Scenario

In this project, I'm developing an automated login attempt analysis system for a specific device. My task involves creating essential variables to track login security, including a device ID, a list of approved usernames, maximum login attempts allowed, current login attempts, and login status. I'll assign appropriate values to these variables, verify their data types, and set up a mechanism to monitor and control login access. The goal is to enhance device security by implementing a structured approach to tracking and managing user authentication attempts.

As part of my cybersecurity project, I'll create a variable device_id to represent a specific device with restricted access. I'll assign the unique identifier `72e08x0` to this variable and verify its contents, which is a critical first step in implementing a secure access control system.

In [1]:
```
# Assign the `device_id` variable to the device ID that only specified users can access
device_id = "72e08x0"
# Display `device_id`
print(device_id)
```

```
72e08x0
```

In this next step of my cybersecurity project, I'll determine and verify the data type of the `device_id` variable. I need to use a Python function to identify the data type, store it in a new variable called `device_id_type`, and then display this new variable. This process ensures that the device_id is stored in the correct format for our security system.

In [10]:
```
# Assign the `device_id` variable to the device ID that only specified users can access
device_id = "72e08x0"
# Assign `device_id_type` to the data type of `device_id`
device_id_type = type(device_id)
# Display `device_id_type`
print(device_id_type)
```

```
<class 'str'>
```

In this task, I'm creating an access control mechanism by developing a `username_list` variable containing approved usernames `"madebowa"`, `"jnguyen"`, `"tbecker"`, `"nhersh"`, `"redwards"` to establish a robust authentication process for restricting device access and enhancing system security.

In [13]:
```
# Assign `username_list` to the list of usernames who are allowed to access the device

username_list = ["madebowa","jnguyen", "tbecker", "nhersh", "redwards"]

# Display `username_list`

print(username_list)
```

```
['madebowa', 'jnguyen', 'tbecker', 'nhersh', 'redwards']
```

In this phase, I'm tasked with identifying and verifying the data type of the `username_list` variable. I'll use Python's built-in function to determine the data type, store it in a new variable called `username_list_type`, and then display this information. This step ensures that our list of approved usernames is stored in the correct format, which is crucial for the proper functioning of our access control system.

In [19]:
```
# Assign `username_list` to the list of usernames who are allowed to access the device

username_list = ["madebowa", "jnguyen", "tbecker", "nhersh", "redwards"]

# Assign `username_list_type` to the data type of `username_list`

username_list_type = type(username_list)

# Display `username_list_type`
print(username_list_type)
```

```
<class 'list'>
```

In this phase, I need to update the `username_list` to include a new employee who has been granted access to the device. I'll first display the current list, then update it with the new information, and finally show the updated list to confirm the changes.

In [20]:
```
# Assign `username_list` to the list of usernames who are allowed to access the device

username_list = ["madebowa", "jnguyen", "tbecker", "nhersh", "redwards"]

# Display `username_list`

print(username_list)

# Assign `username_list` to the updated list of usernames who are allowed to access the device

username_list = ["madebowa", "jnguyen", "tbecker", "nhersh", "redwards", "lpope"]

# Display `username_list`

print(username_list)
```

```
['madebowa', 'jnguyen', 'tbecker', 'nhersh', 'redwards']
['madebowa', 'jnguyen', 'tbecker', 'nhersh', 'redwards', 'lpope']
```

In this stage of, I'm implementing a login attempt limit feature. My task is to create a variable `max_logins` and set it to 3, representing the maximum number of login attempts allowed per user. I'll then determine and store the data type of this variable in `max_logins_type`. Finally, I'll `display max_logins_type` to confirm the correct data type is being used.

```
In [1]: ### Max Login Attempts###
        max_logins = 3

        ### Login Type ###
        max_logins_type = type(max_logins)

        ### Display Login Type###
        print(max_logins_type)
```

```
<class 'int'>
```

In this step, I will create a variable for `login_attempts`, assign it the value `2`, determine its data type, and store that type in a separate variable. I'll then display the data type to understand how Python classifies the login attempt count.

```
In [2]: ### Max Login Attempts ###
        login_attempts = 2

        ### Login Attempts Type ###
        login_attempts_type = type(login_attempts)

        ### Display Login Type ###
        print(login_attempts_type)
```

```
<class 'int'>
```

In this step, I will evaluate whether the user's current login attempts are within the acceptable threshold by comparing the number of attempts made to the maximum number of attempts allowed. The goal is to generate a Boolean value that indicates if the login process should continue or be blocked, ensuring system security while preventing unauthorized access attempts.

```
In [4]: # Assign `max_logins` to the value 3

        max_logins = 3

        # Assign `login_attempts` to the value 2

        login_attempts = 2

        # Determine whether the current number of login attempts a user has made is less than or equal to the maximum number
        # and display the resulting Boolean value

        print(login_attempts<=max_logins)
```

```
True
```

In this step, I'll reassign different values to the `login_attempts` variable, including some that exceed the maximum allowed attempts. By observing how the output changes with these new values, I'll see how the comparison behaves under various conditions, helping me understand the login attempt validation logic.

In [5]:
```
# Assign `max_logins` to the value 3

max_logins = 3

# Assign `login_attempts` to a specific value

login_attempts = 8

# Determine whether the current number of login attempts a user has made is less than or equal to the maximum number
# and display the resulting Boolean value

print(login_attempts <= max_logins)
```

```
False
```

In this step, I'll create a Boolean variable called login_status, set it to False to indicate the user is not logged in, store its data type in `login_status_type`, and then `display login_status_type` to confirm it's a Boolean.

In [6]:
```
# Assign `login_status` to the Boolean value False

login_status = False

# Assign `login_status_type` to the data type of `login_status`

login_status_type = type(login_status)

# Display `login_status_type`

print(login_status_type)
```

```
<class 'bool'>
```