Smart Home Security

A Major Qualifying Project submitted to the faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of requirements for the Degree of Bachelor of Science

Submitted To: Professor Andrew Clark

Submitted By: Noelle Johnson

Submitted On: April 28, 2022

Abstract

This MQP introduces a physics-aware software approach to the safety of Internet Of Things (IoT) systems. The IoT systems have several potential threats due to a lack of industry standards. This project proposes a middleware that is simulated through a python-based simulated smart home air conditioner and controller. The middleware includes a resilient state-estimator and a safety verifier that will ensure the physical safety of cyber-physical systems if control-command or sensor-based threats arise. This middleware can be implemented in a smart home environment as an application on a commodity smart hub.

Table of Contents

Introduction

The Internet of Things (IoT) refers to physical devices that are connected to the internet and can be controlled or monitored by remote users. These devices can range from small personal at-home gadgets to HVAC control in commercial buildings. The applications of IoT devices are almost endless and the industry is growing. The market is projected to grow from USD 381.30 billion in 2021 to USD 1,854.76 billion in 2028 (Fortune Business Insights, 2021). However, with the rise of the IoT industry, comes a rise in security threats to these IoT systems.

The expansion of IoT systems introduces opportunities for security threats in many ways and can threaten the physical safety of IoT devices. The first six months of 2021 have seen a more than 100-percent growth in cyberattacks against IoT devices, with 1.5 billion detected attacks (Seals, 2021). In 2014, hackers utilized spear phishing to gain access to the network of a German steel mill (Cohen, 2021). Once they had access to the network, they were able to reprogram the furnace's programmable logic controllers to improperly shut it down, causing an explosion.

Two potential threats are software vulnerabilities i.e. Data Oriented Programming (DOP), and physical vulnerabilities through the introduction of false sensor data i.e. False Data Injection (FDI). There are many vulnerabilities in the IoT market that lead to these threats. Many of the IoT devices are left without a means for updates or patches. There is a lack of device management and security and protocol standards. Because of the lack of IoT standards, many manufacturers don't focus on security when they create these devices.

One method to overcome such threats is directly monitoring the plant state and switching to a predefined safe mode when the plant reaches an unsafe state due to control software errors. There are two main assumptions of this proposed methodology. First that safety will be ensured by switching between predefined modes to provide safety during different operation conditions. And that the methodology is compatible with existing IoT systems. This paper introduces a python-based controller that communicates with a proposed middleware to develop a software approach for the safe control of IoT systems in the presence of sensor-based and control-command-based cyber attacks.

Related Work

While the IoT business is growing rapidly, a well defined framework and standard for an end-to-end IoT application is not yet available (Hassija et al., 2019). There are several different approaches to the cyber security of IoT devices. One approach monitors program execution for malicious code injection. Orpheus was able to implement this type of program. Other approaches focus on checking the inconsistency of the control commands. The Simplex method monitors a system's current state and if that system reaches unsafe conditions, backup control-commands are given to put the system into a predefined safe mode. ExoSense delivers real-time visual updates about the condition on a connected device and offers condition monitoring, however is only compatible with Exosite's Murano IoT platform. This middleware will differ in that it will be resilient to sensor data attacks, as well as monitoring the safety of the physical plant.

Industrial IoT (IIoT) infrastructure is more complex than consumer-level IoT and requires more elaborate security policies. Rather than having just a few consumer devices, IIoT network connects hundreds to thousands of data points to facilitate operations. "Lack of real-time visibility over connected devices, sensors, endpoints, and their configurations and compliance is perhaps the most persistent security challenge in the IoT landscape" (Koptelov, 2020). This has created the opportunity for cyber-security companies to create programs that will protect these industrial scale cyber-physical systems, such as ExoSense. Most softwares and programs that have been mentioned are aimed more toward the IIoT field which leaves many consumer-level devices such as smart homes vulnerable. There are few cyber security options for smart homes such as UL's cloud-based software program SafeCyber which safety-checks and monitors firmware, but does not guarantee the physical safety of the system. This framework is aimed toward consumer-level devices such as smart homes, but can be applied across several different fields.

