Automated Solar Tunnel Lights: SCADA Project

## **Course:** BFOR 632 – Cyber Vulnerabilities Exploitation

**Professor:** Dominick Foti & Sanjay Goel

**Date:** 13/06/2025

**Team Number:** 2

**Team Members:**

Melpomeni Doutsis,

Steve Correia,

## Leela Pavan

## Sriram Rayala

## Srivarsha Adla

University at Albany, State University of New York

**Contents**

|  |  |
| --- | --- |
| Topics | Page number |
| Project Overview | 2 |
| System Architecture | 2 - 3 |
| Component List & Description | 5 - 6 |
| Logical Flow of Operation | 7 - 8 |
| Security Considerations | 8 - 9 |
| Monitoring & Detection Tools | 9 |
| Security Controls | 9 |
| Detailed exploit and countermeasure | 10 - 16 |
| Future Enhancements | 17 |
| Conclusion | 18 - 19 |
| References | 19 |

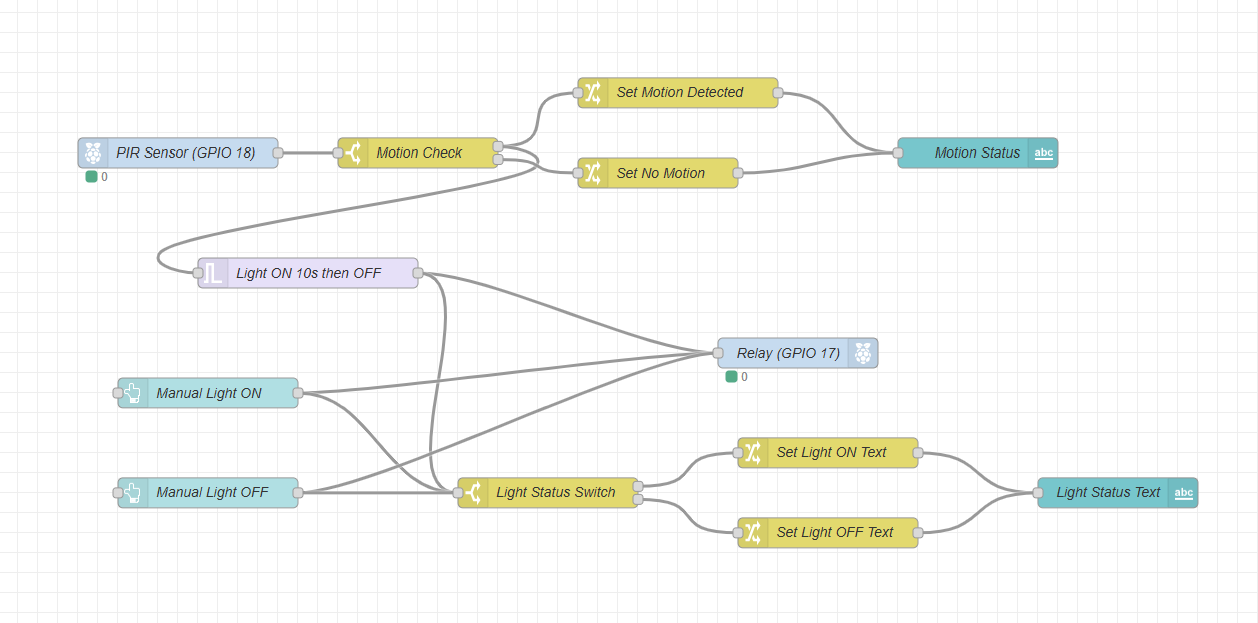
# Project Overview

The *Automated Solar Tunnel Lights* project is a real-world SCADA-based solution aimed at enhancing tunnel safety while improving energy efficiency. Traditional tunnels often consume significant power by keeping lights on 24/7, regardless of vehicle presence or ambient conditions. This project introduces automation through motion sensors that trigger LED lights only when needed, significantly reducing power consumption and promoting sustainable operations.

The system architecture integrates motion sensors to detect vehicle movement and light levels, activating LED lighting via relay circuits accordingly. The entire setup is powered by a solar panel and battery system, enabling off-grid functionality. A Raspberry Pi handles data logging and system management, while a Human-Machine Interface (HMI) built with Node-RED provides real-time monitoring and control of the tunnel lighting environment.

To secure the SCADA system and administrative access, multiple security measures were implemented. SSH hardening with Fail2Ban protects against brute-force attacks, while dashboard access is restricted through authentication mechanisms. Additional protections, including iptables configuration, internal GPIO-based control, and strict local network operation without public internet exposure, collectively ensure system resilience against unauthorized access and cyber threats.

# System Architecture

 Figure 1: System Architecture Diagram

The Tunnel Lighting Automation system operates based on both motion detection and manual override controls. A PIR sensor connected to GPIO 18 detects motion events. When motion is detected, the system sets a "Light ON" status, which activates a relay connected to GPIO 17, controlling the 12V LED strip. The light remains ON for 10 seconds after motion

is detected, after which it automatically turns OFF if no further motion is sensed. Simultaneously, the relay operation updates the Light ON Status Text for monitoring purposes.

Manual override controls are incorporated through "Manual Light ON" and "Manual Light OFF" switches. The "Manual Light ON" directly triggers the relay to turn on the lights, while "Manual Light OFF" updates the light status text accordingly. The Light Status Switch processes manual input to update either the Light ON or Light OFF text, and the final status is displayed on the Node-RED dashboard for real-time visualization.

This architecture ensures seamless integration between automatic motion-triggered lighting with time-based auto-off, manual control overrides, and continuous real-time system status updates for effective tunnel monitoring and energy-efficient operation.

