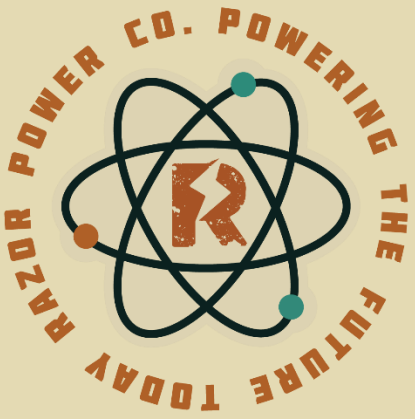
RazorHack 2024 Writeup



“Why is Homer Sampson Always in The Cafeteria”, “You Don’t Have The Necessary Permissions”, & “Employee Policy Manual Rule 5”

RPCO-M-59x02-12

a.k.a. Access Logs

Through your analysis of the email server image, you found out the intruder sent an email to Marzell posing as Ayden asking for a badge to be made for a new employee named Alex Patrick Turner (APT). The next step for the intruder would be to get into the building using the badge. Shane pulled the company access logs, can you find out some more information about the intruder’s movements?

Author: Henry

In this challenge, participants were tasked with analyzing a set of RPCo access logs. The challenge description describes each line as indicating an event that includes a timestamp, a common badge access format, and whether the individual was allowed or denied. Below is a snapshot of what this format looks like.

```
0000000066fbac0011011001000010100011111011101
0000000066fbac1711110001101010100001001111101
0000000066fbac4a1111011100101010000100111100
0000000066fbac7311101001010010110000110011101
0000000066fbac8511011001000111100001101001101
0000000066fbac8811110001111100010111111011101
0000000066fbacb2111010010011110100110010110 1
0000000066fbacdd1101100101110000101100001101
```

The first task is to find the date and time of when the intruder first entered the building using the badge ID you found in the deleted email. However, I think it's important to first analyze the components of each record to better understand the logs. You are told the order of information of the logs in the description, being timestamp, a common badge access format, and whether the user was allowed or denied. We can see that each record is the same length. The first record is the timestamp. Just by looking at the record you can see it switches from hex to binary after the 16th digit, thus the first 16th digits are likely the timestamp. There are many ways you can go about decoding this, but I prefer making a Python script that can later be used to make a file with all the records decoded.

```
timestamp_hex = "0000000066fbac17"
timestamp = int(timestamp_hex, 16)
normal_time = datetime.fromtimestamp(timestamp).strftime('%Y-%m-%d %H:%M:%S')
print(normal_time)

2024-10-01 08:00:23
```

You can get the time by transforming the hex to int and then int (timestamp) to a date time format. The second thing we need to analyze is the next 28 binary bits of the log.

We are told that this is a common badge access format. A quick Google search tells us that the Weigand format is a likely candidate.

common badge access format

All Images Shopping Videos Forums Web News More Tools

Search Labs | AI Overview

The most common format for access badges is the 26-bit Wiegand format, also known as H10301:

Structure

The format includes a facility code, card number, and parity bits for error checking.

Open format

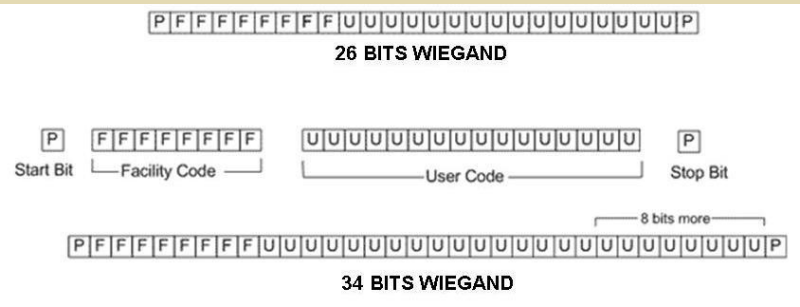
This means that anyone can buy and sell 26-bit Wiegand cards without permission.

Industry standard

Almost all access control systems support this format.

Show more

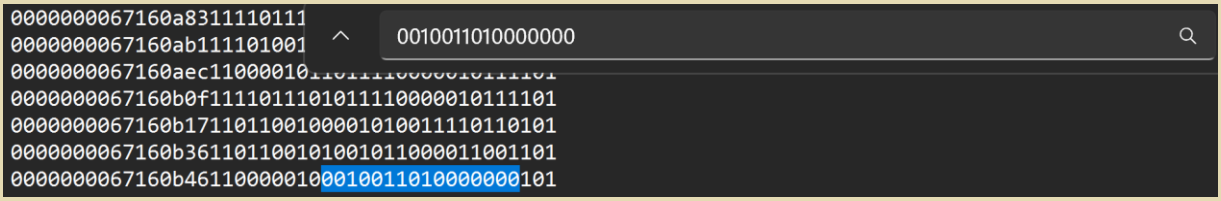
A common proximity format is 26 bit Wiegand. This format uses a facility code, also called a site code. The facility code is a unique number common to all of the cards in a particular set. The idea is an organization has their own facility code and then numbered cards incrementing from 1.