Another huge IoT security threat emerged from the Mirai botnet attack. Mirai is a malware that infects smart devices and turns them into a network of remotely controlled bots, called a botnet. Once the attackers gain access to these devices, they are often used to launch a distributed denial-of-service attack by overwhelming the target with a flood of internet traffic. In 2016 the Mirai botnet launched one of the biggest DDoS attacks against high-profile services such as KrebsOnSecurity and Dyn. It's estimated that the attack on KrebsOnSecurity cost device owners upwards of $300,000. Dyn also suffered from this attack by losing roughly 8% of its customers. The high cost of fending off these attacks led to the development of programs that fend off DDoS attacks, such as Content Delivery Networks (CDN) through the management of internet traffic between servers. The proposed middleware focuses on network layer attacks, rather than application layer attacks.

Background

Internet of Things Background

IoT is a network of physical devices. One branch of IoT systems are cyber physical systems (CPS) which incorporate physical devices or plants with IoT capabilities. CPS have four components: the physical device or plant, sensors or smart devices that can manipulate the physical device, the control server, and the user interface which retrieves and visualizes the data to the user. The architecture of CPS and the threats this paper will focus on is shown in figure 1 below. When the user sends a control command, the control server receives it and generates a control input that is sent to the controller which then carries out the control command via the sensors and actuators. Some examples of CPS are pacemakers, smart lighting systems, cars, and energy harvesting systems such as windmills (Taha et al.,2021).

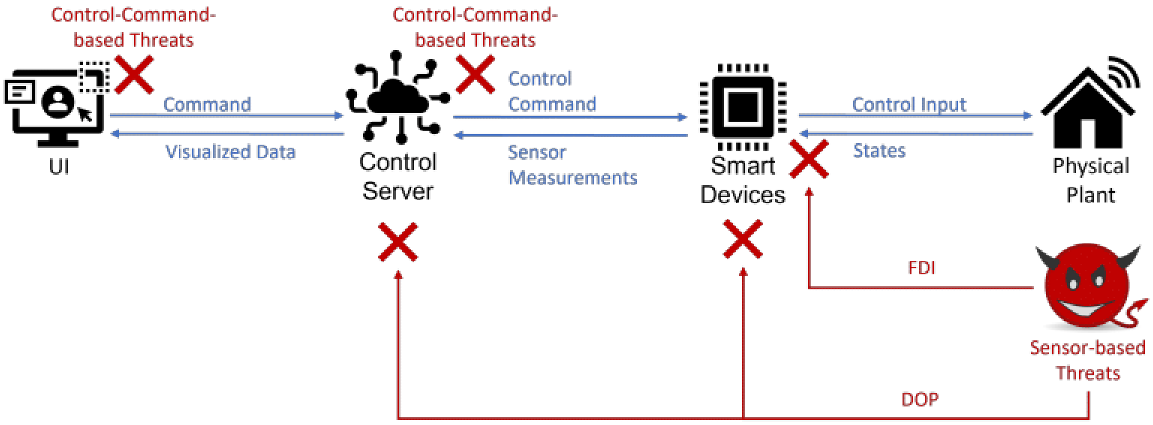

Figure 1: Architecture of IoT Systems and Potential Threats (Li et al., 2022).

Communication protocols are rules that allow the communication of information between two or more entities of a communication system. Most IoT system architecture consists of four main layers: the perception layer, the network layer, the transport layer, and the application layer. The perception layer is responsible for data collection from sensors or actuators, and its transmission. The network layer of IoT systems connects the IoT device to other smart things, network devices, and servers, and it is used for transmitting sensor data. The transport layer is the protocol used to move the data. The application layer is the interface between the network and the IoT device. IoT attacks can be directed at any of these layers.

Threat Model

Although many threats were described above, our approach focuses on attacks that threaten the physical safety of IoT devices and plants. Receiving accurate sensor data is important to produce safe control commands. In the event that sensor data has been spoofed, an unsafe control command could be given based on inaccurate state estimates. False sensor data injection refers to an attack in which the adversary forges or intentionally changes the sensor data being collected to manipulate the output (Sikder et al., 2021). False data can be injected physically or through communication mediums. Gaining physical access to systems can be challenging, so the injection of false sensor data is most commonly implemented through the communication medium.

Command control attacks refers to an attack in which the adversary has access to spoof control commands directly from the user or the control programs. This is usually done by infecting the system with malware. Once the adversary has access to the system, they can change the software of the control program or they can inject malicious code at runtime. These control commands could lead the system to enter an unsafe region.

