CincySecure

by

Kent Magnuson, Jacob Wilson, and Jeremy Wilson

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Kent Magnuson 4/9/2019
Kent Magnuson Date

Jacob Wilson 4/9/2019
Jacob Wilson Date

Jeremy Wilson 4/9/2019
Jeremy Wilson Date

Tyler Hopperton 4/9/2019
Professor Tyler Hopperton, Faculty Advisor Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services

April 2019

Prepared by:

Kent Magnuson, Project Manager
Jacob Wilson, Developer
and Jeremy Wilson, Developer

University of Cincinnati
College of Education, Criminal Justice, and Human Services
School of Information Technology
cech.uc.edu/it

April 2019

Table of Contents

Section **Page**

List of Illustrations

Tables

Figures

Abstract

The rise of smart home devices has allowed a cheap alternative to in-home security systems. With the increasing variety of different branded smart devices such as lights, switches, motion sensors, and IP cameras, issues have become prevalent when trying to get the smart devices to work together. Many smart devices require a user to install a specific app based on the brand of smart device. When more than one smart device from different brands is being used this results in many smart device apps with no easy way to manage all smart devices together. Using an open source application called Home Assistant, our product will allow even a non-tech savvy user to easily manage all of their devices within one easy to use application. The user will be able to fill out an online form which will automatically create a Home Assistant instance that is configured precisely to the smart devices they have. This will allow users to quickly and easily set up a cheap in-home security system.

Problem Statement

Introduction

In 2016, the University of Cincinnati Police reported that there were 231 break-ins in the off-campus area ("2016 Campus Crime Report", 2017). With a number that high, it's hard not to justify the need for a security system. These systems can cost a lot of money that college students may not have. However, most college students may already have smart devices that can be used to be a part of a security system.

Problem

Clifton has a long history of break-ins and thefts. Due to landlords cutting corners and an all-around high crime rate, students have struggled to keep their living space secured from the outside world. While implementing advanced security systems such as Brink's and ADT would help, they are not financially feasible for college students. Luckily with the rise of in-home automation tools and smart devices, security has become much more affordable; however, implementing such technology can prove difficult to set up for a non-tech savvy student. The increasing number of smart devices provides many different options for in-home automation but can be difficult to get them to talk to each other with one easy to use application.

Description

Using an open-source application called Home Assistant, we can pull any Internet of Things (IoT) device into one application and allow users to automate their home from their laptop or smartphone. The setup process for Home Assistant can be a daunting task, requiring installation of Python virtual environments, knowledge of how Python works,

and Yet Another Markup Language (YAML) configuration files among others. Our senior design project would change that. This project would create a front-end that a user would go through, selecting all their IoT devices within their home network, and then compile through a build button. Our application would then take that list of IoT devices and other information (IP address, local network info) translate, transform, and export their results, then deploy it to Raspberry PI using a docker image. This would create a simple, easy to use experience for an average user to feel safe at home. While the Home Assistant application was created as a way for users to put all their smart devices together, our project would also manipulate Home Assistant's automation feature to allow for home security. This is done by using any form of motion sensor, Internet Protocol (IP) camera, webcam, or unused iPhone/Android device among a sound system to alert when motion is detected when no one is home. This would also be cost effective due to the end user not having to buy an expensive security system and instead using cheaper smart devices.

Solution

CincySecure users will be able to setup and configure their smart home security system using a web interface from a computer or mobile device. Our system will provide an easy-to-use user experience regardless of prior technical knowledge.

Technical Requirements

Our requirements included having a product that was easy for any non-technical user to use. Since our product is marketed toward all college students and not just technical users, we decided to make the setup process as automated as possible to limit the amount

of required user input. We also wanted to keep the requirements as cheap as possible as other home security solutions can be expensive.

User Profile

Table 1: User Profile provides the overall user with the overall development needed for CincySecure. This includes the potential users, experience with similar applications, task experience, frequency of use, and key interface designs.

Project Title:
CincySecure

Potential Users:
College students, homeowners, and renters in the Cincinnati Area

Experience with Similar Applications:
CincySecure users may have experience with the below similar applications:

- Alexa Smart Home App
- Google Home App
- IFTTT
- SmartThings App
- Flex Wifi
- Kasa
- E-Control
- Smart Home Manager

Task Experience:
The first task will be navigating to the website using a computer or mobile device. The user will click "get started" which will begin with a form that the user will need to fill with what their smart home should be called along with the number of smart devices they own. Once they have completed this page, Home Assistant will build and send the user an email with the URL to their Home Assistant page.

Frequency of Use:
This setup page is intended to only be used once during setup. It is intended that the end user will only need to setup their smart home devices into Home Assistant once. However, if the user wants to add additional devices, they can go through this setup process again.

Key Interface Design Requirements that the Profile Suggests:

- Simple and compatible webUI
- Easy to use for any student or resident of any age
- Easy to follow instructions

Table 1: User Profile

A user profile was created to discuss the settings and configurations that the users of CincySecure will be interacting with. Using **Table 1: User Profile**, we were then able to identify the requirements in our application's design.

Design Requirements

The design of the application followed the requirements of having a simple and compatible Web User Interface (WebUI) that is to be easy to understand and use. Simplicity was the main focus and was achieved with inviting colors along with keeping elements on the WebUI to a minimum not to overwhelm the user. Our profile also identified easy documentation as key interface requirements. To fulfill this requirement, we broke each section of the setup process into numerical steps that were clear and simple to follow.

Use Case

Figure 1: Use Case provides insight on the various actions performed by the user role as well as what events will happen on the back end. The Use Case expands on the task experienced discussed in **Table 1: User Profile**.

