|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task:** | | **5** | | |
| **Task Title:** | | **Portfolio** | | |
| **Task Code:** | | **AT2 POR-Task-5** | | |
|  | |  | | |
| Assessment type (): | | | | |
|  | Questioning (Oral/Written) | |  | Portfolio |
|  | Practical Demonstration | |  | Project |
|  | 3rd Party Report | |  | Other – Please Specify |

|  |
| --- |
| The base requirements this assessment task include:   * Web server, Python interpreter and database server * IDE or editor for developing Python programs (only PyCharm supported by the college) * Raspberry Pi with SenseHat or other IoT devices, like Arduino Uno or ESP32 * Access to Office 365 & Microsoft Word * Account with Adafruit.io   Use of some of these items may not occur in this part of the assessment task. |
| Assessment Due This assessment is split into components that have several due dates that range from Session 07 to Session 14.  Refer to Blackboard for most accurate dates, which may alter due to unforeseen circumstances.  We also will endeavour to update these document(s) at the same time. |
| Instructions Follow the steps listed in this assessment item.  Submission of the documentation, code, and associated items is at the end of each part of the portfolio.  Each part of the portfolio has a deadline for submission.  It is advantageous to you to attempt to meet the deadline provided. |
| Important If you are using a different configuration of tools and equipment for this assessment item, then assistance in this and subsequent parts of the portfolio to ensure the systems work correctly will be limited. |
| Scenario You are currently working for a small Perth-based start-up company called Incredibly Obvious Technologies.  They are looking to create a presence in the home automation and monitoring market.  Full details in Appendix A: Supporting Details for Portfolio. |

|  |  |  |
| --- | --- | --- |
| **STEP** | Task to perform | |
| 00 | Create Evidence Document Make sure you have followed the instructions on creating the answer document, as given in the General Instructions.  Familiarise yourself with the content and document your progress in this assessment.  Make sure that you complete the title page of the document.  If the question requires you to submit code with your final submission, you may add it as a zip-file **OR in a separate folder**. | |
| 01 | Purpose of application security Research application security in software development. In your own words, describe the purpose of application security in software development. | |
| A01 | This area will expand automatically. What is the purpose of Application Security? [4 – 20 sentences] Application security is the process of developing, adding and testing security features within the applications to prevent security vulnerabilities. In today’s world, most of the applications are connected to the internet, hence making it vulnerable to security threats and breaches. Application security testing can reveal weaknesses, helping to prevent these attacks. Some of the well-known application security features include authentication, authorisation, encryption, and application security testing. | |
| 02 | Policy-based Access Control In this step, you will create a simple API which incorporates a basic form of Policy-based Access Control, or PBAC (sometimes dubbed Attribute-based Access Control, or ABAC).  Follow these instructions:  Create a new project in PyCharm and create a virtual environment. Make sure you are using Python version 3. You may enable Git integration and use version control throughout this assessment.  Install the package **fastapi-authz** using pip. This will also install a few dependencies. Install the package **uvicorn**.  Create a new folder called **api** and within it, create a file called **main.py**.  Create the foundation for a simple API by adding the following lines to the script:  from fastapi import FastAPI  api = FastApi()  @api.get(‘/’)  async def index():  return ‘Hello, world.’  Open a terminal in PyCharm and run the following command from within the **api/** folder:  uvicorn main:api --reload  Open the link [**http://127.0.0.1:8000/docs**](http://127.0.0.1:8000/docs) in your browser. Read the provided information. Try out the newly created endpoint and add a screenshot of the result to this document. | |
| A02 | This area will expand automatically. Place Screenshot Below | |
| 03 | Create the policy files To enable PBAC (more precisely, Role-based Access Control, or RBAC – this is a subset of PBAC), we will first create two files in the **api/** folder: **rbac\_model.conf** and **rbac\_policy.csv**.  Add the following lines to the files:  [**rbac\_model.conf**]  [request\_definition]  r = sub, obj, act  [policy\_definition]  p = sub, obj, act  [role\_definition]  g = \_, \_  [policy\_effect]  e = some(where (p.eft == allow))  [matchers]  m = (p.sub == "\*" || g(r.sub, p.sub)) && (r.obj == p.obj || keyMatch(r.obj, p.obj)) && (p.act == "\*" || r.act == p.act)  The last line is just a single line. Make sure your spacing and punctuation are correct.  [**rbac\_policy.csv**]  p, alice, /ds1/\*, GET  p, alice, /ds1/res1, POST  p, bob, /ds2/res2, GET  Finally, add the following lines to your main script (highlighted):  **import casbin**  from fastapi import FastAPI  api = FastApi()  **enforcer = casbin.Enforcer(**  **‘rbac\_model.conf’,**  **‘rbac\_policy.csv’,**  **)**  @api.get(‘/’)  async def index():  return ‘Hello, world.’  Monitor the output of the **uvicorn** process in your terminal to check for errors (it will try to restart because of **--reload**). Even though the newly create **enforcer** isn’t used, it should load the required files correctly. Fix any errors you encounter.  Answer the following questions:   * Describe the **meaning** of “sub”, “obj” and “act” in the configuration file. * In your own words, describe the authorisations that Alice and Bob have in the policy file. * How do “sub”, “obj” and “act” relate to the various fields in the policy file? | |
| A03 | This area will expand automatically. Describe the meaning of “sub”, “obj” and “act” in the configuration file. [3 – 10 sentences] …  … In your own words, describe the authorisations that Alice and Bob have in the policy file. [3 – 5 sentences] …  … How do “sub”, “obj” and “act” relate to the various fields in the policy file? [3 – 10 sentences] …  … | |
| 04 | Create the required middleware You will now add the required middleware. The package that you are using integrates nicely with any middleware that implement the AuthenticationMiddleware interface.  Follow these instructions.  Create a new file called **basic\_auth.py**. It should reside in the **api/** folder.  Add the following code to the file:  import base64  import binascii  from starlette.authentication import (  AuthenticationBackend,  AuthenticationError,  AuthCredentials,  SimpleUser  )  class BasicAuth(AuthenticationBackend):  async def authenticate(self, request):  if 'Authorization' not in request.headers:  return None  auth = request.headers['Authorization']  try:  scheme, credentials = auth.split()  decoded = base64.b64decode(credentials).decode('ascii')  except (ValueError, UnicodeDecodeError, binascii.Error):  raise AuthenticationError('Invalid credentials')  username, \_, password = decoded.partition(':')  return AuthCredentials(['authenticated']), SimpleUser(username)  As you can see, this class implements a method called **authenticate**.  Answer the following question: In your own words, describe what the method does. You may refer to any documentation that may exist for the class AuthenticationBackend.  Answer in the A04 section provided. | |
| A04 | This area will expand automatically. In your own words describe what the authenticate method does. [3 – 10 sentences] …  … | |
| 05 | Add the required middleware For the access control to work properly, we need to add the appropriate middleware to the api.  Follow these instructions:  Update your main script by adding the following lines (highlighted):  import casbin  from fastapi import FastAPI  **from fastapi\_authz import CasbinMiddleware**  **from starlette.middleware.authentication import AuthenticationMiddleware**  **import basic\_auth**  api = FastApi()  enforcer = casbin.Enforcer(  ‘rbac\_model.conf’,  ‘rbac\_policy.csv’,  )  **backend = basic\_auth.BasicAuth()**  **api.add\_middleware(CasbinMiddleware, enforcer=enforcer)**  **api.add\_middleware(AuthenticationMiddleware, backend=backend)**  @api.get(‘/’)  async def index():  return ‘Hello, world.’   1. Open the following link in your browser: [**http://127.0.0.1:8000/docs/**](http://127.0.0.1:8000/docs/). (It’s the same link as before.)   Add a screenshot of the result to this document and describe why you get the result that you see.   1. Update the policy file to provide **GET** access for a user **anonymous** on the endpoints **/docs** and **/openapi.json**, restart the **uvicorn** server manually, and try loading the page again.   Describe what happened and provide a screenshot.   1. Finally, describe what happens (and why) when you try to open the URL [**http://127.0.0.1:8000/**](http://127.0.0.1:8000/).   Place your answers in the section A05 provided. | |
| A05 | This area will expand automatically. Screenshot of [**http://127.0.0.1:8000/docs/**](http://127.0.0.1:8000/docs/) …  … Describe in your own words why you got the result shown in the screenshot above. [1 – 5 sentences] …  … Screenshot after the policy updates and **uvicorn** restart …  … Describe in your own words why you got the result shown in the post policy update and uvicorn restart screenshot above. [1 – 5 sentences] …  … In your own words, describe what happens when you visit [**http://127.0.0.1:8000/**](http://127.0.0.1:8000/)? [1 – 5 sentences] …  … | |
| 06 | Authenticate a user In Step 5, we have opened the API for abuse by anonymous users by allowing access to certain endpoints for non-authenticated users. You would not do this in a production system!  In this step, we will try to access the endpoint / for an authenticated user, Alice. You can either use the curl tool from the command line (Linux/MAC) or something like Postman (all OS) to add credentials to the request.  Follow these steps:  In a terminal, run the following command:  curl -i -u alice:password <http://127.0.0.1:8000/>  Describe what you see (and why) and provide a screenshot.  Add the following line to the policy file:  p, alice, /, GET  Restart the **uvicorn** server manually (it only watches for changes in the main script).  Run the curl command again. Describe what you see this time and provide a screenshot. |  |
| A06 | This area will expand automatically. Describe the result of the following command. [1-5 sentences] curl -i -u alice:password <http://127.0.0.1:8000/>  …  … Provide a Screenshot … place screenshot here  … Explain why do you see what is shown on screen? [1-5 sentences] …  … After updating the policy file, and restarting uvicorn server, what is the result of the same curl command? curl -i -u alice:password <http://127.0.0.1:8000/>  …  … Explain why you see this output on screen. [1-5 sentences] …  … | |
| 07 | Adding endpoints for Bob too Your policy file should now look like this (some lines have been omitted):  p, alice, /, GET  p, alice, /ds1/\*, GET  p, alice, /ds1/res1, POST  p, bob, /ds2/res2, GET  Now, follow these instructions:  Add two new endpoints (follow the existing code for the endpoint **/** as guidance) **ds1/res1** and **ds2/res2**. The endpoints can just return a string.  Make sure the server restarts without problems after making any changes.  Test the following URLs (**GET** using curl or Postman) and record the results in the table provided in the answer area. If the user is **not** **anonymous**, assume the password is **password**.  The final state of your project must be submitted to Blackboard. Don’t forget to add this document (as a separate file) to your submission. |  |
| A07 | This area will expand automatically. Testing results  |  |  |  | | --- | --- | --- | | **Endpoint** | **User** | **Result** | | ds1/res1 | anonymous |  | | ds2/res2 | anonymous |  | | ds1/res1 | alice |  | | ds2/res2 | alice |  | | ds1/res1 | bob |  | | ds2/res2 | bob |  | | |
|  | Submission of Portfolio Work To submit the portfolio, do the following:   * Save the document with your answers as a MS Word file (.docx). * Put all your code inside a zip-file. * Open Blackboard, and locate the AT2 Portfolio Task 5 assessment * Open the assessment and upload the word-processed document, any video files, and the zip-file as separate items.   Click Submit. |  |

# Appendix A: Supporting Details for Portfolio

This appendix contains details that are important to the assessment, but are placed here to allow students to refer to them as needed, rather than bulking out the front of this assessment document.

### Scenario

TO BE DEFINED

### General Instructions

Once you have downloaded this assessment document from Blackboard, please rename it in the form:

* XXX-InterRIoT-POR-Task-4.docx

Replacing the XXX with your initials.

For example, Adrian Gould would use AG-InterRIoT-POR-Task-4.docx for his submitted filename.

### Answering Questions

When a step includes a question, you must attempt to answer it.

Each answer has an indicated minimum and maximum number of sentences.

If a step has more than one question, these maxima and minima are a total for all the questions in that specific step.

All answers must be in complete sentences unless indicated.

If large amounts of code are required for an answer, make sure to add this code as a separate file to your submission. DO NOT put large amounts of code in this document. Code that is less than 15-20 lines in length is allowed in this document.

### Sources of Information

In industry, it is good practice to keep track of where information was obtained. This is especially true if it is a written document, or even code.

Please include an APA v6 or APA v7 bibliography entry for any resource used when answering a question, or when developing code.

Use a system such as https://MyBib.com to generate the reference.

For example, here are two references using <https://MyBib.com>:

“Sleep Modes - ESP32 - — ESP-IDF Programming Guide Latest Documentation.” *Docs.espressif.com*, docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/sleep\_modes.html.

*ESP32 Deep Sleep with Arduino IDE and Wake up Sources | Random Nerd Tutorials*. 1 Mar. 2019, randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources. Accessed 27 Mar. 2024.

### Code Storage

We advise that you create a GIT repository on GitHub and use this to store a copy of your work.

You may also use OneDrive within your college Office365 to store a backup of your code or keep a copy on a USB thumb drive.

# Appendix B: Code Style Guidelines

Your code will follow the PEP 8 standard.

Readability Counts  
- Zen of Python

Explicit is better than implicit.  
- Zen of Python

Other code standards available in the Presentation, “Python Coding Standards for North Metropolitan TAFE”.

# Appendix C: Useful Resources

“Casbin/Pycasbin.” *GitHub*, 27 Mar. 2024, github.com/casbin/pycasbin. Accessed 27 Mar. 2024.

*How to Implement Role-Based Access Control RBAC in Python: A Comprehensive Guide - Quickread*. 22 Oct. 2023, www.quickread.in/how-to-implement-rbac-in-python/. Accessed 27 Mar. 2024.

“Best Practices for Authorization in Python | Permit.” *Www.permit.io*, www.permit.io/blog/best-practices-for-authorization-in-python. Accessed 27 Mar. 2024.

“Flask/Python API: Role-Based Access Control Code Sample.” *Auth0 Developer Resources*, developer.auth0.com/resources/code-samples/api/flask/basic-role-based-access-control. Accessed 27 Mar. 2024.

“Best Practices for Authorization in Python | Permit.” *Www.permit.io*, www.permit.io/blog/best-practices-for-authorization-in-python. Accessed 27 Mar. 2024.