State Space Model

State Space model is a state-space representation of a physical system with a set of inputs, output, and state variables that are related by first order differential equations. The state variables change over time with respect to the input, and the output depends on the value of the state and input variables. The continuous time form of a state-space model is given by

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

Where ẋ(t) is the differential state vector, x(t) is the state vector, u(t) is the input vector, and y(t) is the output vector, A is the system matrix, B is the input matrix, C is the output matrix, and D is the feed-forward matrix.

Methodology

This section will introduce the methodology for how the system decides whether or not to change the control action of the IoT system, and how it will change it at each time step. Figure 3 below shows the components of the proposed design. The user sends commands to the high level control program, which sends the control command to the middleware.

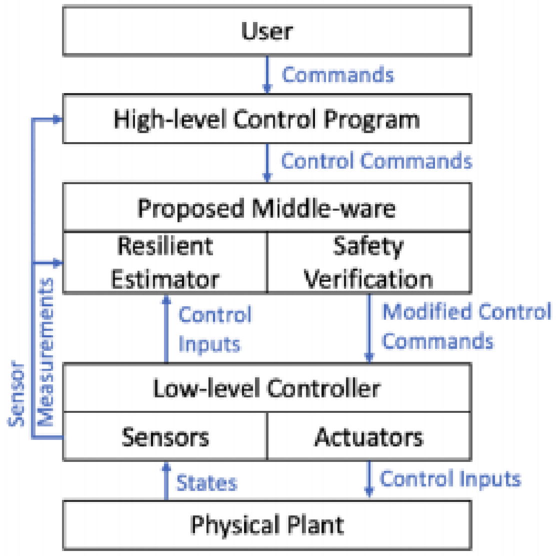

Figure 3: Proposed middleware design (Li et al., 2022).

The middleware consists of two components: the resilient estimator, and the safety verification. The input to the Resilient estimator comes from the controller. The controller collects the sensor measurements and control inputs and transports them to the resilient estimator as input through wired or wireless channels. The resilient estimator uses this input to calculate the state estimates and sends them to the safety verifier. The safety verifier monitors the safety status of the physical plant. It does this by using the state estimates from the resilient estimator and then decides whether or not to modify the commands from the control program in order to guarantee safety.

The middleware will detect unsafe control commands and modify the control command to keep the system in a safe operating region. In the event that there is a control command attack on the system, the middleware will calculate the safety status of the system and modify the control command to keep the system from entering an unsafe region.

In the event of a sensor-based threat, the resilient estimator will be able to maintain accurate estimates based on the uncorrupted sensor measurements. These estimates will then be sent to the safety verifier to decide whether or not to modify the control command if the system approaches the unsafe region.

In addition to the middleware there is the low-level controller and the simulated plant that will be set to regulate the temperature. The low level controller receives the control input from the middleware. The control input is then calculated via a linear quadratic tracking controller or it is set to a discrete mode that will heat the system as quickly as possible.

Evaluation and Results

Evaluation

In this section the middleware, the low level controller, and the virtual plant will be simulated. The simulation will consist of a smart air conditioning system whose diagram is shown in figure 3 below. The control program is written in NodeJS and deployed on a third party cloud and communicates with the SmartThings Cloud. The control program connects to the middleware through ThingSpeak. The middleware itself is written in Python and will be running on the Raspberry Pi 4. The virtual smart device and controller will be a python based program running on a PC. The low level controller and simulated smart device were originally simulated in matlab, however I ported the code from matlab to python in order to reduce the total amount of time that it takes for the program to run. The middleware and the controller will transmit data via TCP/IP protocol.

