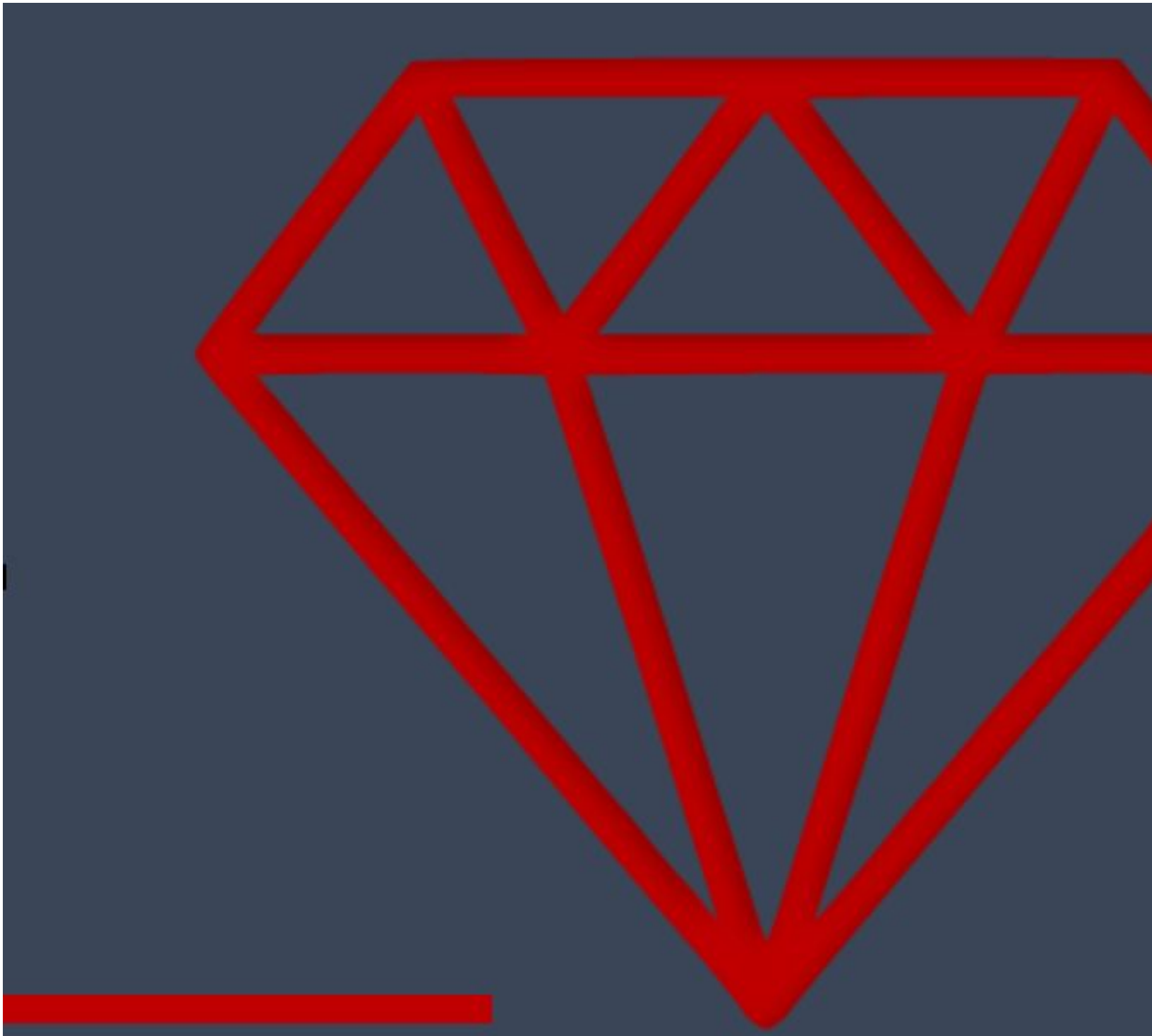


SCADA Diamond Vault

Operating Manual

Jeremy Boyd
Hannah Hanback
Owen Sanders
Evan Stewart

Table of Contents

Introduction

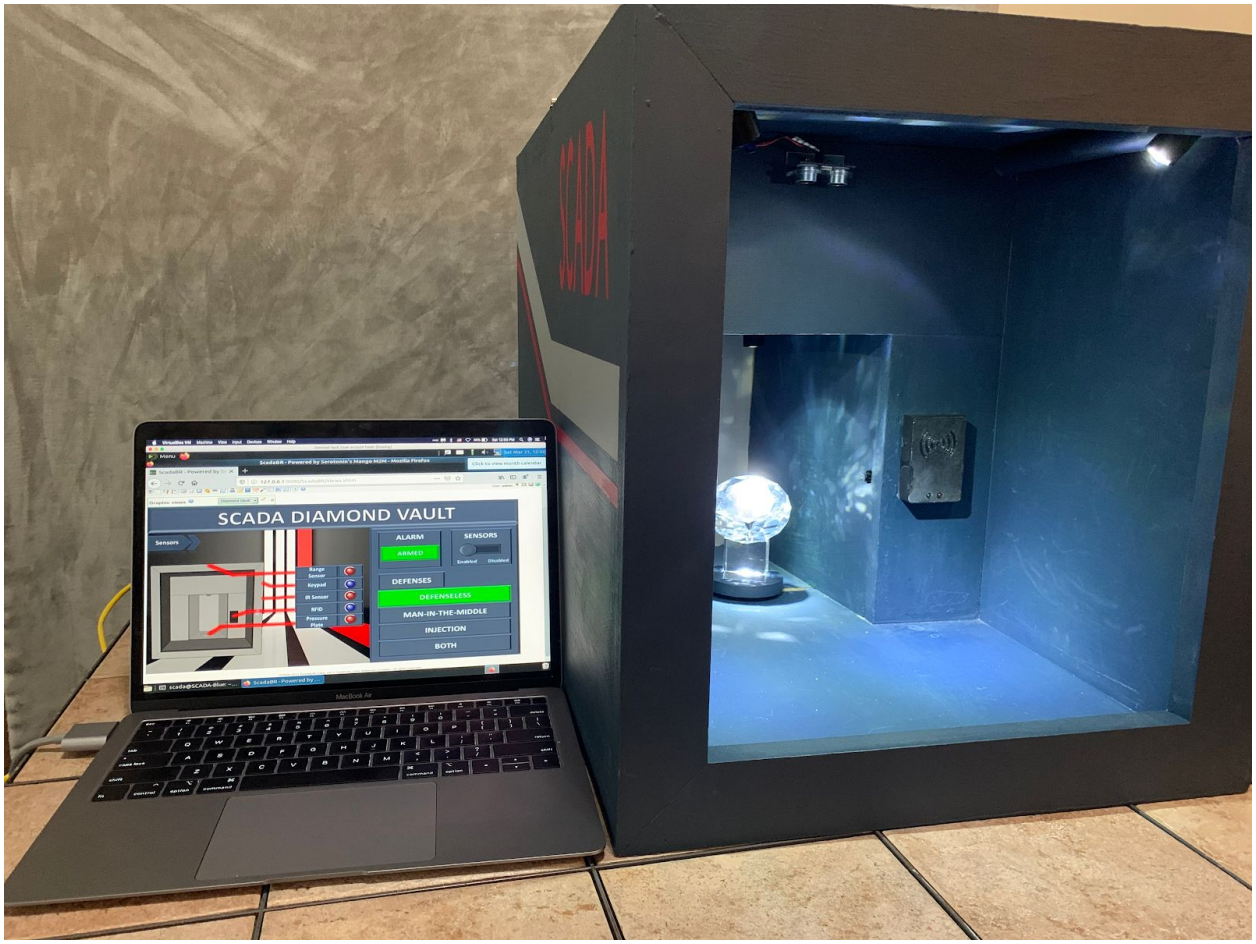
Summary

Cybersecurity is a continuously growing and ever-changing field inseparable from our everyday lives. Everything we do has some sort of cybersecurity concern from health records at the doctor's office, to sending an email to your colleague, to buying lunch with our credit cards, everything is vulnerable. Unfortunately, not many people are aware of the dangers present in the cyber world. There is an increasing need to educate the world on these threats and how to protect themselves. For this purpose, the SCADA Blue Team created a demonstration of cybersecurity attacks to display the implications in a real-life situation. The project fits inside a portable flight case and is controlled using Arduinos and Raspberry Pis running OpenPLC software. This demonstration imitates a museum diamond vault guarded by an IR laser alarm system, an RFID reader, an ultrasonic range sensor, and a pressure plate underneath the diamond. The cybersecurity attacker/demonstrator will implement Man-in-the-Middle and Injection attacks to bypass the security and steal the diamond by interfering with the logic controller and/or the Human Machine Interface. Togglable defenses have also been implemented to show how these attacks could be prevented. In this way, we hope to educate more people about vulnerabilities in current systems and why cybersecurity is so important.

