Analysis and Prevention of Security Vulnerabilities in a Smart City

Ben Lupton, Mackenzie Zappe, Jay Thom, Shamik Sengupta, Dave Feil-Seifer
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: benjaminlupton_2023@depauw.edu, mzappe@nevada.unr.edu, jthom@unr.edu, ssengupta@unr.edu, dave@cse.unr.edu

2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC) | 978-1-6654-8303-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/CCWC54503.2022.9720824

***Abstract*—In recent years, there has been a growing interest in so-called *smart cities*. These cities use technology to connect and enhance the lives of their citizens. Smart cities use many Internet of Things (IoT) devices, such as sensors and video cameras, that are interconnected to provide constant feedback and up-to-date information on everything that is happening. Despite the benefits of these cities, they introduce a numerous new vulnerabilities as well. These smart cities are now susceptible to cyber-attacks that aim to "alter, disrupt, deceive, degrade, or destroy computer systems." Through the use of an educational and research-based IoT test-bed with multiple networking layers and heterogeneous devices connected to simultaneously support networking research, anomaly detection, and security principles, we can pinpoint some of these vulnerabilities. This work will contribute potential solutions to these vulnerabilities that can hopefully be replicated in smart cities around the world. Specifically, in the transportation section of our educational smart city several vulnerabilities in the signal lights, street lights, and the cities train network were discovered. To conduct this research two scenarios were developed. These consisted of *inside the network* security and *network perimeter* security. For the latter we were able to find extensive vulnerabilities that would allow an attacker to map the entire smart city sub-network. Solutions to this problem are outlined that utilize an Intrusion Detection System and Port Mirroring. However, while we were able to exploit the city's Programmable Logic Controller (PLC) once inside the network, it was found that due to dated Supervisory Control and Data Acquisition (SCADA) systems, there were almost no solutions to these exploits.**

***Index Terms*—Iot, Iot test-beds, Software Defined Networks, IoT Security, Cybersecurity, PLC, SCADA.**

I.I NTRODUCTION

Smart city technologies are rapidly increasing around the world, allowing for the interconnection and cooperation of multiple devices within a system. The main goal of these technologies are to increase the well-being and resource efficiency of the city's citizens. With everything interconnected, a safer and more efficient environment is made possible. These smart technologies include a wide range of sectors spanning from smart traffic controls, parking, street lighting, public transportation, energy management, water management, waste management, and overall physical security [1]. All the sectors intersect and provide for the betterment of the lives of the cities' residents. Furthermore, these cities will work to protect residents from things like crime, terrorism, and civil unrest. The benefits are clear and have been shown to accomplish these goals when implemented correctly [2]. Benefits of urbanized use of smart cities are an increased efficiency of city planning and development, increased economic opportunities, and service expedition through services such as waste management and energy production [3].

978-1-6654-8303-2/22/$31.00 ©2022 IEEE

Unfortunately, when implementing new smart city devices city officials have often overlooked one of the most crucial pieces of the city's infrastructure: the network integrity of which these technologies reside. Many of these systems are easily manipulable and have become an attractive target for hackers. Recently, there has been an uptick in cyber-attacks on these smart cities [4]. The three primary forms of attack include availability attacks, confidentiality attacks, and integrity attacks. Availability attacks attempt to close or deny service to a system, confidentiality attacks attempt to steal information or surreptitiously monitor activity, and integrity attacks attempt to enter a system to alter information and settings [5]. As a consequence of attempting to make cities as interconnected and safe as possible, smart city developers have also opened a wide range of new security concerns.

Previous research on smart city concerns, has revealed that the most common of these problems are related to human negligence. Primarily, many of these technologies are installed out-of-the-box, meaning the equipment is set up with very basic and easy-to-access levels of security [4]. The default settings mean that even a low-level hacker could exploit vulnerabilities in these smart cities that could cause major disruptions. As prior research explains, these disruptions can range from stealing citizens' personal information to shutting down entire city services. Both disruptions can cause massive financial and physical damages to the city itself. The range of attack consequences demonstrates the magnitude of impact attacks have on the individual and collective citizens of the affected city. Furthermore, the reputation of the city is damaged by the attacks as it reflects poorly on their abilities to protect their citizens' information and livelihoods. An example of this was demonstrated when a hacker was able to shut down Ukraine's entire electrical grid for several hours, leaving a quarter of a million customers without service [1].