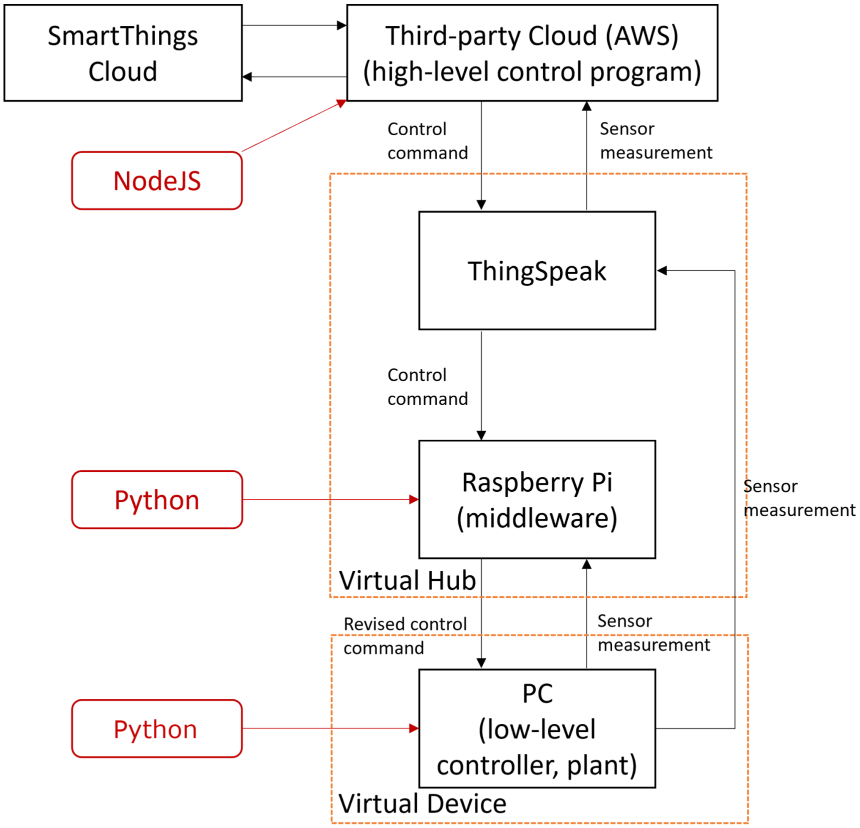

Figure 3: Smart-home air conditioning system diagram.

The linearized equation for the setpoint is defined as $\dot{x}(t) = Ax(t) + Bu(t)$. $\dot{x}$(t) is the system state and represents the deviation from the operating point where the six dimensions represent the deviations of the air temperature leaving the evaporator, the indoor temperature, the air temperature leaving the dry-cooling region of the evaporator, the temperature of the evaporator wall, the moisture content of air leaving the evaporator, and the indoor moisture content. u(t) represents the deviation from the operating point where the two dimensions are the air volumetric flow rate and the compressor speed.

To implement a control-command based attack, the control program will send a setpoint that is within the unsafe region. The sensor-based attack will be implemented via DOP. An attack signal will be added to each sensor, one at a time to increase the state estimate. The incorrect state-estimate will lead the system to start approaching the unsafe region. Once the temperature approaches the unsafe region, the safety verifier will modify the control command to keep it from actually reaching the unsafe region.

Results

After porting the matlab controller and simulated smart-home air conditioner to python, I ran the simulation to confirm that both programs were running the same. I did this by taking out the noise added to the output. After running the simulation 20 times, I found that the python simulation ran for an average of 442.212 seconds with a standard deviation of 5.655 seconds. This is an improvement to the matlab program which had an average runtime of 558.755 with a standard deviation of 10.417 seconds.

The unsafe region was set to any value less than a 7℃ below the setpoint. The control command was set to lower the temperature to 8℃ below the setpoint. As the real state of the system approached -7℃, the control command was modified to heat the system as quickly as possible. The real state did not reach the unsafe region at all during the simulation. With the controller and simulated plant simulated in matlab, the resilient state estimate and the real state overlapped throughout the simulation. With the python-based controller and simulated plant, the resilient state estimate started at 7℃ below the setpoint and did not coincide with the real state until after about 5000 time steps, after which they overlapped. The system ran for about 35000 time steps. Figure 4 shows the impact of both the sensor-based and control-command based attacks.

