Vigil – Security Console and Network Scanner

by

Austin Vaive, Maurice Jones, & Matthew Cooley

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

© Copyright 2019 Vigil Inc

The author grants to the School of Information Technology permission
to reproduce and distribute copies of this document in whole or in part.

Austin Vaive 4/14/2019

Austin Vaive Date

Maurice Jones 4/14/2019

Maurice Jones Date

Matthew Cooley 4/14/2019

Matthew Cooley Date

Tyler Hopperton 4/14/2019

Tyler Hopperton, Faculty Advisor Date

University of Cincinnati
College of
Education, Criminal Justice, and Human Services
April 2019

TABLE OF CONTENTS

List of Illustrations

TABLES

FIGURES

ABSTRACT

According to mediapost.com, the average American will own 13 internet connected devices by 2021[1]. That means a family of 4 will have close to 50 devices connected to their home network. In the current climate of large-scale vulnerabilities, ransomware, and other digital threats, how can we protect our devices, and those of their family members, when we may not even know exactly what is connected to our network? Vigil is a home network sentry that constantly monitors and scans every device connected to your wireless network and sends an alert when it detects an outdated or vulnerable device and when a new, unrecognized device connects to the network. Vigil allows its users complete transparency into the devices on their network, and also offers easy to understand, actionable information to non-technical users. This paper summarizes the process and technical details for the development of Vigil and its real-world usage data

[1] Martin, Chuck. “North American Consumers to Have 13 Connected Devices.” MediaPost, 11 June 2017, www.mediapost.com/publications/article/302663/north-american-consumers-to-have-13-connected-devi.html.

1. Problem Statement

1.1 Introduction

The number of connected devices is rising at a rapid rate, and so is the frequency of ransomware, botnets, and other large-scale malicious software attacks. Every device connected to a home Wifi network has the potential to be compromised if it is not regularly patched with Security updates. Vigil will automatically scan the network and notify the user when a device is out of date or vulnerable to attack.

1.2 Project Description

Vigil allows its users to install a simple home network monitor into their network. Once the device is powered on, and connected to the network, it will begin to silently scan the devices that are connected to the same network, and interface back to the user via an intuitive smartphone application. Once the scanner detects a noteworthy anomaly, such as an outdated device that may become vulnerable to attack, Vigil sends the user a notification to alert them that action is required, and they should update their device as soon as possible. Vigil also enables visibility into a network that the non-tech savvy user has never had before, with a simple interface that allows users to view a listing of all the devices connected to their network, as well as receive alerts when new devices join.

1.3 Problem

Vulnerability and network scanners are already commercially available products, and open-source variants of these tools are already available, but these tools are largely command-line based and are generally available on Linux operating systems. For the average home user, these tools may as well not exist, because they require technical

skills that they most likely do not possess. Also, large families will have a large number of devices. It can be difficult to remember to update devices that may not get used every day. With the recent surge of popularity of Internet of Things (IoT) devices, it can be difficult to keep track of exactly what is connected to the network, and when the last time it was checked for updates.

1.4 Solution

Vigil will remove the barriers that are currently stopping average home users the ability to see important information about their wireless network. By building a smartphone application that will interface directly with our network scanning appliance, Vigil will allow a user with any level of technical experience to see the same kind of information that was previously only available to the most tech-savvy of consumers.

Vigil will be installed as a two-piece solution, with one piece being the network monitor itself. The network monitoring portion will be a small System of a Chip (SoC) computing device that once connected to the network, will begin to interface with the smartphone app, and collect information about the network automatically. Past initial setup of the monitor (plugging into power, connecting to the network, etc), the user will be able to control everything directly through the companion Vigil smartphone application.

The Vigil app will present the user with basic information about their network by default, with easy to understand terms and graphics to show how many devices are out of date and other information but will allow the user to view more advanced information by selecting a specific device. This allows more tech-savvy users to still find value in our solution because they can still receive the detailed information they need, while it is hidden from view by default so less tech-savvy users do not feel overwhelmed.

1.4.1 Technical Approach

For the back-end of the application there were two design requirements set. The first requirement was that the backend would gather the information from the security scan logs and parse them into a format that was consumable by both the database and the front-end. The second requirement was to transfer that information from the Vigil device to the database and from there to the front-end application. These goals outlined the steps and procedures the back-end needed to follow, gather and interpret the information and pass it along.

1.5 User Profile

The end user is the only person that will be interacting directly with our solution, as it is a self-contained, Do-It-Yourself approach to network monitoring. There will be no cloud services required. The user itself retains total control over the solution, and they are the only ones that will be able to interact with it, aside from software updates to the application or scanner itself.

1.5.1 Project Title

Vigil - Security Console and Network Scanner

1.5.2 Potential Users

- Parents that wish to ensure their children's electronic devices are not vulnerable and out of date
- Anyone looking to improve the security of the devices connected to their home network.

1.5.3 Software and Interface Experience

Because our project is aimed at the “average joe”, the software and user interface will be easy to use and understand for users with a wide variety of technical ability. It will include basic, easy to understand information, with the ability to view more detailed and technical information if the user desires to see it. A basic understanding of how to navigate a modern smartphone app is all that is required to begin using Vigil.

1.5.4 Experience with Similar Applications

Vigil is unique, but experience in the following applications will be similar to Vigil’s user experience:

- Apple iCloud Device Manager
- Fing mobile application
- Consumer internet router control panels

1.5.5 Task Experience

The user of Vigil must be comfortable with:

- Connecting a new device to home Wi-Fi network
- Navigating to a website and logging in

1.5.6 Frequency of Use

Users of Vigil will open the app either whenever they would like to view the information about their network, or whenever they receive a push notification that a device is out of date or if a new device joins the network. This could be every day if a user is very interested in the information the app provides, but the majority of users will most likely check the app every few weeks.