SCADA Background

Supervisory Control and Data Acquisition (SCADA) is a system for collecting and analyzing information in real-time. The typical setup for this consists of a Programmable Logic Controller (PLC), a Human Machine Interface (HMI), and one or more sensors. A common communication method for these systems is known as Modbus TCP. Through data packets, the HMI can communicate with slave devices by reading or writing to registers. However, there is no real integrity check with these packets: introducing vulnerabilities to the system. Packets can be manufactured by someone pretending to be a Master device or can even be manipulated in transit through a Man-in-the-Middle attack. These SCADA systems are used throughout many industrial applications such as power plants, nuclear facilities, and manufacturing. Though awareness has been increasing for these vulnerabilities, many legacy systems are still in place: whether for ease of use or monetary concerns.

SCADA Diamond Vault



Team Members

Jeremy Boyd - jsb0028@uah.edu: Team Lead, Attack/Defense development

Hannah Hanback - hyh0001@uah.edu: HMI/Vault design, Attack/Defense research

Owen Sanders - wos0001@uah.edu: Sensor hardware/software design, PLC ladder logic design, Attack/Defense development

Evan Stewart - ens0013@uah.edu- Vault construction, Attack/Defense research

Startup Guide

This project has been designed for easy and quick setup. If any issues come up during this process, please consult the troubleshooting guide.

Setting up the Vault

Place the diamond and its stand on the pressure plate. Then plug in the vault: the power cable is on the back. The lights should immediately turn on. If they don't, refer to the Troubleshooting Guide.

After around **30** seconds, the vault should be set up, and the alarm active. Do a quick test by putting your hand through the opening on the front. If the alarm doesn't go off, refer to the Troubleshooting Guide.

Setting up the HMI

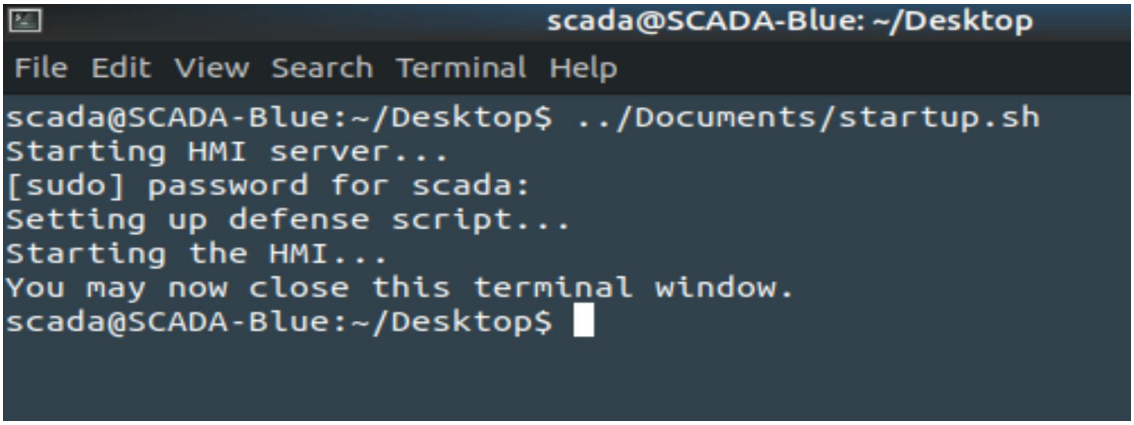
Plug your computer's ethernet port into the blue port on the back of the vault.

Select the "Human Machine Interface" VM image in VirtualBox, click start, and it should automatically log in.
The credentials for this system are:

> User: scada
> Password: scadablue

Go to the settings and select the network tab in Virtualbox. Make sure it is set to a bridged adapter, and the vault's Ethernet connection is selected.
Once the desktop has loaded, right-click on it and click "*Open in Terminal*."

In the terminal window, type ../Documents/startup.sh. Type the password from above, hit enter, and wait for a few seconds. ScadaBR will take a while to load after firefox starts up.

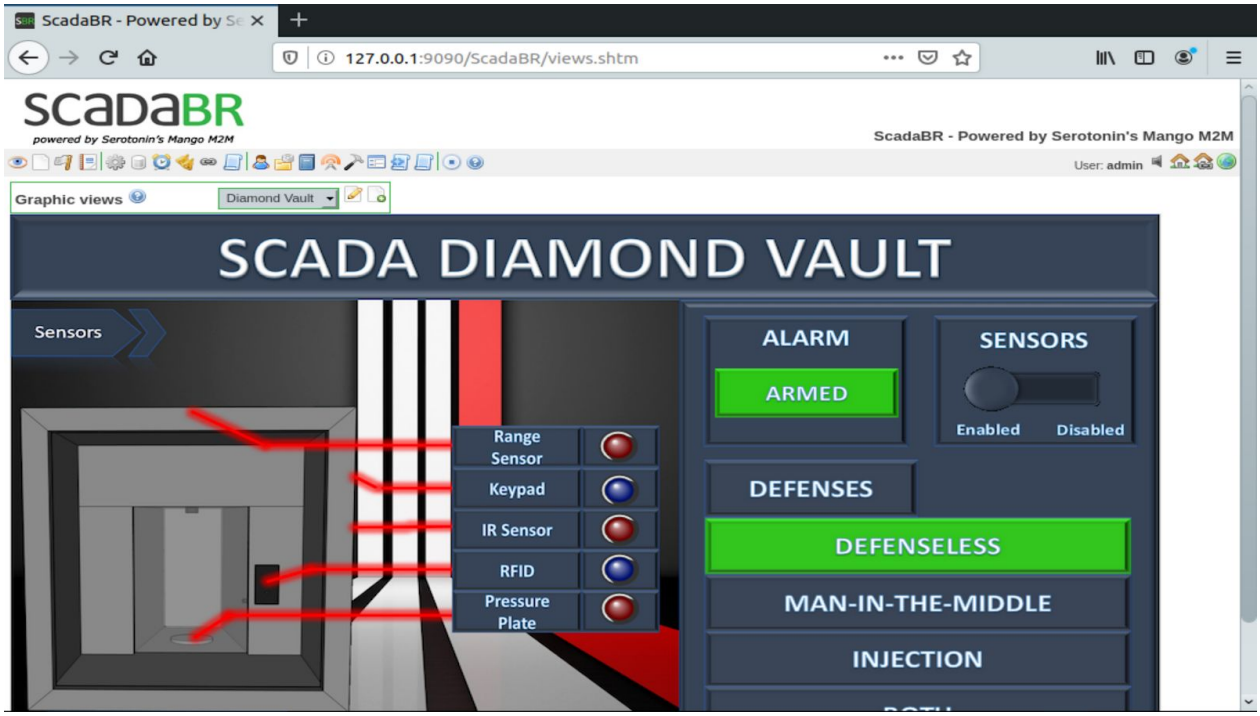


If you do not see the screen below after a few minutes, refer to the Troubleshooting Guide.

