**QualityControllers – SQA2023 – AUBURN**

**Team 15**: Christopher Casey, Nihitha Reddy Seelam, Pooja Chilukuru​, Yinbo Chen

**Contents:**

1. Security Weakness
2. Fuzzing
3. Forensics
4. Conclusion

1)Security Weakness:

------Body-----

Screenshot

2)Fuzzing:

**Target Functions:**

For performing the Fuzzy tests. We chose five functions from the component ‘scanner.py’. They are ‘isValidUserName()’, ‘isValidPasswordName()’, ‘isValidKey()’, ‘ checkIfValidSecret()’, ‘checkIfValidKeyValue()’. Each function has been confined and only specified input can go through the execution. All functions’ input values are strings. The input and output values are shown below:

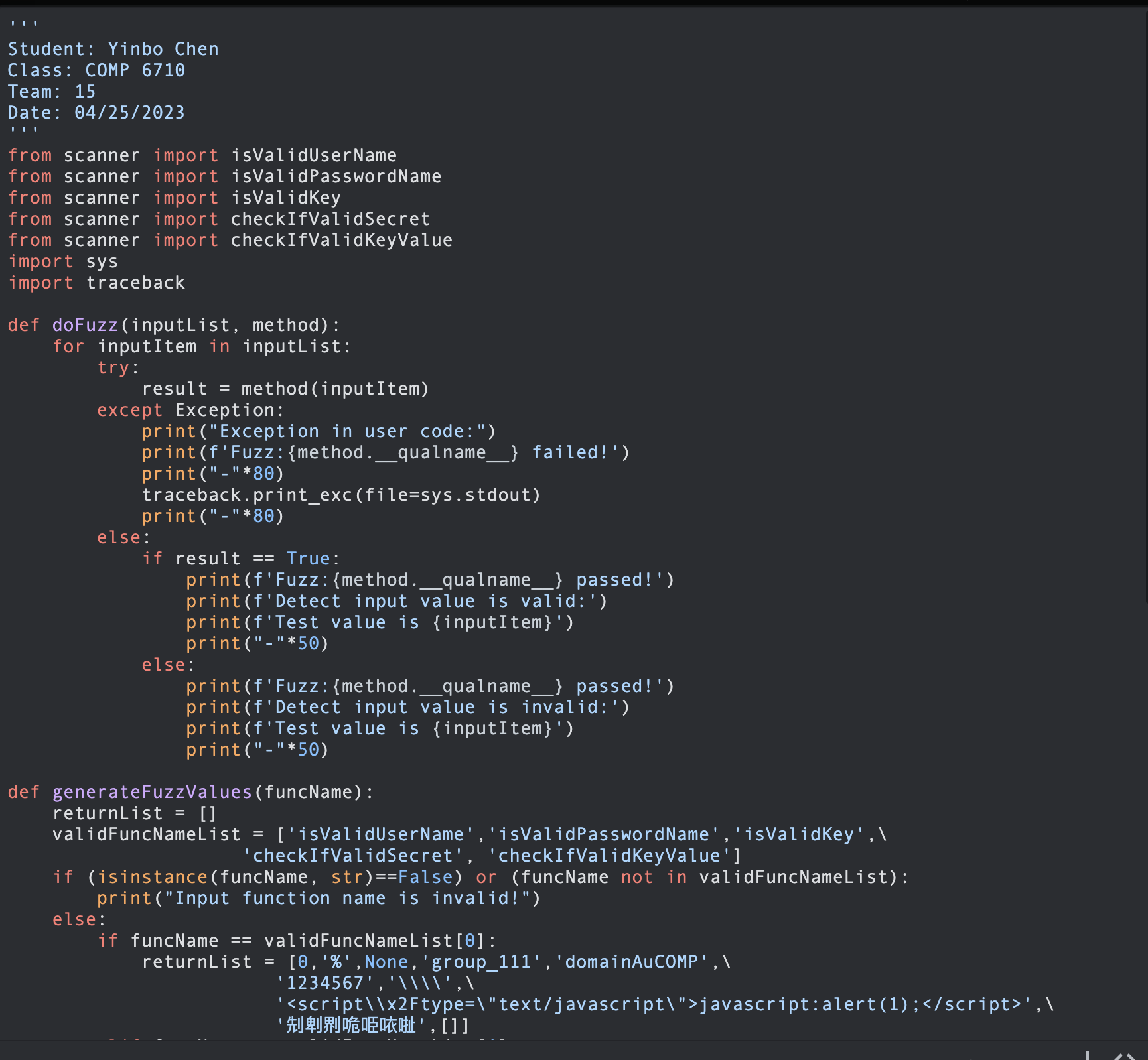
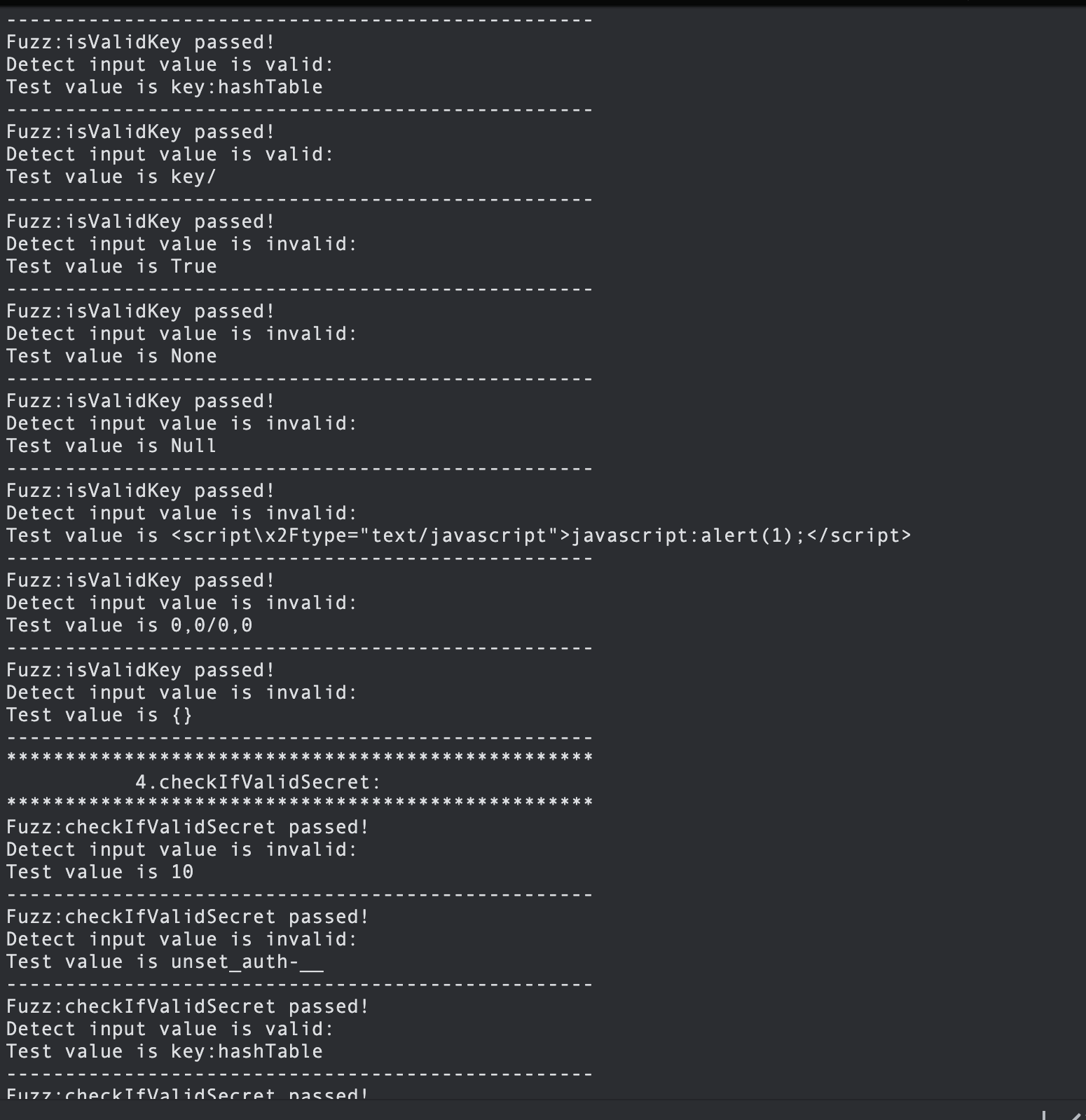