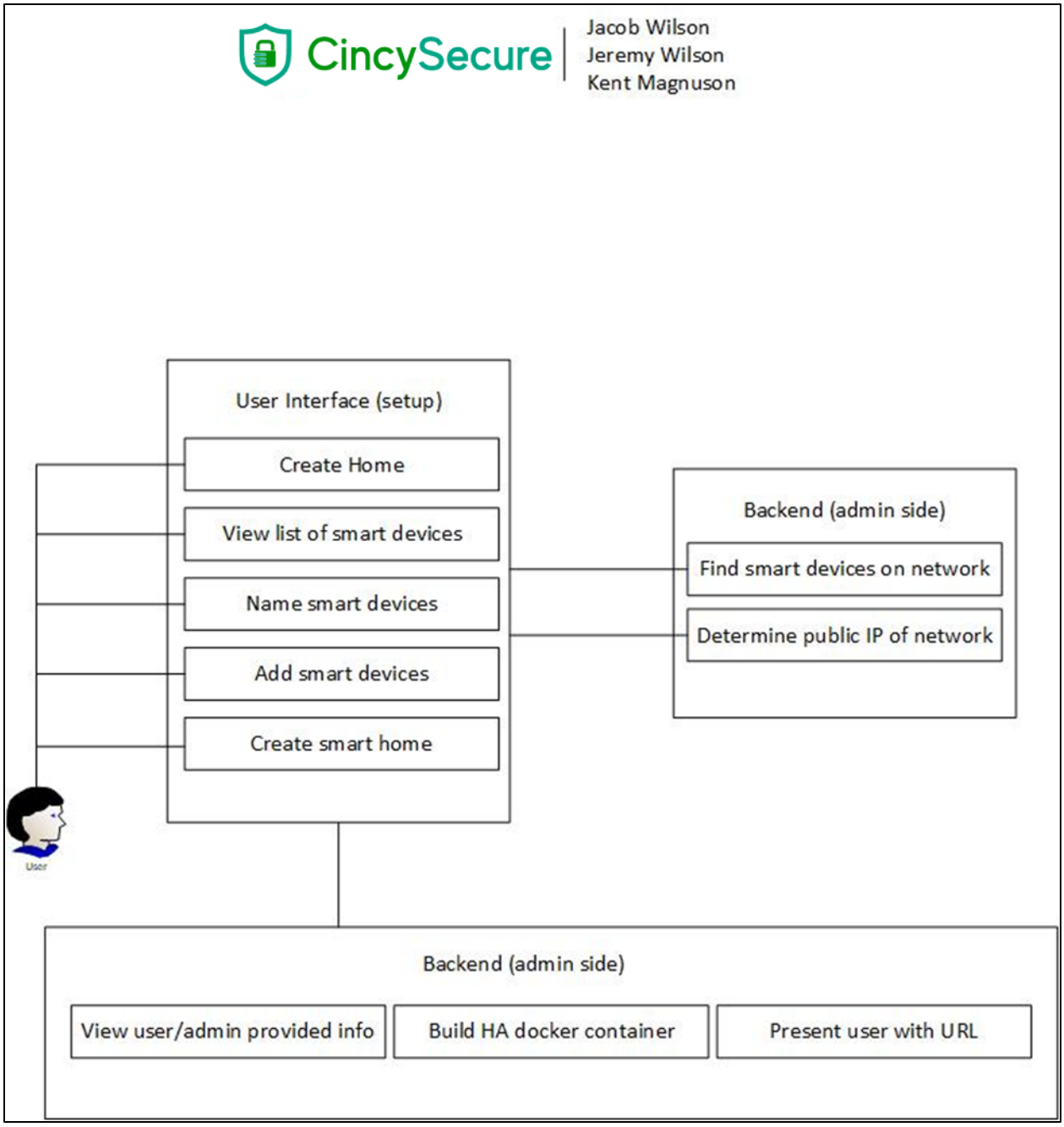


Figure 1: Our Use Case

Figure 1: Use Case provides context around how our application will work throughout the beginning (user interface) and the backend (admin side).

Procedure

A prerequisite to beginning the setup process was that the user needed to have all of their smart devices plugged in and connected to their home network. Once these IoT devices were operational, the user would be ready to begin.

First, the user goes to www.cincysecure.com and starts the setup process by copying and pasting a command from our website into their terminal. The command would clone our repository from Github and begin a series of scripts which would create containers for MySQL and our WebUI using Docker Compose. After the Docker containers were created, a Python script would scan their home network for devices. The script would identify only the smart device by cross referencing their Media Access Control (MAC) address to the known manufacture's MAC address reference. Once the script completed, a web portal with the user's personal setup application would open through a browser. The user would then be able to begin the process of identifying and naming their smart devices.

Once the web portal was up and running, the user would set a name to call their home along with a password. These credentials are used to later log into the created Home Assistant instance. After the name and password were set, the application would pull the detected smart devices into their own setup pages. Each page would consist of a detected smart device, displaying it's IP address and a friendly name set from the Python script.

To determine what smart device, they are currently setting up, the user would then click a button on the web page which would toggle the device to turn on and off. This would allow the user to easily see which smart device they are naming without having to manually look up its IP and MAC address. Based on what device is being set up, the user would select a box if the smart device is either a light, switch, or motion sensor. The smart

device is then named and submitted. If other smart devices were discovered, the web site would direct you to the next device setup page.

Once done setting up the smart devices the user clicks build home which would take all the smart device configurations they entered in and put it in a Home Assistant instance. The Home Assistant web page is then presented to the user where they could log in with the password they previously created. Once logged in, the user can see all of the smart devices they added, they now can set up automation and control all the smart devices from one central application.

Technical Architecture

Figure 2: Technical Diagram outlines how our product interacts with a user's devices within their home network.