One of the most significant vulnerabilities researchers have found in smart cities is in the area of transport management systems [6]. This includes activities like disrupting the flow of traffic or a ransomware attack on ticketing services that can

completely shut the ticketing system down. As an example, it is reported that “the University of Michigan managed to hack and manipulate more than a thousand wire-less-accessible traffic signals in one city using a laptop, custom-software, and a directional radio transmitter” [5]. The research revealed that any system that relied on supervisory control and data acquisition (SCADA) software had vulnerabilities that could be taken advantage of by hackers because almost all commands sent to and from SCADAs are transmitted in plain text. Sending sensitive instructions in plain text is extremely vulnerable as it is easily intercepted and manipulated.

In our research, we utilize the IoT testbed built here at the University of Nevada, Reno, to demonstrate how easily the software and hardware systems supporting these cities can be exploited. With the exposed exploits, we show how the vulnerabilities can be easily be avoided and remedied when implementing correct and appropriate cybersecurity measures.

II. Background

Previous research has placed primary emphasis on exposing the vulnerabilities of smart cities. One of the greatest challenges in conducting research on this topic has been replicating the smart city itself. Attempting to discover vulnerabilities in an actively employed smart city could lead to devastating consequences; therefore, any educational attempt to discover vulnerabilities must be done on a simulated environment. Research of this nature is typically conducted through the utilization of virtual test-beds.

Several simulated test-beds have been proposed in the literature. One of the most well-known and heavily researched test-bed projects is the SmartSantander Project [7]. Its name derived from its orignial location, the Santander test-bed Project is located in the city of Santander, Spain. The Santander test-bed consists of IEEE 802.15.4 devices that are used to replicate wireless personal area networks (WPANs), GPRS modules, and joint RFID tag/QR code labels deployed at various locations in the city. It supports several applications concerning environmental monitoring, precision irrigation, augmented reality, and participatory sensing. Within Santander, the primary research goals relate to the implementation of the different applications; there is less emphasis on network security and the subsequent ramifications of proposed attacks on the network of the smart city.

Another test-bed example comes from Antwerp, Brussels. This test-bed setup, which has been replicated by numerous groups with various changes, allows for experiments on the network level wherein researchers have deployed their network protocols on top of existing nodes. They then evaluate their solutions in a realistic city-wide network [8]. It also facilitates experiments at the data level, allowing for research on the nodes implemented and provides for continual monitoring of the city’s parameters. The Antwerp test-bed allows experiments on the user level providing for input on novel smart city applications.

Finally, there is literature based on test-beds setups most similar to the setup developed at the University of Nevada, Reno. Our test-bed is on a much smaller scale, and is referred to as an "educational, research driven IoT test-bed" [9]. Test-beds of this nature subscribe to the build it, break it, fix it philosophy [9]. Our test-bed serves as a training ground for students who are aspiring to understand attacker behavior by scanning and footprinting network environments. Additionally, the test-bed provides an environment for students to practice identifying honeypot devices. It utilizes open-source tools and commercial off-the-shelf materials to emulate a real-world IoT environment. The smart city test-bed at the University of Nevada, Reno is designed to accommodate multiple users performing research and analysis on IoT device networking and security. It provides a complex multi-layer network topology, a software-defined network, and numerous physical and virtual devices emulating real and decoy machines. While the university’s smart city is not as complex as other test-beds in the literature, it does allow for testing on a smaller scale which could then be translated and tested further on much larger projects similar to those in Antwerp and Santander. A benefit to the smaller scale setup is that it reduces the processing of overwhelming amounts of data that can come with larger-scale test-beds.

From the test-beds presented, there is a general consensus that the vulnerabilities in smart cities are urgent and need to be addressed immediately; most of the vulnerabilities are easily exploitable. Research has found that the security capabilities of IoT devices are highly variable. Some systems are lacking the computational capacity to manage encryption for basic access credentials such as usernames and passwords. Other systems are susceptible to infection from malware and firmware modification [4]. As IoT networks increase in complexity, the risk of exposure proportionally increases. The networks can expose a large attack surface and numerous vulnerabilities. In the Journal of Urban Technology, Kitchin and Dodge discover at least 14 different attack surfaces within the IoT networks, ranging from mobile applications to various device web interfaces [5].

