**TSTP for Unique Password/Tokens Detection (2.1.A.3 CSR)**

<DUT Details: > Ex: Router

<DUT Software Version:>

<Digest Hash of OS>

<Digest Hash of Configuration>

<Applicable ITSAR: >

<ITSAR Version No:>

<OEM Supplied Document list: >

1. **<ITSAR Section No & Name>**  **Section 1: Authentication**
2. **<Security Requirement No & Name > 2.1.A.3 Unique Password Detection**
3. **<Requirement Description: >**

Where passwords are used and, in any state, all consumer IoT device passwords shall be. unique per device or defined by the user. If password-less authentication

is used, the same principles of uniqueness apply.

[Ref: ETSI EN 303 645 V2.1.0 (2020-04) Provision 5.1-1, IoT SF IoT Security assurance framework Release 3.0 November 2021 2.4.8.3]

1. **DUT Confirmation Details**:

Smart energy meter model no MB074TI

1. **DUT Configuration:**

Smart energy meter model no MB074TI

1. **Preconditions**
2. One DUT (Smart energy meter)
3. The tester has privileges to create an interactive session between the end user and the supplier both locally and remotely.
4. Vendor should specify the mechanism for hardware tempering.
5. **Test Objective**

To ensure the detection of Non-Unique or Weak Password / token-based Authentication on Govt. aided sites to reduce cyber-attacks and hacking using Python Script Code.

1. **Test Plan**
   1. Set Up Environment: We’ll be starting with installing the required libraries.

\*Ensure you have Python installed (preferably version 3.6+).

\*Install any necessary libraries, such as “regional expression” for testing

* 1. Tools Required: Python Compiler, Code execution experts, Computer Software, Python version preferably above 3.6.
  2. Detailed Steps for Running Tests:
* **Define Password Criteria**: Confirm the password policy requirements from the government guidelines.
* **Set Up Testing Environment**: Ensure Python and necessary libraries are installed Create a Python script file (e.g., test\_password\_strength.py)
* **Write Test Cases**: Identify test scenarios for all password rules.
* **Automated Testing**: Example test cases include valid, invalid, and edge case passwords.
* **Execute Tests**: Run the script in the terminal: python test\_password\_strength.py.

Check for any failed tests.

* **Review and Debug**: Analyze failures and fix issues in the password validation function.
* **Repeat Testing**: Rerun tests after fixing bugs until all pass.

***This process ensures that the password detection code is thoroughly tested and meets the necessary security standards.***

1. **Expected Results for Pass:**

There are certain passwords which can be used against as a test cases to check

the performance of code.

1. **Test Execution:**

**Test Case Number:** 01🡪 “#GovtofInd123”

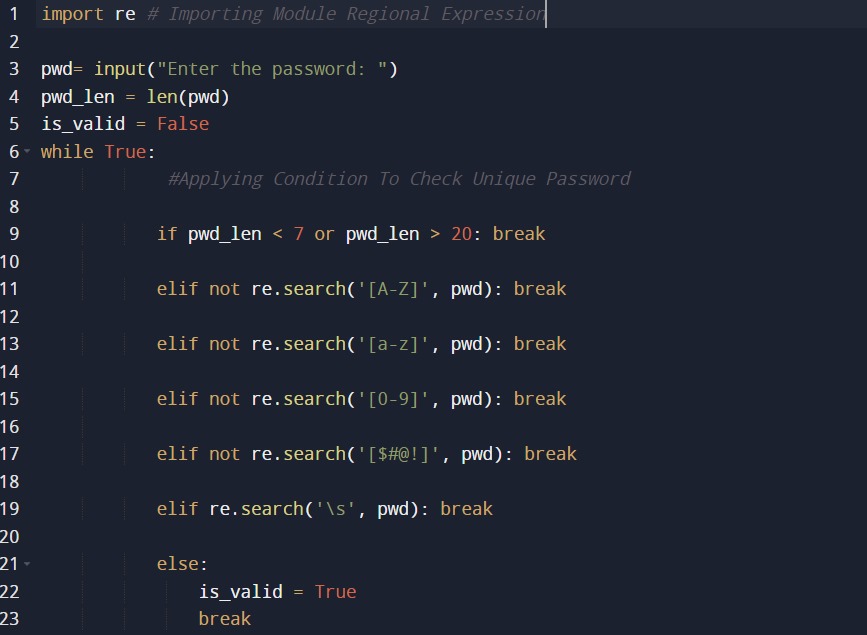


Figure 3.1: Python Script for Unique Password Detection.