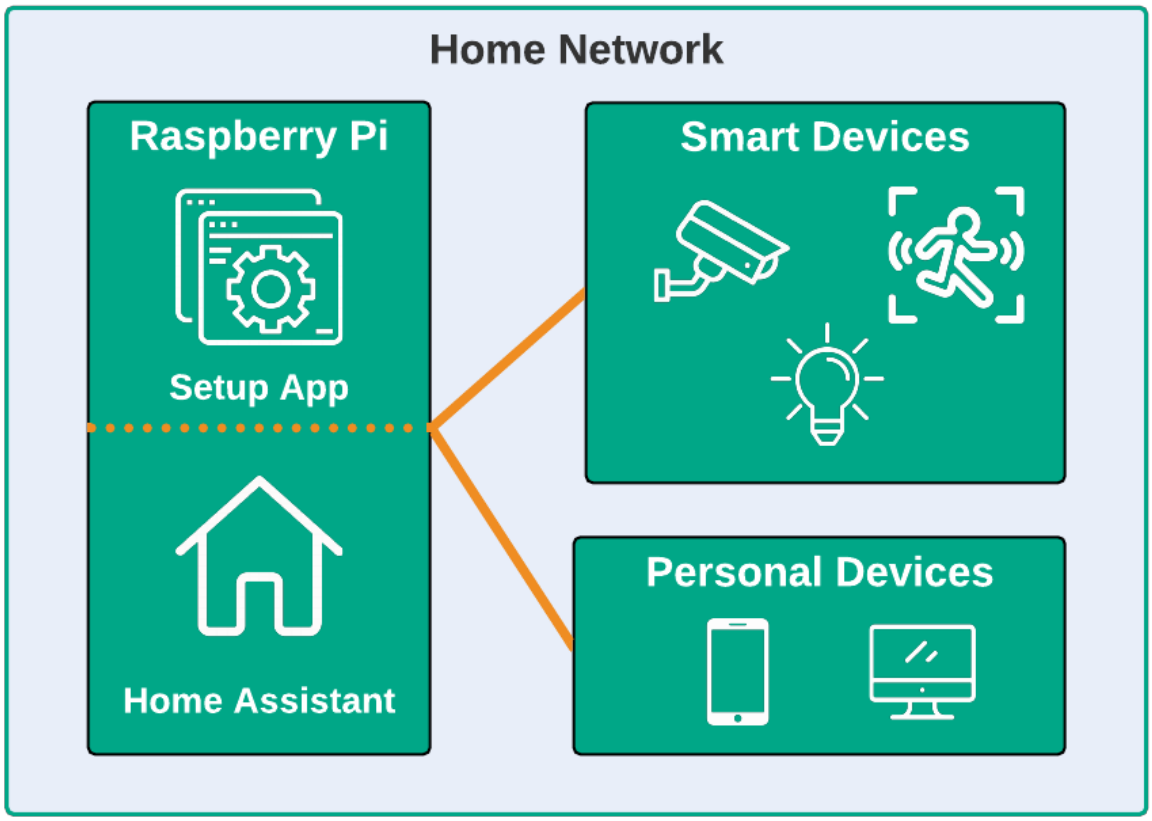


Figure 2: Technical Diagram

The Raspberry Pi was split between the setup application that runs through the WebUI and the Home Assistant instance that it creates. The “Smart Devices” section represents various IoT devices that will be configured and added into Home Assistant. The “Personal Devices” section represents the devices such as smart phone or laptops that the user can use to access the Home Assistant instance running on the Pi.

Project Management

Budget

Table 2: Original Budget illustrates the associated risks within our project such as using Google Cloud Platform. Our budget displays the estimated labor costs as well as any additional hardware we needed to purchase.

Estimated Cost				**Annual Support**			Hardware/Misc Costs:	
	Rate Per/Hr	Work Effort	Costs	Rate Per/Hr	Work Effort	Support Cost	**Item:**	**Cost:**
Labor - IT	$25.00	400	**$10,000.00**	20	100	**$2,000.00**	Raspberry Pi 3	$39.99
Software			-			-	Philips Hue A19	14.99
Hardware			**79.95**			-	GoSound Mini Smart Plug	9.99
Misc			-			-	White Board	9.99
TOTAL			**$10,079.95**			**$2,000.00**	Markers	4.99
							TOTAL:	**$79.95**

Table 2: Original Budget

This table explained our estimated labor cost while working on the project, software needed, hardware and other miscellaneous items. We also detailed out on the right side the different hardware requirements. The original budget was created during the Fall semester.

Table 3: Final Budget illustrated our projects final budget once the project was completed.

Estimated Cost				**Annual Support**			Hardware/Misc Costs:	
	Rate Per/Hr	Work Effort	Costs	Rate Per/Hr	Work Effort	Support Cost	**Item:**	**Cost:**
Labor - IT	$25.00	400	**$10,000.00**	20	100	**$2,000.00**	Raspberry Pi 3	$39.99
Software			-			-	Philips Hue A19	14.99
Hardware			**79.95**			-	GoSound Mini Smart Plug	9.99
Misc			9.99			-	White Board	9.99
						-	Markers	4.99
TOTAL			**$10,079.95**			**$2,000.00**	Candy for Expo	$9.99
							TOTAL:	**$89.94**

Table 3: Final Budget

Table 3: Final Budget explained the increased overall cost of CincySecure. This table provides relevant information on all of our expenses. The final budget was created during the Spring semester and provide a more accurate vision of our budget for the total project.

Objectives and Deliverables

Figure 3: Deliverables showed major project milestones and phases of our project. We determined the timing of these phases using **Figure 4: Gannt Chart**. The right side of the table indicated assignment milestones and their due dates. The left side listed our project's development phases and their due dates.

MAJOR PROJECT MILESTONES (DELIVERABLES)			
FALL OF 2018 MILESTONES			
Initiation Phase	9/23/2018	Team Contract Written	9/24/2018
Research Phase	10/4/2018	Project Abstract Drafted	10/15/2018
Planning Phase	10/15/2018	User Profile Drafted	10/22/2018
Infrastructure Phase	11/22/2018	Elevator Speech	10/22/2018
Implementation Phase	11/26/2018	Fall Presentation	11/19/2018
SPRING OF 2019 MILESTONES			
Test Phase #1	1/8/2019	Spring Presentation	3/11/2019
Deployment Phase	1/20/2019	Tech Expo	4/9/2019
Security Testing	2/1/2019		
Test Phase #2	2/17/2019		
Redesign/Software Update Phase	3/3/2019		

Figure 3: Deliverables

Many of the milestones for the Spring semester were from testing and updating our project. Using **Figure 4: Gannt Chart** our tasks were broken down and allowed us to stay on target.