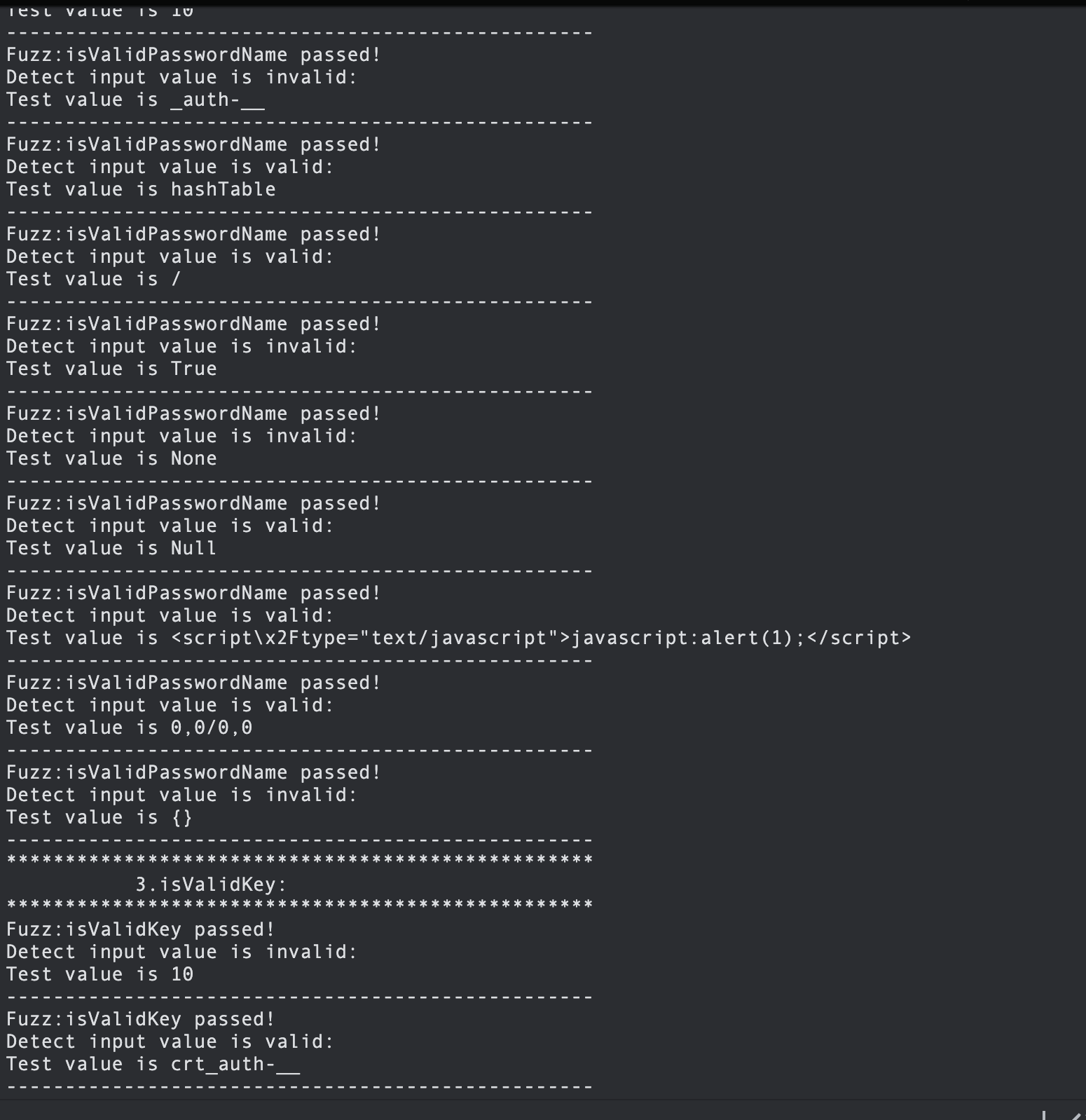
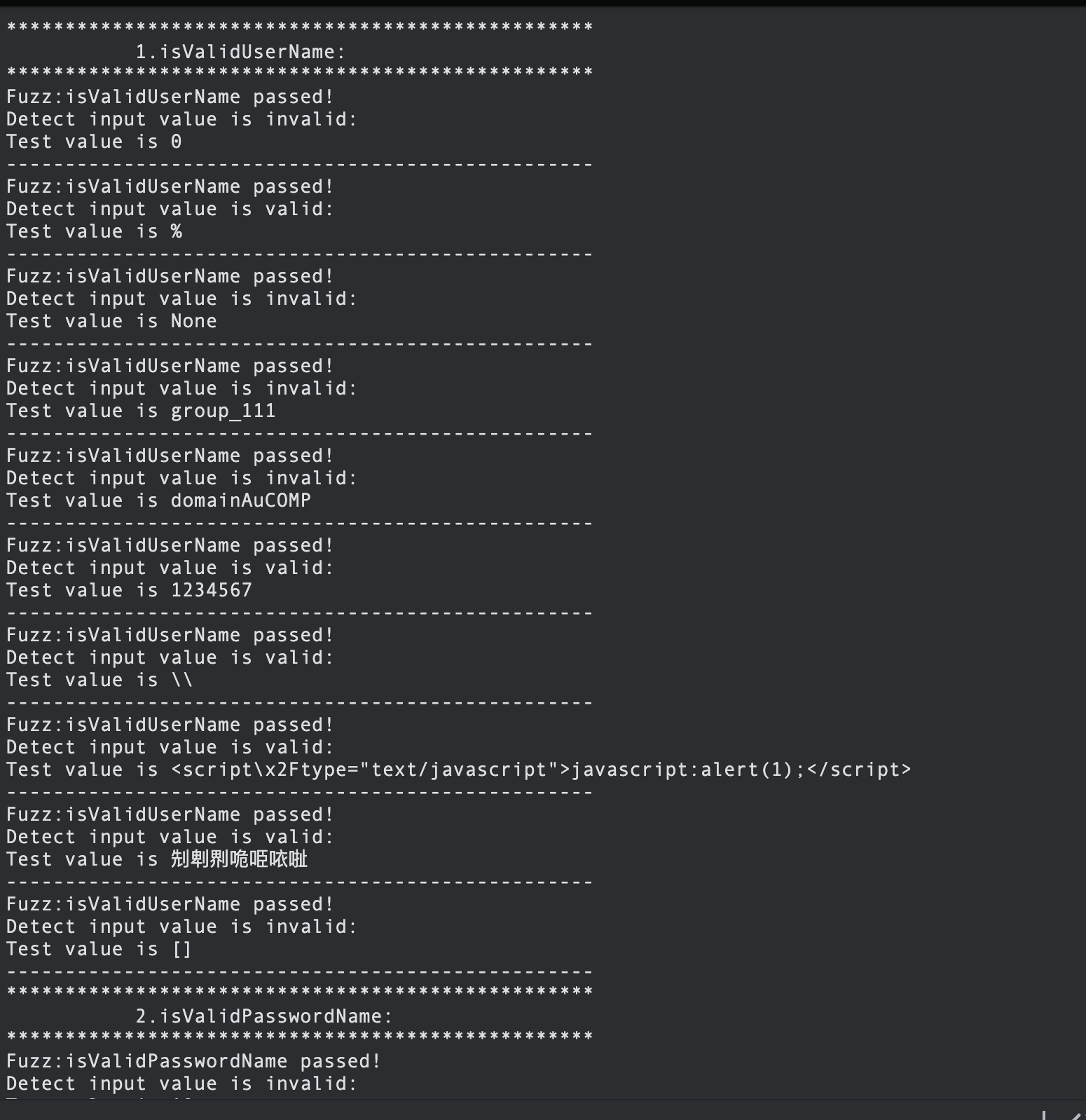
1. The ‘isValidUserName()’ takes a string as input, then checks the input value with constant values to see if it is getting any matching (constant values be treated as a substring of input string). After checking the input data format, if it’s a correct string type then continually execute to match strings. The return value is either ‘True’ or ‘False’.
2. The ‘IsValidpasswordName()’ function’s execution process is like the first one. The only difference is that using different constant comparison lists.
3. The ‘IsValidKey()’ is the same as the previous two functions.
4. The ‘CheckIfValidSecret()’ function’s input is a string too. It transformed input letters to lowercase and removed spaces if there were any. Then do matching and return ‘True’ or ‘False'.
5. The ‘CheckIfValidKeyValue()’ function’s logic is similar to the first one above.

**Fuzz Testing Input:**

The input data structure is a list. The possible input value types are ‘int’, ‘float’, ‘None’, ‘strings (without constant substring)’, ‘string (with constant substring)’, and symbols.

**Result:**

All functions returned correct results. We didn’t find any bugs or defects. The screenshots are shown below.



Text

Description automatically generated

Graphical user interface, text

Description automatically generated

3)Forensics:

Forensics is a way to integrate logs into the software. Which is useful when unwanted event occurs it is easy to detect and trouble shoot.

Mainly issues like access issues, change issues, exceptions and so on are detected.

Software forensics should be implemented from the initial stage of developing an application/Software feature.

As part of our project, logging\_forensics.py has the giveMeLoggingObject()function that provides a way creation of logging object using logging module. Function sets up a basic configuration for the logging object, including the format of log message and the file to which the logs will be written.

All the logs got stored in the logger.log.

logging is done in more than 5 methods in each s/w. **Logs are implemented** in parser.py, main.py, scanner.py and graphtaint.py.

Following are the snapshots of the implementation:

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated