hr@wazuh.com
https://wazuh.com

Introduction

Wazuh Inc. is an American IT security company focused on Host-based Intrusion Detection Systems (HIDS). Specifically, we contribute to the world of Open Source Security developing and improving the capabilities of Wazuh software.

Below, there are some tasks that the candidate must perform. Solve them giving the requested **source code and an explanation** of the proposed solution. The source code must be written using Python. Thanks in advance for your time working on these tasks.

Do as much as you can! Never surrender! We look for quality rather than quantity.

Thanks in advance for your time working on these tasks.

Best regards,

Wazuh, Inc.

Python tasks

Task 1.1

Given the code below, implement `MyCustomManager` giving the minimum expected functionality:

```
with MyCustomManager('/path/to/file.csv', sep=',') as o:
    o.set_columns(['Column 0', 'Column 1'])
    try:
        o.add_rows(data=[['test1', 'test2'], ['test3',
'test4']])
        print(f"Number of rows: {len(o.rows)}
        print(f"Number of columns: {len(o.columns)})
    except BadRowError as e:
     print('Tried to add a row with incorrect columns')
```

Deliverables

- Python script

Task 1.2

Using pytest, write the best unit tests you can for the following function:

```
import socket

def write_to_socket(data):
    """
    Write data to socket
    :param data string to write to the socket
    :return boolean
    """

    socket_path = '/some/hardcoded/path'
    if exists(socket_path):
        try:
            conn = socket.socket(socket.AF_UNIX,
socket.SOCK_DGRAM)
            conn.connect(socket_path)
        except socket.error:
            raise ValueError(f"Error sending data to
{socket_path}")
    else:
     raise ValueError(f"Socket {socket_path} does not
exist")

    try:
        conn.send(data.encode())
        conn.close()
    except socket.error as e:
        raise ValueError(f"Error sending data")

    return True
```

Deliverables

- Python script

Task 1.3

We need to add support for users and privileges in our web application. These privileges are called roles and they define permissions for a specific action. We are looking to generate a database with these entities and to provide a way to manage them.

A user is composed of `user_id (int)`, `username (str)`, `password (str)`, `creation date (datetime)`, `deleted (bool)`. As for the roles, they are composed of `role_id (int)`, `role_name (str)`, `description (str)`, `creation date (datetime)` and `deleted (bool)`.

Users will obtain privileges based on the roles assigned to them. A user may or may not have any roles assigned to them.

- Develop the logical relational schema.
- When saving passwords in a database, how would you do it?
- Implement the necessary classes for the database administration, as well as three basic methods to add, modify and delete entities and relationships.

HINT: It is possible to map relational database structures to Python objects with tools such as ORM, SQLAlchemy, etc.

Deliverables

- Python script

API tasks

Task 2

Attached to this document you will find an api.tar.gz file containing the minimum necessary files to run a simple but operational OpenAPI REST server with python. This API provides an endpoint that should only allow reading files inside the data folder. Using the attachment, try to complete the following tasks:

1. Run the api service (api.py) completing any necessary steps.
2. Using cURL or any API client (Insomnia, postman, etc) run an API request to read file test.txt content (the file is contained in the attachment).
3. There are two potentially hazardous bugs in the current implementation. One regarding data access and a known vulnerability, are you able to find them?
4. Make any necessary changes to api.py and api_spec.yaml to add another endpoint that would allow deleting files from the data folder.

Deliverables

- Output examples
- Modified API files

Docker tasks

Task 3

Create and deploy a docker environment using `docker-compose v3.7` which follows the next guidelines. Please deliver any necessary files to run the environment.

1. The environment must have 3 containers:
   - `server_node`: use a fresh **CentOS 7** image as base.
   - `client_node_1`: use a fresh **Ubuntu 18** image as base.
   - `client_node_2`: use the same image as the first client container.
2. The server container image must have **Python v3.9** installed and running properly. For the client containers, their image should have **cURL** installed.
3. Both client services must wait for the server container to be done before starting.
4. Add the configuration file from below to the client images. Once the containers have started, replace the **wildcards** with the correct values keeping in mind that each client would connect to the server and the **second client** would be **disabled** using the same entrypoint file. DNS are allowed.
5. Add a healthcheck for the server container that checks that it has connection to the client nodes.

   NOTE: You do not have to use this configuration file for any connection task.

```
<-- Configuration file template. Wildcards cannot be modified
manually. -->

<configuration>

  <enabled>IS_ENABLED</enabled>

  <server>SERVER_IP</server>

  <protocol>tcp</protocol>

<configuration>
```

configuration.xml

Form of delivery

- PDF containing each task answer written in English, screenshots and any other information you consider in forming your response.
- ZIP file containing all the deliverables from each task.
- All files must be sent to hr@wazuh.com.

Due date

Your completed tasks are due seven days from the receipt of this document.