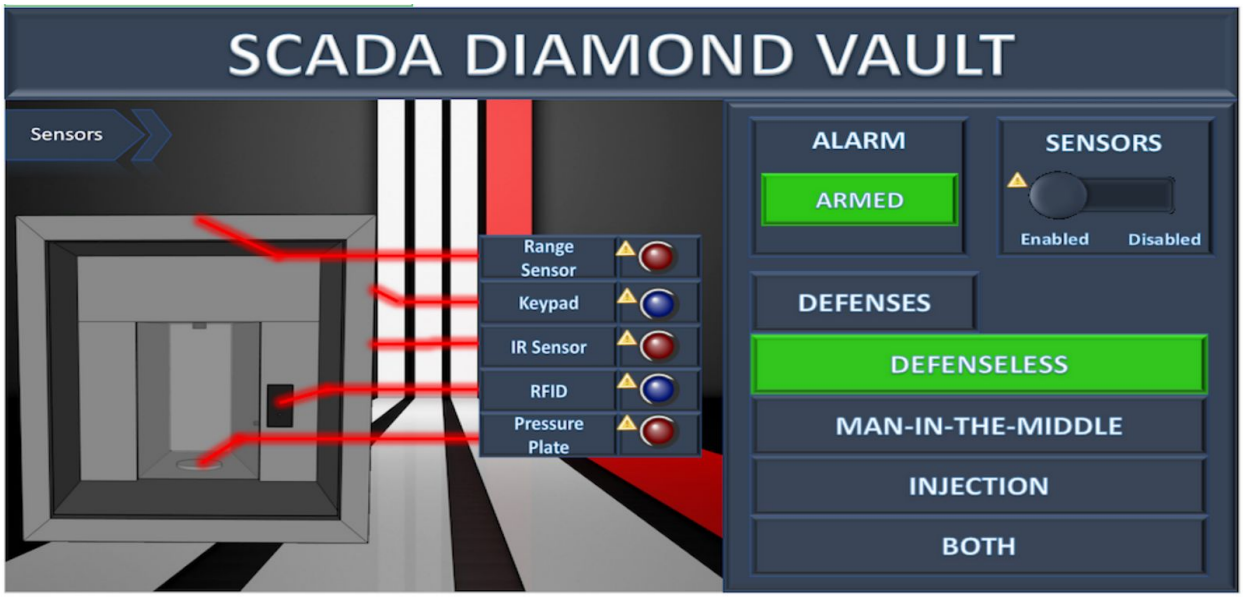
Login to ScadaBR with the credentials:

Username: admin
Password: admin

You should now see the main screen for the HMI. If not, click the paper icon in the bar at the top of the screen.



The HMI should now be showing the information being sent from the PLC. If you see yellow warnings around the HMI, like the picture below, refer to the Troubleshooting Guide.



Setting up the Attacker

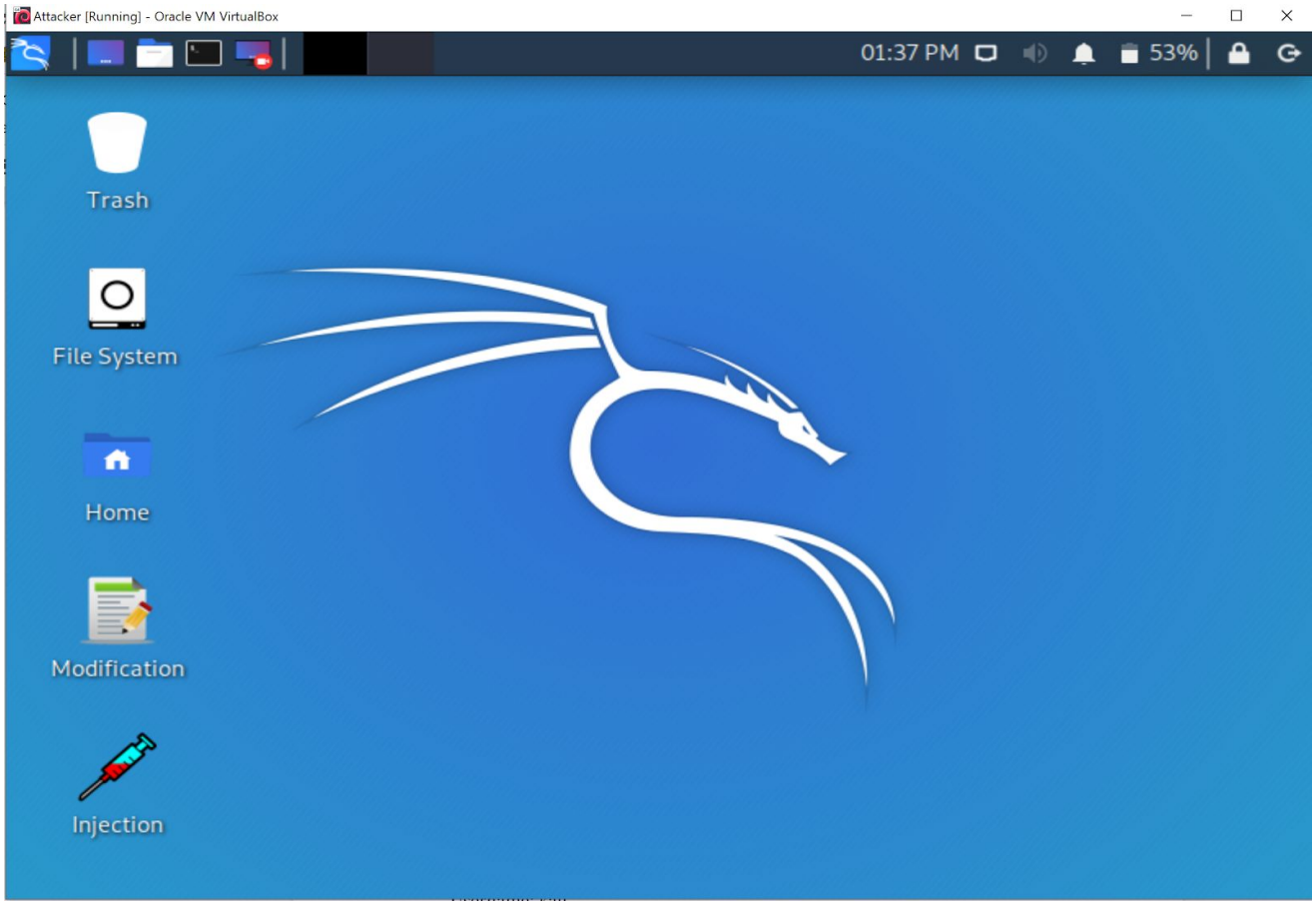
If you are using a separate machine for the attacker, plug it into the red ethernet port on the back of the vault. If you plan to use the same computer to run both the HMI and attacker, move on to the next step.

On the machine running the attacker (either the same as the HMI or another), open up the “Attacker” VM image in VirtualBox. Go to the settings and select the network tab in Virtualbox. Make sure it is set to a bridged adapter, and vault’s Ethernet connection is selected.

Click start and log in with the credentials:

Username: kali
Password: kali

You should now see the attacker’s desktop.



All systems are now set up, and you are ready to move on to the attack demonstrations.

Operation

Normal Operation

After following the steps laid out in the Startup Guide, the vault should be in its normal operating state. The alarm will be triggered if you put your hand inside the vault, and this will be reflected on the HMI screen. To disable the alarm and allow access to the second area, you must type the correct code into the keypad. The code is **2487**. After typing this in, the green light will turn on, indicating the alarm has been disabled for all but the pressure plate. To enable the alarm, you will type the same code. If an incorrect code is entered, a red light and buzzer will be active for about a second, and you will be able to try again.

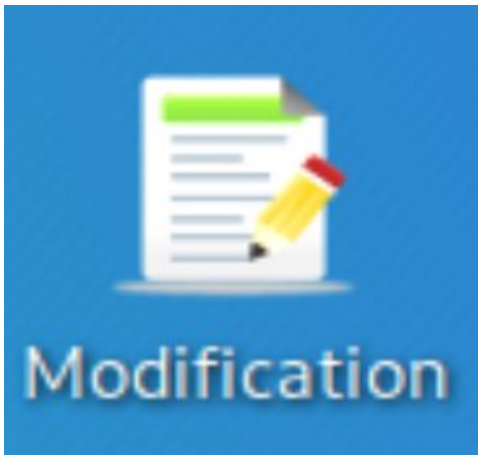
To disable the pressure plate, you will need to use the correct RFID card. A correct card will turn the green light on, and the diamond will then be safe to remove. To enable the pressure plate, scan the correct RFID card again. The green light will now turn off. Keep in mind that the alarm will go off if the diamond is not on its stand at this point.

