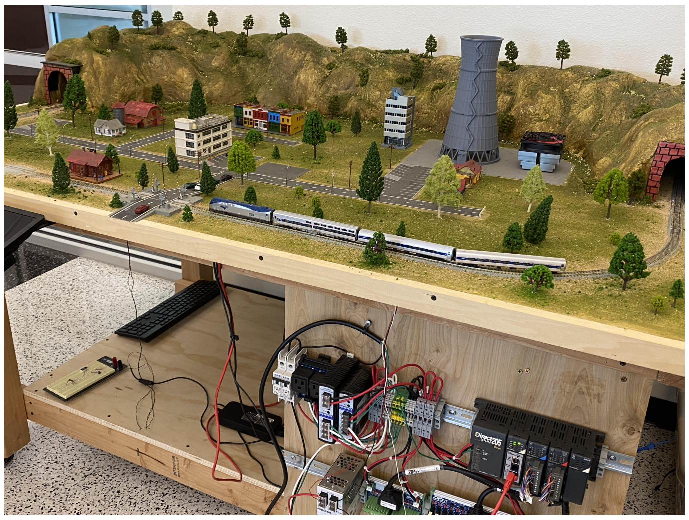


Fig. 3: Smart City Network

IoT devices are much more vulnerable than traditional compute devices. The Mirai botnet, which was responsible for the largest distributed denial of service (DDoS) attack recorded, was able to amass a large number of IoT devices such as IP cameras, routers, printers, DVRs etc. [22]. This demonstrates the presence of critical vulnerabilities in IoT devices which can be exploited. SDN can be utilized to leverage a global view of the network to understand the behavior of IoT devices and and to leverage this to employ attack detection and prevention mechanisms at the network level. Our proposed IoT test-bed caters to the goal of understanding the behavior of IoT devices and build better solutions to actively detect and prevent security attacks using SDN. Fig. 2 shows the SDN network currently running in our test-bed which can be customized according to the simulation requirements.

D. Traffic Analysis and Network Probing

The test-bed provides multiple points where traffic monitoring and data collection can be performed. The use of a virtual environment provides for a simplified means to probe the traffic directly in the device or host as required. A device (either virtual or Raspberry Pi-hosted) is located within each network segment to act as a gateway device which students can be given access to. In this way, they are able to gain a foothold within the network firewall to act as a starting point for network foot-printing. Using scanning tools such as *nmap* or *Zmap*, students can probe the test-bed to discover network topology, device types, etc. and to attempt brute-force attacks or vulnerability exploits on the various elements. Simultaneously, honeypot devices are able to record some of this activity, and analysis can be performed on honeypot logs to learn attacker behavior and to provide research and development opportunities on honeypot technologies. Skills and understanding can be gained for both improved device stealth, as well as skills in device fingerprinting. To test the setup, university cyber-club students can be given access to the network from both an offensive and defensive perspective for exercises in attack and network defense.

V. Simulations

In the following section simulations are performed on the test-bed environment to demonstrate functionality and proof-of-concept. The test-bed is designed to support multiple experiments simultaneously by students and researchers for network analysis, security analysis, and education/training. The following is a detail of such activity.

A. *Traffic and Flow Analysis*

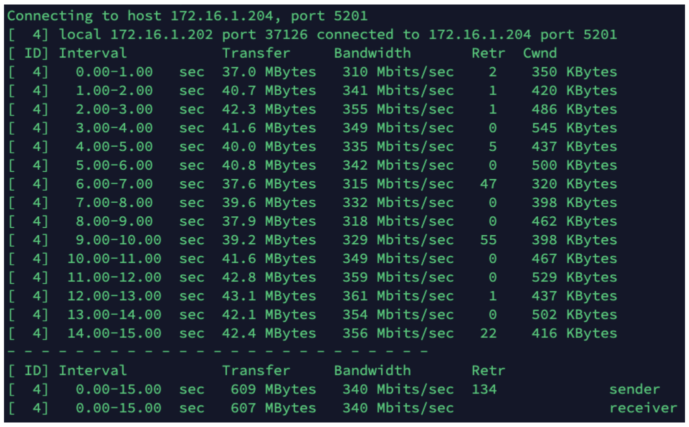
To showcase the functionality of the developed IoT test-bed, IoT traffic and flow information within the software defined network is captured. There are two case scenarios that are analyzed to highlight the operability of the proposed network:

- When network operations are uninterrupted
- When network operations are interrupted

Interruptions can refer to any network traffic that may influence normally accepted operations. This can include multiple traffics flows through the same network device, causing additional congestion and increased resource sharing. Another instance of interruption can be device failure, malfunction, or tampering which can cause variability in network traffic statistics. For our simulations, a focus is placed on the network traffic between Host-B and Host-D.

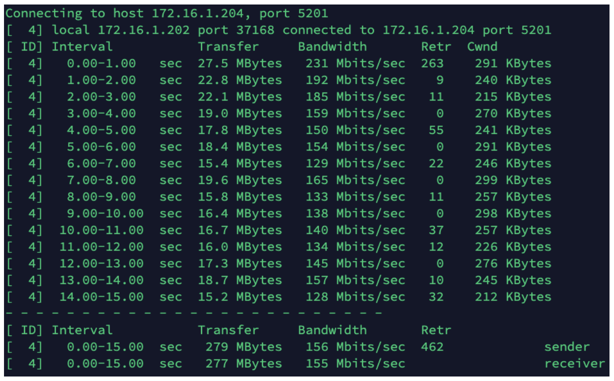
To generate network statistics related to uninterrupted flow, network flow information is captured from Host-B to Host-D. During this time there is no other traffic flow occurring between any other hosts in the entirety of the network. Fig. 4 illustrates the observed uninterrupted network bandwidth, transferred bytes, packet re-tries, and TCP congestion from Host-B to Host-D during a predetermined time interval. It can seen that during uninterrupted network operations, the bandwidth from host-B to host-D remains between 310 - 361 megabits per second. We also observe that the TCP congestion remains consistent with packet loss in the transmission, as TCP congestion window drops whenever there are packet losses.

Along the same lines, network statistics are generated for interrupted flows. To create the interruption, simultaneously transfers are performed between host-A and host-C, while host-B and host-D were also running. Fig. 5 shows the observed interrupted network bandwidth, transferred bytes, packet re-tries, and TCP congestion from host-B to host-D during a certain time interval. It can seen that during interrupted network operations, the bandwidth from host-B to host-D is reduced to between 128 - 231 megabits per second, less than half of uninterrupted network statistics. It can also be observed that TCP congestion remains consistent with the packet loss in the transmission, as TCP congestion window drops whenever there are packet losses.



```
Connecting to host 172.16.1.204, port 5201
[  4] local 172.16.1.202 port 37126 connected to 172.16.1.204 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  37.0 MBytes   310 Mbits/sec    2    350 KBytes
[  4]   1.00-2.00   sec  40.7 MBytes   341 Mbits/sec    1    420 KBytes
[  4]   2.00-3.00   sec  42.3 MBytes   355 Mbits/sec    1    486 KBytes
[  4]   3.00-4.00   sec  41.6 MBytes   349 Mbits/sec    0    545 KBytes
[  4]   4.00-5.00   sec  40.0 MBytes   335 Mbits/sec    5    437 KBytes
[  4]   5.00-6.00   sec  40.8 MBytes   342 Mbits/sec    0    500 KBytes
[  4]   6.00-7.00   sec  37.6 MBytes   315 Mbits/sec   47    320 KBytes
[  4]   7.00-8.00   sec  39.6 MBytes   332 Mbits/sec    0    398 KBytes
[  4]   8.00-9.00   sec  37.9 MBytes   318 Mbits/sec    0    462 KBytes
[  4]   9.00-10.00  sec  39.2 MBytes   329 Mbits/sec   55    398 KBytes
[  4]  10.00-11.00  sec  41.6 MBytes   349 Mbits/sec    0    467 KBytes
[  4]  11.00-12.00  sec  42.8 MBytes   359 Mbits/sec    0    529 KBytes
[  4]  12.00-13.00  sec  43.1 MBytes   361 Mbits/sec    1    437 KBytes
[  4]  13.00-14.00  sec  42.1 MBytes   354 Mbits/sec    0    502 KBytes
[  4]  14.00-15.00  sec  42.4 MBytes   356 Mbits/sec   22    416 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-15.00  sec   609 MBytes   340 Mbits/sec  134             sender
[  4]   0.00-15.00  sec   607 MBytes   340 Mbits/sec                  receiver
```