While there is a variety of research on the vulnerabilities of smart cities, there is less concrete literature about potential solutions or patch options. Most research attributes outdated control systems, that contain legacy components, to the smart city vulnerabilities. These legacy components use outdated software which has not been regularly patched [4]. Remedies to common vulnerabilities can be categorized into technical implementations, preparation tactics, and educational resources.

The technical implementations involve five primary approaches. First, it is suggested to use proper access controls such as usernames and passwords that abide to secure standards, two-stage authentication, and biometric identifiers. Next they recommend to properly maintain and place firewalls. Prior research also encourages end-to-end strong encryption. Then virus scanners and removers, generally known as malware checkers, are expected to be implemented. Finally, the literature suggests to establish consistent procedures to ensure routine software patching.

In addition to the technical implementations, cities are recommended to heavily monitor activity and prepare for cyber-attacks. By monitoring consistently, systems would be able to rapidly detect and then eradicate intrusions. Preparing for cyber-attacks is significant because the city cannot afford to be completely offline in the event of an attack. Tactics to execute monitoring and preparation include: responding with urgent updates to close exploits as they occur, auditing trails of usage and changelogs, having effective offsite backups, and establishing emergency recovery plans.

Finally, literature recommends extended education of the professionals working in smart city systems. Education takes the form of consistent and frequent training of cybersecurity awareness. The overall proficiency on topics such as adopting stronger passwords, routinely updating software, encrypting files, and avoiding phishing attacks is strengthened.

Since so many cities use similar network structures and Programmable Logic Controllers (PLCs), this work will address a few of these vulnerabilities [6], specifically in the transportation sector. For our research we employ our educational test-bed to demonstrate these vulnerabilities and strategies on how to eliminate them. By accomplishing this research, it prepares the way for further research to be done with respect to the solutions' scalability towards larger systems, like those in Santander and Antwerp. This research will open the potential for real-world implementation of solutions from the test-bed examples into actual smart city networks that are similar to the test-bed at the University of Nevada, Reno.

III. Methodology

This work utilizes the University of Nevada, Reno smart city test-bed which implements both physical and virtual devices on various platforms. The testbed includes several 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 as seen in Fig. 1. A virtual pfSense router is incorporated as a network gateway and firewall, and an OpenDayLight SDN controller using OpenFlow10 manages the software-defined network [9]. A *SmartCity* model utilizing a *DirectLogic* PLC that is attached to the network (Fig. 2). We are going to break down these methods into four different segments: finding the vulnerabilities on the perimeter of the network, finding vulnerabilities inside the network, finding solutions for the perimeter vulnerabilities, and finding solutions for vulnerabilities that exist inside the network.

A. Vulnerabilities on the Perimeter

To find vulnerabilities on the perimeter of the smart city network, we used nmap to create a map of the network [10]. Nmap allowed us to conduct two types of low-level scans, and also a much more comprehensive scan. To do this we initiated scans from a computer located on a different subnet than that of the smart city. For our specific research, this was the Debian-10 machine (2). The low-level scans provided for a rough overview of the network, including finding where the