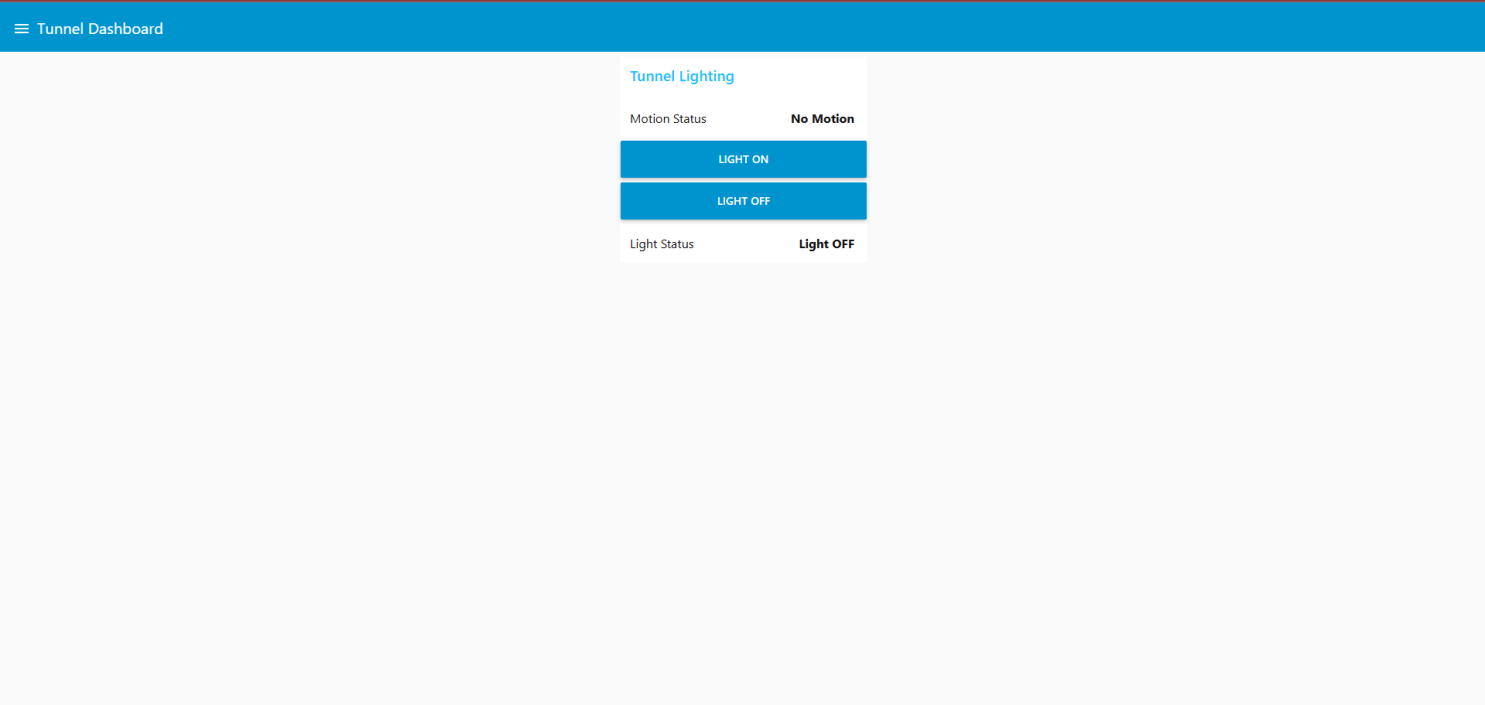


Figure 2: HMI Dashboard

The screenshot (motion\_log.txt) shows motion detection logs from your Raspberry Pi system. These logs are **evidence of system functionality and real-time behavior**



Figure 3: motion\_log.txt

#### **Challenges faced during implementation included:**

**Sensor Calibration:** Ensuring the PIR sensor reliably detected motion within the desired range and angle.

**Relay Configuration:** Correctly wiring and programming the relay module to avoid false switching or delayed response.

**Network Configuration:** Setting up secure local network access to the Raspberry Pi without exposing it to public internet access.

**Dashboard Security:** Implementing user authentication on the Node-RED dashboard and ensuring it functioned without disrupting usability.

**Testing under Realistic Conditions:** Simulating consistent motion detection and observing system behavior in different lighting conditions to confirm performance.

Despite these challenges, the system was successfully deployed as a working prototype with both automatic and manual lighting control, powered by solar energy and secured using basic cybersecurity measures.