Fig. 4: Bandwidth Captured from Host B to Host D during uninterrupted operation



```
Connecting to host 172.16.1.204, port 5201
[  4] local 172.16.1.202 port 37168 connected to 172.16.1.204 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  27.5 MBytes   231 Mbits/sec  263    291 KBytes
[  4]   1.00-2.00   sec  22.8 MBytes   192 Mbits/sec    9    240 KBytes
[  4]   2.00-3.00   sec  22.1 MBytes   185 Mbits/sec   11    215 KBytes
[  4]   3.00-4.00   sec  19.0 MBytes   159 Mbits/sec    0    270 KBytes
[  4]   4.00-5.00   sec  17.8 MBytes   150 Mbits/sec   55    241 KBytes
[  4]   5.00-6.00   sec  18.4 MBytes   154 Mbits/sec    0    291 KBytes
[  4]   6.00-7.00   sec  15.4 MBytes   129 Mbits/sec   22    246 KBytes
[  4]   7.00-8.00   sec  19.6 MBytes   165 Mbits/sec    0    299 KBytes
[  4]   8.00-9.00   sec  15.8 MBytes   133 Mbits/sec   11    257 KBytes
[  4]   9.00-10.00  sec  16.4 MBytes   138 Mbits/sec    0    298 KBytes
[  4]  10.00-11.00  sec  16.7 MBytes   140 Mbits/sec   37    257 KBytes
[  4]  11.00-12.00  sec  16.0 MBytes   134 Mbits/sec   12    226 KBytes
[  4]  12.00-13.00  sec  17.3 MBytes   145 Mbits/sec    0    276 KBytes
[  4]  13.00-14.00  sec  18.7 MBytes   157 Mbits/sec   10    245 KBytes
[  4]  14.00-15.00  sec  15.2 MBytes   128 Mbits/sec   32    212 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-15.00  sec   279 MBytes   156 Mbits/sec  462             sender
[  4]   0.00-15.00  sec   277 MBytes   155 Mbits/sec                  receiver
```

Fig. 5: Bandwidth Captured from Host B to Host D during interrupted operation

Two metrics are gathered to measure the network flow performance between the interrupted and uninterrupted flows: throughput and round trip time. To capture the throughput, network traffic is captured and analyzed under the two case scenarios. The results are shown in Fig. 6. It can be seen that introducing the additional flow reduced the throughput below 50% of that of the uninterrupted flow. In fact, the mean throughput measured for the uninterrupted flow was 306.13 Mb/s, while that of the interrupted flow was 122.46 Mb/s.

Another metric that was gathered was the round trip time for the network packets under interrupted and uninterrupted flows. Both simulations were run for a period of 50 seconds. This interval was chosen as it provided an adequate time for generalization of the network activity under both circumstances. The data gathered is illustrated in Fig.7. From this, we noted that there were ≈52,000 network packets transferred during the uninterrupted case, while the packets dropped to ≈23,000 during the interrupted case. This demonstrates that the uninterrupted flow was able to transmit more network packets than the interrupted flow. This is because of the reduced throughput and congestion on the network. Also, it is noted that the magnitude of the round trip times are lower in the uninterrupted circumstance than that of the interrupted.