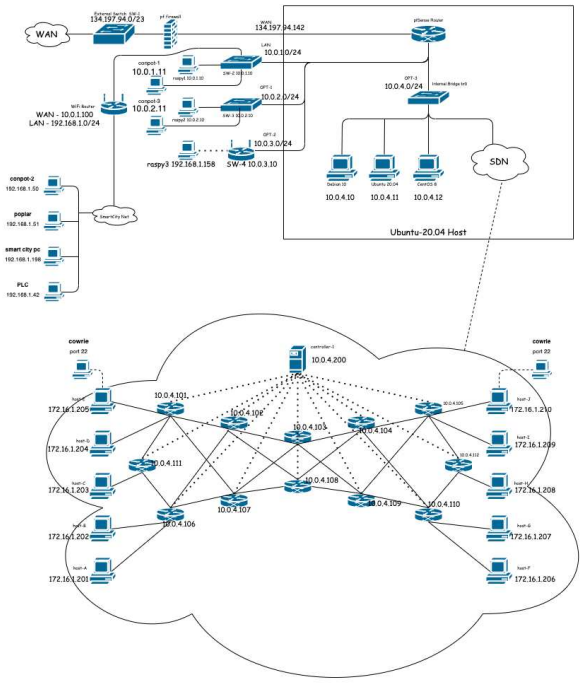

Fig. 1. Physical/Virtual Topology of the UNR IoT Testbed Network

smart city subnet was in relation to the rest of the network. The purpose of the low-level scan was to reveal how many "hosts" or computers were on each subnet and where the routers were on the subnet. An example of the command we would send to conduct these low-level scans was *sudo nmap 192.168.x.x/24*. The more thorough Nmap scans were used later in the research process, after we had established a rough map of the network. We did not initially use the more thorough scans due to the length of time it takes for these scans to run. However, once we found what we hypothesized to be the smart city subnet, we could send the more thorough scan command to extrapolate which ports were open on each machine. The thorough command accomplished this by scanning each of the 65,535 TCP Ports. The port we were searching for specifically was port 503, the Modbus port. An example of the more thorough command was *sudo nmap -p- -sV 192.168.x.x/24* [10].

B. Vulnerabilities Inside the Network

To identify vulnerabilities inside the network, we used a Linux Command Line Utility called "mbpoll" to communicate with the Modbus port [11]. Modbus is a data communications protocol that is used to communicate with PLCs. Modbus has become a standard communication protocol and is now a common way of connecting industrial electronic devices, including our *DirectLogic205* PLC that we are using in our smart city [6]. Once we gain access to this port, we could theoretically control and manipulate any device that is controlled through the PLC. The only requirement is that the device attacking and manipulating this port must be connected to the Smart City

network. In our case,the device could be established through a wired or wireless connection. To exploit the port we connected a simple Dell laptop running Ubuntu to the city's network, and then sent commands directly to the PLC over the network. A successful command is shown in Fig. 3.

C. Solutions for Perimeter Vulnerabilities

To remedy the perimeter vulnerabilities, we had to test a variety of potential options before we found the method that worked. We will cover the intricacies of the testing we went through in the results section and will only cover the successful method in this section. To catch intruders trying to conduct network scans, using software like Nmap, we set up a laptop wired directly into the smart city router and configured it to run the Intrusion Detection System (IDS) called Snort [12] [10]. An IDS was chosen to solve the perimeter vulnerabilities we found surrounding the simple and complex nmap scans, as these vulnerabilities are dependent on network scanning. The IDS would allow a network administrator to vet suspicious traffic as it comes in.

We configured the Snort IDS to capture TCP intrusions on the smart city's subnet. To do this, we ran the following command from the Linux command line: *sudo gedit /etc/snort/snort.conf* [12]. This command opens the Snort configuration window. In the Snort configuration window, we modified the network interface name and the IP range we wanted Snort to be monitoring. In our case, the network interface name was *enp4s0* and the IP range of the smart city subnet was *192.168.1.0/24*. While Snort has a set of preconfigured rules to catch most network intrusions, we added a local rules to assure that we would catch all Nmap scans so they would not pass through as normal network traffic. To do this, we had to modify the Snort local rules using the command *sudo gedit/etc/snort/rules/local.rules* [12]. We added 3 rules following the standard Snort rule configuration of *alert tcp any any -> any (msg:"TCP Scan"; sid:1000001; rev:1;)*. Following this rule syntax, modifications can be made to the rule action, source IP, source port, direction, destination IP, destination port, Snort message, Snort rule ID, revision number, and class type. Once we had the Snort IDS rules configured, we then had to ensure the system would catch all traffic directed at the PLC. To confirm this, we removed the preinstalled operating system from our Linksys AC1900 V2 router that is the router for the smart city network. We then installed the open-source router operating system called OpenWRT. OpenWRT allows for port mirroring to be configured in the interfaces window when accessing 192.168.1.1 in any web browser. Then allow the Enable Mirroring of Incoming and Outgoing setting under he Network tab, then the Switch options. We then configured the router so that the Snort machine would be the receiving machine and the PLC would be the monitored machine. Finally, we turned off the wireless functionality on the smart city router so that the only way to access the network is through a direct, wired access to the router.

D. Solutions for Inside the Network Vulnerabilities

Unfortunately, as we will cover in more detail in the results section, we do not have a concrete solution to the vulnerabilities we discovered inside the network. We will instead cover the temporary methods we put in place until a better solution is discovered. The first thing we did was turn off the wireless functionality on the smart city router so that the only way to access the network is through a direct, wired access to the router. In the case that the smart city is unable to be removed from wireless access, we recommend a very complex router password that only trusted individuals know. Another security method we used for inside the network vulnerabilities was configuring a password on the PLC configuration software. For our *DirectLogic* PLC we configured the password for the "Do-More Designer" software that is used to program the PLC. We recommend this password follow the same requirements as

Fig. 2. University of Nevada, Reno Educational Smart City

```
iot-user@iot-laptop:~$ mbpoll 192.168.1.42 -t 0 -r 5 0
mbpoll 1.4-12 - FieldTalk(tm) Modbus(R) Master Simulator
Copyright © 2015-2019 Pascal JEAN, https://github.com/epsilonrt/mbpoll
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'mbpoll -w' for details.

Protocol configuration: Modbus TCP
Slave configuration...: address = [1]
                        start reference = 5, count = 1
Communication.........: 192.168.1.42, port 502, t/o 1.00 s, poll rate
1000 ms
Data type.............: discrete output (coil)

Written 1 references.

iot-user@iot-laptop:~$ mbpoll 192.168.1.42 -t 0 -r 2 0
mbpoll 1.4-12 - FieldTalk(tm) Modbus(R) Master Simulator
Copyright © 2015-2019 Pascal JEAN, https://github.com/epsilonrt/mbpoll
This program comes with ABSOLUTELY NO WARRANTY.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'mbpoll -w' for details.

Protocol configuration: Modbus TCP
Slave configuration...: address = [1]
                        start reference = 2, count = 1
Communication.........: 192.168.1.42, port 502, t/o 1.00 s, poll rate
1000 ms
Data type.............: discrete output (coil)

Written 1 references.
```

Fig. 3. Example of exploiting the city's PLC through the Modbus port on the Linux command line

those for the router password, both adhering to secure security standards.

IV. Results

The results section will be divided into the same format as the methods section. To determine the method used to find the results presented, please refer to the corresponding methods subsection.

A. Results for Perimeter Vulnerabilities

Using the low-level Nmap scans as described in the methods section, we were able to determine where the smart city subnet was in comparison to the rest of the network [10]. While we already knew how the network was mapped because we implemented the network topology ourselves, we used the Nmap software as if we did not. Then we used the actual network map to verify our findings. Our low-level scans showed us all the "hosts" on the smart city and allowed us to pinpoint the subnet where we should run our more thorough scans. When conducting the scan on the network, the IP address of 192.168.1.42 was the only host machine that had port 503, the Modbus port [6]. Using the more thorough scans we were able to determine that the PLC was the IP address of 192.168.1.42. The map of the network we were able to create derived from the Nmap scans matched exactly that of the network blueprint itself.

B. Results for Inside the Network Vulnerabilities

Once connected to the Smart City router, we were able to send the "mbpoll" command to the Modbus port on the PLC and control almost the entire city [11]. We were able to do this both using a laptop that was connected wirelessly and a RaspberryPi machine that was connected using a Cat5e Ethernet cable to the smart city's router. The "mbpoll" command was used to read Modbus coils, write to Modbus coils, read Modbus registers, and write to Modbus registers. The command syntax for reading/writing to Modbus coils was *mbpoll <host> -t 0 -r <coil number> 0/1* [11]. The command syntax for reading/writing to Modbus registers was *mbpoll <host> -t 4 -r <register number> 0/1/other* [11]. The easiest way to tell the difference between what Modbus coils do and what Modbus registers do is to think of it in terms of binary. For Modbus coils, the change is only to either 0 or 1, or turning a system off or on. However, for Modbus registers, these are values that can be overwritten to something other than 0 or 1. In our smart city, the Modbus coil examples were turning off or on the main power to the city, the train power, the streetlight power, the traffic signal power, or turning off/on each traffic signal light (turning on specifically the red light on the signal light). For our Modbus register examples, were able to do a smaller number of more complicated things, like changing the direction of the train to off, forward, or backward using the values 0/1/2 respectively. Or, we could change the target temperature of the reactor to an integer, which could cause the nuclear reactor to trigger a false overheat. We could also change the temperature limit to an integer. If the temperature went beyond this limit, the main power would turn off. Finally, we could change the traffic signal mode, so that they either acted like normal signal lights, stop signs, or our custom configuration, like having all the signal lights illuminate the green light at the same time. All of these vulnerabilities, if manipulated in an actual smart city by attacking the PLC, could have potentially life-threatening consequences.