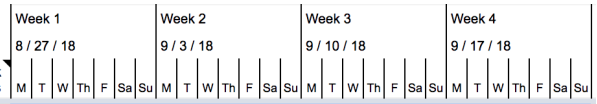
Project Schedule

Our project schedule was broken down into major phases included in **Figure 4: Gannt Chart**. We separated our charts into different phases based on the milestones discussed in **Figure 3: Deliverables.**

Initiation:

Display Week: 1

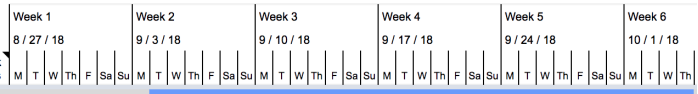
WBS	Task	Start	End	Cal Days	% Done	Work Days
1	**Intiation**	Mon 8/27/18	Sun 9/23/18	28		20
1.1	Team Assembly	Mon 8/27/18	Sun 9/02/18	7	100%	5
1.2	First Team Meeting	Mon 9/03/18	Mon 9/03/18	1	50%	1
1.3	Idea Brainstorm	Tue 9/04/18	Sat 9/08/18	5	25%	4
1.4	Idea Sent to Jim	Sun 9/09/18	Mon 9/10/18	2	0%	1
1.5	Advisor Approval	Tue 9/11/18	Wed 9/12/18	2	0%	2
1.6	Team Contract	Thu 9/13/18	Sun 9/23/18	11	0%	7



Research:

Display Week: 1

WBS	Task	Start	End	Cal Days	% Done	Work Days
2	**Research**	Tue 9/04/18	Thu 10/04/18	31		23
2.1	Identify Requirements	Tue 9/04/18	Mon 9/10/18	7	0%	5
2.2	Research React (Web UI)	Tue 9/11/18	Fri 9/14/18	4	0%	4
2.3	Research Docker	Mon 9/17/18	Thu 9/20/18	4	0%	4
2.4	Research GCP Enivornment	Fri 9/21/18	Mon 9/24/18	4	0%	2
2.5	Research IoT devices	Tue 9/25/18	Fri 9/28/18	4	0%	4
2.6	Research HomeAssistant	Mon 10/01/18	Thu 10/04/18	4	0%	4



Planning:

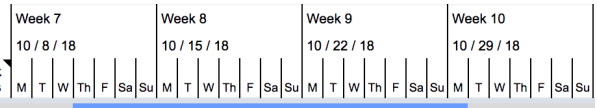
Display Week: 6

WBS	Task	Start	End	Cal Days	% Done	Work Days
3	**Planning**	Thu 10/04/18	Mon 10/15/18	12		8
3.1	Develop roadmap	Thu 10/04/18	Sun 10/07/18	4	0%	2
3.2	Develop architecture design/overview	Mon 10/08/18	Thu 10/11/18	4	0%	4
3.3	Define "champions" of each technology	Fri 10/12/18	Mon 10/15/18	4	0%	2

Stage 1:

Display Week: 7

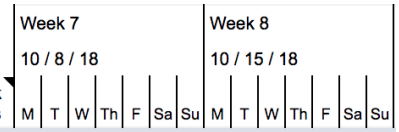
WBS	Task	Start	End	Cal Days	% Done	Work Days
4	**Stage 1 (Front end / GCP infra)**	Thu 10/11/18	Mon 10/29/18	19		13
4.1	Create GCP account	Thu 10/11/18	Thu 10/11/18	1	0%	1
4.2	Assign IAM rules for admin users	Fri 10/12/18	Fri 10/12/18	1	0%	1
4.3	Build out core instance(s)	Sat 10/13/18	Wed 10/17/18	5	0%	3
4.4	Begin work on building out Web UI	Thu 10/18/18	Mon 10/22/18	5	0%	3
4.5	Build user provided form	Tue 10/23/18	Sat 10/27/18	5	0%	4
4.6	Build OpenVPN Form	Sun 10/28/18	Mon 10/29/18	2	0%	1



Stage 2:

Display Week: 7

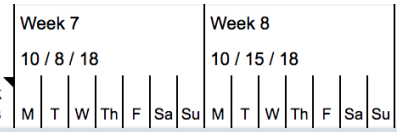
WBS	Task	Start	End	Cal Days	% Done	Work Days
5	**Stage 2 Docker**	Thu 10/11/18	Wed 10/17/18	7		5
5.1	Understand how web UI form results	Thu 10/11/18	Thu 10/11/18	1	0%	1
5.2	ETL results into docker build file	Fri 10/12/18	Sun 10/14/18	3	0%	1
5.3	Export results into GCP API	Mon 10/15/18	Mon 10/15/18	1	0%	1
5.4	Upload build file	Tue 10/16/18	Tue 10/16/18	1	0%	1
5.5	Prepare file for upload	Wed 10/17/18	Wed 10/17/18	1	0%	1



Stage 3:

Display Week: 7

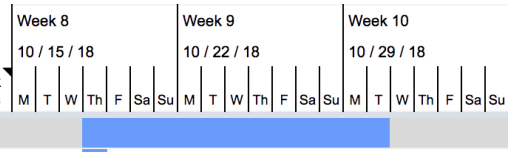
WBS	Task	Start	End	Cal Days	% Done	Work Days
6	**Stage 3 User provided GCP Instanc**	Thu 10/11/18	Thu 10/18/18	8		6
6.1	Using GCP API, create instance	Thu 10/11/18	Thu 10/11/18	1	0%	1
6.2	Using API, instance update itself	Fri 10/12/18	Fri 10/12/18	1	0%	1
6.3	Using API, install required applications	Sat 10/13/18	Sun 10/14/18	2	0%	0
6.4	Push docker build file to GCP	Mon 10/15/18	Mon 10/15/18	1	0%	1
6.5	Docker build	Tue 10/16/18	Thu 10/18/18	3	0%	3



Stage 4:

Display Week: 8

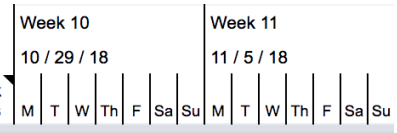
WBS	Task	Start	End	Cal Days	% Done	Work Days
7	**Stage 4 HomeAssistant on GCP**	Thu 10/18/18	Tue 10/30/18	13		9
7.1	Detect build complete	Thu 10/18/18	Thu 10/18/18	1	0%	1
7.2	Detect build fail	Fri 10/19/18	Fri 10/19/18	1	0%	1
7.3	API to setup DNS name	Sat 10/20/18	Wed 10/24/18	5	0%	3
7.4	API to build HTTPS	Thu 10/25/18	Mon 10/29/18	5	0%	3
7.5	Provide back completed URL	Tue 10/30/18	Tue 10/30/18	1	0%	1



Stage 5:

Display Week: 10

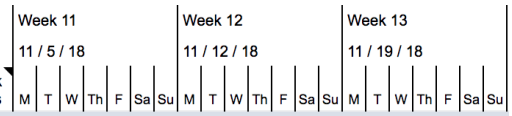
WBS	Task	Start	End	Cal Days	% Done	Work Days
8	**Stage 5 FAQs**	Tue 10/30/18	Thu 11/22/18	24		18
8.1	Create page with provided URL	Tue 10/30/18	Thu 11/01/18	3	0%	3
8.2	Provde API password for HA	Fri 11/02/18	Fri 11/02/18	1	0%	1
8.3	Provide GCP instance ID	Sat 11/03/18	Sat 11/03/18	1	0%	0
8.4	Privode docs on automation setup	Sun 11/04/18	Tue 11/06/18	3	0%	2
8.5	Provide support plan	Wed 11/07/18	Fri 11/09/18	3	0%	3



Infrastructure:

Display Week: 11

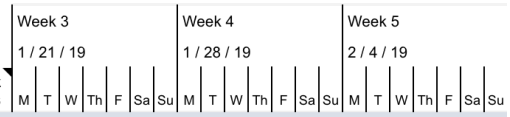
WBS	Task	Start	End	Cal Days	% Done	Work Days
9	**infrastructure Plan**	Fri 11/09/18	Thu 11/22/18	14		10
9.1	Define Support Plan	Fri 11/09/18	Fri 11/09/18	1	0%	1
9.2	Define and create DR for web UI	Sat 11/10/18	Sat 11/10/18	1	0%	0
9.3	Define SLA for product	Sun 11/11/18	Sun 11/11/18	1	0%	0
9.4	Documentation on overall product	Mon 11/12/18	Wed 11/21/18	10	0%	8



Testing:

Display Week: 3

WBS	Task	Start	End	Cal Days	% Done	Work Days
1	**Testing**	Mon 1/07/19	Sat 2/09/19	34		25
1.3	Regroup with team	Tue 1/15/19	Tue 1/15/19	1	25%	1
1.4	Testing of Product	Wed 1/16/19	Tue 1/29/19	14	0%	10
1.5	Define major bugs	Wed 1/30/19	Sat 2/02/19	4	0%	3
1.6	Fix major bugs	Sun 2/03/19	Sat 2/09/19	7	0%	5



Testing Phase 2:

Display Week: 3

WBS	Task	Start	End	Cal Days	% Done	Work Days
1	**Testing Phase 2**	Sat 2/09/19	Tue 3/19/19	39		27
1.3	Continuation of testing	Sun 2/17/19	Sat 3/02/19	14	25%	10
1.4	Define Bugs	Sun 3/03/19	Tue 3/05/19	3	0%	2
1.5	Fix bugs	Wed 3/06/19	Tue 3/12/19	7	0%	5
1.6	External testing	Wed 3/13/19	Tue 3/19/19	7	0%	5

Expo Preparation:

Display Week: 3

WBS	Task	Start	End	Cal Days	% Done	Work Days
1	**Expo Prep**	Tue 3/19/19	Tue 4/16/19	29		21
1.3	Preparation of table	Wed 3/27/19	Tue 4/02/19	7	25%	5
1.4	Final practice runs	Wed 4/03/19	Mon 4/08/19	6	0%	4
1.5	Expo	Tue 4/09/19	Tue 4/09/19	1	0%	1

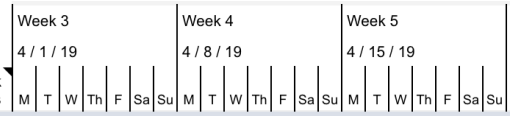


Figure 4: Gannt Chart

Technical Elements

Application

The CincySecure setup application was built off a WebUI using a Hypertext Markup Language (HTML) and Personal Home Page (PHP) code base. The overall idea of the project was to provide a user with an easy to use front-end setup to remove the hassle of learning how to use the more complicated Home Assistant configurations. Along with an easy to use

WebUI, our back-end completed the build in order to provide the user with the completed Home Assistant instance.

This project was largely dependent on the build stage of our application. Early on, we believed that the best way to do this would be through the user results in an Extensible Markup Language (XML) format. As we began working on this, we soon realized that this idea did not provide the best results. Parsing XML and then inserting those results into Home Assistant configuration became complicated. To resolve this issue, we moved away from XML and focused on using a database instead.

Our architecture consisted of Docker containers which allowed us to transfer all of the device configurations and store them within MySQL database container. From there, our WebUI was able to easily pull these devices into the setup page. This allowed the user to not have to know technical terms like IP and MAC address of each of their smart devices.

When creating the front-end, we began thinking of the best way to provide an easy to use interface for our users. With Home Assistant already being over complicated, we wanted to be sure that our WebUI was easy to use for anyone. A decision was made to have the setup pages show one device per page. On each page, the IP and MAC address would be already filled in and grayed out to prevent the user from making changing to the discovered device. From there, the user would toggle a button that triggered the smart device to turn on or off depending on the previous state of the smart device. This allowed for the user to understand which device they were connecting to. For example, the scan would find a TP-Link light bulb, but the user would not know which light bulb it was. They would then toggle the button to see that their bedroom or kitchen lamp would turn on. Once they confirmed which device it was, they would name it and move to the next page.