# 

# Component List and Description

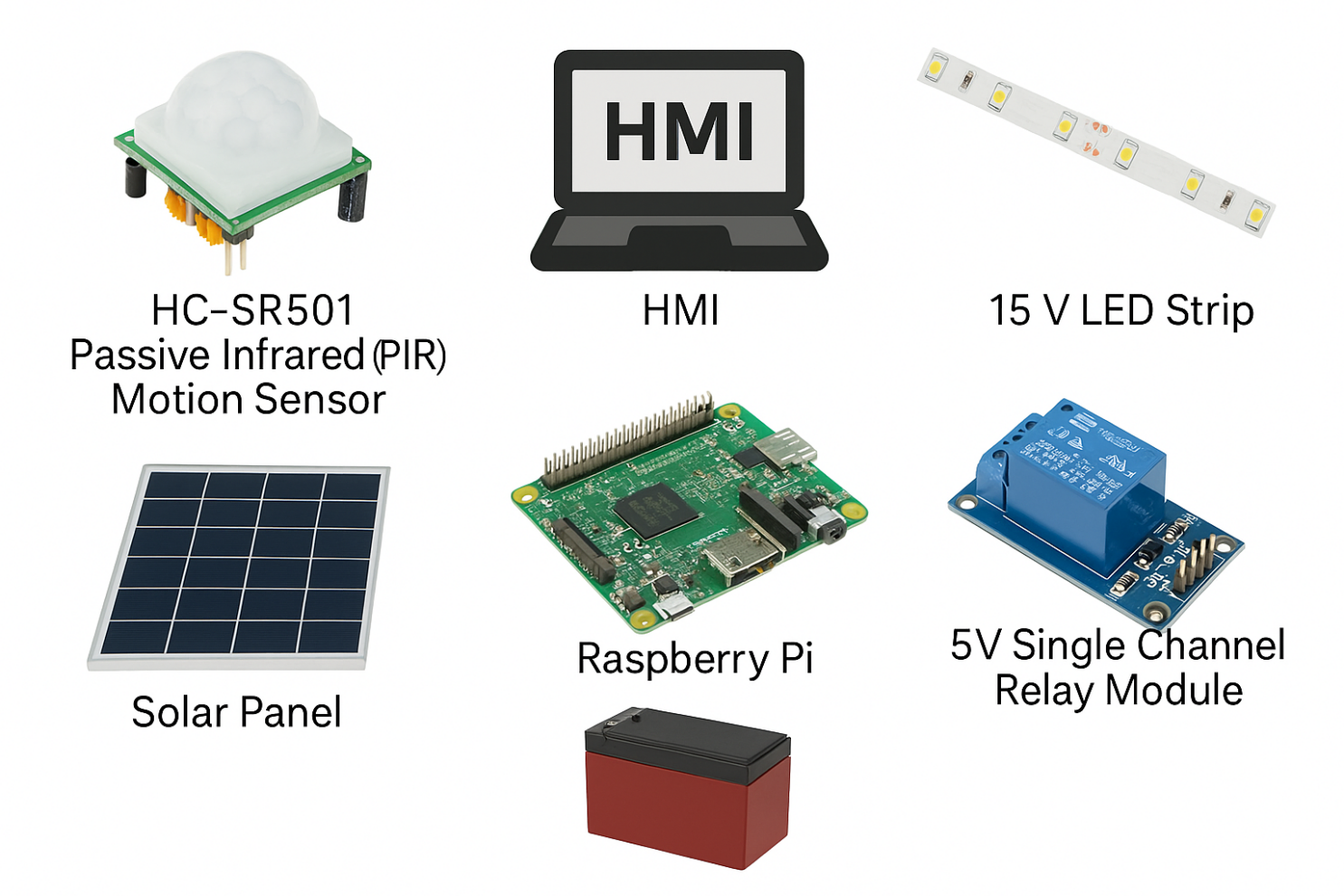


Figure 4: Key Components Used in the Automated Solar Tunnel Lights Project

The following components are shown in Figure [2](#_bookmark3) and are used in the design and implementation of the SCADA-based solar tunnel lighting system:

### 🔹 HC-SR$0 Passive Infrared (PIR) Motion Sensor

The HC-SR501 Passive Infrared (PIR) Motion Sensor detects motion by sensing infrared radiation changes caused by moving objects, particularly humans or vehicles. In the tunnel lighting system, it activates the lights when motion is detected, enabling automation based on vehicle presence.

### 🔹Node-Red

Advanced HMI is a graphical user interface (GUI) platform used to monitor and control industrial systems. In this project, it provides a real-time visual dashboard that displays system status and allows manual control of tunnel lights via a web interface.

### 🔹 12V LED Strip

The 12V LED strip provides efficient and bright lighting for the tunnel. Controlled by sensors and relays, it illuminates the tunnel only when required, ensuring both energy savings and maintaining visibility for vehicles.

### 🔹 5V Single Channel Relay Module

The 5V Single Channel Relay Module acts as a switch that controls the electrical connection to the LED strip. It receives signals from the processing unit (Raspberry Pi) based on sensor inputs and physically turns the lights ON or OFF as needed.

### 🔹 Solar Panel

The solar panel captures solar energy and converts it into electrical power. It supplies renewable energy to the entire tunnel lighting system, allowing it to operate independently of the traditional electrical grid.

### 🔹 Battery

The battery stores excess energy generated by the solar panel during the day. It supplies backup power to the system at night or during periods of low sunlight, ensuring continuous and uninterrupted operation.

### 🔹 Raspberry Pi

The Raspberry Pi serves as the central controller and data logger for the system. It processes sensor data, manages the relay operations, updates the HMI dashboard, and records system performance metrics for analysis.

# Logical Flow of Operation

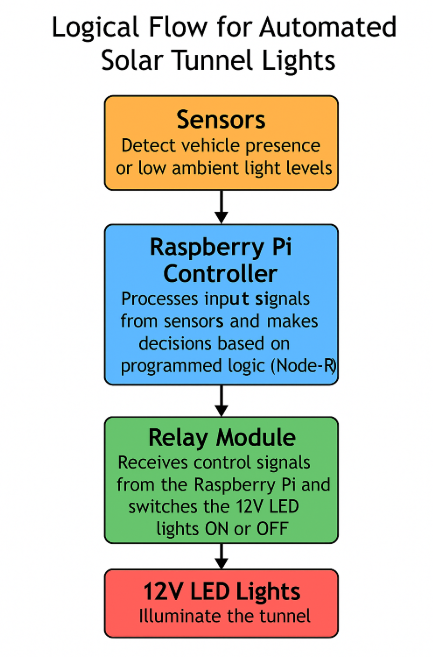


Figure 5: Logical Flow Diagram for Automated Solar Tunnel Lighting System

The operation of the Automated Solar Tunnel Lights system is structured as follows:

**Sensor (Motion):** These sensors form the input stage of the system. Motion sensors detect vehicle presence, while light sensors monitor ambient brightness. When motion is detected or ambient light levels drop below a threshold, a signal is sent to the **Raspberry Pi controller**.

**Raspberry Pi Controller:** The Raspberry Pi processes the sensor inputs and based on programmed logic, decides whether to activate or deactivate the lighting system.

**Relay Module:** The Raspberry Pi sends control signals to the relay, which acts as a switch for the LED lights, allowing low-power signals to control high-power lighting devices safely.

**LED Lights:** Once triggered by the relay, the LED lights illuminate the tunnel. They remain active as long as motion is detected or ambient light remains low, ensuring safety and visibility inside the tunnel.

# Security Considerations

**1. Vulnerabilities in Local Network Communication**

The Automated Solar Tunnel Lights system operates over a local network without direct exposure to the public internet. However, any device connected to even a private network can still be vulnerable to internal threats if not properly secured. A compromised local device could attempt to SSH into the Raspberry Pi hosting the tunnel control system. Therefore, SSH access is hardened using Fail2Ban to block repeated failed attempts, IP whitelisting, and secure password policies. Additionally, iptables are configured to limit unauthorized connections within the local network.

**2. Weak Authentication on Web-Based HMI**

The Node-RED dashboard, acting as the system's HMI (Human-Machine Interface), could present a critical attack point if improperly secured. Weak passwords, lack of HTTPS, or poor login configurations could allow unauthorized access to the tunnel lighting controls. To mitigate this, user authentication has been enabled on the Node-RED dashboard with strong password policies. Future hardening should include options such as OAuth2 authentication, account lockout mechanisms after repeated failed login attempts, encrypted connections (SSL/TLS), and continuous monitoring of access logs.

**3. Insecure Sensor and Relay Control**

Since GPIO control mechanisms are used directly through the Raspberry Pi to control relays and read sensors, an attacker gaining SSH or terminal access could potentially override sensor inputs or manually trigger relays. Local GPIO access is restricted only to authorized users at the OS level. The Raspberry Pi is operated within a segmented internal network with no public exposure to minimize the risk. Additionally, limiting sudo/root privileges further protects against unauthorized manipulation of critical system functions.