Attacks

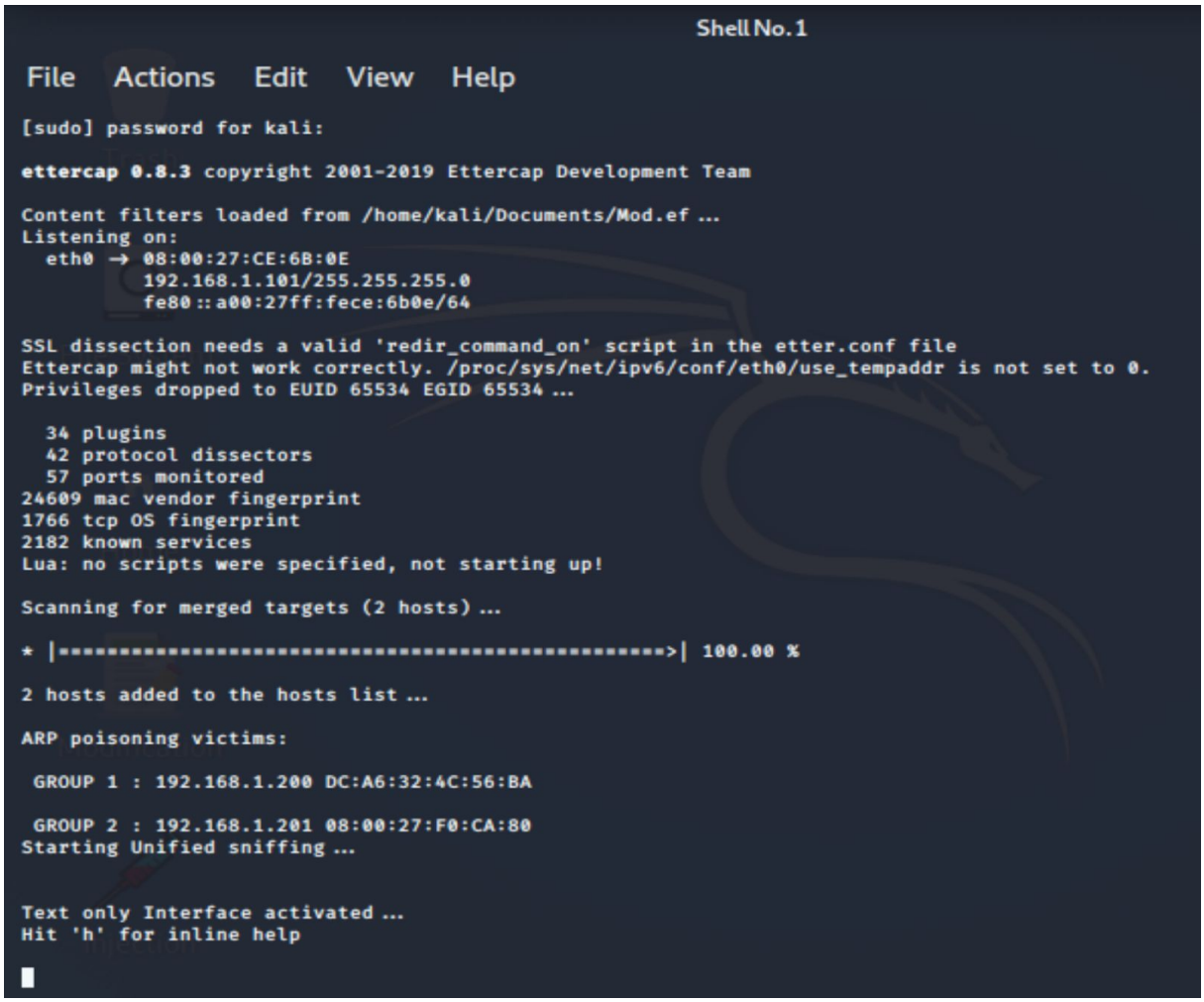
Modification Attack:

This attack will cause the HMI to see all inputs and coil values from the PLC as operating normally. The vault will still function properly, but no alarms will be reflected in the HMI's screen.

To run this attack, double click the desktop icon named **Modification** on the attacker VM.



Doing this will open a terminal requesting the user password - type **kali** and enter. The attack will now be running.



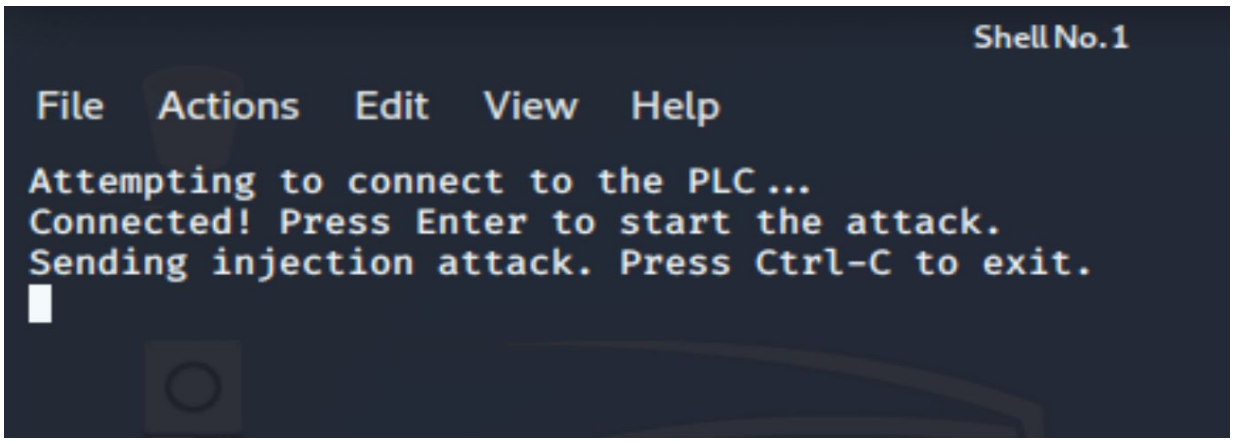
To stop the attack, press **q**.

Injection Attack:

This attack continually sends packets from the attacker to the PLC to set a value and disable the sensors.

To start this attack, double click the desktop icon named **Injection**.

A terminal window will appear, and the attacker will attempt to connect to the PLC. If it succeeds, press Enter to launch the attack.



To stop the attack, press Ctrl-C in the terminal window.

Defenses

The vault will default in defenseless mode. To turn on a specific defense mode, type **#[Letter]** and the code **2487** to the vault.

The letter you choose corresponds to a certain defense:

- **#A (Defenseless)**
- **#B (Man-in-the-Middle Defense)**
- **#C (Injection Defense)**
- **#D (Both Defenses)**

For example, if you would like it to be put in Injection defense mode, you would enter **#C** and then **2487**.

These defenses will be described further in the Attacks and Defenses Explained section below.

Attacks and Defenses Explained

Modification Attack:

Modification attacks are considered a type of integrity attack in which the data has been modified or changed from what it originally was. These can happen in the realm of networks or on a system's files. In our case, we are modifying the Modbus packets that are traveling from our Raspberry Pi to our Virtual Machine (VM) running our Human Machine Interface (HMI). These modified packets trick the HMI into thinking that the alarm and sensors are not going off without alerting it that there is an attack taking place.

This attack is done from our attacker VM, where we launch a script by clicking an icon titled 'Modification' on the desktop. This script runs the following command:

sudo ettercap -Tq -M arp /192.168.1.200/ /192.168.1.201/ -F Mod.ef

The command calls Ettercap, a software capable of performing Man-in-the-Middle attacks, that ARP poisons the target. ARP poisoning is where an attacker responds to packets every time a 'Who is <IP address>?' packet is sent out. Basically, the attacker will always respond to these packets quickly, saying that their MAC (Media Access Control) address is connected to all the target IP (Internet Protocol) addresses in the network. Then it makes all the packets that would flow between the target IP address be sent to the attacker first. Then the attacker can forward these packets just to eavesdrop or they might stop the packets from reaching the intended target by not sending anything received. The attacker can even change the contents of a packet to anything they want and then forward it. What we are doing for our Modification attack is the last option, which is explained below.

Once the targets are ARP poisoned, the attacker will be receiving all the packets from the two poisoned targets. We then use an Ettercap filter that modifies the actual data in the packets. This attack is possible due to Modbus/TCP's lack of integrity checking. So, for our modification attack, we modify the packets coming from the Pi to the HMI and change the last few bits in the packets. These bits contain all the sensor data coming from OpenPLC running on our Pi. So, when we modify these bits, we can set them all to be 0's to make it seem like the sensors aren't being triggered.

Once the bits are set, we can easily reach into the vault and take the diamond without fear of an administrator being alerted. While the physical alarm in the vault will still be active, the HMI will not show that the alarm is going off or that any sensors are tripped.

Modification Defense:

To defend against this modification attack, we can use static ARP table entries.

As explained in the Modification attack portion of this document, the ARP poisoning will respond to the requests of 'Who is <IP address>'. Connected devices use these responses to update a local table, called the ARP table. These tables contain IP addresses and the corresponding MAC addresses for the other devices on the network. This is how each device knows where to send their packets and who they received a packet from. The way to stop this is to have statically defined ARP tables entries. These entries cannot be changed by requests and responses to 'Who is <IP address>' packets.