Once all the devices had been named from the setup pages, the user would get to the build page. This would take all the personalized device names with their IP and MAC addresses and configure them into a new Home Assistant instance. After the build stage would complete, the user would then be able to log in and control all their smart devices in one simple Home Assistant application.

Network

The network device behind our idea started out with a mix of Google Cloud Platform (GCP) and a home Virtual Private Network (VPN) client. Once the user had selected their devices, it would have created a VPN tunnel within the user's home. We quickly found that this process was not as easy as it sounded. With many different home routers, it became difficult to setup configurations to provide a simple setup that would work on in every situation. Instead of using GCP and a VPN, we moved to keep the solution within the home network for both ease of use and security. We also thought about having the user download a custom script to run from their computers. However, we then ran into the issue of different computer systems. To resolve all of these issues, we decided to move away from the cloud and back within the home network.

To provide the user a list of devices within their network, we performed an Address Resolution Protocol (ARP) scan to determine the number of smart devices within a user's home network. The ARP scan was run through a Python script that ran during the setup stage. This script collected the friendly name of the device, it's IP address, and it's MAC address. Once the scan of the home network was completed, the script would send the collected devices to a MySQL database.

Security

The security concerns revolved around how the application and host will be kept secure. The IP would remain on the user's home network and not exposed to the internet. We included auto updates of Home Assistant and security checks to prevent the application from malicious attacks. Since our WebUI was built within the host during setup, the website would never be exposed to the outside internet. Thus, the risk for an attack is signifyingly less. The Home Assistant application itself is safe and secure as it will only existed within the user's home network.

Visuals

Flow Chart

Figure 5: Story Board showed our initial outline of how our system will operate and the steps that will take place for the user's point of view. It starts at the very beginning with the user purchasing a Pi and ends with the finished product.

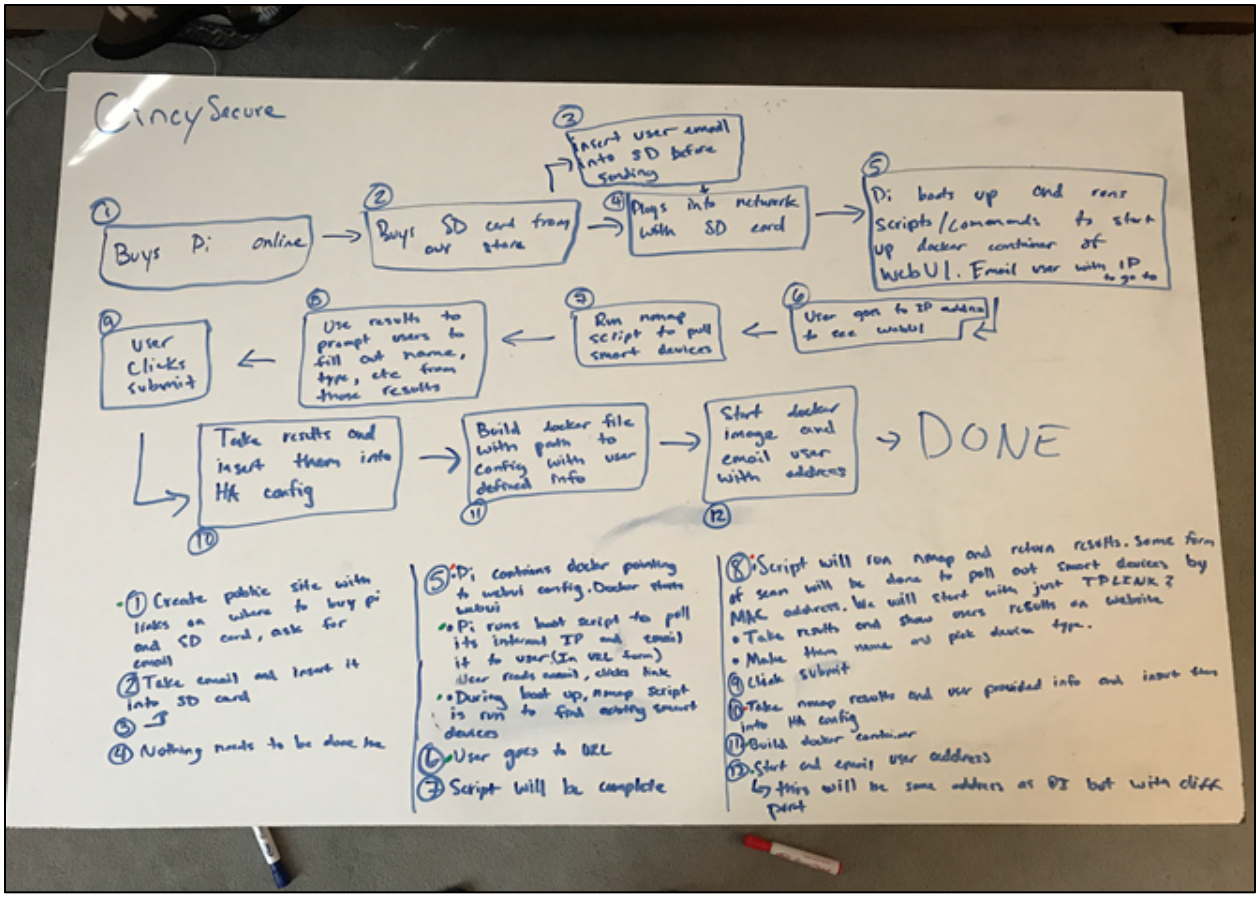


Figure 5: Flow Chart

This flow chart was created in the beginning of the Fall 2018 semester and was our first iteration of the project. The project flow greatly changed from the beginning to the end of CincySecure.

Raspberry Pi

Figure 6: Raspberry Pi shows our Raspberry Pi with a micro Secure Digital (SD) card which hosted our service. The Raspberry Pi 3 was the version that was used.

Figure 6: Raspberry Pi

The third model was used as it was the latest model with the most up to date hardware. The Docker images we used all supported the Raspberry Pi 3 firmware.

Expo Poster

Figure 7: Tech Expo Poster was the poster we displayed during the tech expo. The poster discussed the problem and proposed solution that CincySecure offered.