C. Results for Perimeter Solutions

Finding the solutions for the vulnerabilities discovered on the perimeter security proved to be the most challenging task. The main issues we encountered were with the technology itself; we will cover those in this section. We will start with the results for the solution that was presented in the method section, that we found to be the most effective. Using the OpenWRT port mirroring feature, our Snort machine was able to capture almost all potentially manipulative network scans. To confirm the network monitoring abilities of the OpenWRT port mirroring function, we used the WireShark software to monitor the actual network traffic compared to what Snort was catching [13]. When we sent the thorough Nmap scans directed toward the PLC or the entire Smart City subnet, we caught all the traffic on WireShark and most of the Nmap TCP pings were caught by Snort [10] [12] [13]. We then were able to block the IP address of the would-be manipulator and eliminate the threat. This type of detection does require monitoring of the Snort machine on a fairly regular basis. In the context of a smart city, we recommend having a trusted IT professional monitoring the city's network traffic.

The issues we encountered in implementing perimeter vulnerabilities came from the Linksys AC1900 V2 router we are using for our smart city's network. We originally wanted to set up a Snort Machine in a Demilitarized Zone (DMZ). Our original thought was that the DMZ would enable access to the Snort machine from an external, untrustworthy network while securing the rest of the network behind a firewall [14]. However, we determined that when configuring a two-firewall system DMZ, it would become unnecessarily complicated; we planned to turn off the wireless functionality of the router anyways to eliminate other vulnerabilities from the research. Instead, we tried to configure the Snort machine so that if any Nmap scanning traffic was directed towards any device on the network, it would be captured [14]. Unfortunately, because we were using a router and not a switch that can have port mirroring configured on it, the only Nmap scans that the Snort machine would catch were scans directed at the Snort machine's IP. Thus, defeating the purpose of catching traffic that was trying to scan the PLC. So, to solve this issue we removed the Linksys preinstalled router operating system. We then installed the OpenWRT operating system. The new router operating system was chosen since it has a port mirroring feature, while continuing to function as a router. Once the router operating system was changed to OpenWRT, we were able to create three unique Snort rules to detect different attempted scans of the PLC.

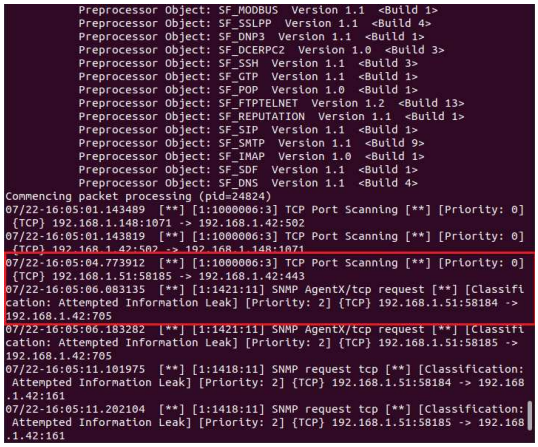

Fig. 4. Example of catching an intruder (192.168.1.51) scanning the PLC (192.168.1.42) using OpenWRT Port Mirroring and and Snort IDS

D. Results for Inside the Network Solutions

Inside the network, we did not accomplish the results that we had originally hypothesized were possible. Using the *DirectLogic 205* PLC, there is a lack of security measures that can be added to it. This is because it, like many other PLCs, are deployed with outdated SCADA systems. These SCADA systems' purpose is to monitor and control devices, like our IoT devices, at remote sites. SCADA systems are necessary because they help maintain efficiency by collecting and processing real-time data [6]. They also allow for real-time manipulation and adjustments so that systems can stay online while updates are sent out to devices. Despite these benefits, they lack security features. The United States Department of Energy has even acknowledged their weakness, stating that "performance, reliability, flexibility and safety of distributed control/SCADA systems are robust, while the security of these systems is often weak" [15]. There is even a specific advisory published about the *DirectLogic* PLCs [16]. In our specific smart city setup, using the corresponding methods section about Inside the Network solutions, we found that was the most secure we could make our PLC. We had hoped to add encryption or even a machine learning Intrusion Protection System (IPS), but our options were limited. We did find that turning off the wireless functionality for the router that the PLC is connected to was a good temporary solution and allowed for strict monitoring of who was on the network. We also found that configuring a password for the "Do-More Designer" software that is used to program the PLC allowed for at least some added security, but the passwords themselves are very basic. The "Do-More Designer" only allows for eight-character numeric passwords, and most of the time, the password is preset to all zeros.