**4. Physical Security Risks**

Given that solar panels, sensors, relays, and Raspberry Pi are installed in a physical outdoor environment (tunnel infrastructure), there is a risk of physical tampering. Securing the hardware in tamper-resistant enclosures, monitoring for unauthorized access using motion alerts, and periodically auditing the physical setup are crucial to maintaining system integrity.

# Monitoring and Detection Tools

The following tools and methods were used to monitor and detect security incidents within the system:

* **Manual Log Monitoring**:  
   Regular manual inspection of system logs (/var/log/syslog) and authentication logs (/var/log/auth.log) was performed to detect unauthorized access attempts and system anomalies.
* **Fail2Ban Alerts**:  
   Fail2Ban was not only used as a security control but also served a monitoring role by providing alerts when repeated failed SSH login attempts were detected and banned.
* **Periodic SSH Session Review**:  
   SSH login sessions and active connections were periodically reviewed to ensure no unauthorized users were logged into the system

# Security Controls

The following security controls were implemented to protect the Automated Solar Tunnel Lights system:

* **SSH Hardening**:  
   Root login over SSH was disabled, and strong password authentication was enforced to prevent unauthorized remote access.
* **Fail2Ban Deployment**:  
   Fail2Ban was configured to monitor SSH login attempts and automatically block IP addresses after multiple failed login attempts, reducing the risk of brute-force attacks.
* **iptables Firewall Rules**:  
   iptables was configured to allow only trusted IP addresses to establish SSH connections, while all other traffic was blocked to minimize the attack surface.
* **Node-RED Dashboard Authentication**:  
   Access to the Node-RED Human-Machine Interface (HMI) was protected with a login system, restricting control access to authorized users only.
* **Private Local Network Operation**:  
   The system was operated within an isolated, private network without exposure to the public internet, further enhancing overall security.

# Detailed exploit and countermeasure

## **Reconnaissance Phase**

* Connected to the unsecured Wi-Fi network used by the Solar Pi.
* Used Wireshark to capture network traffic and identified the Solar Pi's IP and MAC address.
* Performed an Nmap scan on the Solar Pi.
* Found open ports for SSH and Node-RED web interface.
* Identified potential weaknesses like unprotected access and exposed dashboards.