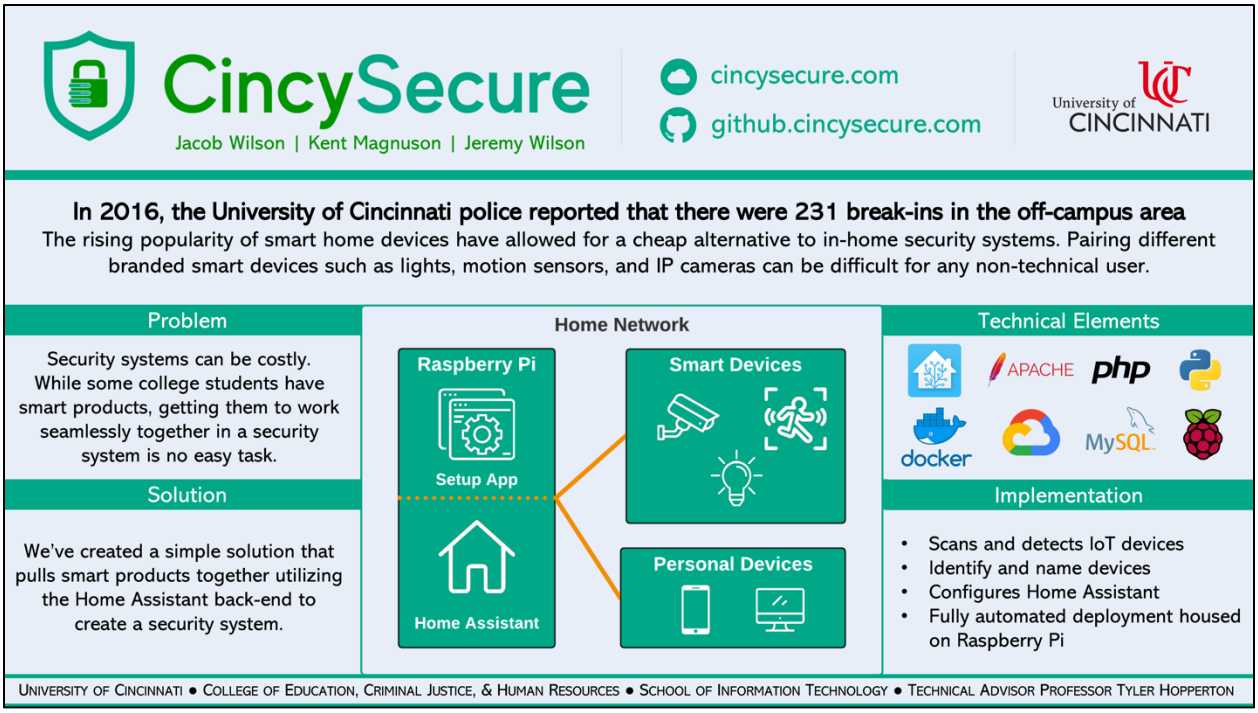


Figure 7: Tech Expo Poster

We put a technical diagram of how the Pi interacted with the smarts devices. It also showed the user's personal devices. We tried to keep this diagram simple so any user could understand it. We also included our technical elements as well as how it is implemented.

Test Plan

Overview

Our testing phases began in the beginning of the Spring semester. We carried out two different internal tests to identify major bugs then conducted external testing from other students. External testing was performed by each member of the group. We recruited two people for external testing with different levels of technical knowledge and no prior understanding of how CincySecure worked.

Objective

Our objective was to determine how easy and automated our setup process was. We recruited users with various levels of technical experience to conduct the tests on. Testing was also performed to ensure CincySecure can be used with multiple different brands of IoT devices.

Scope

The scope of testing was run through CincySecure's WebUI. The tests were run without guidance from any team member. However, directions to complete each step were given.

Test Plan

Four different tests were conducted on users to determine the usability of the setup process of our application and user interaction with the product. The steps required to complete each test were given to the user before the testing began. The testers used a Raspberry Pi within a room which had multiple smart devices on the same network as the Pi. The tester was informed before the test started on the different smart devices within the room. This included the brand and name of each smart device.

Test Results

Table 3: Stability Testing focused around how stable the CincySecure WebUI was when run through the Raspberry Pi when testers interacted with it.

Tester	Date	Item #	Expected	Actual	Pass/Fail	Bug
User 1	1/12/19	1.1 Pi Stability	Installs and uses application	Installed and uses application	Pass	None
User 2	1/14/19	1.2 Pi Stability	Installs and uses application	Installed and uses application	Pass	None

Table 4: Stability Testing

The main objective of this test was to confirm that the hardware we were using could handle our software. The biggest concern was that the multiple Docker containers that we used would cause the Raspberry Pi to crash. After running the stability test, we determined the Pi was able to handle the setup and build.

Table 4: Setup Test focused on the accuracy of the how well the testers read through the instructions and installed the WebUI onto the Pi.

Tester	Date	Item #	Expected	Actual	Pass/Fail	Bug
User 1	1/12/19	2.1 Launch UI	User launched local WebUI page and begins setup	Installed and used WebUI	Pass	None
User 2	1/14/19	2.2 Launch Ui	User launched local WebUI page and begins setup	Had problems understanding how to install WebUI	Fail	None

Table 5: Setup Test

This test was performed to make sure even non-technical users could understand our documentation and complete the setup. Since CincySecure was marketed towards all users regardless of their technical experience, this was an important test. After we concluded these tests, we determined to re-work our WebUI to make it more simplistic and automated. We decided to limit the number of items on each webpage with a very visible

next and submit buttons. This allowed our webpage to be user friendly in what we wanted the user to do and outlined our steps. With most of the work being done on the back-end, we put more of an emphasis on making sure our build scripts were working as best as possible.

Table 5: User Interface Test was performed after the testers installed and began using the WebUI. Testing was done on how well they interacted with each step of the setup.