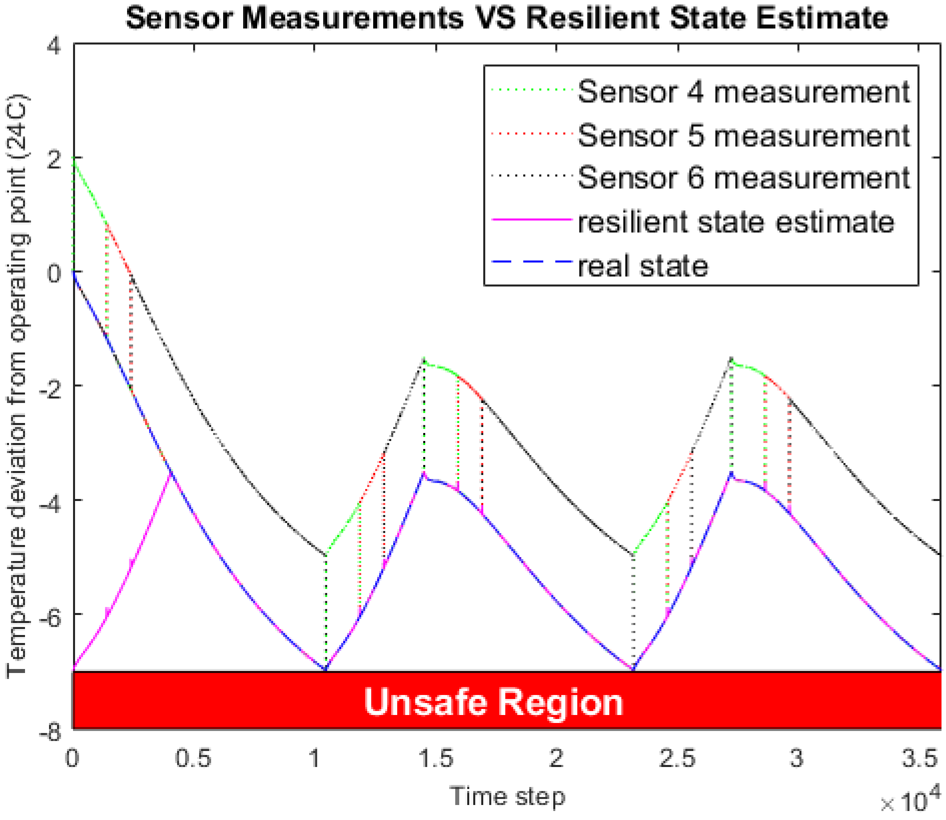

Figure 4: Indoor temperature and resilient state estimates under DOP attacks which target sensor 4, sensor 5, and sensor 6.

Conclusion and Future Work

In this paper, I presented a software-based frame-work for the safe control of IoT devices. The framework includes a middleware between the low-level controller and the control programs. The middleware consists of a resilient state-estimator and a safety verifier. The safety verifier switches between different modes when the system is approaching an unsafe region. The simulation proved that the frame-work provides safety guarantees of IoT systems under the threat of both physical sensor-based attacks and software command-control attacks. This was evaluated through a smart-home air conditioning case study. Further research into this project may include further testing on a real physical system rather than a simulated physical plant to evaluate and confirm that the framework will still work correctly.

References

Cohen, G. (2021, June 10). Throwback attack: A cyberattack causes physical damage at a german steel mill. Retrieved April 14, 2022, from https://www.industrialcybersecuritypulse.com/throwback-attack-a-cyberattack-causes-physical-damage-at-a-german-steel-mill/

Fortune Business Insights. (2021, November 10). *IOT market size, share, growth, trends, business opportunities, IOT companies, statistics, report 2028: Internet of things industry report- fortune business insights*. GlobeNewswire News Room. Retrieved April 21, 2022, from https://www.globenewswire.com/news-release/2021/11/10/2331267/0/en/IoT-Market-Size-Share-Growth-Trends-Business-Opportunities-IoT-Companies-Statistics-Report-2028-Internet-of-Things-Industry-Report-Fortune-Business-Insights.html

Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P., & Sikdar, B. (2019). *A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures*. IEEE Xplore Full-text PDF: Retrieved April 21, 2022, from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8742551

Koptelov, A. (2020, September 29). *Industrial IOT security: Protecting your enterprise*. Itransition. Retrieved April 22, 2022, from https://www.itransition.com/blog/industrial-iot-security

Li, Z., Zhang, H., Clark, A. (2022) *Safe and Resilient Switching Control of Hybrid Systems.* Submitted to IEEE Conference on Decision and Control (CDC).

Seals, T. (2021, September 6). IOT attacks skyrocket, doubling in 6 months. Retrieved April 14, 2022, from https://threatpost.com/iot-attacks-doubling/169224/

Sikder, A. K., Petracca, G., Aksu, H., Jaeger, T., & Uluagac, A. S. (2021, March 8). A survey on sensor-based threats and attacks to smart devices and applications. Retrieved April 14, 2022, from https://ieeexplore.ieee.org/document/9372295

Taha, W.M., Taha, AE.M., Thunberg, J. (2021). What is a Cyber-Physical System?. In: Cyber-Physical Systems: A Model-Based Approach. Springer, Cham. https://doi.org/10.1007/978-3-030-36071-9_1