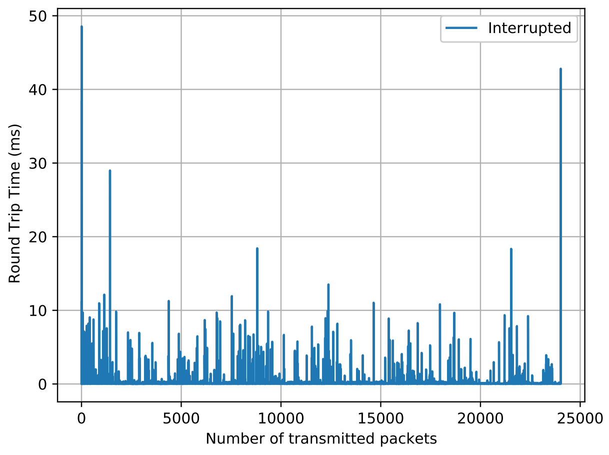
This can also be attributed to the congestion on the network. In fact, the mean round trip time on the uninterrupted flow was measured to be 0.079 ms while that of the interrupted measured 0.127 ms.

B. Network Foot-printing and Honeypot Detection

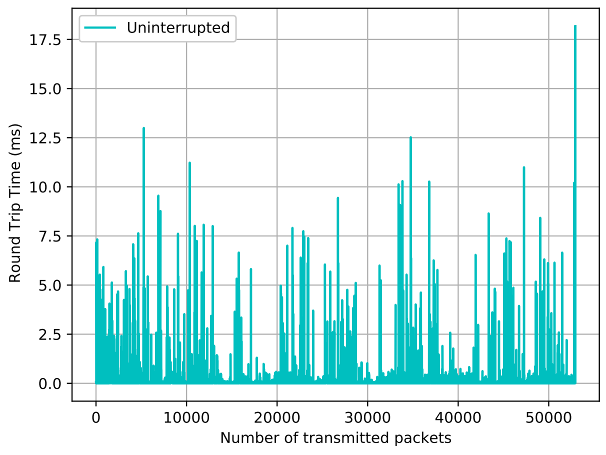
By granting a user (student) access to a gateway machine inside of the test-bed firewall, it is possible to practice techniques for discovering network topology and makeup with the help of scanning tools such as nmap or Zmap and potentially discover vulnerable hosts or services on specific ports. In this case the objective is to identify honeypots that have been placed on the network, namely Cowrie honeypots that will be running SSH and FTP on ports 22 and 23. By running the Zmap command: ***zmap –interface=ens3 -p 22 10.0.4.0/24 –output-file=targets.txt*** or the nmap command: ***nmap -p- 10.0.4.0/24*** a list of hosts can be assembled for further analysis. Honeypots can be fingerprinted in a number of ways by identifying key attributes, in particular those associated with default settings included with the Cowrie installation. One such feature is the key exchange algorithms presented by a host during the SSH handshake. By using the -v flag (for *verbose, or debug* a listing of available algorithms is given to allow hosts to agree on a suitable algorithm. The default algorithms available in Cowrie by default under the *kex-input-ext-info: server-sig-algs* tag are *rsa-sha2-256* and *rsa-sha2-512*. For a non-honeypot SSH installation this list is typically (although not necessarily always) longer, containing several possible key-exchange algorithms. This short list is one possible means of fingerprinting a potential instance of a Cowrie honeypot.

Cowrie can be set to respond to login attempts by reading from a default list of username and password combinations, although in most cases (by default) it is set to accept any combination of username and password after a pre-set random number of attempts. The goal is to give the appearance of security, but ultimately it is desirable to allow an attacker to access the honeypot. Typically an SSH installation will be given a good password, will not allow root logins, and will often block login attempts for a certain amount of time after three failed entries. By brute-forcing suspected instances of Cowrie based on what is returned in the key-exchange stage of the handshake it is possible to successfully log into a honeypot.

Cowrie is capable of logging all activities received, and keeps track of the various attack-command types such as successful and failed login attempts, usernames and passwords for each attempt, file downloads, system commands, etc. By examining these logs it is possible to learn attacker behaviors, and to discover some of the activities an attacker is conducting prior to, during, and after an attack. These logs are stored in .json format and are easily parsed either with user-defined script, or by using tools such as the ELK stack for storing, parsing, and displaying log information. An example of a Cowrie honeypot indicating a failed login attempt can be seen in Fig. 8. In this example the user *www* attempted to log in with the password *123123* unsuccessfully. This is very useful



