TryHackMe CTF Task 3 BREACH

[THM CTF] Industrial Intrusion Warm-up — Breach Writeup

Overview

In the **Industrial Intrusion** room, the goal is to exploit a weakness in the control infrastructure and **open a security gate** that's controlled via a badge and motion detection system. Let's dive in on how to tackle this room!

Target: 10.10.40.202 Objective: Bypass badge authentication to open the gate.

Phase 1: Reconnaissance & Initial Enumeration

The first step was network scanning to uncover open ports and services on the target system using Nmap. The scan revealed several interesting ports, including:

```
nmap -p- --min-rate=1000 -T4 10.10.40.202 -oN full-port-scan.txt
```

Open Ports Identified:

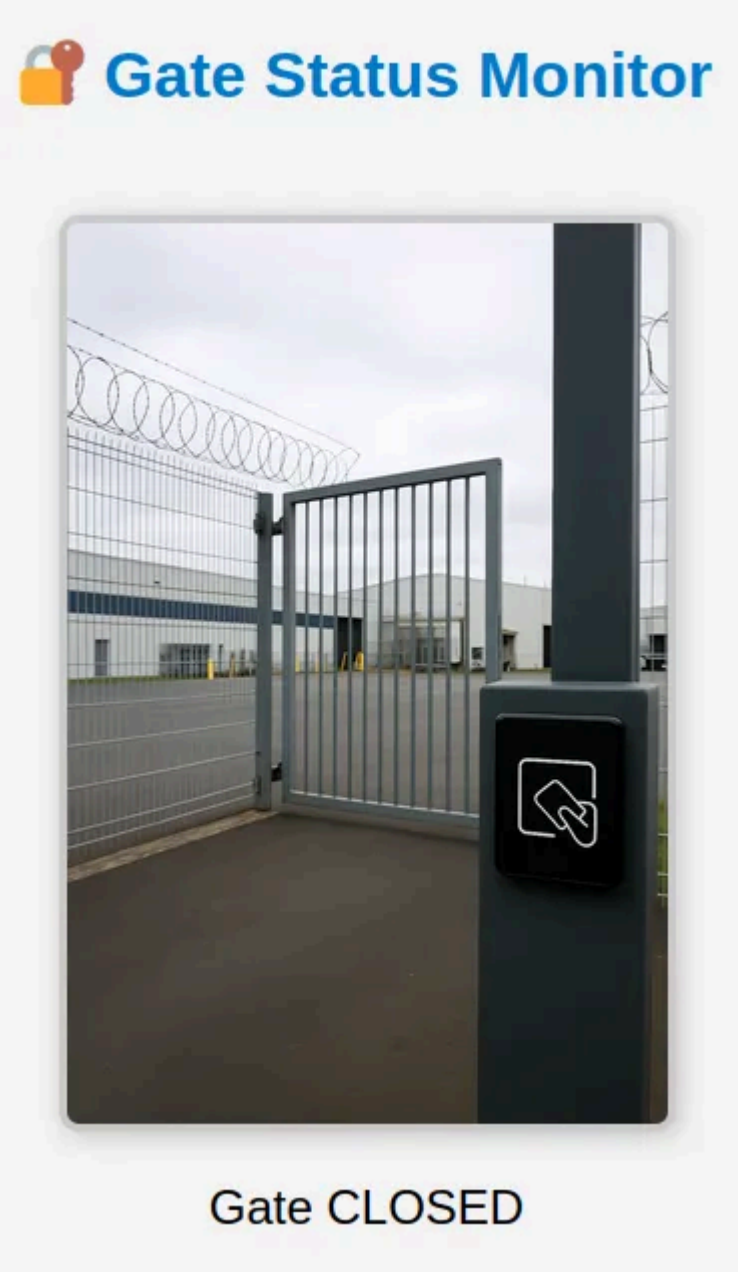
- 22/tcp (SSH)
- 80/tcp (HTTP)
- 102/tcp (iso-tsap)
- 502/tcp (Modbus)
- 1880/tcp (vsat-control)
- 8080/tcp (HTTP Proxy)
- 44818/tcp (EtherNetIP-2)