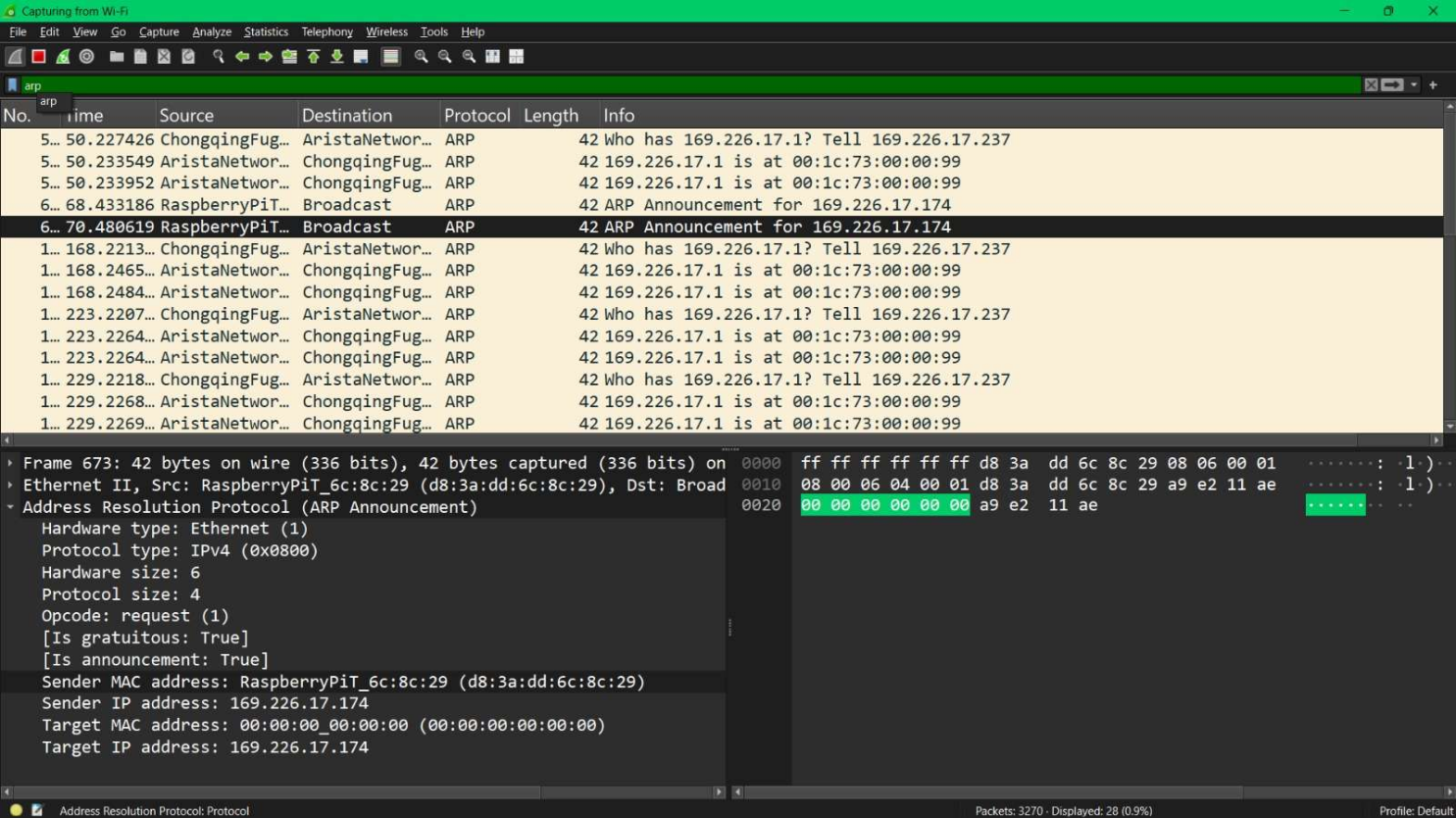


Figure 6: Network Traffic Analysis using Wireshark

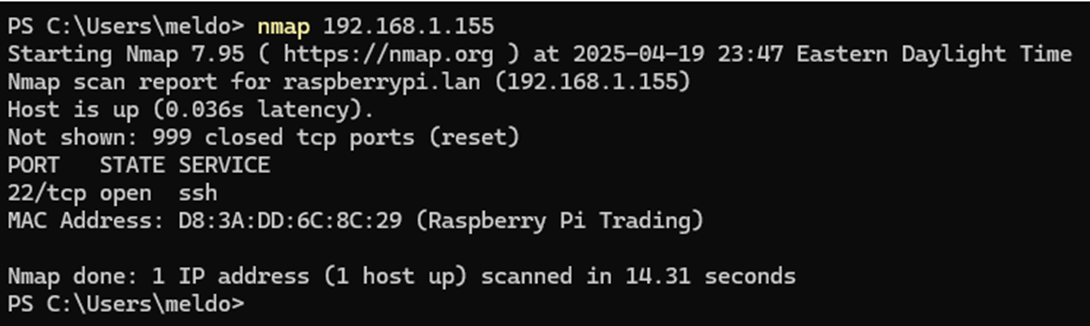


Figure 7: NMAP Scan

**Detailed Explanation:**

The reconnaissance began with passive traffic monitoring using Wireshark on the unsecured Wi-Fi network. Through traffic capture, the attacker identified the IP address and MAC address of the Raspberry Pi (Solar Pi). Following this, an Nmap scan was conducted against the Solar Pi, as shown in the screenshot. It revealed open ports 22 (SSH) and 1880 (Node-RED web interface), both potential vectors of attack. This network footprinting provided information about available services, indicating weak access control configurations and a lack of port hardening. From the screenshots, we can also infer that no firewall was installed to prevent untrusted access attempts, as SSH was left open to the outside world.

# Attack and Exploitation Phase

* Identified the open SSH port discovered during the reconnaissance phase.
* Planned to exploit weak authentication by performing a brute-force attack using Hydra.
* Attempted to gain system access by targeting SSH with a dictionary of common usernames and passwords.
* Successfully logged into the SCADA Raspberry Pi after brute-forcing.
* Gained full system access, allowing interaction with services and configuration files.
* Modified Node-RED dashboard credentials to secure access.
* Demonstrated potential impact by altering Node-RED logic flows that controlled tunnel lighting.