(a) Interrupted Traffic Flows



(b) Uninterrupted Traffic Flows

Fig. 7: Round Trip Times for Network Traffic Flows

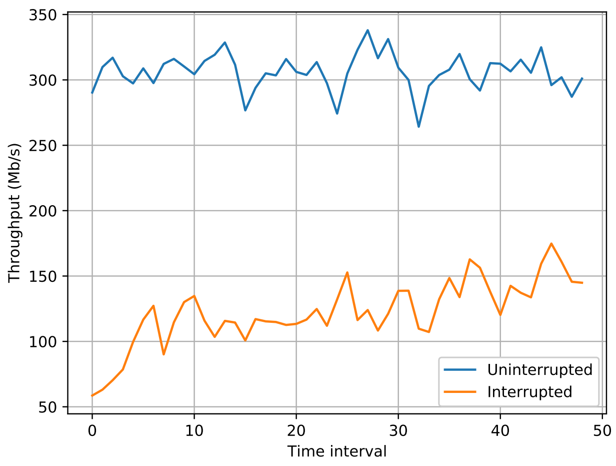
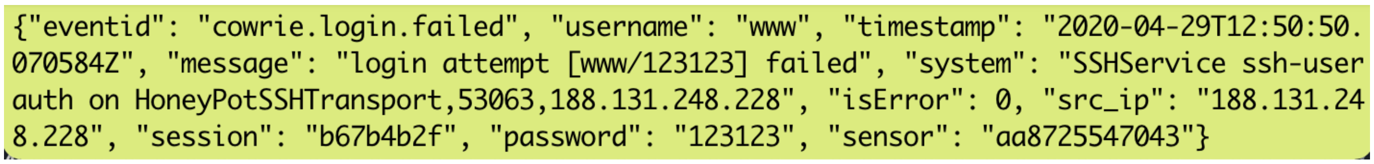


Fig. 6: Measured Throughput between Interrupted and Uninterrupted flows



```
{"eventid": "cowrie.login.failed", "username": "www", "timestamp": "2020-04-29T12:50:50.
070584Z", "message": "login attempt [www/123123] failed", "system": "SSHService ssh-user
auth on HoneyPotSSHTransport,53063,188.131.248.228", "isError": 0, "src_ip": "188.131.24
8.228", "session": "b67b4b2f", "password": "123123", "sensor": "aa8725547043"}
```

Fig. 8: Cowrie .json log segment showing a failed login attempt on a honeypot.

for helping students to understand the makeup of a cyber attack, and to view real-time attacks while they are in progress. In a honeypot exposed to the Internet these attacks often come in a near-continuous stream from bots (and occasionally live attackers) seeking to compromise exposed servers and IoT devices. By making these tools accessible in a testbed environment, they can be used for instruction and training, and can prepare students for deploying and analyzing honeypots and honeypot data in the real world.

A second honeypot deployed in our testbed is the Conpot honeypot, which emulates a number of IIoT devices on ports supporting device-specific services and protocols. Access to these devices allows researchers and students to develop techniques for fingerprinting devices, and also for learning to mask their presence in an effort to learn as much about an attacker as possible before they determine they are dealing with a honeypot and not a *real* device.

VI. Conclusion and Future Work

In this paper a testbed for education and research was presented which utilizes open-source tools and COTS materials to emulate a real-world IoT environment. It is designed to accommodate multiple users performing research and analysis on IoT device networking and security. It can simultaneously serve as a training ground for students learning to understand attacker behavior by scanning and footprinting network environments, identifying honeypot devices through fingerprinting, and using these devices to better understand attacker behavior. It provides a complex multi-layer network topology, a software defined network, and numerous physical and virtual devices emulating both real and decoy machines. In the future additional devices will be added to allow for development of efficient and secure IoT environments, and will help students and new researchers develop the necessary skills to maneuver these new technologies. It can also be expanded to provide access for research to industry partners and the education community to stimulate interest in IoT development and security awareness.

Acknowledgment

This research is supported by the National Science Foundation (NSF) Award #2019164.

References

[1] Steward, J. (2021, April 21). 21+ Internet of Things Statistics, Facts and Trends for 2021. Findstack. https://findstack.com/internet-of-things-statistics/.