The standout was 1880/tcp, running Node-RED, a flow-based tool commonly used in Industrial Control Systems (ICS). This hinted that Node-RED could be central to controlling the system.

```
┌──(fc0d3x_guest㉿kali)-[~]
└─$ nmap -p- -min-rate=1000 -T4 10.10.40.202 -oN full_port_scan.txt
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-29 15:28 EEST
Nmap scan report for 10.10.40.202
Host is up (0.063s latency).
Not shown: 65528 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
102/tcp   open  iso-tsap
502/tcp   open  mbap
1880/tcp  open  vsat-control
8080/tcp  open  http-proxy
44818/tcp open  EtherNetIP-2

Nmap done: 1 IP address (1 host up) scanned in 25.15 seconds
```

Phase 2: Web Enumeration (Gobuster) & Service-Specific Research



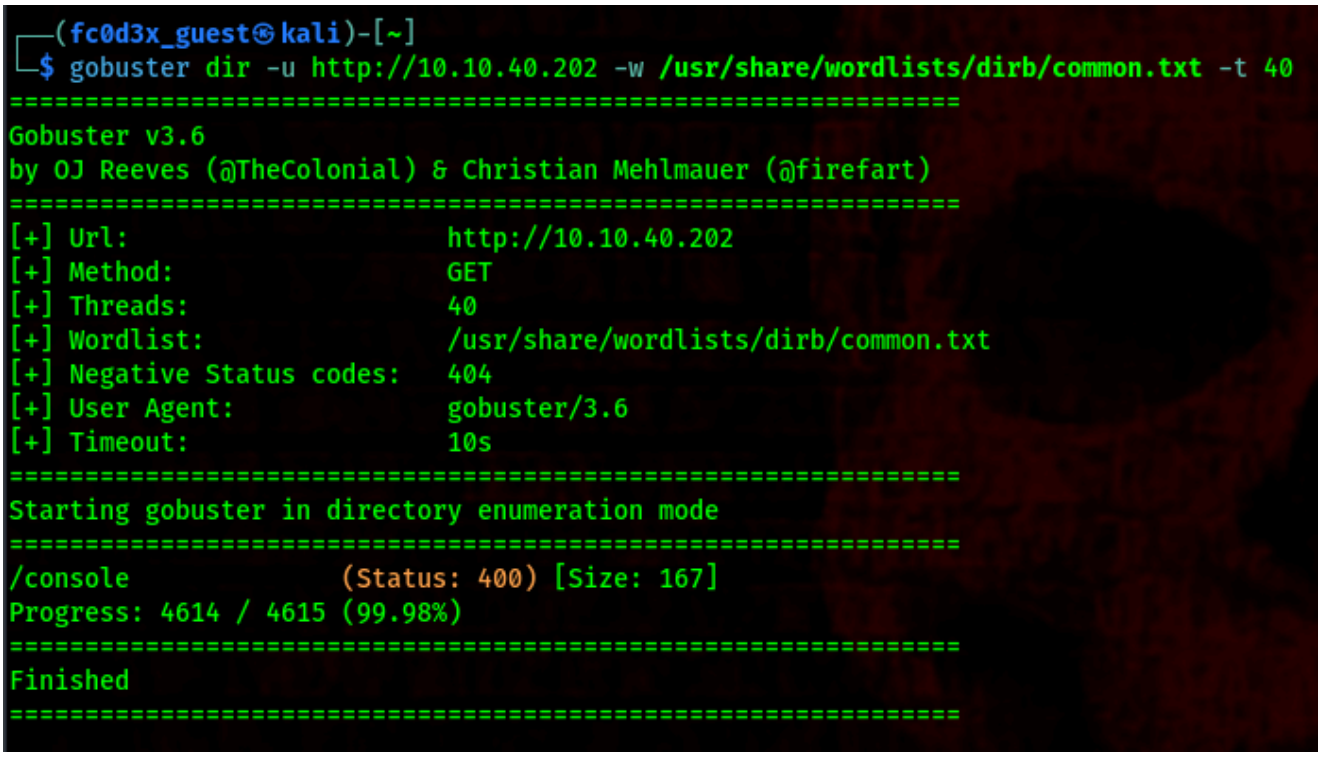
To further investigate the target system, I used Gobuster to run a web directory scan on port 80, searching for hidden directories or endpoints:

```
gobuster dir -u [http://10.10.40.202](http://10.10.40.202/) -w
/usr/share/wordlists/dirb/common.txt -t 40
```

This scan uncovered a /console path, but nothing related to gate control or Node-RED was exposed directly. I then researched Node-RED default ports and paths and found:

- Node-RED often runs on port 1880.
- The Node-RED editor is typically located at /.
- The dashboard UI elements are located at /ui.