1.5.7 Key Interface Design Requirements

- Ability to quickly view time-sensitive information such as vulnerable devices and recently connected devices
- View all devices connected to the network, their hostname, IP address, and Operating System version.

1.6 Use Case Diagram

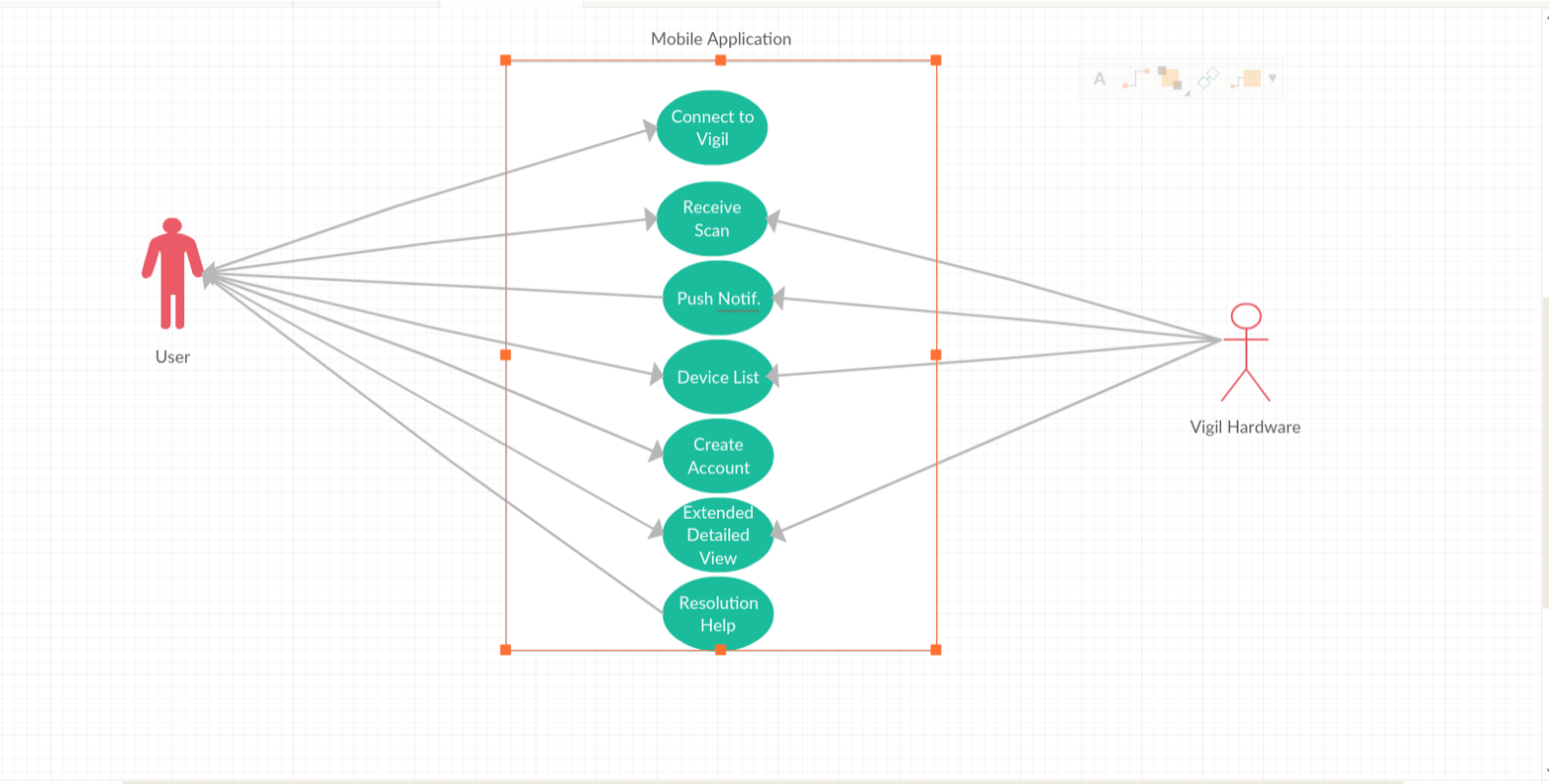


Figure 1: Use Case Diagram

1.7 Architecture

The Vigil system is separated into three distinct structures with defined communication pathways between each structure. The structures are the security scanner, the back -end and the front-end. Between these the security scanner may pass information using its log files that the back end will transform and transfer to the front end. The back-end may communicate with the front-end by passing JSON objects with user and scan data. The front-end may interact with the back end by requesting a specific dataset from the database.

1.7.1 Security Scanner

The security scanner portion of the project takes the form of a Python script that utilizes a set of scanning tools such as OpenVAS to run periodic scans of the local network. It will then take the results of these scans and output them to a log.

1.7.2 Back End

The back-end of the application is comprised of two parts. The first part is a NodeJS application running on the Vigil device. This application will take the security logs created by the scanner and parse them, creating a useable JSON object. The application will then open a connection to the second part, the Google Firebase Database. The Firebase database is a NoSQL database that stores user information and log data in JSON format.

1.7.3 Front End

The Front-end of the application is an Apache Cordova web application that allows both mobile app creation as well as desktop instances to be developed at the same time. The front-end uses the Google Firebase APIs to request data from the back end. Once the JSON objects are received it will dynamically display the needed information to the user.