This defense can be activated by typing '#B' (or '#D' for all the defenses) and then the password on the keypad. Typing this runs the following command on the Raspberry Pi:

arp -s 192.168.1.201 08:00:27:F0:CA:80

The Raspberry Pi will then send a message to the HMI on a different port (port 5005) to have the HMI run a script with the following command:

arp -s 192.168.1.200 DC:A6:32:4C:56:BA

These commands will set static ARP table entries for both the Raspberry Pi and the HMI. The '-s' is a built-in arp command that assigns the first argument, the IP, to the table and matches it with the second argument, the MAC address.

If you want these static ARP table entries removed and want to go back to a defenseless mode, you will type '#A' and then the password on the keypad. Typing this will run the following script on the Raspberry Pi:

arp -d 192.168.1.201

And then once again, the Pi will send a message to the HMI on the same port, 5005, to disable the static entry, by running the following command:

arp -d 192.168.1.200

Once again, these commands will remove the static ARP table entries for both the Raspberry Pi and the HMI. The '-d' command is a built-in arp command that deletes the ARP entries that correspond to the first argument given, the IP address.

Injection Attack:

Injection attacks are a type of attack where you inject packets into a machine from an attacker. These can happen in the realm of networks to a specific system. In our case, we are injecting Modbus packets into our Raspberry Pi. These injected packets are made so that they are constantly sending the signal for the sensors to be turned off. Modbus Protocol does not care where commands are coming from, so this will cause the alarm to never be set off, even if the sensors are triggered.

This attack is launched from our attacker VM where we launch a script by clicking on an icon labeled 'Injection' on the desktop. The script contains:

```
import socket

TCP_IP = "192.168.1.200"
TCP_PORT = 502

BUFFER_SIZE = 1024
ON_MESSAGE = b"\x00\x00\x00\x00\x00\x06\x01\x05\x03\x21\xff\x00"
OFF_MESSAGE = b"\x00\x00\x00\x00\x00\x06\x01\x05\x03\x21\x00\x00"
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
print("Attempting to connect to the PLC...")
try:
        s.connect((TCP_IP,TCP_PORT))
except:
        input("Could not connect! Press Enter to exit.")
        exit()
input("Connected! Press Enter to start the attack.")
print("Sending injection attack. Press Ctrl-C to exit.")
while True:
        try:
                s.send(ON_MESSAGE)
                s.recv(BUFFER_SIZE)
        except:
                s.send(OFF_MESSAGE)
                s.recv(BUFFER_SIZE)
                s.close()
                exit()
```

This script is a python code that creates a TCP message and mimics the Modbus protocol. We then add the exact message we want to the data portion of the TCP message we have created, to tell OpenPLC that the sensors should be disabled.

Once a message has reached OpenPLC, we will be able to reach into the vault and take the diamond out without setting off the alarm. The HMI shows that the sensors have been disabled, and when you reach into the vault, the alarm signal will not show that it was triggered. The administrator can still see that something is wrong, but the physical alarms will not be going off.

Injection Defense:

To defend against the injection attack, we will use IP table rules in our local firewall.

While the injection attack is active, the PLC is receiving the fabricated packets from the attacker whose IP is different from the normal IP of the HMI. Therefore, if we limit packets coming into the Raspberry Pi to only ones from the HMI's IP address, then we can stop injection attacks. This is what the IP table rules can do.

To activate this defense, you will need to type '#C' (or '#D' for all the defenses), followed by the password on the keypad. Typing this will run a script enabling the rules to be set by running this command:

sudo iptables-restore < /etc/iptables.test.rules

The rules we are running are in the file in the /etc directory called *iptables.defenses.rules*. The contents of this file are as follows:

```
*filter

#Drop everything but our output to the internet
-P FORWARD DROP
-P INPUT DROP
-P OUTPUT ACCEPT

#Allow established connections (the responses to our outgoing traffic)
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

#Allow local programs that use loopback
-A INPUT -s 127.0.0.0/8 -d 127.0.0.0/8 -i lo -j ACCEPT

#Allow only the HMI (ScadaBR) to talk to this PLC
-A INPUT -s 192.168.1.201 -p tcp --dport 502 -m state --state NEW -j ACCEPT

#Allow SSH connections
-A INPUT -p tcp -m state --state NEW --dport 22 -j ACCEPT
```

#Allow ping
#-A INPUT -m icmp -m --icmp-type 8 -j ACCEPT

#Allow HTTP and HTTPS connections
-A INPUT -p tcp --dport 80 -j ACCEPT
-A INPUT -p tcp --dport 443 -j ACCEPT

COMMIT

If you then want these IP table rules removed, type '#A' followed by the password on the keypad, which will run the following:

sudo iptables-restore < /etc/iptables.reset.rules

This command will set the IP table rules from the file *iptables.reset.rules* to the following rules:

**filter*

#Accept everything
-P FORWARD ACCEPT
-P INPUT ACCEPT
-P OUTPUT ACCEPT

Both Attacks:

We can implement both the Modification and Injection attacks at the same time. The Modification attack will allow us to get the diamond out without the HMI alerting any administrators/monitors that something is happening, and the Injection attack will let us take the diamond without the physical alarm on the vault going off. So, if we launch both of the attacks, we will be able to get the diamond out without the alarm or the HMI knowing that an attack is happening.

The best way to do this is to launch the Modification attack first to stop the HMI from showing anything, then launch the Injection attack to stop the physical alarm since the Injection attack will change the toggle switch of the sensors being disabled. Once these are launched, grab the diamond and go.

Both Defenses:

Now, just like we can launch both attacks, we can also have both defenses. To launch both, you will need to type '#D,' followed by the password on the keypad. Typing this will apply and send the signals explained above and run both defenses at once.

Once both defenses are set, the attacker can run any combination of the two attacks, and the HMI and the alarm on the vault will still work.

Troubleshooting Guide

Raspberry Pi Issue

1. Setting Up the Pi
    a. To set up the image for the Pi, download and flash the desired sd card with Raspbian Buster.
    b. Change the default password to scadablue
    c. Set up to Boot into CLI at startup
2. If you are having trouble viewing the screen of the Pi using the HDMI cable, make sure you are using a source that can display it. The Pi 4s at the time of this project was having difficulty displaying on certain screens like TVs.
    a. To log into the Pi remotely, its IP address is 192.168.1.200 and its login info is:

        Username: pi

        Password: scadablue
3. Pinout, and Where to Hook-up the Wire
    a. If any wires come undone, use the following table and figure to know where the wires go. Also, the wires have little tape flags on them to tell you the pin number.

#	Name	Class	Type	Location
1	IR_Sensor	Local	BOOL	%IX0.3
2	US_Sensor	Local	BOOL	%IX0.4
3	PressurePlate	Local	BOOL	%IX0.5
4	Keypad	Local	BOOL	%IX0.6
5	RFID	Local	BOOL	%IX0.7
6	Alarm	Local	BOOL	%QX100.0
7	SensorControl	Local	BOOL	%QX100.1
8	Buzzer	Local	BOOL	%QX0.2
9	Lights	Local	BOOL	%QX0.3
10	Lock	Local	BOOL	%QX0.4
11	TON0	Local	TON	
12	TOF0	Local	TOF	
13	TON1	Local	TON	

Hardware Issues

1. If the lights don't turn on, try the following:
    a. Unplug and replug the box from the power source.
    b. Check to make sure the power strip is in the RESET position and the 12V power supply is plugged in.
    c. Check if the wires have a good connection to the relay and 12V source.
2. If the alarm does not go off when sticking your hand through the front opening:
    a. Unplug and replug the box from the power source and allow about 30 seconds for the pi to start OpenPLC
    b. If other sensors are working, check that the IR LEDs (located on the right and top of the opening) are still in place.
        i. The top one should be in line with the emitter below
        ii. The right one should be in the direct vertical center of the box
    c. If the other sensors are not working, the issues most likely lie in the Pi

i. Make sure all blue wires are connected to the Pi and in the correct spot (See raspberry Pi issues)

ii. Refer to OpenPLC Problems

OpenPLC Problems

1. To Install OpenPLC
    a. OPENPLC Runtime
        - git clone https://github.com/thiagoralves/OpenPLC_v3.git
        - cd OpenPLC_v3
        - ./install.sh rpi
    b. Go to Web Browser: localhost:8080
        - Log in with:

            Username: openplc

            Password: openplc
    c. Click Hardware
        - Select Raspberry Pi in the drop-down
        - Click “save these setting” at the bottom
    d. Pi Problems
        - GPIO Pins code may need updating (wiringPi)
        - https://openplc.discussion.community/post/openplc-on-raspberry-pi-no-control-of-gpio-pins-10358436
        - http://wiringpi.com/wiringpi-updated-to-2-52-for-the-raspberry-pi-4b/
2. To Install OpenPLC Editor
    a. Go to: https://www.openplcproject.com/plcopen-editor
    b. Run the install script
    c. Some python libraries may not have been installed, the following were the ones we had to install:
        - lxml - (sudo apt-get install python-lxml)
        - pyro - (pip install pyro)

- sslpsk - (pip install sslpsk)

3. Sometimes the webserver does not work as intended. If this happens, follow these steps to add and run new programs through OpenPLC runtime in the terminal:
    a. Navigate to the OpenPLC runtime folder
    b. Navigate to /webserver/st_files and place the file(s) you would like here.
    c. Go back a directory and Navigate to /scripts
    d. In here run ./compile_program.sh [name of your st file]
    e. If compilation was successful, back out a directory and read the contents of the "active program" file to make sure it matches your file name
    f. Back out a directory and run ./start_openplc.sh: Your code should now be uploaded and running.