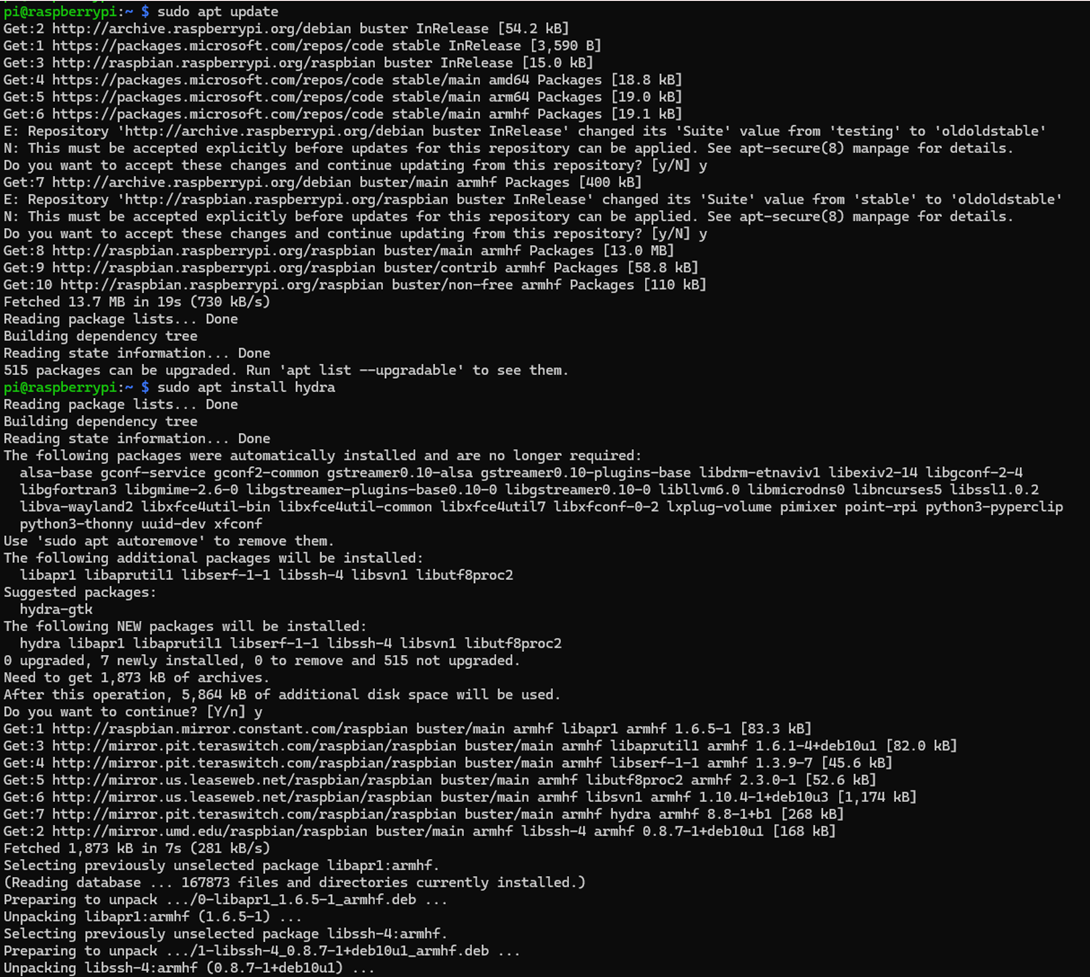


Figure 8: Hydra Installation

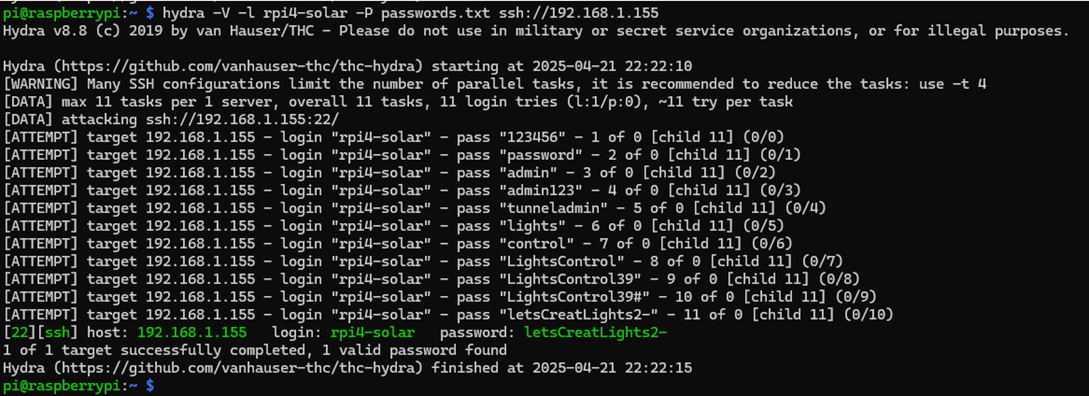


Figure 9: Hydra brute-force attack revealing valid SSH credentials

Inserting image...

Figure 10: Accessing Node-RED settings file for credential modification

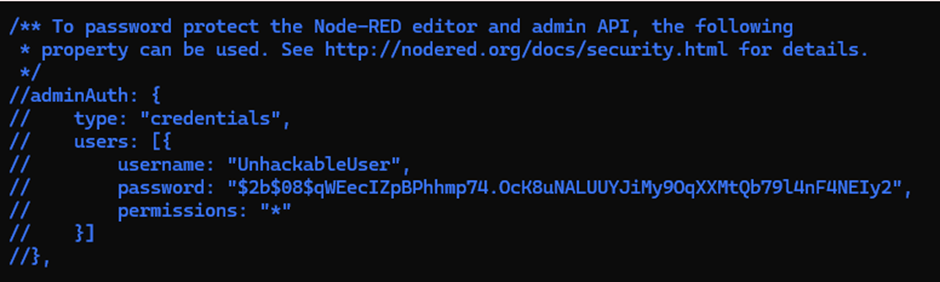


Figure 11: Inserting new admin credentials into Node-RED settings file.

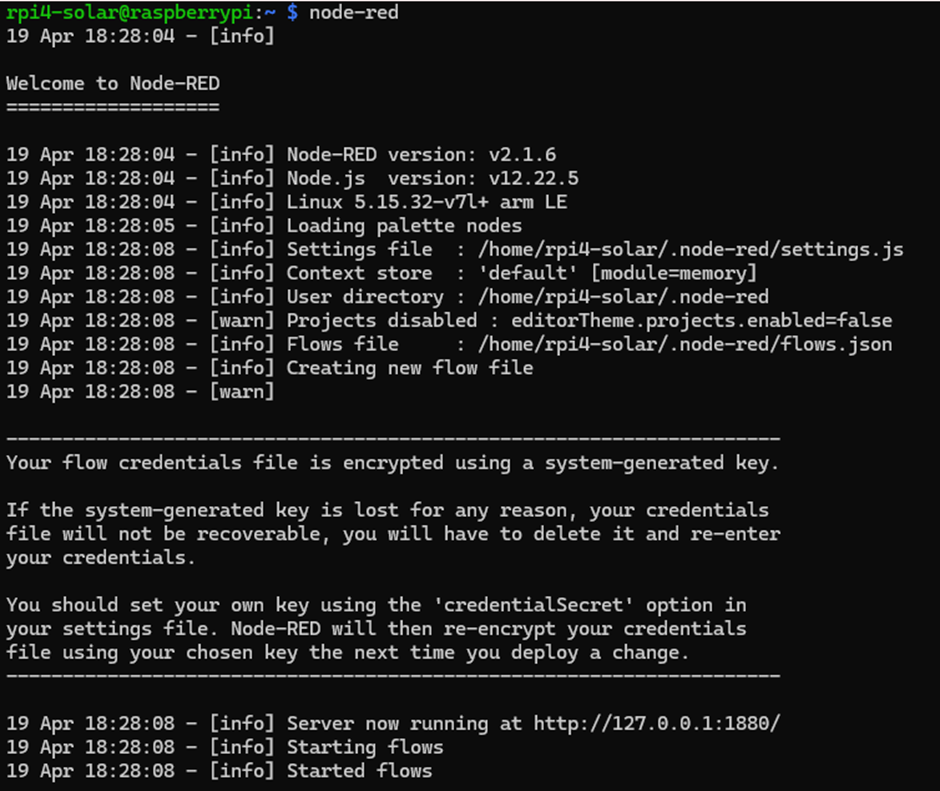


Figure 12: Restarting Node-RED service after modifying settings and credentials

**Detailed Explanation:**

Once SSH was identified as a potential point of entry, we attempted to brute-force it using Hydra, a well-known password attack tool. Hydra was configured with a dictionary of weak passwords and default usernames, and within just

11 attempts, we successfully broke into the system. This revealed that the Solar Pi was using easily guessable credentials — a major security flaw.

With legitimate SSH access established, we logged into the SCADA Raspberry Pi and gained full control over its operating system. This access allowed us to interact directly with its services and configuration files, providing nearly administrative-level privileges.

One of the initial steps taken after gaining access was to secure the exposed Node-RED service. We used command-line tools to generate a new, secure password hash and replaced the existing one in the settings.js file. After restarting the Node-RED environment, we successfully logged in using the new credentials, confirming control over the interface.

To demonstrate the potential risks, we modified the Node-RED logic flows that control the tunnel lights. This exercise illustrated how easily an attacker could manipulate critical infrastructure — such as turning tunnel lights on or off — if basic security measures were not properly implemented.

**System Hardening Phase**

* Focused on hardening the system after successful exploitation.
* Installed Fail2Ban to monitor and block repeated failed SSH login attempts.
* Configured iptables firewall rules to restrict SSH access to trusted IP addresses only.
* Tested firewall and Fail2Ban configurations to verify they were correctly protecting the system.