[2] Yamin, Muhammad Mudassar, et al. ”Make it and break it: An IoT smart home testbed case study.” Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control. 2018.

[3] Nguyen, Trung, et al. ”A miniature smart home testbed for research and education.” 2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER). IEEE, 2017.

[4] Alsukayti, Ibrahim S. ”A Multidimensional Internet of Things Testbed System: Development and Evaluation.” Wireless Communications and Mobile Computing 2020 (2020).

[5] AbdelHafeez, Mahmoud, and Mohamed AbdelRaheem. ”AssIUT IOT: A remotely accessible testbed for Internet of Things.” 2018 IEEE Global Conference on Internet of Things (GCIoT). IEEE, 2018.

[6] Waraga, Omnia Abu, et al. ”Design and implementation of automated IoT security testbed.” Computers and Security 88 (2020): 101648.

[7] Oyewumi, Ibukun A., et al. ”Isaac: The idaho cps smart grid cybersecurity testbed.” 2019 IEEE Texas Power and Energy Conference (TPEC). IEEE, 2019.

[8] Siboni, Shachar, et al. ”Security testbed for Internet-of-Things devices.” IEEE transactions on reliability 68.1 (2019): 23-44.

[9] Li, Yuhong, et al. ”A SDN-based architecture for horizontal Internet of Things services.” 2016 IEEE International Conference on Communications (ICC). IEEE, 2016.

[10] Guo, Zhengxin, et al. ”An implementation of multi-domain software defined networking.” (2015): 5-5.

[11] AbdelHafeez, Mahmoud, Ali H. Ahmed, and Mohamed AbdelRaheem. ”Design and Operation of a Lightweight Educational Testbed for Internet-of-Things Applications.” IEEE Internet of Things Journal 7.12 (2020): 11446-11459.

[12] Guo, Terry, et al. ”IoT Platform for Engineering Education and Research (IoT PEER)–Applications in Secure and Smart Manufacturing.” 2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI). IEEE, 2018.

[13] Čeleda, Pavel, et al. ”Kypo4industry: A testbed for teaching cybersecurity of industrial control systems.” Proceedings of the 51st ACM Technical Symposium on Computer Science Education. 2020.

[14] Munoz, Jonathan, et al. ”OpenTestBed: Poor Man's IoT Testbed.” IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2019.

[15] Raglin, Adrienne J., et al. ”Smart CCR IoT: Internet of Things Testbed.” 2019 IEEE 5th International Conference on Collaboration and Internet Computing (CIC). IEEE, 2019.

[16] Sauer, Felix, et al. ”LICSTER–A Low-cost ICS Security Testbed for Education and Research.” arXiv preprint arXiv:1910.00303 (2019).

[17] Akbari, Iman. ”SDN part 1: What is Software-defined Networking (SDN) and why should I know about it?” from Technology Networks, Technology Networks, 20 Aug. 2019, www.technologynetworks.com/informatics/articles/a-new-dawn-for-security-vulnerabilities-in-hpc-322974.

[18] Akbari, Iman. “SDN Part 2: Building an SDN Playground on the Cloud Using Open VSwitch and OpenDaylight.” Medium, Medium, 13 June 2019, medium.com/@blackvvine/sdn-part-2-building-an-sdn-playground-on-the-cloud-using-open-vswitch-and-opendaylight-a0e2de029ce1.

[19] Benzekki, Kamal, Abdeslam El Fergougui, and Abdelbaki Elbelrhiti Elalaoui. ”Software-defined networking (SDN): a survey.” Security and communication networks 9.18 (2016): 5803-5833.

[20] Kreutz, Diego, et al. ”Software-defined networking: A comprehensive survey.” Proceedings of the IEEE 103.1 (2014): 14-76.

[21] Bhunia, Suman Sankar, and Mohan Gurusamy. ”Dynamic attack detection and mitigation in IoT using SDN.” 2017 27th International telecommunication networks and applications conference (ITNAC). IEEE, 2017.

[22] Antonakakis, Manos, et al. ”Understanding the mirai botnet.” 26th USENIX security symposium (USENIX Security 17). 2017.