V. Future Research

When comparing the internal vulnerabilities exploited by the "mbpoll" command to the internal vulnerabilities solutions, it is apparent that the solutions were minimal. As examined before, the most secure options for securing these vulnerabilities were to isolate the smart city network or employing strong password etiquette. To isolate the smart city network, the network administrators must turn off or hide the current wireless functionality of the entire smart city. Additionally, network administrators should ensure that any password associated with the PLC software abides by secure standards. Future research can be conducted to further discover and test new solutions for the internal vulnerabilities. The security implications of interconnected smart cities also can be taken into consideration for future research.

A. OpenPLC

In our search for internal vulnerability solutions, we found a software called OpenPLC which is an alternate to the legacy components of the SCADA software based systems. OpenPLC is an "open source" software referring to its available and modifiable capabilities available to public users. Traditional PLC hardware architectures have reserved their documentation which makes it difficult for researchers and educators to completely explore the existing vulnerabilities and test developing solutions to these exploits. Contrary to the traditional PLC hardware architectures, the open source capabilities of OpenPLC would allow researchers to assess network vulnerabilities and test solutions with a hands-on methodology at the hardware level. After extensive additional research, this software could replace the security shortcomings of the outdated SCADA software. The OpenPLC project was created specifically for this purpose [17].

A key functionality embodied in the OpenPLC software is its aptitude for cryptography. The most traditional sense of security stems from well-developed and unique cryptography in place for static and dynamically operating networks [18]. OpenPLC adheres to a AES-256 implementation encryption process. This encryption technique requires that both the sender and receiver of data must have the same secret key in order to gain access to the information, which creates what is known as a symmetric cipher. Because both the sender and receiver of data are required to be able to decrypt information in the same way, there is an additional step needed to allow the PLC system to benefit from the OpenPLC. The additional step requires the OpenPLC project to implement a Localhost Gateway to allow the supervisory software of the PLC to be able to decrypt the data encrypted by OpenPLC. By enabling the OpenPLC Localhost Gateway and further designating the PLC IP address within the supervisory system as "localhost", the supposed unsecured channel between the designated gateway and the main components of the PLC is nonexistent [18]. Thus the system is overall more protected.

B. Interconnected Smart Cities

Another challenge created by the emergence of smart cities is the evaluation of security threats within the scope of multiple smart cities being interconnected. The extensive inter-connectivity of smart cities exponentially increases the systems' endpoint complexity. In order to be completely secure

each additional device to the network has to be operated to the same standard of security as all other devices on the network as "the level of security [is] only guaranteed by the weakest link" [5]. Another consideration for the system of systems approach for interconnected smart cities, is the increased complexity of maintaining the vast quantity of devices. This allows for a single-points of failure in the event of routine program bugs or human mistakes that would have a "cascade effect" [5] on the entire system. The disastrous consequence of this possibility is that the entire system would be wiped out rather than a single segment of the system. Future research can be applied to investigate mitigation techniques of interconnected smart cities.

VI. Conclusion

Using the educational IoT smart city test-bed at the University of Nevada, Reno, we researched vulnerabilities and solutions that will hopefully be used in future research or actual cities. We broke down our research into two different subsections. Vulnerabilities and solutions for Network Perimeter security, and vulnerabilities and solutions for Inside the Network security. Using the Nmap network mapping software, we were able to expose valuable network information [10]. We found where the smart city subnet was located and which device on the subnet would be exploitable. Thankfully, we were able to eliminate most of those vulnerabilities by setting up an Intrusion Detection System that allowed us to isolate the IP address of a would-be hacker and block it from the network. Once inside the network, we were able to show that through the Modbus port on our PLC, we were able to send the "mbpoll" command [11]. This command, if modified correctly, could exploit almost every infrastructure of the city and result in life-threatening consequences. While we were able to find these vulnerabilities, we determined that if a hacker could get inside the network, there is not much that can be done. This is due to the dated SCADA systems used in almost every PLC [6]. Our most successful method of eliminating these vulnerabilities was to turn off or hide the wireless functionality of the router that the PLC is connected to and to make sure a custom password is used to protect the programming software for the PLC.