Tester	Date	Item #	Expected	Actual	Pass/Fail	Bug
User 1	1/12/19	3.1 Uses WebUI	Select and name smart devices found	User named devices	Pass	None
User 2	1/14/19	3.2 Pi Stability	Select and name smart devices found	User named devices	Pass	None

Table 6: User Interface Testing

This test confirmed that users could recognize which devices they were adding and naming. Testing was done to see if the user could toggle the on or off button to make their selected light bulb or switch turned turn. Once they would see which device they were configuring, they would give it a relevant name.

Table 6: Functionality Test was focused around the features of the application. This was to confirm that each feature was working as expected and in a logical way.

Tester	Date	Item #	Expected	Actual	Pass/Fail	Bug
User 1	1/12/19	4.1 Overall use of application	Overall experience on the WebUI	User seemed able to understand how to use the tool	Pass	None
User 2	1/14/19	4.2 Overall use of application	Overall experience on the WebUI	User had issues with the install script but once passed that, the user was able to name devices	Pass	None

Table 7: Functionality Test

Functionality testing determined if our product would be ultimately useful. It tested CincySecure at a broader scope to see if the scripting and automation were accurate, as well as how the build stage performed. Once the build stage completed, we monitored how the user interacted with their created Home Assistant instance. For this test, we used screen recording software to track where they were clicking and what grabbed their attention the most.

Future Recommendations

Problems Encountered and Solutions

One of the largest issues with CincySecure was the reliability of the network scanning script. The script ran an ARP scan and cross referenced the MAC addresses it found with a database of know devices. Sometimes the scan would not find certain devices on the network for seemingly no reason. We had to modify the script to run the ARP collection scan for a longer amount of times as well as multiple times. The script would

then drop any duplicates it would find. This allowed the script to be more reliable in discovering smart devices on the network.

After the testing phase we decided to rework our WebUI. Our first iterations had too much to do on each of the setup pages causing users to get confused on what the objective of each page was for. After configuring our script to translate the MAC address into a friendly name, we created another script that pulled in the IP and name of each smart device into the setup pages automatically. We then removed the ability to change these fields. This made it clearer to the user that all that was needed to be done was to confirm which device was selected and name it.

Another issue came with running certain Docker images on the Raspberry Pi. Since Docker can be very intense on a machine's memory and central processing unit (CPU), we needed to use Docker Compose to customize resources used on each docker container. We pulled Pi specific images from Docker Hub and created our own DockerFiles which resolved the issue of containers crashing and not consistently starting.

During the Tech Expo our home router we were using for the demonstration was not working. The Service Set Identifier (SSID) of the router was displayed, however, no smart devices or computers could connect to it. We had a back-up on hand that also had the exact same issue. After talking with someone from the University of Cincinnati's Network Operation team we found that this issue may have been from the fact that we were too close to other access points causing our signal to get jammed. After trying to emit our routers broadcast on different channels, we ran out of time and had to instead use a mobile hotspot. We were able to get our demo up and running, however, the hotspot's network was considerably slower.

Recommendations for improvement

If more time was given, more testing would be conducted on various different home networks. This would be to confirm the usability of the network scanning scripts. We could also look into better documentation for technical users that want to customize their own instance. They would be able to fork our current repository and make additions and improvements.

Conclusion

Fall 2018 Conclusion

The Fall semester we conducted research on different ways we could go about reaching our solution. After purchasing a whiteboard and visualizing out the structure of our design, we were able to take a deeper look into what we needed to do.

First, we investigated hosting through Google Cloud Platform as well as locally on a Raspberry Pi. After some consideration, we found that using a Pi will be more secure as well as cost effective. Switching from cloud to local, greatly changed the structure of our design which required some steps to be completely changed.

Our biggest change was switching from using a React WebUI to a HTML and PHP web form which would import results into a Structured Query Language (SQL) database. This decision was made due to the complications that came with trying to parse and store the XML results of the React page. Using a SQL database made it much easier to store and retrieve data.

We also spent much time researching how we were going to scan the user's network to find all smart devices. We looked into using an ARP table as well as flashing the home router, however, we found using a modified ARP scan was the best solution. Along with the scan we will use a Python script to parse the output to a comma-separated value (CSV) file and import into our database.

A shell of the web page was completed as well as the PHP form. We also developed a working ARP scan with a Python script to export the data. Our next steps were to be able to import the results in the MySQL database.

Spring 2019 Conclusion

With the success of creating the WebUI and network scanning script during the Fall, our next task was the be able to import the results into MySQL. Many modifications to the script were needed in order to import the data into MySQL cleanly. This took a bulk of the first month of the Spring semester.

Once we had a working version of CincySecure we started to conduct our internal testing. When each group member agreed that we had a proper working product, we moved towards external testing. Having users try out our product and receiving feedback was paramount in our development stages.

Being one of the first groups to present in the Spring, some pressure was on the team in order to get a well-working prototype done. After completion of our first prototype, the bulk of the semester was spent testing out the scripting and WebUI.

The goal of the rest of the semester was to make sure our product was simple to use. During the Tech Expo, we wanted to make sure people would be able to walk up and test the setup themselves to get a better grasp of how it works. In preparation, we also recorded a demo video to be used for our presentation and to be displayed during the expo.

During the expo, the judges and attendees enjoyed our demonstration and the product we had to offer. Despite having some issue getting the home router to work, we had a great looking booth. In our eyes the expo was a great success as we won the “Best of IT Expo” and “Presentation in Cyber Security Solutions” awards. After months of work and sacrificing many weekends, we are proud of CincySecure and believe it was an amazing accomplishment.

Appendix

References

[1] “Configuring Home Assistant,” 2018. [Online]. Available: https://www.home-assistant.io/docs/configuration/

[2] “Internet of Things With Raspberry Pi,” 2018. [Online]. Available: https://www.instructables.com/id/Simple-IOT-project-for-Beginners/

[3] “Nmap from Beginner to Advanced,” 2018. [Online]. Available: https://resources.infosecinstitute.com/nmap/

[4] “2016 Campus Crime Report,” 2017. [Online]. Available: https:// www.uc.edu/content/dam/uc/publicsafety/docs/2016%20Campus%20Crime%20Report_FINAL.PDF