This common badge access format includes an 8-bit facility code and 16-bit user code padded by one bit on either side. If we are counting bits from left to right, which would be in the order provided in the description, that would mean we have two bits left. These two bits must represent whether an employee is denied or allowed access to the door. You can notice this in the first ten to twenty lines of logs, where denied is represented by '00' and allowed is represented by '01'.

```
0000000066fbae01110110010101010110100110010
0000000066fbae25101100011111000101111101100
0000000066fbae50111010010011010000111101010
```

Now that we know how the logs are formatted let's go back to the first challenge. This challenge asks us to find the first time that the intruder entered the building using the provided badge number in the email. For this challenge we can simply CTRL+F {badge ID} and translate the first entry we find to get this time. That gets us to the following line.



Using the information, we found before we can create code to translate each log record into a readable format, this will help with the next two challenges. Using this code (which is listed below) we can find that the datetime for the first entry is the following.

```python
from datetime import datetime
import re

def parse_log(log_entry):
    """
    Parses a log entry and returns the normal time, door code, badge code, and access status.

    Log format:
    - Timestamp (64-bit hex)
    - Start digit '1'
    - Door identifier (8-bit binary)
    - Badge code (16-bit binary)
    - Stop digit '1'
    - Access status ('01' for allowed, '00' for denied)
    """
    log_pattern = r"^(?P<timestamp>[0-9a-f]{16})1(?P<door_code>[01]{8})(?P<badge_code>[01]{16})1(?P<access_status>01|00)$"
    match = re.match(log_pattern, log_entry)

    if not match:
        raise ValueError("Invalid log entry format")

    timestamp_hex = match.group('timestamp')
    door_code = match.group('door_code')
    badge_code = match.group('badge_code')
    access_status_code = match.group('access_status')

    # Convert timestamp to datetime
    timestamp = int(timestamp_hex, 16)
    normal_time = datetime.fromtimestamp(timestamp).strftime('%Y-%m-%d %H:%M:%S')

    access_status = 'Allowed' if access_status_code == '01' else 'Denied'

    return {
        'Time': normal_time,
        'Door Code': RazorPowerCoAccessTree.get_node(door_code).tag,
        'Badge Code': badge_code,
        'Access Status': access_status
    }
```

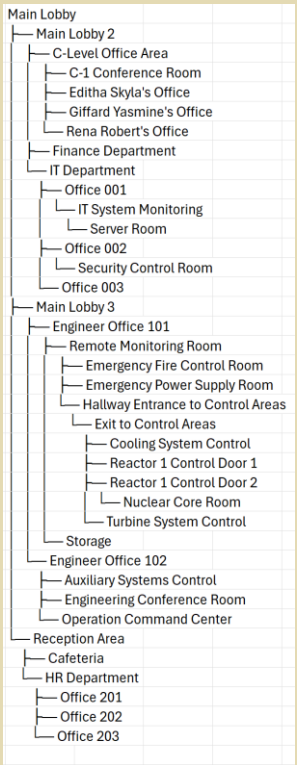
The second challenge in this series asks you to find out whether the intruder was denied access to any rooms during his first visit to the nuclear power campus. This would be important information to know as it would tell you what level of privilege the attacker has on his badge as well as what controls he might be interested in. This time you are given more information about the RPCo layout with the inclusion of an Excel sheet which gives you the names and code corresponding to each door as well as a tree of how they are connected. This challenge asks you to provide the name of the door and time they tried to access it. Similarly to the previous question, CTRL+F is your friend. Using this and looking at the access status we see the first entry is for

```
{'Time': '2024-10-21 09:16:16', 'Door Code': 'Main Lobby 3', 'Badge Code': '1110001011111011', 'Access Status': 'Allowed'}
{'Time': '2024-10-21 09:16:40', 'Door Code': 'IT Department', 'Badge Code': '0010011010000000', 'Access Status': 'Denied'}
{'Time': '2024-10-21 09:16:43', 'Door Code': 'Engineer Office 101', 'Badge Code': '1011110000010111', 'Access Status': 'Allowed'}
```

the IT Department at 9:16:40 on the day of their arrival. The second entry can be found by continuing to scroll through found IDs. Here we can see that they failed to get into the Engineering Office 101 at 10:30:28.

```
{'Time': '2024-10-21 10:30:20', 'Door Code': 'Office 003', 'Badge Code': '0011110000110100', 'Access Status': 'Allowed'}
{'Time': '2024-10-21 10:30:28', 'Door Code': 'Engineer Office 101', 'Badge Code': '0010011010000000', 'Access Status': 'Denied'}
{'Time': '2024-10-21 10:30:29', 'Door Code': 'Main Lobby', 'Badge Code': '0101010000100111', 'Access Status': 'Allowed'}
```