1.7.4 Architecture Diagram

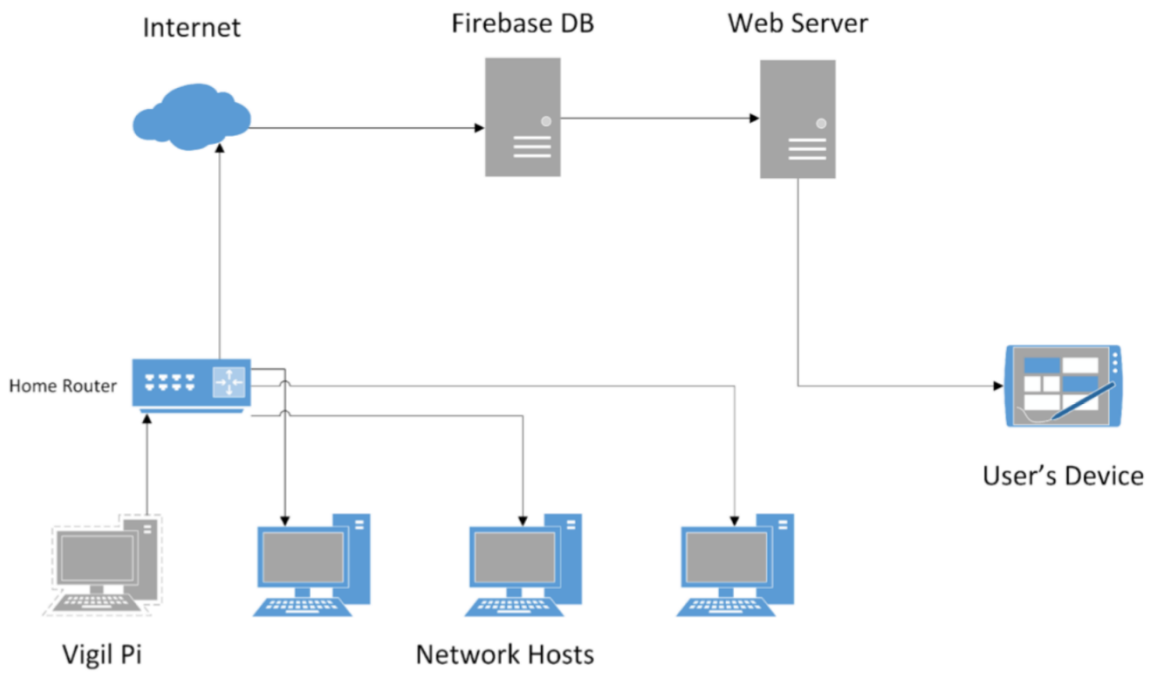


Figure 2: Architecture Diagram

2. Project Management

2.1 Budget

Table 1: Budget

Item	Unit#	Cost	Total
Raspberry Pi	1 unit	$35	$35
Labor- Front End	300 hours	$20/hour	$6,000
Labor- Back End	250 hours	$20/hour	$5,000
Labor- Cyber Security	350 hours	$20/hour	$7,000
Virtual Network for Testing	1 unit	$350	$350

Table 1: Budget

Grand Total: $18,385

2.2 Objectives / Deliverables

2.2.1 Objectives

For Vigil to be considered a success, we wanted to ensure that we met several key objectives:

- Ease of use
- Accurate data
- Actionable information

I believe that we accomplished the majority of these goals with our project. The user-facing application is very easy to use, and it provides accurate data as well as actionable information. The major concern that we are currently facing is the unknowns of each user's network. Vigil relies on the data it receives from the open source scanning

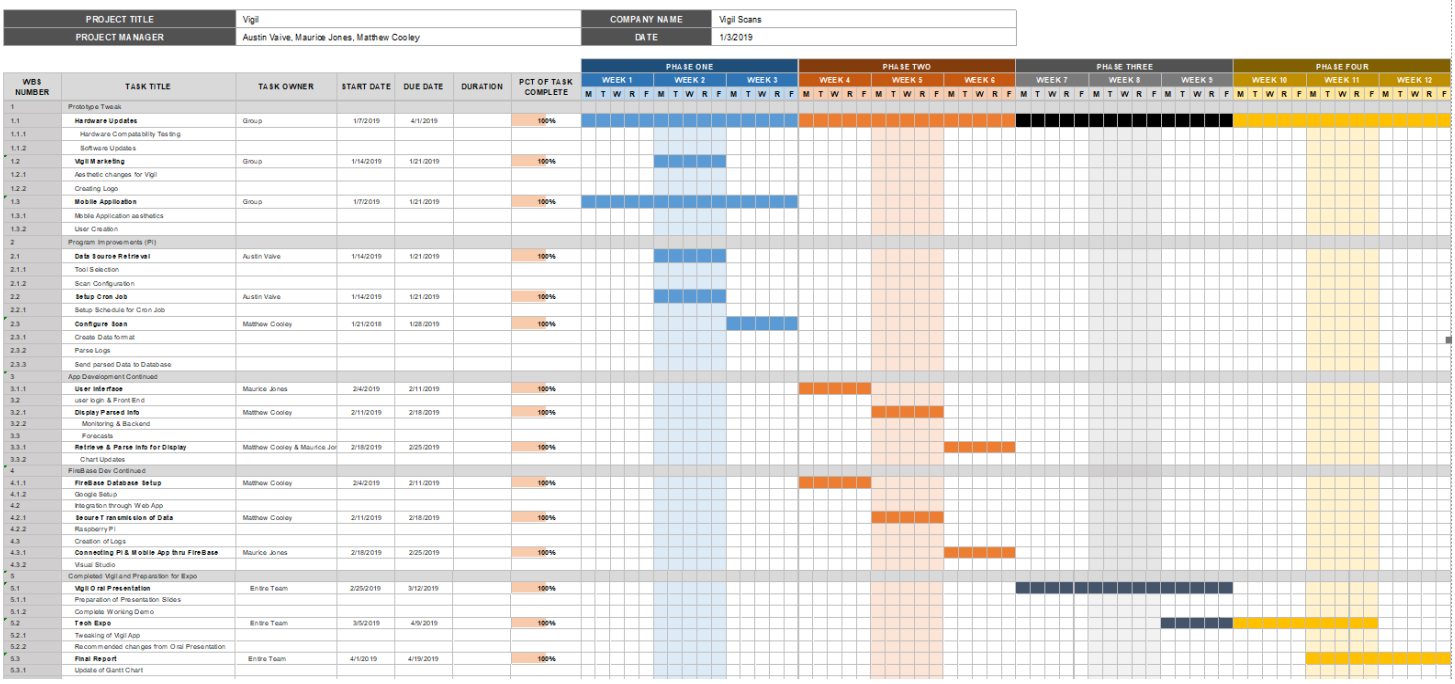
applications, and these applications can sometimes provide false-positives, or not provide detailed enough information. In that sense, we were somewhat limited to what information we were able to provide to the user. All things considered, however, even if Vigil could not provide the most detailed information about a certain network or device, it is still providing the user with more information than they originally had, so in that sense, it is a success.