SCADA-BR & HMI Issues

1. Install ScadaBR
    a. SCADA-BR
        - git clone https://github.com/thiagoralves/ScadaBR_Installer.git
        - cd ScadaBR_Installer
        - ./install_scadabr.sh
    b. Go to Web Browser: localhost:9090/SCADABR
        - Log in with:
            Username: admin
            Password: admin
2. Yellow Triangles Appear in HMI
    a. If yellow triangles  appear next to the data sensors in the HMI:
        i. Make sure the computer running the VM is connected to the vault.
        ii. Make sure the IP address assigned to the VM is 192.168.1.201
        iii. Make sure the IP address assigned to the Pi is 192.168.1.200
            - If not, in SCADA-BR, go to Data Sources
            - In the Vault category, edit the connection to the Pi's IP
        iv. Check the Urgent alarms in the Alarm section if nothing else works, to see why it is saying that the data from OpenPLC can not be displayed.
3. Graphics
    a. How to Create a Graphics View
        - → "Graphical Views" tab → "New View"
        - Name the View
        - Optional: Add a background image
        - Save (ALL THE TIME)

b. Adding existing images to view
    - Select the type of component from “Components:” drop-down tab
    - Press the puzzle piece image “Add component to view”
    - A lego piece will be in the view. Hover over it and select “edit point component settings”
        -- Click the “point” drop-down menu and choose a point and save
    - Hover again and select “edit graphical renderer”
        -- Select the “Image Set” drop-down menu and choose your images
        -- Set a default if needed
c. Adding Your Own Image Set
    - Navigate to /opt/tomcat6/apache-tomcat-6.0.54/webapps/ScadaBR/graphics
    - Create a new folder and name it whatever you want
    - Add a text file called info.txt to this new folder and add the following information:
        1. name=<insert name>
        2. type=imageSet
        3. width=<insert width>
        4. height=<insert width>
        5. text.x=<insert value>
        6. text.y=<insert value>
    - In the same directory add your images (preferably pngs or gifs)
    - Navigate to /opt/tomcat6/apache-tomcat-6.0.54/bin and run the following commands:
        1. Sudo ./catalina.sh stop
        2. Sudo ./catalina.sh start
    - The images should now be available to use in the graphics view
d. Adding Data Sources
    - Click on the Data Sources (it looks like a silver cylinder)
    - On the drop-down menu select Modbus IP
    - Click add
    - Fill in the Host IP with the PLC’s IP address
    - Save and enable the Data Source
e. Adding Points
    - After creating a data source click edit on the Data Source
    - Go down to the bottom and click the brick to add a point
    - Specify in the Register Range drop-down what type of point (Coil or Input)

- Type in the offset which comes from the value after the decimal point of your point's location
- Click save and then enable the point under status.

Attack and Defense Problems

1. Modification Attack
    a. If nothing happens, or you encounter a problem when you launch the Modification Attack, make sure that the IP addresses of the Pi and HMI have not changed.
        i. Pi: 192.168.1.200
        ii. HMI: 192.168.1.201
2. Injection Attack
    a. If nothing happens, or you encounter a problem when you launch the Injection Attack, make sure the IP address of the Pi has not changed.
        i. Pi: 192.168.1.200
3. Defenses
    a. If neither defense is working, restart the system to have the startup scripts run again.

Ettercap

1. Installing Ettercap (version: 0.8.3)
    a. sudo apt-get install librtmp-dev libidn11-dev libgtk-3-dev cmake flex libnet1-dev libpcap0.8-dev libncurses5-dev libssl-dev libgeoip-dev
    b. download ettercap file
    c. mkdir build
    d. cd build
    e. cmake ../
    f. make install
2. ARP Poisoning Example
    a. Run:
        - sudo ettercap -T -i eth0 -M arp /192.168.1.100/ /192.168.1.101/
    b. If you want to add a filter, add:
        - -f <filtername>
3. Create a Filter
    a. Create a file:
        - [filename].filter
        - Add the filter code

b. Run:
    - etterfilter <file> to create the filter file

4. Important Notes for Filters
    a. if/else statements are allowed, but no loops.
        - All statements need to have {} even if only 1 instruction
        - You can use && and || for compound statements however you cannot use parentheses to group conditions
    b. There must be a space between if and (, but no space between func and (
    c. You can check for the protocol by doing eth.proto == [IP/IP6]
    d. tcp.src and tcp.dst do work
    e. IP addresses must be in single quotes
5. Important Built-in Functions for Filters
    a. search(where,what) - find the string 'what' in the buffer 'where' (either DATA.data or DECODED.data) and returns true if found
    b. replace(what,with) - always performed on DATA.data
    c. inject(what) - injects into DATA.data (use drop() right before to replace a packet)
    d. log(what,where) - writes information to a file (can be used for a replay attack)
    e. msg(message) - displays in the messages center
    f. drop() - drops all packets
    g. exit() - exits

Parts and Pieces

Sensors & Alarm

Infrared Lasers

The Emitter circuit:

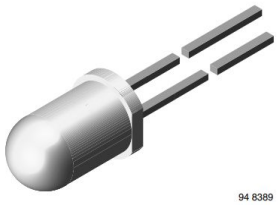


Figure 1: Emitter

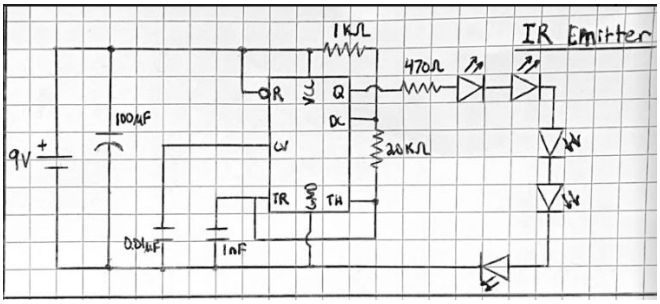


Figure 2: Emitter Circuit Diagram

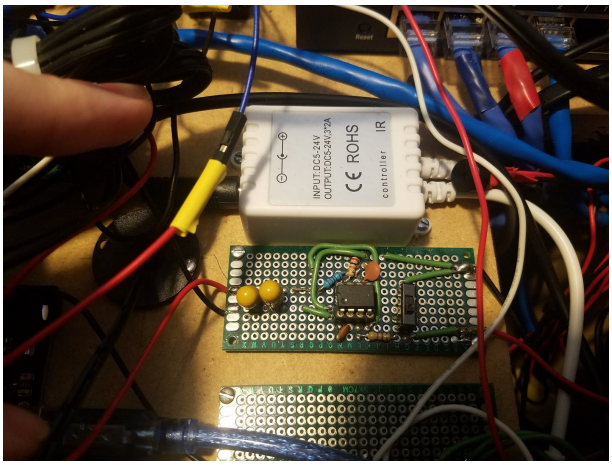


Figure 3: Emitter Circuit

For our infrared laser system, we built an emitter circuit, Figure 2, that uses the 555-timer to modulate the light coming out of the emitters at the correct frequency they needed to be for the receiver to pick it up. We built the circuit using a prototyping board and the parts shown in the circuit diagram. The IC shown in the middle of the diagram is the 555 timer.