In the final challenge Shane suspects that the intruder may have copied someone's badge and returned at a different time, given the fact that new badge ID for Alex Patrick Turner doesn't show up after that initial day. Your job is then to find whose ID was cloned. This question requires some ingenuity. There are a couple of things you can knock out of the way immediately. The first thing you can ignore is any log before the lasy log of the first day for APT. This is because we are only looking for instances after the intruder was initially there. The next thing to notice is the door tree in the Excel file given in the previous challenge. An employee has to sequentially go through each door on the way down and up the tree. An engineer coming into work in engineering must go through the Main Lobby, Main Lobby 3, and then the Engineer Office 101, and to go back out they must go back through Main Lobby 3 and Main Lobby. This tree can be seen below.



With these two assumptions we can start to work through the employee IDs to find discrepancies. The main discrepancy we are looking for is when an employee seemingly jumps from one area to another or is in two different places at nearly the same time. There are two ways to go about this, either manually or with code. Either way I think the best way to approach this problem is to look at the problem per employee ID badge. This challenge is limited to 5 guesses for a reason, there are only 11 employee codes in this log. Using the tree provided in Access_Door_Codes.xlsx we can filter the logs by employee and check to see if each path traversed was allowed.

An example of this code and how the access code logs were made can be found here: https://colab.research.google.com/drive/1XuUn8Q-484W3ISZIPxJBSYfqwoEfXCva?usp=sharing

This code tells us that the only possible option for whose badge the intruder stole is 1010000100010000, also known as Homer Sampson's badge.

```
Processing employee logs...
Processing employee with badge code: 0001010001111011
Processing employee with badge code: 0101010000100111
Processing employee with badge code: 1001011000011001
Processing employee with badge code: 0011110000110100
Processing employee with badge code: 1110001011111011
Processing employee with badge code: 0111101001100101
Processing employee with badge code: 1110000101100001
Processing employee with badge code: 0110100001111010
Processing employee with badge code: 0001010011110110
Processing employee with badge code: 1010000100010000
The last node (Engineer Office 102) is not a parent, child, or sibling of the current_node (Hallway Entrance to Control Areas)
Found at time 2024-10-23 09:32:56
The last node (Remote Monitoring Room) is not a parent, child, or sibling of the current_node (Auxiliary Systems Control)
Found at time 2024-10-23 10:30:49
The last node (Auxiliary Systems Control) is not a parent, child, or sibling of the current_node (Engineer Office 101)
Found at time 2024-10-23 11:14:22
The last node (Remote Monitoring Room) is not a parent, child, or sibling of the current_node (Auxiliary Systems Control)
Found at time 2024-10-23 11:54:24
The last node (Engineer Office 102) is not a parent, child, or sibling of the current_node (Hallway Entrance to Control Areas)
Found at time 2024-10-23 11:55:47
The last node (Exit to Control Areas) is not a parent, child, or sibling of the current_node (Engineering Conference Room)
Found at time 2024-10-23 12:04:45
The last node (Main Lobby 3) is not a parent, child, or sibling of the current_node (Cooling System Control)
Found at time 2024-10-23 12:35:19
The last node (C-Level Office Area) is not a parent, child, or sibling of the current_node (Exit to Control Areas)
Found at time 2024-10-23 14:03:02
Processing employee with badge code: 1100111010111100
Processing employee with badge code: 1011110000010111
Processing employee with badge code: 0010101100110100
Processing employee with badge code: 1010101101001100
Processing employee with badge code: 0000111001011010
Processing employee with badge code: 1001111011010110
Processing employee with badge code: 1110110001101010
Processing employee with badge code: 0101011110111000
Processing employee with badge code: 1001101110111010
Processing employee with badge code: 0000111111110001
Processing employee with badge code: 1101110001001101
Processing employee with badge code: 1000110111111001
Processing employee with badge code: 0000111101010100
Processing employee with badge code: 1011111111001010
Processing employee with badge code: 0010011010000000
```