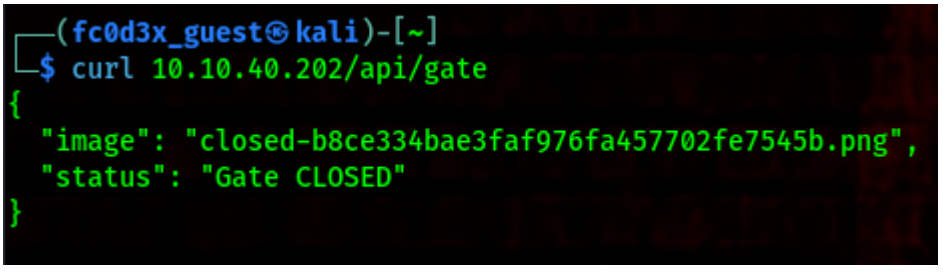
This gave me insight into the potential target area for the badge authentication.



Phase 3: Initial Gate Status Check & Exploitation

In this phase, I was looking to interact with the gate control system. After examining the web application, I found an API endpoint:

```
curl [http://10.10.40.202/api/gate](http://10.10.40.202/api/gate)
```

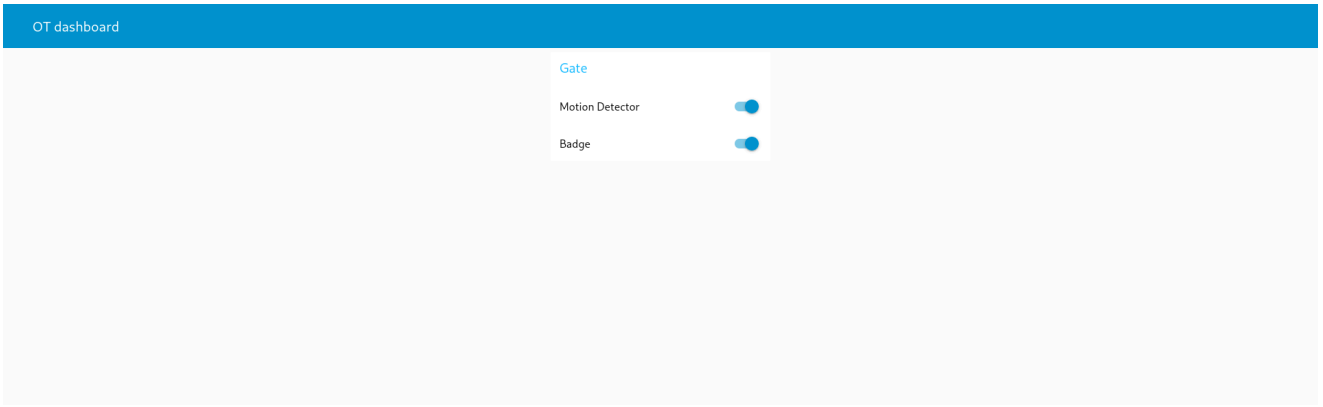


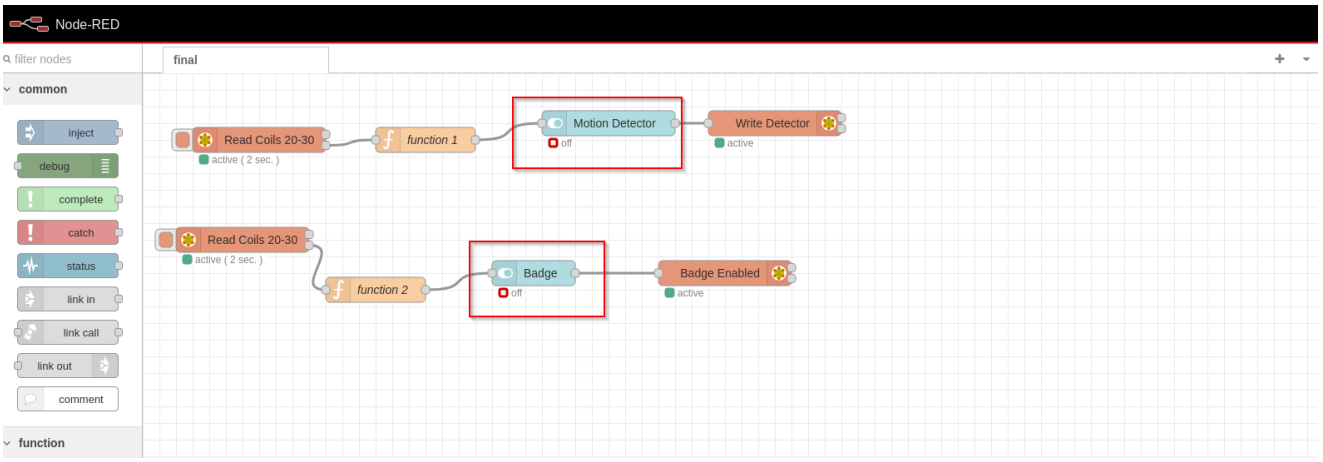
Next, I accessed the Node-RED dashboard UI directly at:

```
[http://10.10.40.202:1880/ui](http://10.10.40.202:1880/ui)
```

The interface was simple and included a "Badge" toggle switch, likely associated with the gate control.
The best part?
The dashboard was unauthenticated, meaning I could toggle the switch without needing to log in.

With a quick toggle of the switch, I triggered the internal flow responsible for controlling the gate, successfully opening it.





Phase 4: Verification

In Phase 4, the system's functionality was verified to ensure that all components were working as intended, including confirming the gate's open status and the interaction with the dashboard.

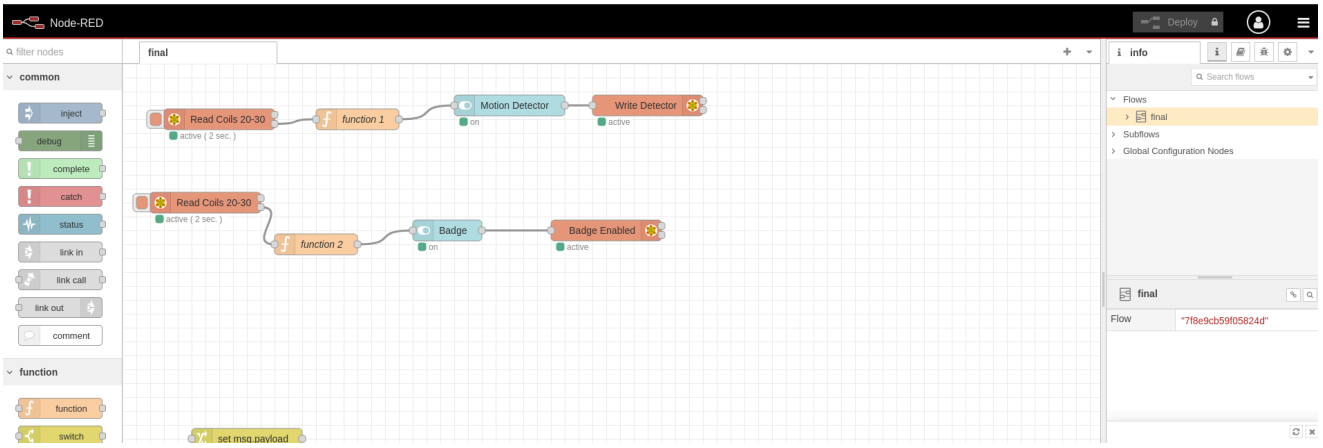
Node-RED Flow Configuration:

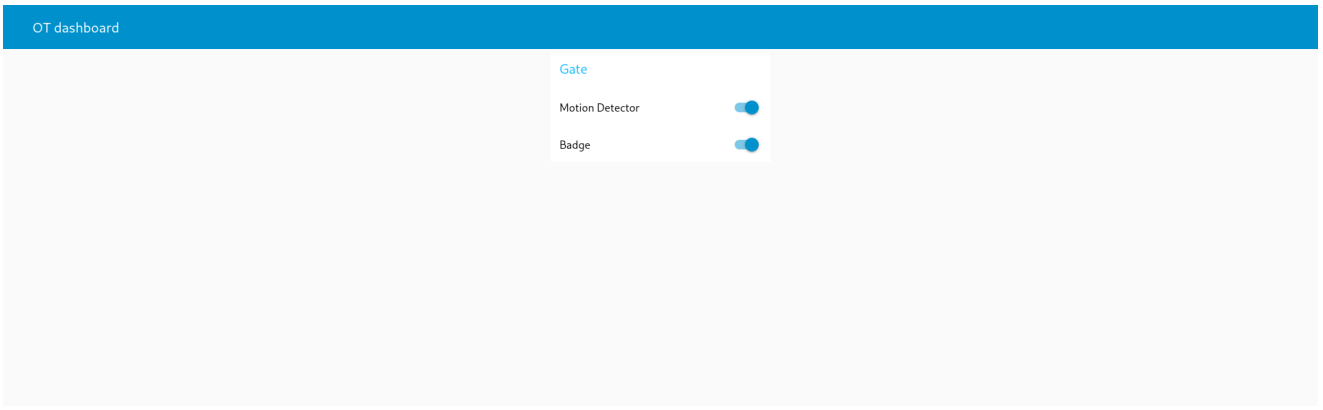
I reviewed the Node-RED configuration, which was set up to control multiple sensors and features. The flow included:

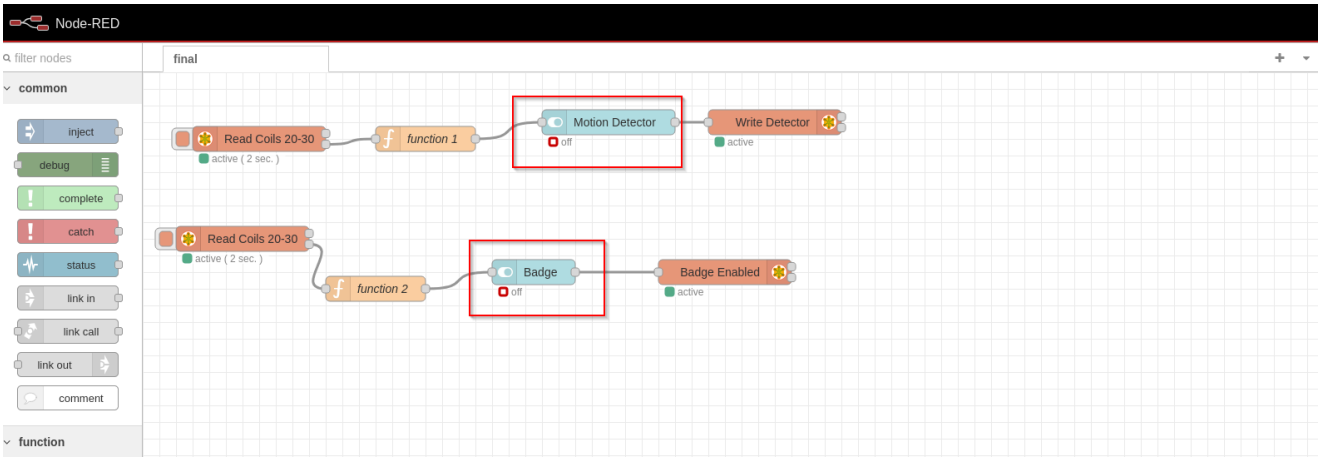
- Read Coils 20-30: Nodes read the state of various sensors.

- Motion Detector and Badge: These were toggled to trigger internal actions in the system.

This flow was structured to control the motion detector, badge system, and other interactions such as the gate status.





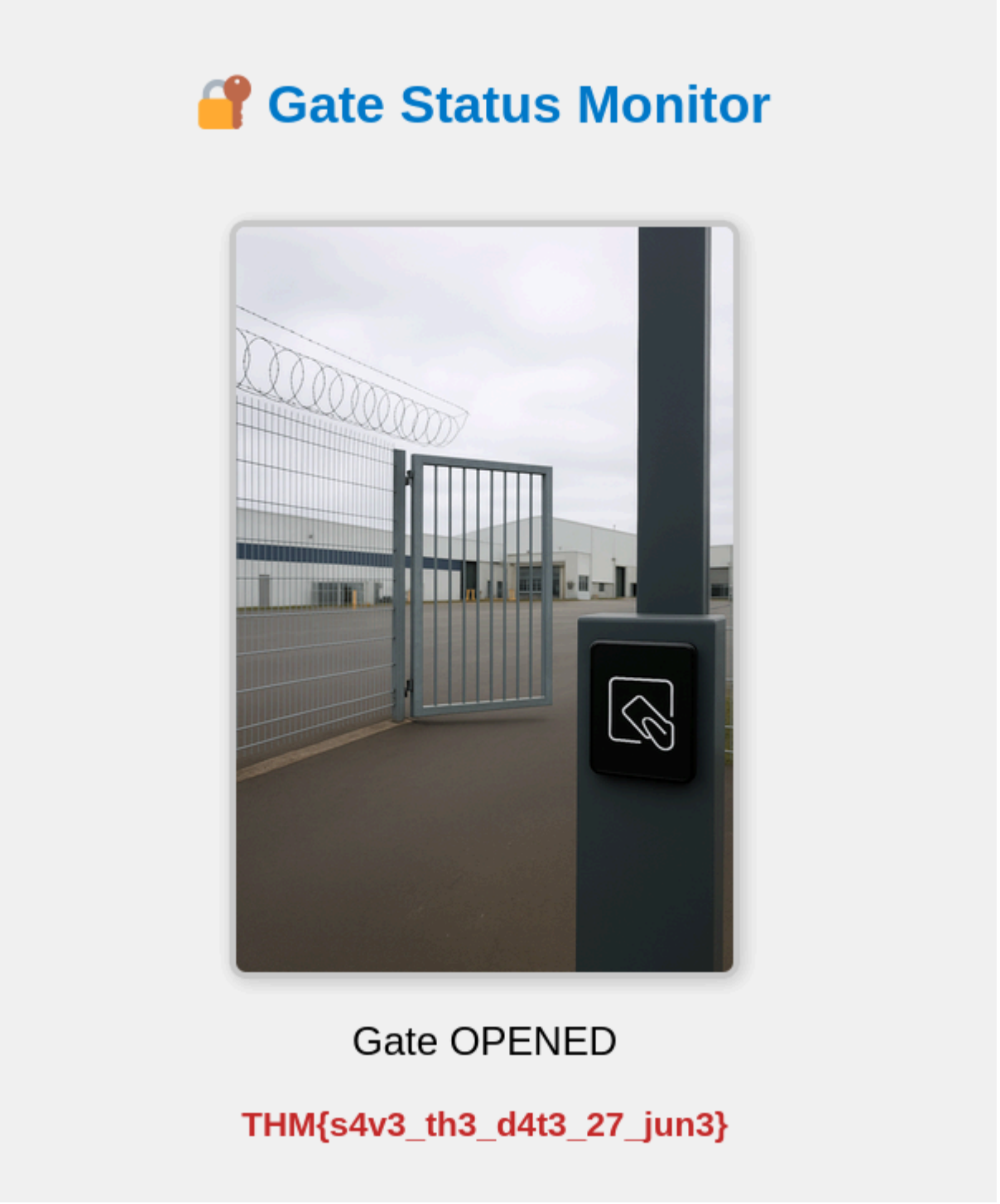


Dashboard Interaction:

I accessed the OT dashboard again to visually confirm the state of the gate. The Gate Status was reflected accurately on the dashboard, showing that the gate was open.

Final Gate Status:

- The badge system was properly enabled, and the motion detector was functioning.
- The CTF flag was displayed as part of the verification:




```
THM{s4v3_th3_d4t3_27_jun3}
```

This successful verification confirmed that all systems were operating correctly, including the badge system and gate control.

Conclusion:

In Phase 4, the system was successfully verified by interacting with the Node-RED flows and confirming that the gate control and related systems worked as intended. The CTF flag was

successfully retrieved, demonstrating a full compromise and successful exploitation of the system.