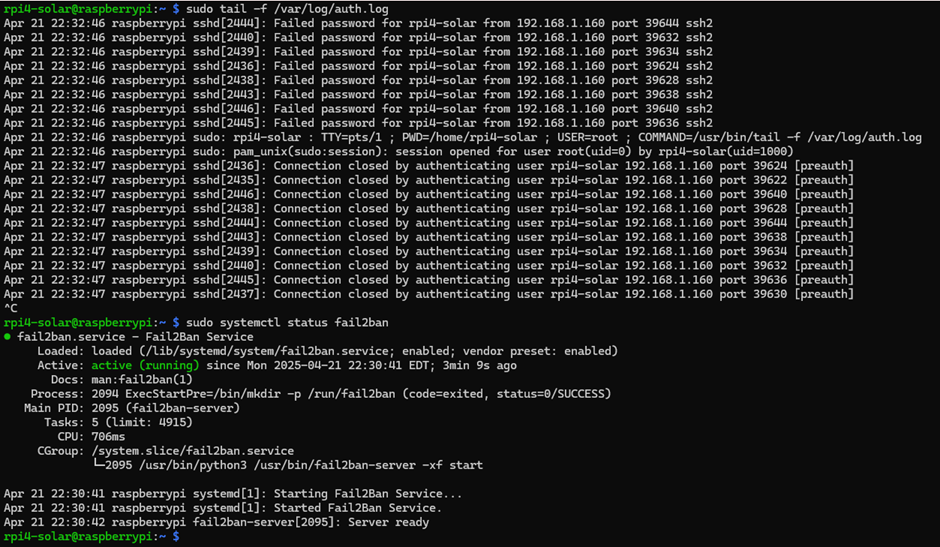


Figure 13: Fail2Ban detecting repeated failed SSH attempts and actively protecting the system

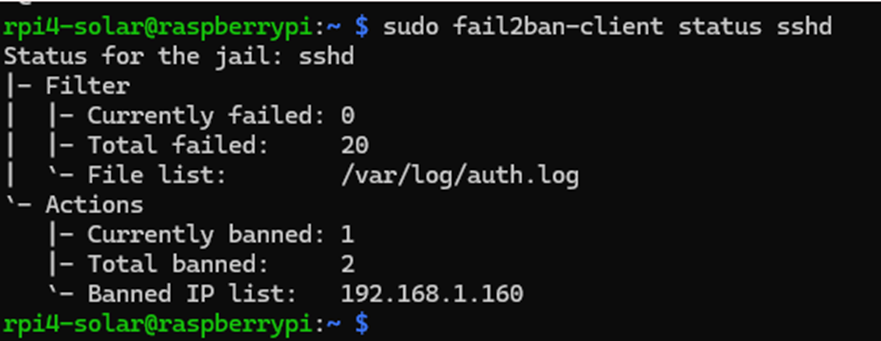


Figure 14: Fail2Ban showing banned IP after multiple failed SSH login attempts

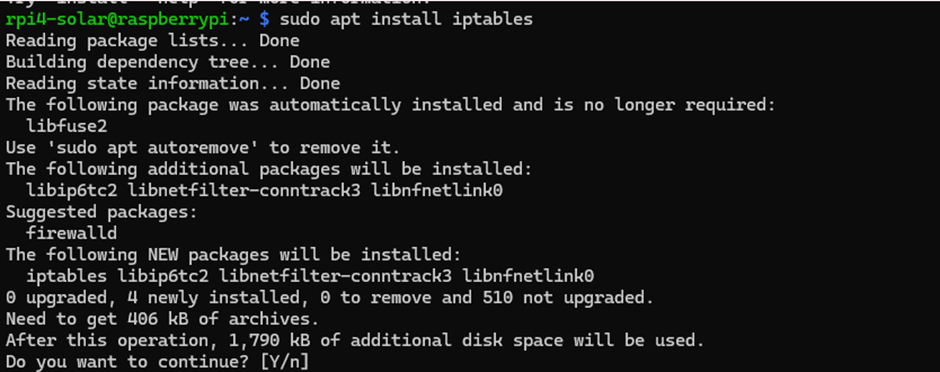


Figure 15: Installing iptables on the Solar Pi to configure firewall rules

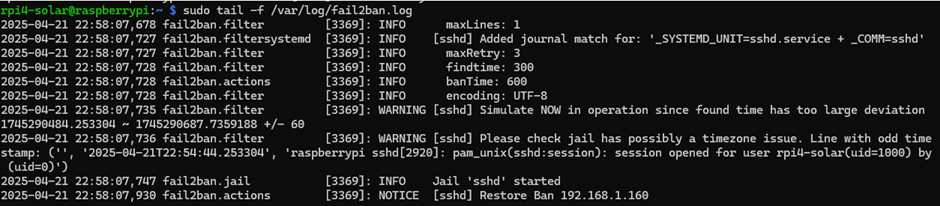


Figure 16: Fail2Ban log showing restoration of ban for repeated SSH login failures

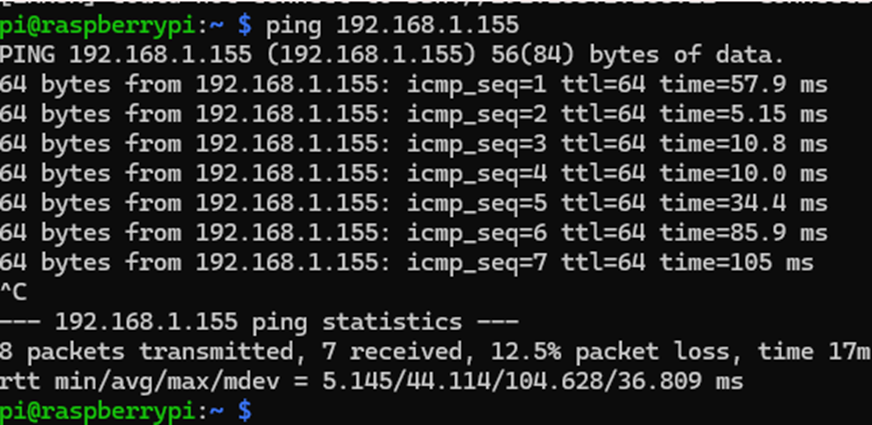


Figure 17: Ping test confirming Solar Pi remains reachable after SSH access was restricted

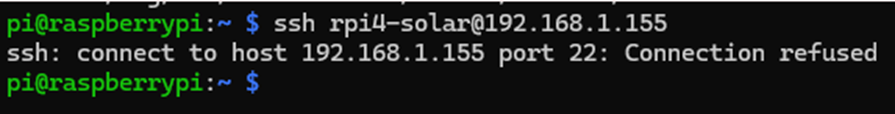


Figure 18: SSH connection attempt refused, confirming firewall and Fail2Ban protections are active

**Detailed Explanation:**

Following exploitation, we focused on system hardening. Our first reaction was to install Fail2Ban, which closely monitors login attempts and bans IPs on repeated consecutive failed attempts. To test this, we initiated some failed SSH login attempts from another machine. As expected, the source IP was immediately banned, and confirmed this in the log.