2.3 Project Schedule

WORK BREAKDOWN STRUCTURE WITH GANTT CHART

PROJECT TITLE	Vigil	COMPANY NAME	Vigil Scans
PROJECT MANAGER	Austin Vaive, Maurice Jones, Matthew Cooley	DATE	9/24/18

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	Dev Environement					
1.1	**Setup Environment**		10/14/18	10/20/18		0%
1.1.1	Purchase of Hardware	Austin Vaive				
1.1.2	Installing OS & Software	Austin Vaive				
1.2	**Git Setup**		10/21/18	10/27/18		0%
1.2.1	Mobile Application Repository	Maurice Jones				
1.2.2	Raspberry Pi Repository	Matthew Cooley				
1.3	**Initialize Mobile Application**		10/21/18	10/27/18		0%
1.3.1	Starting Application	Group				
1.3.2	Device	Group				
2	Program Development (Pi)					
2.1	**Data Source Retrieval**	Maurice Jones	11/4/18	11/11/18		0%
2.1.1	Tool Selection					
2.1.2	Scan Configuration					
2.2	**Setup Cron Job**	Austin Vaive	11/11/18	11/17/18		0%
2.2.1	Setup Schedule for Cron Job					
2.3	**Configure Scan**	Matthew Cooley	11/11/18	11/17/18		0%
2.3.1	Create Data format					
2.3.2	Parse Logs					
2.3.3	Send parsed Data to Database					
3	App Development					
3.1.1	**Create User Interface**	Maurice Jones	10/21/18	10/27/18		0%
3.2	user login					
3.2.1	**Display Parsed Info**	Matthew Cooley	11/18/18	11/24/18		0%
3.2.2	Monitoring					
3.3	Forecasts					
3.3.1	**Retrieve & Parse Info for Display**	Matthew Cooley & Maurice Jone	11/18/18	11/24/18		0%
3.3.2	Chart Updates					
4	FireBase Dev					
4.1.1	**FireBase Database Setup**	Matthew Cooley	11/4/18	12/4/18		0%
4.1.2	Windows 10					
4.2	Purchase Manager					
4.2.1	**Secure Transmission of Data**	Matthew Cooley	11/4/18	12/4/18		0%
4.2.2	MS Exchange					
4.3	MS Office					
4.3.1	**Connecting Pi & Mobile App thru FireBase**	Maurice Jones	11/4/18	12/4/18		0%
4.3.2	MS Windows Server 2016					
5	Prototype of Vigil		12/7/18			



PROJECT TITLE	Vigil	COMPANY NAME	Vigil Scans
PROJECT MANAGER	Austin Vaive, Maurice Jones, Matthew Cooley	DATE	1/3/2019

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION	PCT OF TASK COMPLETE
1	Prototype Tweak					
1.1	**Hardware Updates**	Group	1/7/2019	4/1/2019		100%
1.1.1	Hardware Compatability Testing					
1.1.2	Software Updates					
1.2	**Vigil Marketing**	Group	1/14/2019	1/21/2019		100%
1.2.1	Aesthetic changes for Vigil					
1.2.2	Creating Logo					
1.3	**Mobile Application**	Group	1/7/2019	1/21/2019		100%
1.3.1	Mobile Application aesthetics					
1.3.2	User Creation					
2	Program Improvements (Pi)					
2.1	**Data Source Retrieval**	Austin Vaive	1/14/2019	1/21/2019		100%
2.1.1	Tool Selection					
2.1.2	Scan Configuration					
2.2	**Setup Cron Job**	Austin Vaive	1/14/2019	1/21/2019		100%
2.2.1	Setup Schedule for Cron Job					
2.3	**Configure Scan**	Matthew Cooley	1/21/2018	1/28/2019		100%
2.3.1	Create Data format					
2.3.2	Parse Logs					
2.3.3	Send parsed Data to Database					
3	App Development Continued					
3.1.1	**User Interface**	Maurice Jones	2/4/2019	2/11/2019		100%
3.2	user login & Front End					
3.2.1	**Display Parsed Info**	Matthew Cooley	2/11/2019	2/18/2019		100%
3.2.2	Monitoring & Backend					
3.3	Forecasts					
3.3.1	**Retrieve & Parse Info for Display**	Matthew Cooley & Maurice Jor	2/18/2019	2/25/2019		100%
3.3.2	Chart Updates					
4	FireBase Dev Continued					
4.1.1	**FireBase Database Setup**	Matthew Cooley	2/4/2019	2/11/2019		100%
4.1.2	Google Setup					
4.2	Integration through Web App					
4.2.1	**Secure Transmission of Data**	Matthew Cooley	2/11/2019	2/18/2019		100%
4.2.2	Raspberry Pi					
4.3	Creation of Logs					
4.3.1	**Connecting Pi & Mobile App thru FireBase**	Maurice Jones	2/18/2019	2/25/2019		100%
4.3.2	Visual Studio					
5	Completed Vigil and Preparation for Expo					
5.1	**Vigil Oral Presentation**	Entire Team	2/25/2019	3/12/2019		100%
5.1.1	Preparation of Presentation Slides					
5.1.2	Complete Working Demo					
5.2	**Tech Expo**	Entire Team	3/5/2019	4/9/2019		100%
5.2.1	Tweaking of Vigil App					
5.2.2	Recommended changes from Oral Presentation					
5.3	**Final Report**	Entire Team	4/1/2019	4/19/2019		100%
5.3.1	Update of Gantt Chart					