Future research could involve doing smart city test-bed development with a software called OpenPLC. This software does not use SCADA systems and is instead open source. This means that encryption and a machine learning IPS could be added to it [18]. The only downside is that our city would have to be reprogrammed to work with this new PLC. In terms of increasing security, future research might prove that it is worth the time commitment. Future research might also explore whether cities that are collectively working together against cyber-attacks and to build smart cities like ESPON (European Spatial Planning Observation Network), are more successful against cyber-attacks [19]. It is clear that cyber-attacks targeting smart cities are not going away anytime soon, but working together we can hopefully share how to protect these cities for the good of all.

VII. Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1757929. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] A. Gomez, H. Shahriar, V. Clincy, and A. Shalan, "Hands-on lab on smart city vulnerability exploitation," in *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2020, pp. 1777–1782.

[2] D. Hadden, "Do smart cities improve citizen well-being," vol. 4, pp. 389–413, 2018.

[3] P. S. Ebenezer Okai, xiaohua Feng, "Smart city survey," pp. 1726–1730, 2018.

[4] C. Cerrudo, "An emerging us (and world) threat: Cities wide open to cyber attacks," *Securing Smart Cities*, vol. 17, pp. 137–151, 2015.

[5] R. Kitchin and M. Dodge, "The (in) security of smart cities: Vulnerabilities, risks, mitigation, and prevention," *Journal of Urban Technology*, vol. 26, no. 2, pp. 47–65, 2019.

[6] R. Khatoun and S. Zeadally, "Cybersecurity and privacy solutions in smart cities," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 51–59, 2017.

[7] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis *et al.*, "Smartsantander: Iot experimentation over a smart city testbed," *Computer Networks*, vol. 61, pp. 217–238, 2014.

[8] S. Latre, P. Leroux, T. Coenen, B. Braem, P. Ballon, and P. Demeester, "City of things: An integrated and multi-technology testbed for iot smart city experiments," in *2016 IEEE international smart cities conference (ISC2)*. IEEE, 2016, pp. 1–8.

[9] J. Thom, T. Das, B. Shrestha, S. Sengupta, and E. Arslan, "Casting a wide net: An internet of things testbed for cybersecurity education and research," in *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), 2021*, 2021.

[10] Gordon Lyon. Nmap. [Online]. Available: https://nmap.org/

[11] epsilonRT and Pascal, Jean. mbpoll. [Online]. Available: https://github.com/epsilonrt/mbpoll

[12] Marty Roesch. Snort ids. [Online]. Available: https://www.snort.org/

[13] Gerald Combs. Wireshark. [Online]. Available: https://www.wireshark.org/

[14] Infosec and A. Yadav. (2020) Network design: Firewall, ids/ips. [Online]. Available: https://resources.infosecinstitute.com/topic/network-design-firewall-idsips/

[15] U. DOE, "21 steps to improve cyber security of scada networks," 2002.

[16] U. CISA, "Ics advisory (icsa-12-102-02) koyo ecom modules vulnerabilities," 2018.

[17] T. Alves and T. Morris, "Openplc: An iec 61,131-3 compliant open source industrial controller for cyber security research," vol. 78, pp. 364–379, 2018.

[18] T. Alves, T. Morris, and S.-M. Yoo, "Securing scada applications using openplc with end-to-end encryption," in *Proceedings of the 3rd Annual Industrial Control System Security Workshop*, 2017, pp. 1–6.

[19] M. Batty, K. W. Axhausen, F. Giannotti, A. Pozdnoukhov, A. Bazzani, M. Wachowicz, G. Ouzounis, and Y. Portugali, "Smart cities of the future," *The European Physical Journal Special Topics*, vol. 214, no. 1, pp. 481–518, 2012.