Then we added access control through the use of iptables. We wrote policies to allow SSH access from certain known IPs and deny all others. This added another strict firewall-based control mechanism.

To check the firewall, we tried to ping the Pi from a distant machine it responded (indicating the Pi was still running), but our SSH was denied. This indicated our defenses were working: only authorized users could access SSH, but availability of the system was maintained.

# 

# Future Enhancements

## **System Design Improvements**

* **Integration of Weather Sensors**:  
   Incorporating ambient light sensors (e.g., BH1750), rain sensors, and fog detection sensors into the existing system can enable adaptive lighting based on real-time environmental conditions. Lighting intensity can be adjusted dynamically to improve visibility during low-light or adverse weather conditions while conserving energy during bright daylight.
* **Traffic-Based Lighting Optimization**:  
   Adding basic traffic sensors at tunnel entry and exit points could allow the system to adjust lighting levels based on real-time traffic flow. Light-weight traffic flow prediction models, using simple machine learning tools integrated within Node-RED, can further optimize lighting schedules without requiring complex external infrastructure.
* **Scalability to Multi-Tunnel Management**:  
   Expanding the system design to manage multiple tunnels through centralized monitoring would be a logical step. Each tunnel could retain local control via its Raspberry Pi, while centralized dashboards connected securely (e.g., over VPN or encrypted protocols like MQTT) could provide remote monitoring and control capabilities across different locations.

## **Security Enhancements**

* **Deployment of Security Onion for Network Monitoring**:  
   Security Onion can be introduced to provide full packet capture, intrusion detection, and real-time network traffic analysis within the tunnel lighting system’s network. This would help detect scanning attempts, brute-force attacks, or unusual network behaviors targeting critical devices.
* **Use of Next-Generation Firewalls (NGFWs)**:  
   Replacing or supplementing basic iptables rules with NGFWs would offer deeper packet inspection, application-level control, and threat prevention. This would improve defense against modern cyberattacks beyond basic IP and port filtering.
* **Centralized Log Management and SIEM Integration**:  
   System event logs from the Raspberry Pi devices and network equipment can be forwarded to a centralized Security Information and Event Management (SIEM) system. This would enhance monitoring, alerting, and forensic investigation capabilities in case of a security incident.
* **Improved Authentication and Access Control**:  
   Implementing two-factor authentication (2FA) for administrative SSH access and Node-RED dashboards would further harden the system against unauthorized access. Additionally, segmenting the network through VLANs can restrict communication paths and limit the lateral movement of attackers.

1. **Conclusion**

The Automated Solar Tunnel Lights project successfully demonstrated the design, implementation, and security hardening of a SCADA-based tunnel lighting system powered by solar energy. By integrating motion and ambient light sensors with a Raspberry Pi and Node-RED HMI, the system achieved both automation and energy efficiency goals.

The project also highlighted key cybersecurity vulnerabilities typically present in SCADA systems, such as weak SSH authentication and unsecured dashboard interfaces. Through practical exploitation and the implementation of countermeasures—including Fail2Ban, iptables firewall rules, and dashboard authentication—the system was effectively secured against common threats.

Future enhancements, such as integrating weather and traffic sensors, machine learning optimization, centralized multi-tunnel management, and advanced network security tools like Security Onion and next-generation firewalls, have been identified to further improve system intelligence, scalability, and resilience.

This project serves as a practical foundation for developing smarter, safer, and more sustainable tunnel lighting systems, while maintaining strong cybersecurity standards.This project provided valuable hands-on experience in building and securing a small-scale SCADA system. By implementing an automated tunnel lighting setup powered by a solar panel and controlled by a Raspberry Pi, we gained technical knowledge in sensor integration, GPIO programming, relay control, and HMI development using Node-RED. The process of configuring the system and observing its real-time behavior helped us understand how control logic and feedback loops function in industrial settings.

A major learning outcome was the exposure to cybersecurity concerns in SCADA environments. By simulating reconnaissance and brute-force attacks using tools like Wireshark, Nmap, and Hydra, we observed the vulnerabilities present when proper security configurations are not in place. Implementing Fail2Ban, iptables rules, and SSH hardening measures taught us how to protect system components from common attack vectors.

#### **Suggestions for Next Year’s Class:**

**Weekly Project Review by Group Members**  
Introduce a weekly review session where one member from each group presents a brief summary of the work done that week. This rotation can help ensure equal participation from all team members and reduce the burden on a single person to report project progress.

**Dedicated Project Workdays**  
It would be beneficial to have one lecture/lab day and a separate day reserved solely for project development every 2–3 weeks. Due to time constraints and conflicting personal schedules, working extensively outside class is difficult. A dedicated class period for project work would improve focus and progress.

**Faster Delivery of Equipment**  
Ensuring that hardware components (like sensors, relays, and Raspberry Pis) are distributed earlier in the semester would give students more time to troubleshoot and complete testing without delays.

#### **Teaching Assistant Support:**

The Teaching Assistants were helpful and responsive, especially during regular class sessions and labs. They were available to answer technical questions and assist with hardware-related issues. However, since most of the in-class time was occupied by lab exercises, there were limited opportunities for TAs to engage with individual projects in depth. Increasing their involvement during the dedicated project workdays (as suggested above) would likely enhance support and outcomes for future students.

1. **References**

**<https://www.isaca.org/resources/isaca-journal/past-issues/2014/scada-cybersecurity-framework>**

**<https://www.isaca.org/resources/isaca-journal/issues/2023/volume-2/protecting-industrial-control-system-software-with-secure-coding-practices>**

**<https://www.robertmlee.org/a-collection-of-resources-for-getting-started-in-icsscada-cybersecurity/>**

**<https://www.progress.com/blogs/network-visibility-in-the-scada-ics-environment>**