Figure 3: Project Schedule

2.4 Problems Encountered

This project, like any other had its share of problems and setbacks. Here we will note some of the more major setbacks and challenges that the Vigil team faced in the process of creating this project.

2.4.1 Android Studio

A few weeks into the project it was discovered that Android Studio required a Intel processor to run its emulator. This led the team to search for different development paths for the front-end. The solution was making a web application that ran with Apache Cordova. Not only did this provide a work around for Android Studio, it allowed for the simultaneous development of Android, IOS and desktop versions of the application.

2.4.2 Log Formats

Midway through the project the format of the security logs needed to be changed from XML to CSV. This allowed for use of better scanning tools. However, the back-end needed a new plugin to process the new format of CSV files.

2.4.3 Learning new Systems

As this project is a vehicle for learning, each member ended up taking on items that they had never used before: Python, NodeJS, Apache Cordova. Because of this the ramp up time was longer than expected.

3. Technical Elements

3.1 Scanning Appliance (Hardware)

For our proof of concept, we needed a small device capable of connecting to a network and running an operating system that is compatible with the scanning tools we planned on using. This led to us deciding to use a Raspberry Pi 3 B loaded with Kali Linux as our network scanning appliance

3.2 Application

The Vigil application utilizes both hardware (Raspberry Pi) and software (our scanning software, database and user interface). We have also created a mobile application that passes the information gathered by the scanner to the user via an intuitive user interface. The mobile application, the database and the Raspberry Pi are in constant communication.

3.3 Database

For the backend database that connects the Pi and mobile application side of Vigil we are using Google Firebase. We are using this service for multiple reasons. First of all Firebase was developed with mobile applications as a priority. Because of this it is optimal for receiving notifications.

3.4 Security

All logs generated by Vigil will be encrypted while at rest, and in transit using industry standard methods

4.0 Application

4.1 Application Architecture

The user-facing application was created in VisualStudio and was designed to be hosted as a web application for compatibility with a large number of devices. This does limit our ability to send push notifications like a traditional application, but we found the benefit of device compatibility was a worthy tradeoff.

4.2 Screen Shots and Examples

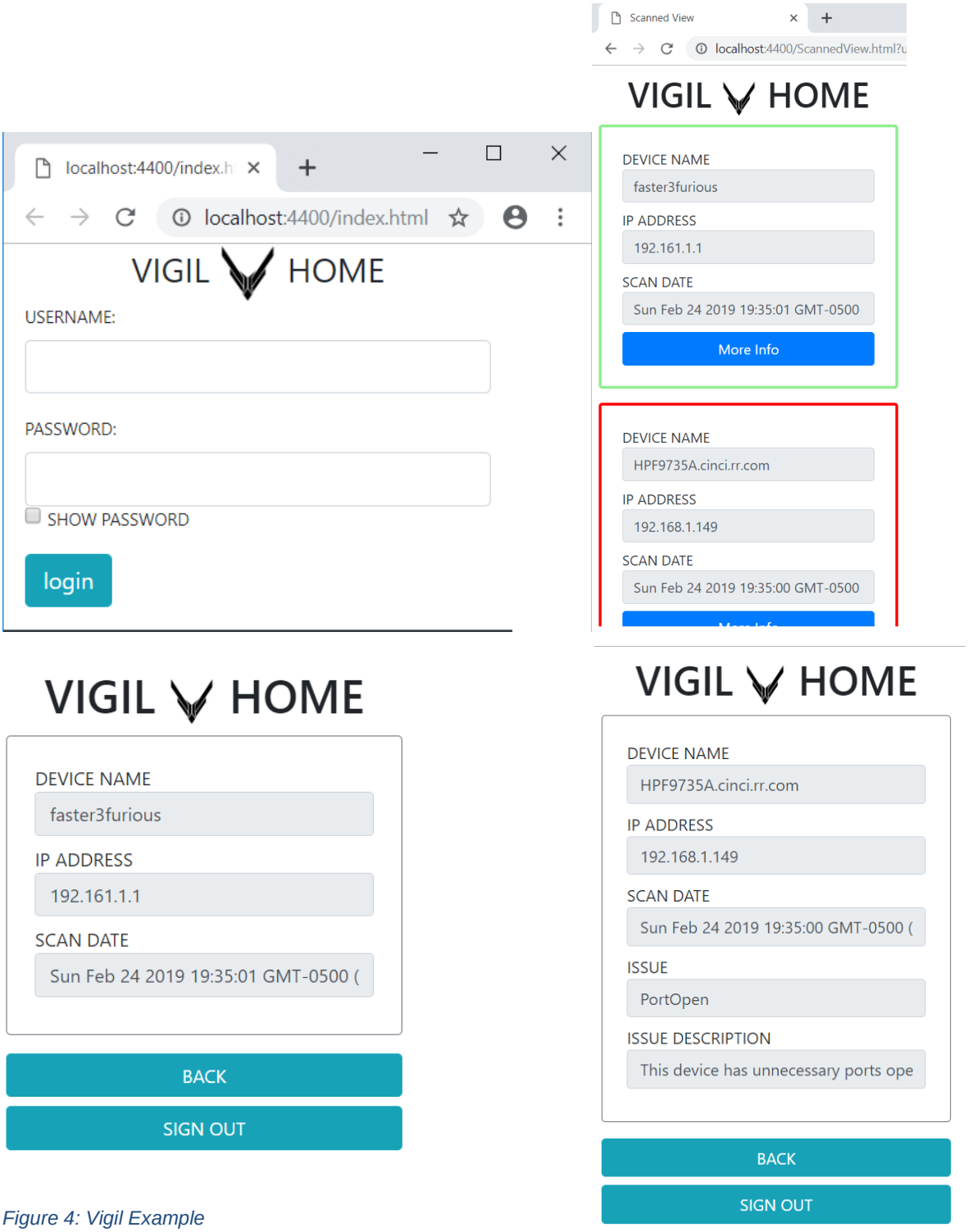


Figure 4: Vigil Example

4.2.1 Application Homepage

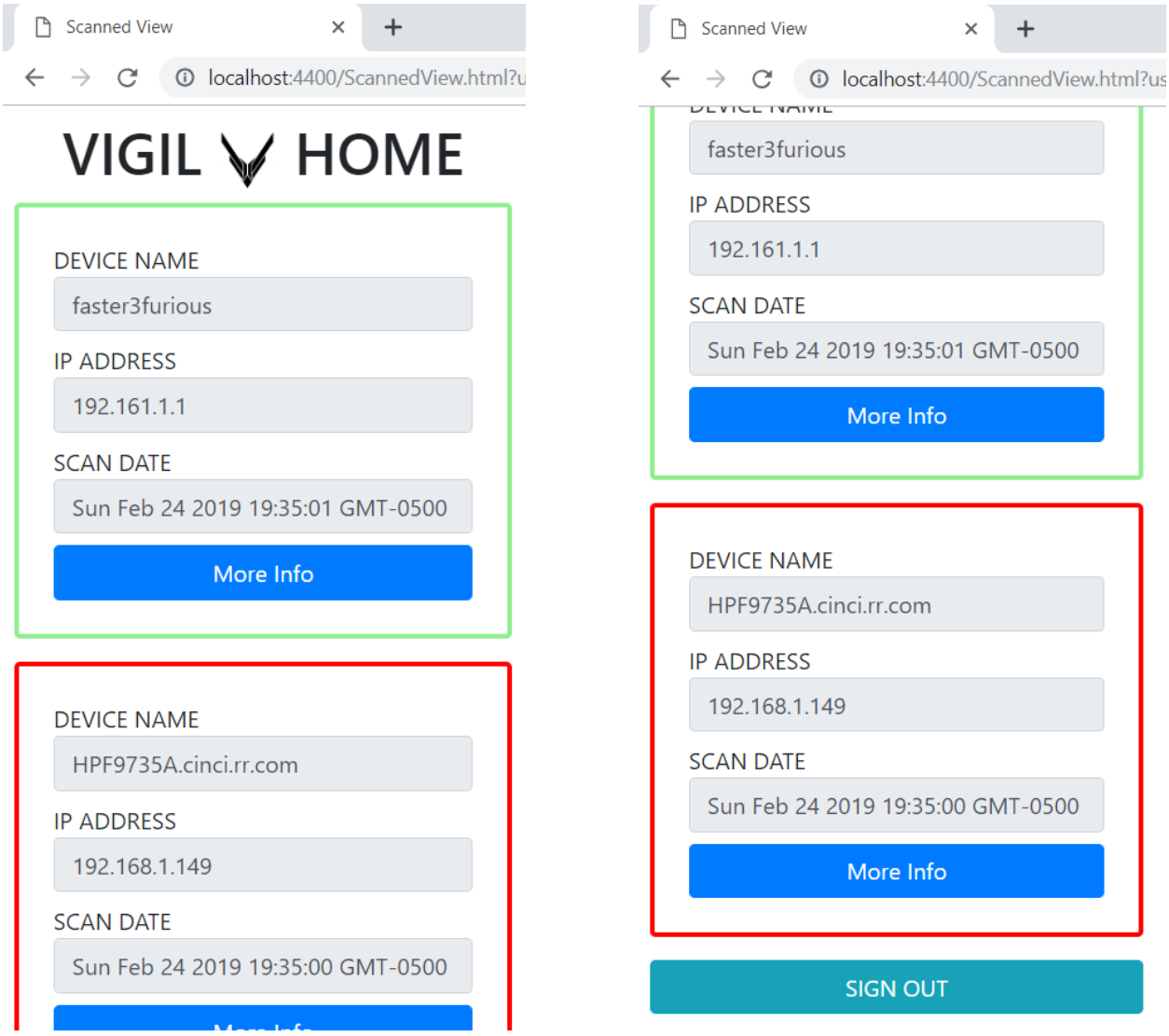


Figure 5: Vigil Home Page

5. Test Plan

5.1 Overview / Methodology:

In order to test Vigil, we wanted to simulate a typical home network, which consists of a variety of devices in varying degrees of vulnerability. In order to ensure that Vigil will catch vulnerable devices, we tested it in a real home network that was seeded with several intentionally out of date Virtual Machines. We will run Vigil on the network for enough time for the scanners to fully map all of the devices, and then compare the results it found to what we already know about the network.