The exact emitters we used were the TSAL6100 (High Power Infrared Emitting Diode, 940 nm, GaAlAs, MQW), Figure 1. Their datasheet can be found here:

https://www.mouser.com/datasheet/2/427/tsal6100-279822.pdf

You can order the emitters from Mouser Electronics at this link:

https://www.mouser.com/ProductDetail/Vishay-Semiconductors/TSAL6100?qs=%2Fha2pyFaduic3dX61YfbCQlbZQyNkMI60fCdVJu5Wr0%3D

The Receiver circuit:

Figure 4: Receiver

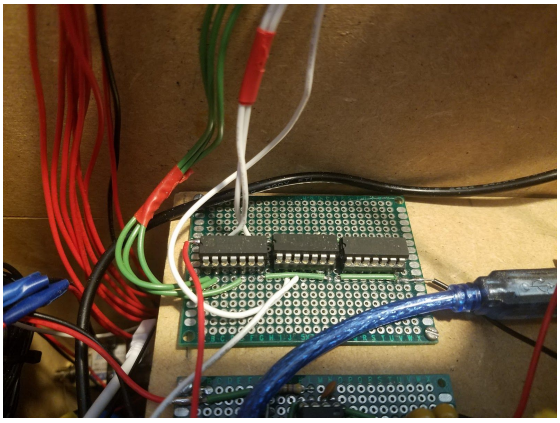


Figure 5: Receiver Circuit

For our Receivers, we built another circuit, Figure 5, that we also made using a prototyping board and OR-gates. This circuit takes in the outputs from our five receivers around the front of the vault and sends an output signal if any receiver is not seeing the light from the IR Emitter.

The exact receivers we used were the TSSP93038 (IR Sensor Module for Reflective Sensor, Light Barrier, and Fast Proximity Applications), Figure 4. Their datasheet can be found at:

https://www.mouser.com/datasheet/2/427/tssp930-1767223.pdf

You can order these receivers from Mouser Electronics at the following link:

https://www.mouser.com/ProductDetail/Vishay-Semiconductors/TSSP93038?qs=%2Fha2pyFaduiSrGdBGwJ7Ed1nVtxk41ayDzd4cAKje5c%3D

The 555 Timers:

Figure 6: 555 Timer IC

We used a 555 timer, as mentioned above, to modulate the infrared emitters to the correct frequency needed for the receivers to pick up the signal. If you refer back to Figures 2 and 3 you can see the placement of this IC in the emitter circuit.

The 555 timer we used was the LN555CM (Single Timer), Figure 6. This timer's datasheet can be found at:

https://www.jameco.com/Jameco/Products/ProdDS/27423.pdf

You can order these timers at Jameco Electronics at the following link:

https://www.jameco.com/z/LM555CN-R-Major-Brands-IC-LM555-Standard-Timer-Single-8-pin-DIP_27423.html

Ultrasonic Sensor

Figure 7: Ultrasonic Sensor

For our Ultrasonic Sensor, we used the HC-SR04 (Smraza Ultrasonic Module HC-SR04 Distance Sensor), Figure 7. This sensor's datasheet can be found at:

https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

You can order the Ultrasonic Sensor from Amazon at the following link:

https://www.amazon.com/Smraza-Ultrasonic-Distance-Mounting-Duemilanove/dp/B01JG09DCK/ref=sr_1_5?keywords=ultrasonic+sensor&qid=1576721655&sr=8-5

Keypad

Figure 8: Keypad

Figure 9: Installed Module

For our keypad we used Adafruit's 4x4 Matrix Keypad, Figure 8, which can be found on Adafruit at the following link:

https://www.adafruit.com/product/3844

Pressure Plate

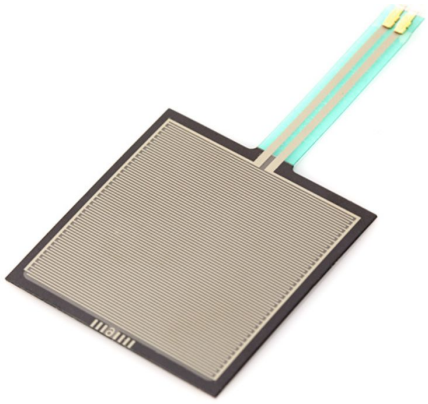


Figure 10: Pressure Plate

Figure 11: Diamond on the Pressure Plate

For our pressure plate, we used the SEN-09376 (Force Sensitive Resistor - Square), Figure 10. This pressure plate circuit also uses a 3300 Ohm resistor tied to ground, per the specifications on the datasheet. To find the datasheet follow this link: https://cdn.sparkfun.com/assets/c/4/6/8/b/2010-10-26-DataSheet-FSR406-Layout2.pdf

You can order this Pressure Plate from SparkFun at the following link:

https://www.sparkfun.com/products/9376

Electronic Lock

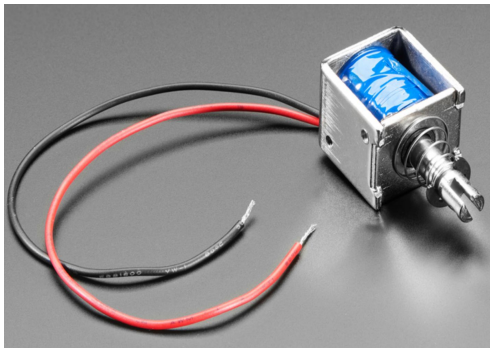


Figure 12: Solenoid Lock

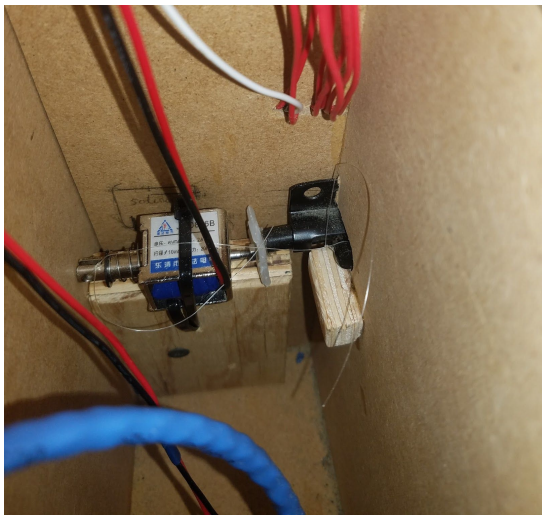


Figure 13: Picture with Lock

For our electronic lock, we used the 6V 2A, Medium Push-Pull Solenoid, Figure 12. This solenoid pushes a standard lock, to lock the door. The solenoid can be found from Adafruit at the following link:

https://www.adafruit.com/product/3992

NOTE: This ended up not being used, due to transient issues with the solenoid, but this system is still installed inside the vault.

RFID Reader

Figure 14: RFID Reader

Figure 15: Picture with Case

For our RFID system, we used the RFID-RC522 (MFRC RC522 RFID-RC522 RF IC Card Reader Sensor Module), as seen in Figure 14. This RFID reader reads in RFID signals when they are held up to it. The kit we bought came with one S50 13.56MHz RFID Smart Card and 5 similar Smart Keyrings. The datasheet for this module can be found at:

https://vetco.net/datasheets/VUPN6326/MFRC522%20datasheet.pdf

You can find the exact kit we used from Amazon at the following link:

https://www.amazon.com/IZOKEE-RFID-RC522-13-56MHz-Arduino-Raspberry/dp/B076HTH56Q/ref=pd_sbs_23_4/142-5037047-5948600?_encoding=UTF8&pd_rd_i=B076HTH56Q&pd_rd_r=5a2ab53c-aa53-4180-a8a5-5e43aa8419a7&pd_rd_w=tfzGh&pd_rd_wg=CFZHm&pf_rd_p=5873ae95-9063-4a23-9b7e-eafa738c2269&pf_rd_r=PQYAS2AQP2DFCKBBC2JB&psc=1&refRID=PQYAS2AQP2DFCKBBC2JB

Alarm

Figure 16: LED Lights

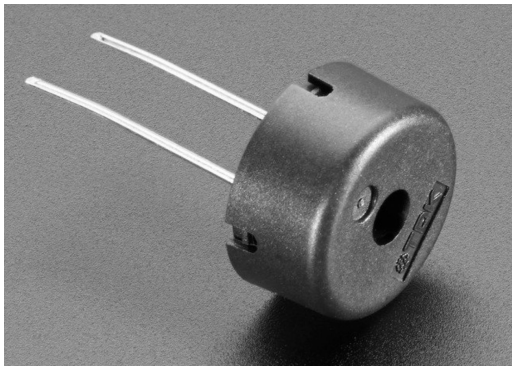


Figure 17: Piezo Buzzer

For our Alarm system, we used several strands of Red LED Lights (Red LED, 3528 SMD, 300LED 5M,12V 2A 24W Lights) as seen in Figure 16 to add to the ambiance of the alarm. We also used passive Piezo Buzzers, PS1240, as seen in Figure 17, for our Alarm sound by modulating them with code. The LEDs we used were from Amazon, and can be found at the following link:

https://www.amazon.com/XKTTSUEERCRR-Waterproof-300LED-Flexible-Light/dp/B00EKEODA0?ref_=fsclp_pl_dp_2