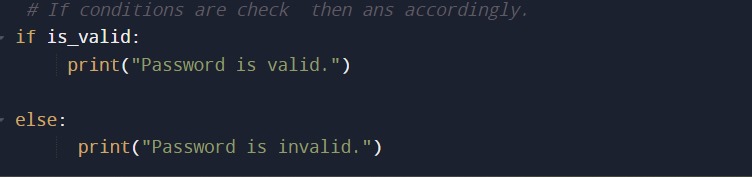


Figure 3.2: Python Script for Unique Password Detection.

1. **Test Case Result:**

**Case 1-🡪 \*#GovtofInd123\***

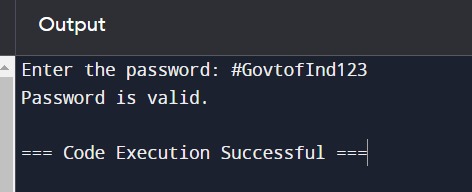


Figure 4.1 --- Output of TestCase-1 (Pass)

**Case 2 🡪 \*GovtofInd123\***

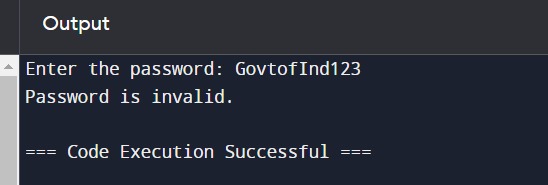
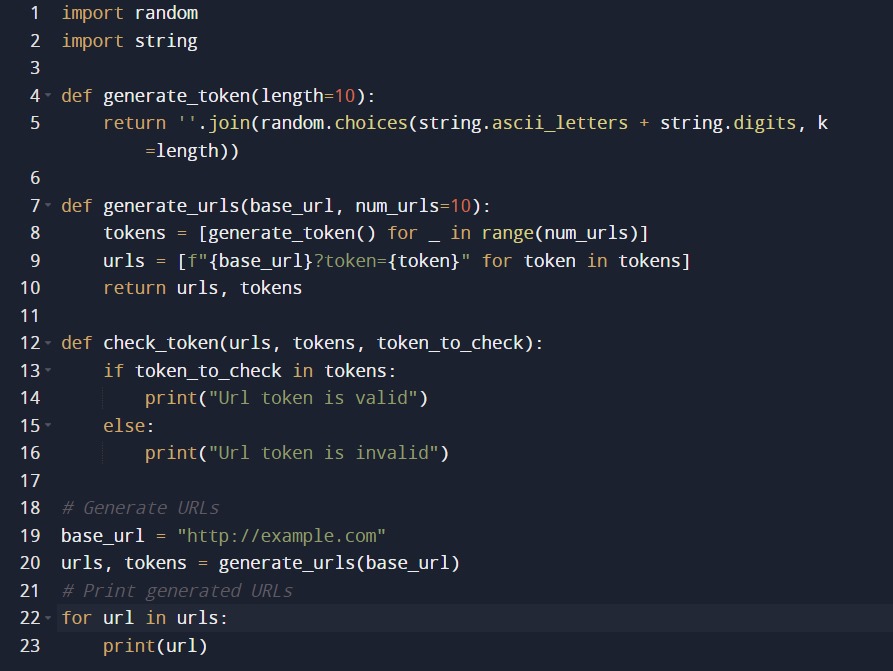


Figure 4.2 -🡪 Output of TestCase-2(Fail)

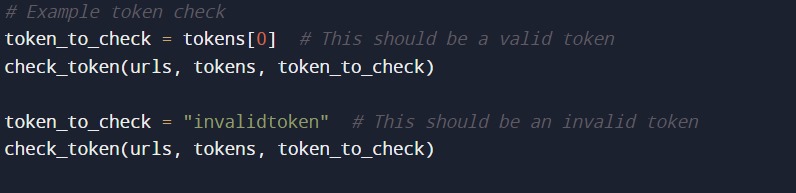
**Conclusion:**

1. *Test Case1* ***Passes*** *Correctly As all conditions of Unique Password are satisfied.*
2. *Test Case 2* ***Fails*** *due absence of using Special symbols like “@, #, $, %” etc. which is one of the prominent conditions of Unique Password Checking*.

* To Encounter the case of Passwordless authentication Code Flow is as follows 🡪

**

***Figure 1.1🡪Code to generate Test Urls with a unique token***



***Figure 1.2 🡪 Test Cases to detect a valid token***

* ***OUTPUT OF CODE(Fig 1.1 & Fig 1.2)***🡪



* ***Conclusion:***
* Test Case 1 (Token from generated Urls)🡪 **PASS.**
* Test Case 2 (Random Token from outsource)🡪**FAIL**.