5.2 Scope of Testing:

In this test, we will be examining what we determine to be a realistic mix of devices on a realistic home network. We will not be testing an exhaustive list of devices and operating systems, as this would be resource and time intensive. We will test devices using the major consumer operating systems (Windows and macOS), as well as other standard devices such as printers, smart phones, and smart home / IoT devices.

5.3 Objective:

The objective for this test is to verify that our product will successfully detect vulnerable devices and display the results as expected in the app and that there are no anomalies or devices that are skipped.

5.4 Entry and Exit Criteria

5.4.1 Entry

- Scanning program is running reliably
- Scanning program is formatting results properly
- Application is displaying results correctly

5.4.2 Exit

- Scans are completed properly
- Any bugs or hiccups are documented

5.5 Logging Test and Reporting

After the completion of the test, the team discussed the outcome and determined what bugs required the most attention to resolve. We then divided up the tasks based on severity, worked to resolve the bugs, and tested again.

5.6 System Testing

We first tested the scanner module separately from the rest of the project, because any bugs that occur with the scanner will stop the rest of the system from working as intended.

Once the scanning module was confirmed to work properly, we then tested the system as a whole to ensure it was working properly.

5.7 Testing Procedures

The test that we will conduct will follow these steps:

5.7.1 Reliability Test:

Will the program start / run reliably without crashing or hanging up?

- Yes, the program was left running for several days running constant scans and there were no issues.

5.7.2 Accuracy Test:

Will the scanner catch all vulnerable and out of date devices on the network to a reasonable degree of error?

- The scanner caught any device we connected to the test network, but could only provide the information it was able to collect from the scanning plugins.

5.7.3 User Interface Test:

Is the interface easy for the average user to understand and navigate?

- Yes. We went through several phases of our application's design and settled on one that both gave the user enough detail when required, but also still maintained a clean design language fitting of a modern application.

5.8 Pass / Fail Conditions

It is expected that Vigil will pass all tests to be determined successful. If a test is failed, the team will work to resolve the issue.

- All above tests were determined to have passed.

5.9 Risks

The following are risks that we must address during testing:

- Network availability on the test network and devices
- Launching penetration tests against devices could result in instability on the devices being scanned.

Since we wrote the program to have a delay built in between scans, we did not witness any noticeable negative effects on the network itself or the devices being scanned.

5.10 Test Results

Our test process resulted in Vigil successfully scanning the network and detecting all present devices. There were no observed network or device unreliability issues like we

had originally anticipated. The scanner detected every test device that we placed on the network and was able to display their IP address and hostnames.

ip	hostname(s)
192.168.1.1	faster3furious
192.168.1.149	HPF9735A.cinci.rr.com
192.168.1.165	Google-Home-Mini.cinci.rr.com
192.168.1.166	LGwebOSTV.cinci.rr.com
192.168.1.212	Courtneys-MBP-2.cinci.rr.com
192.168.1.228	AustinVesiPhone.cinci.rr.com
192.168.1.238	Chromecast.cinci.rr.com
192.168.1.240	kali.cinci.rr.com
192.168.1.242	iPhone.cinci.rr.com
192.168.1.69	

Table 2: Test Devices

The user interface test was also passed, which was verified by allowing several non-tech savvy users navigate the application and recording any difficulties encountered.

Test Procedure	Pass	Fail
Reliability Test	X	
Accuracy Test	X	
User Interface Test	X	
Overall Pass/Fail	X	

Table 3: Test Results

6. Conclusion

Designing Vigil was a challenge for all of us on our team. We knew from the very beginning that the concept and theoretical function of the project would be possible with the tools at our disposal. We also knew that we could fill a genuine need with our project if we were able to successfully complete our goals. In the future, we hope to continue producing Vigil and hopefully, we will be able to help protect the networks of a large number of people.

During the course of this semester we have learned lessons in group work as well as lessons of completing a project of this magnitude. Over the course of the fall semester we were able to learn about what is possible within scope of a project and to be able to be flexible when things do not go the way that we had originally planned. We faced several issues where no one on our team felt that they were an expert in a particular subject, but we took those opportunities and used them to learn. For example, no one on our team had much experience with programming in Python. We knew that would be the ideal language for the scanner because of the packages available, but it was a large hurdle that Austin (Cybersecurity) had to overcome in order to get the scanner running.

During the spring semester we learned the importance of preparation and the presentation of a finished product We all individually had the pieces of the project working independently but making all the individual parts of the project come together and flow was more difficult than we originally anticipated. We also focused on making our front-end appealing to a user, and making our app appear modern and easy to use. None of us had much experience with designing a UI, but Maurice did research and made a clean, simple UI that still displays the necessary information. Getting the Firebase Database to reliably pull data from the Pi was also a struggle, due to the limited format that we were able to export the logs with from the Python program. Matt had to develop a program that would parse the data in a .csv file and update fields in the Firebase database.

We finalized our work by presenting to our peers and the general public at IT Expo. Expo gave us the opportunity to present our project in an organic way and get insight and feedback from other people. Overall, this project has improved our technical abilities, our abilities to participate in large-scale group projects, as well as our presentation and professional communication abilities.

Appendix A: Additional Information

The team contacted the following individuals for guidance and additional information:

1. Tyler Hopperton, Advisor

Hardware Utilized:

- We utilized a Raspberry Pi Model 3 B+ as our scanning appliance because it is relatively inexpensive but still feature-rich. Due to the built in Wireless compatibility, it made it an even better choice because the end user does not necessarily even need to have the scanner directly wired to their router.

Software Utilized:

- The Raspberry Pi was running Kali Linux due to our familiarity and its reputation as a secure Linux OS.
- The scanner was written in Python 3 and called the python-nmap package as well as the OpenVAS CLI package. These two Python packages allowed us to easily implement existing open source tools into our solution.
- Our backend database used Google's Firebase as is has a quick response time and is easily tied to all the other software we are using.
- The Raspberry Pi also had a Node JS application running on it to extract information for the security logs and post it to the database.

Appendix B: References

1. Martin, Chuck. “North American Consumers To Have 13 Connected Devices.” MediaPost, 11 June 2017, www.mediapost.com/publications/article/302663/north-american-consumers-to-have-13-connected-devi.html.

Appendix C: Poster and Photos

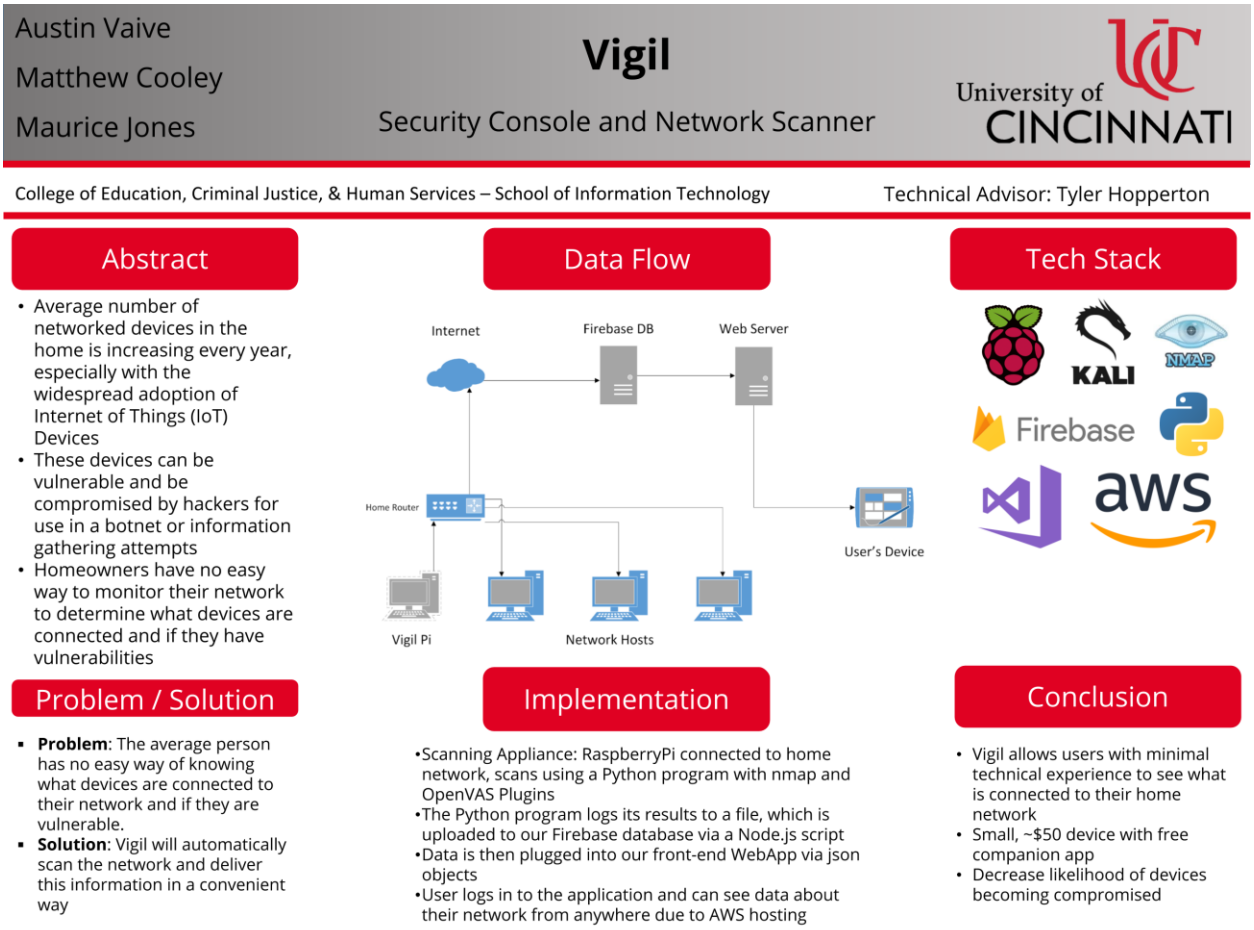


Figure 6: Tech Expo Poster

Figure 6: Vigil Pi Device

```python
#!/usr/bin/env python3

import os, sys, nmap, argparse, datetime, ipaddress, socket, csv, datetime, time
from tempfile import NamedTemporaryFile
import shutil
from lib.cLogging import Logger
from modules.data_manager import DataManager

class Scanner():
    def __init__(self, ips):
        self.logger = Logger()
        self.ips = ips
        self.hostL = []
        self.scriptPath = os.path.dirname(os.path.realpath(sys.argv[0]))
        self.scanPath = os.path.join(self.scriptPath, 'scans')

    def scan(self):

        nm = nmap.PortScanner()
        self.logger.writeToLog('STarting nmap scan', 'debug')
        nm.scan(hosts=self.ips, arguments='-n -sP -PE -PA21,23,80,3389')
        hosts_list = [(x, nm[x]['status']['state']) for x in nm.all_hosts()]
        dataM = DataManager()

        print()
        print('[*] Processing scan info...')
        self.logger.writeToLog('Processing scan info...')

        for host, status in hosts_list:
            self.hostL = dataM.process(host, status)

        dataM.write(self.hostL)
```

Figure 7: Code Snippet from Scanner