The Piezo Buzzers we used were from Adafruit, and can be found at the following link:

https://www.adafruit.com/product/160?gclid=Cj0KCQjw6eTtBRDdARIsANZWjYYodd_UMCkPlW74KDDcsDiTpRWW0a0m7DNtlpCcqSvNEUdiE9pF6t0aAkpREALw_wcB

Hardware

Raspberry Pi

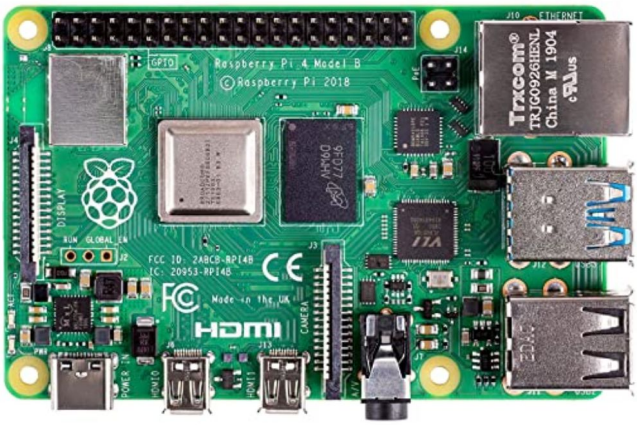


Figure 18: Raspberry Pi 4

Our main controller board we used on this project was the Raspberry Pi 4, with 2 GBs of RAM, as seen in Figure 18. This is where OpenPLC is run and all of the sensors and Arduinos send information. This Pi is the brain of our system. We purchased a Canakit from Amazon that had all the items we needed for the Pi. You can find it here: https://www.amazon.com/gp/product/B07TXMDVPQ/ref=ppx_yo_dt_b_asin_title_o09_s01?ie=UTF8&psc=1

Arduinos

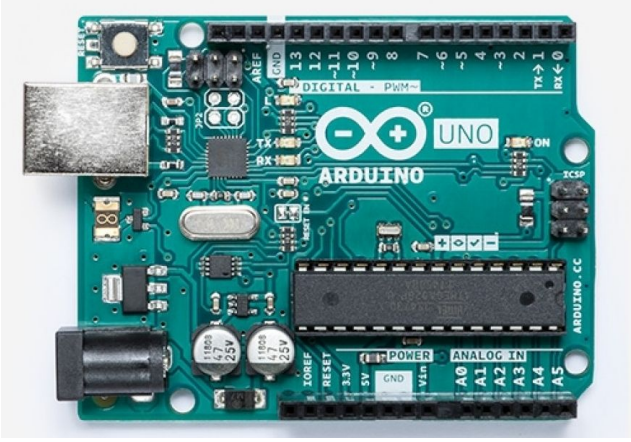


Figure 19: Arduino

We used two Arduino Uno boards for dealing with the inputs and output of many of our sensors like the one shown in Figure 19. These boards run a lot of the code for different modules. For example, one Arduino handles the code for the RFID reader and then provides the information to the Pi on whether the card it read was correct or not. These two boards were regular Arduino Unos that we were given by one of our sponsors, Dr. Morris.

Network Switch

Figure 20: Network Switch

Figure 21: Ports on the Vault

To communicate with the Pi and the HMI and to have the attacker get into the network, we decided it would be best to use a network switch. This is so the communications are not limited by the location's Wifi rules and strength. This network switch, shown in Figure 20, has one port that goes directly to the Pi and ports that connect to Ethernet ports on the back of the Vault as shown in Figure 21. Two of these ports are connected to regular communication ports on the switch: the red for the attacker and the blue for the HMI. The last port, the orange one, is used to connect the whole network to the internet if needed. We were given this network switch by one of our sponsors, Dr. Coe.

Power Strip

Figure 21: Power Strip

All the circuitry in our vault is being powered by the power strip shown in Figure 21. We got this power strip from Amazon at the following link: https://www.amazon.com/Fire-Proof-Mountable-SUPERDANNY-Protection-Extension/dp/B07GZNHVT4/ref=sr_1_12?keywords=power+strip&qid=1575997363&s=electronics&sr=1-12

LED Spotlights

Figure 22: LED Spotlights

We are providing light in the vault by 3 LED Mini Spotlights (6000k Cold White 1.5W 12VDC LED Mini Spotlight), like the one shown in Figure 22. We have two of these situated at the front of the vault shining into the vault,= and one directly over the

diamond which gives it a nice glimmering effect. The LED spotlight kit we bought can be found at Amazon at the following link:
https://www.amazon.com/Jorunhe-Surface-Mounted-Spotlight-Ac85-265v/dp/B07QBZ3DHX/ref=sxbs_sxwds-stvp?keywords=Micro%2BPivoting%2BLED%2BSpotlight&pd_rd_i=B01DP873GM&pd_rd_r=6f32bad7-fb67-4a71-97ba-8335a49c5c91&pd_rd_w=VjBe7&pd_rd_wg=qE3h5&pf_rd_p=a6d018ad-f20b-46c9-8920-433972c7d9b7&pf_rd_r=T5H0ZFG97AMSSNQN9FNG&qid=1575925763&th=1
We also purchased two more of these lights from Amazon at the following link:
https://www.amazon.com/gp/product/B07RM5HJ5P/ref=ppx_yo_dt_b_asin_title_o02_s00?ie=UTF8&psc=1

Relay

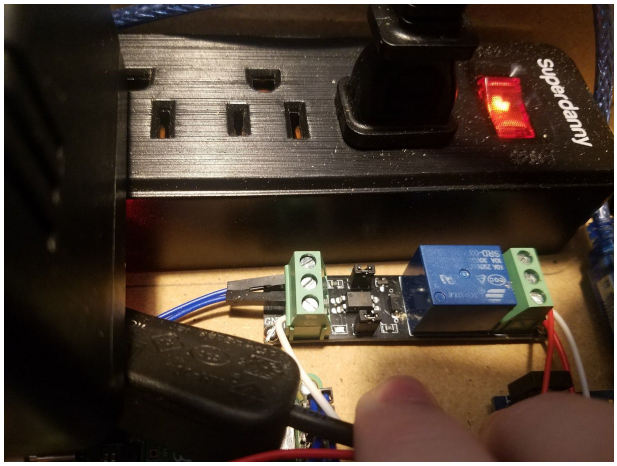


Figure 23: 5V Relay

To be able to control the LED alarm lights and the mini LED spotlights, we used a 3.3V controlled relay. With this, our Raspberry Pi can toggle the lights on and off. The relay we used is shown in Figure 23. We were given these by one of our sponsors, Dr. Coe.

Cables & Wire

In this vault, we used many cables to connect things. All the cables in the vault are 22 gauge. We mainly tried to use red wires for voltage, black for ground, and white or green for data connections. BUT, this is not always the case! Some red wire is also

used for data connections, and some wires may just have had the wrong color due to limited supply. In general though, red is voltage and black is ground.

The other cables include:

- The ethernet cables which were all Cat5 cables.
- The 12V power supply for our LED lights and spotlights
- The white cables and vault for the mini LED spotlights
- The brick and adjoined cable for the network switch.
- The USB power cables for the Raspberry Pi and the Arduino's.

Vault

Paint

Figure 24: Dark Blue Paint

Figure 25: Red Paint

Figure 26: Gray Paint

Figure 27: Polyurethane Clear Coat

The paint we used for the three different colors on the vault, can be seen above in Figures 24, 25, and 26. These are all BEAR Premier Paint and Primer in One paints. We also used a Polyurethane Clear Coat to help prevent scratches of the paint and to help protect the vault from water or other possible damages. It is shown in Figure 27. These were all bought from Home Depot.

Wood

The vault itself is made out of about ⅔ of a 4-foot by 8-foot sheet of ¾ inch MDF. This sheet of MDF was purchased from Home Depot.

Pipe

We used a ½ inch pipe on each side to try and hide some of the wires that run along the top of the vault back into the compartment containing all the hardware. This pipe was purchased from Home Depot.

3D Printed Parts

Figure 28: Grommet

We had two different parts 3D Printed for the vault. The first was the grommet used to have the power cable come through on the back of the vault, as seen in Figure 28. The second, which can be seen up in Figure 15, is the case we mounted our RFID Reader into.