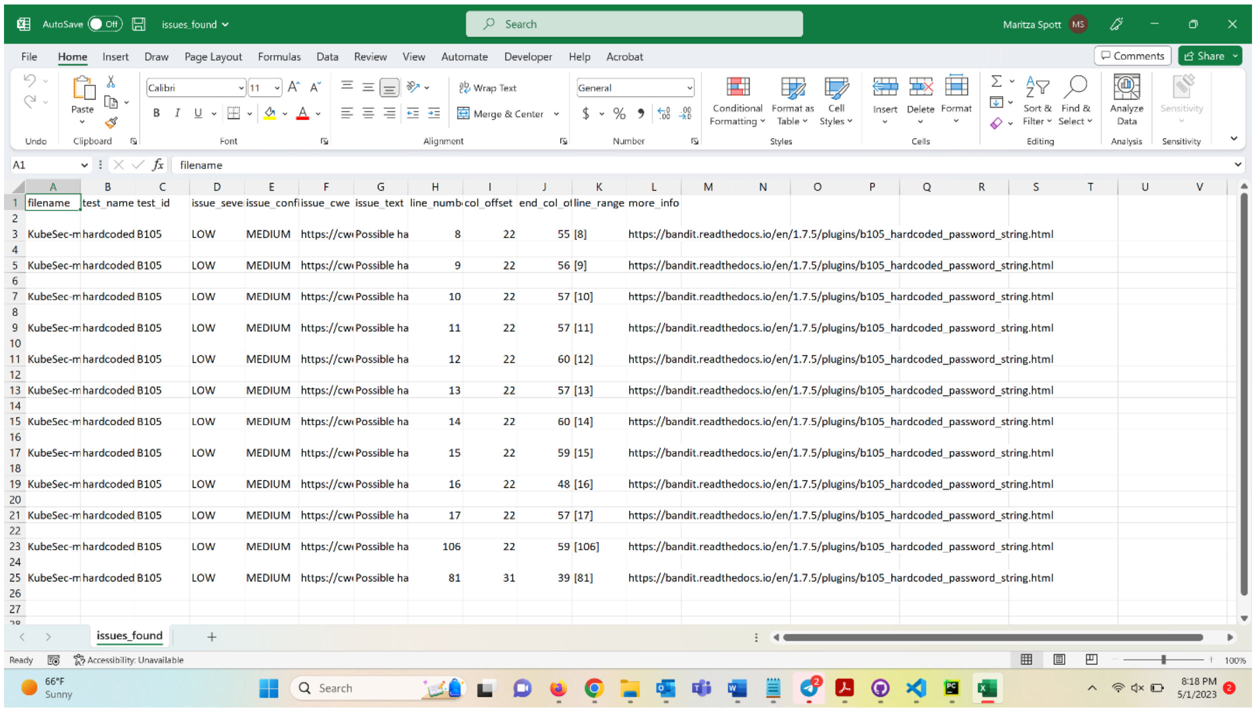
Software Quality Assurance Final Project
Maritza Spott

4A. Git Hook

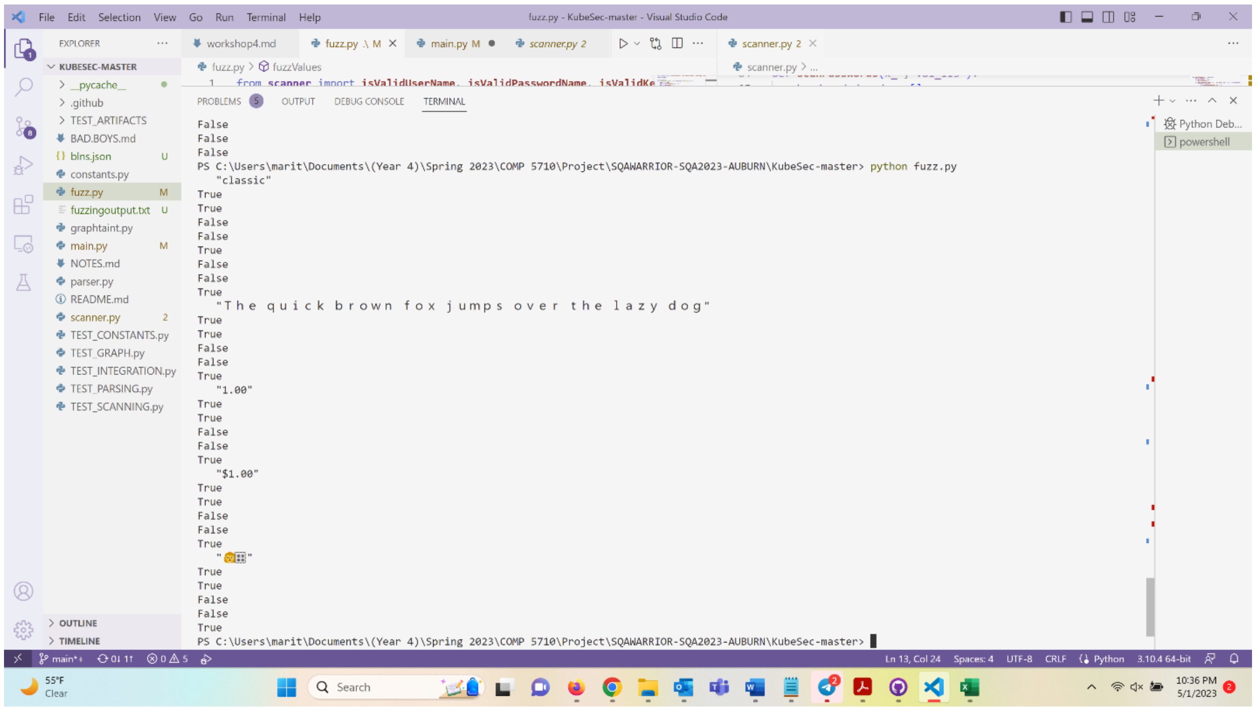
The CSV file created is labeled ‘issues_found.csv’. It is currently in .gitignore with pre-commit, but a copy of the file from one upload is in the main folder (SQAWARRIOR-SQA2023-AUBURN). A screenshot is also below.



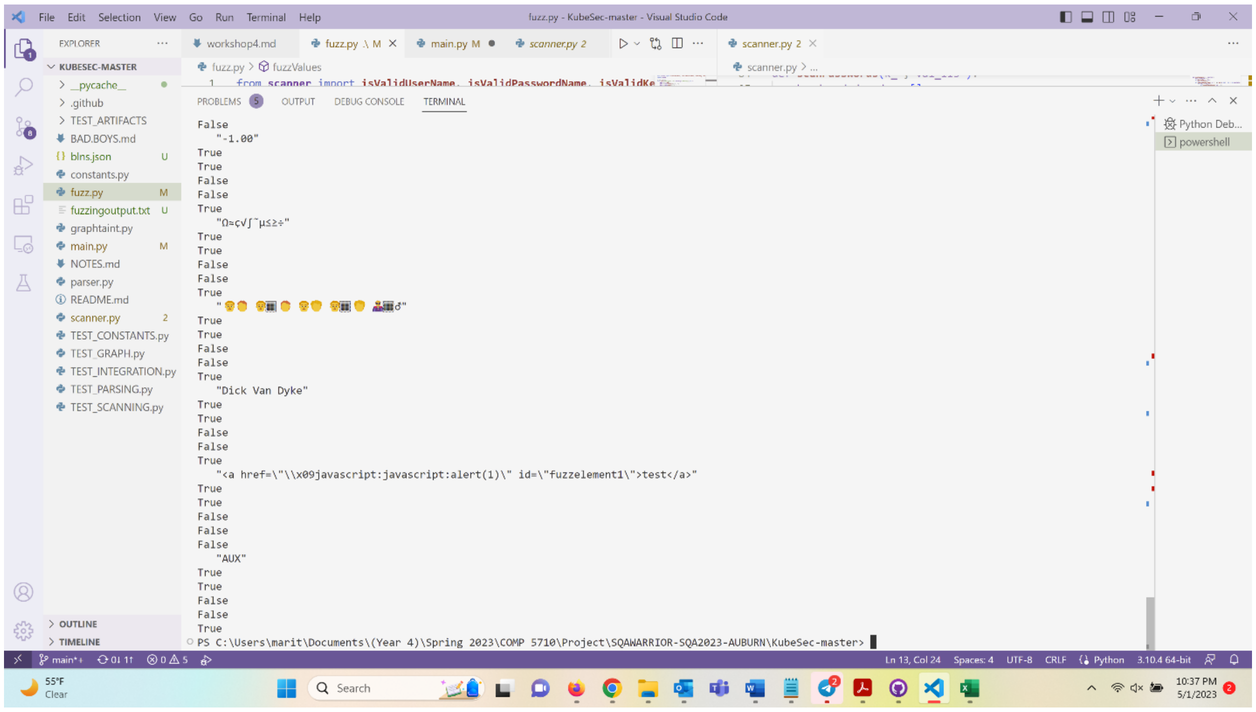
	A	B	C	D	E	F	G	H	I	J	K	L
1	filename	test_name	test_id	issue_seve	issue_confi	issue_cwe	issue_text	line_numb	col_offset	end_col_o	line_range	more_info
2												
3	KubeSec-m	hardcoded	B105	LOW	MEDIUM	https://cw	Possible ha	8	22	55	[8]	https://bandit.readthedocs.io/en/1.7.5/plugins/b105_hardcoded_password_string.html
4												
5	KubeSec-m	hardcoded	B105	LOW	MEDIUM	https://cw	Possible ha	9	22	56	[9]	https://bandit.readthedocs.io/en/1.7.5/plugins/b105_hardcoded_password_string.html
6												
7	KubeSec-m	hardcoded	B105	LOW	MEDIUM	https://cw	Possible ha	10	22	57	[10]	https://bandit.readthedocs.io/en/1.7.5/plugins/b105_hardcoded_password_string.html
8												
9	KubeSec-m	hardcoded	B105	LOW	MEDIUM	https://cw	Possible ha	11	22	57	[11]	https://bandit.readthedocs.io/en/1.7.5/plugins/b105_hardcoded_password_string.html
10												
11	KubeSec-m	hardcoded	B105	LOW	MEDIUM	https://cw	Possible ha	12	22	60	[12]	https://bandit.readthedocs.io/en/1.7.5/plugins/b105_hardcoded_password_string.html
12												
13	KubeSec-m	hardcoded	B105	LOW	MEDIUM	https://cw	Possible ha	13	22	57	[13]	https://bandit.readthedocs.io/en/1.7.5/plugins/b105_hardcoded_password_string.html
14												
15	KubeSec-m	hardcoded	B105	LOW	MEDIUM	https://cw	Possible ha	14	22	60	[14]	https://bandit.readthedocs.io/en/1.7.5/plugins/b105_hardcoded_password_string.html
16												
17	KubeSec-m	hardcoded	B105	LOW	MEDIUM	https://cw	Possible ha	15	22	59	[15]	https://bandit.readthedocs.io/en/1.7.5/plugins/b105_hardcoded_password_string.html
18												
19	KubeSec-m	hardcoded	B105	LOW	MEDIUM	https://cw	Possible ha	16	22	48	[16]	https://bandit.readthedocs.io/en/1.7.5/plugins/b105_hardcoded_password_string.html
20												
21	KubeSec-m	hardcoded	B105	LOW	MEDIUM	https://cw	Possible ha	17	22	57	[17]	https://bandit.readthedocs.io/en/1.7.5/plugins/b105_hardcoded_password_string.html
22												
23	KubeSec-m	hardcoded	B105	LOW	MEDIUM	https://cw	Possible ha	106	22	59	[106]	https://bandit.readthedocs.io/en/1.7.5/plugins/b105_hardcoded_password_string.html
24												
25	KubeSec-m	hardcoded	B105	LOW	MEDIUM	https://cw	Possible ha	81	31	39	[81]	https://bandit.readthedocs.io/en/1.7.5/plugins/b105_hardcoded_password_string.html
26												
27												

4B. Fuzzing

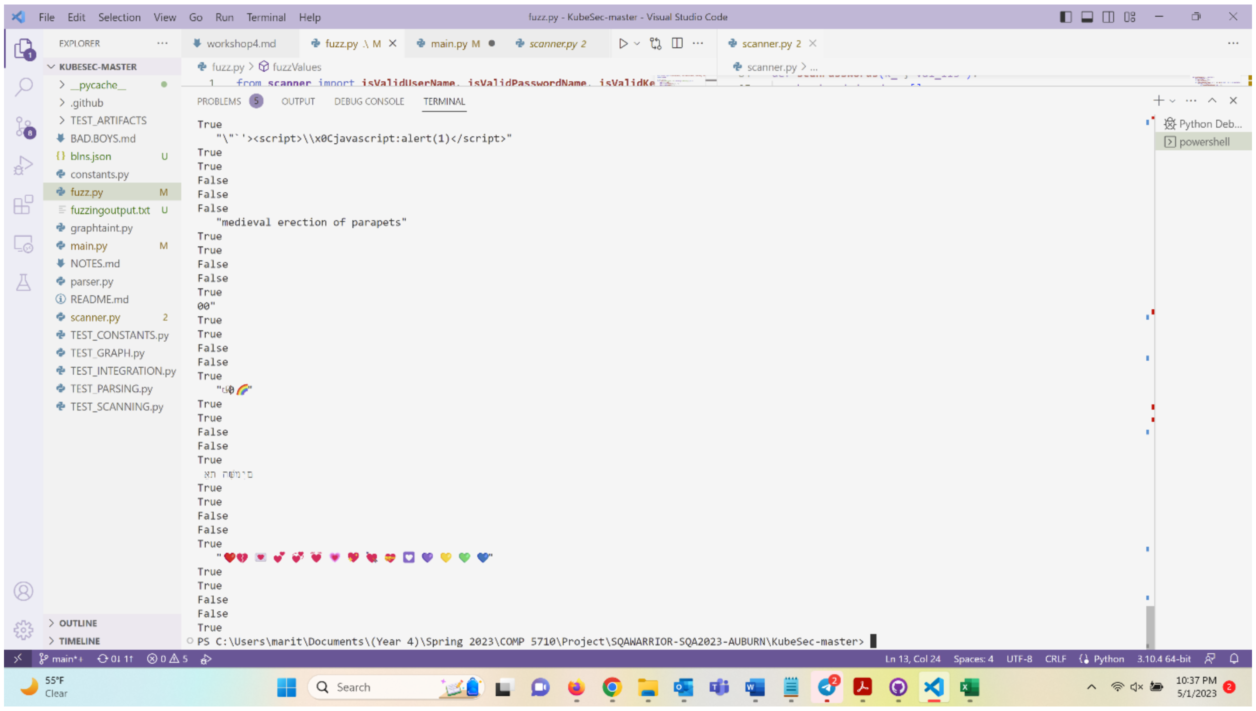
I created a fuzzing method that selects seven random phrases from blns.json file to run through five functions in scanner.py. The five functions are `isValidUserName`, `isValidPasswordName`, `isValidKey`, `checkIfValidKeyValue`, and `checkIfValidSecret`. Screenshots of output are below where some of the methods allow emojis to be valid.



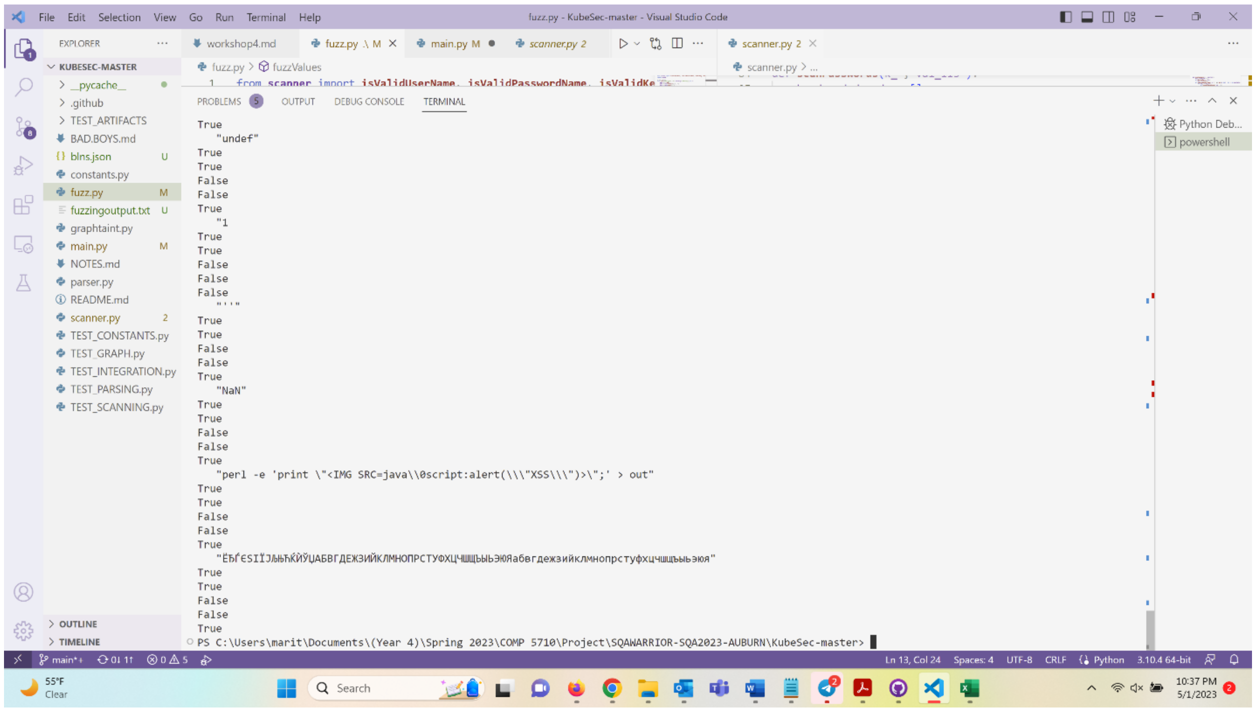
```
False
False
False
PS C:\Users\marit\Documents\(Year 4)\Spring 2023\COMP 5710\Project\SQAWARRIOR-SQA2023-AUBURN\KubeSec-master> python fuzz.py
   "classic"
True
True
False
False
True
False
False
True
   "T h e  q u i c k  b r o w n  f o x  j u m p s  o v e r  t h e  l a z y  d o g"
True
True
False
False
True
   "1.00"
True
True
False
False
True
   "$1.00"
True
True
False
False
True
   "🧑▦"
True
True
False
False
True
PS C:\Users\marit\Documents\(Year 4)\Spring 2023\COMP 5710\Project\SQAWARRIOR-SQA2023-AUBURN\KubeSec-master>
```



```
False
   "-1.00"
True
True
False
False
True
   "Ω≈ç√∫˜µ≤≥÷"
True
True
False
False
True
   "👨👊 👨▦👊 👨👊 👨▦👊 🧑▦♂"
True
True
False
False
True
   "Dick Van Dyke"
True
True
False
False
True
   "<a href=\"\\x09javascript:javascript:alert(1)\" id=\"fuzzelement1\">test</a>"
True
True
False
False
False
   "AUX"
True
True
False
False
True
PS C:\Users\marit\Documents\(Year 4)\Spring 2023\COMP 5710\Project\SQAWARRIOR-SQA2023-AUBURN\KubeSec-master>
```



```
True
   "\"`'><script>\\x0Cjavascript:alert(1)</script>"
True
True
False
False
False
   "medieval erection of parapets"
True
True
False
False
True
00"
True
True
False
False
True
   "ꝍ0🌈"
True
True
False
False
True
 אֵת הַשָּׁמַיִם
True
True
False
False
True
   "❤️💔💌💕💞💓💗💖💘💝💟💜💛💚💙"
True
True
False
False
True
PS C:\Users\marit\Documents\(Year 4)\Spring 2023\COMP 5710\Project\SQAWARRIOR-SQA2023-AUBURN\KubeSec-master>
```



```
True
   "undef"
True
True
False
False
True
   "1
True
True
False
False
False
   "'''"
True
True
False
False
True
   "NaN"
True
True
False
False
True
   "perl -e 'print \"<IMG SRC=java\\0script:alert(\\\"XSS\\\")>\";' > out"
True
True
False
False
True
   "ЁЂЃЄЅІЇЈЉЊЋЌЍЎЏАБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯабвгдежзийклмнопрстуфхцчшщъыьэюя"
True
True
False
False
True
PS C:\Users\marit\Documents\(Year 4)\Spring 2023\COMP 5710\Project\SQAWARRIOR-SQA2023-AUBURN\KubeSec-master>
```

4B. Forensics: Logging

For 4B, I used logging to identify vulnerabilities from poisoning attacks and model tricking.

The five functions scanned were `getYAMLfiles`, `scanForHTTP`, `scanForMissingSecurityContext`, and `getItemFromSecret` in scanner.py and `getYAMLfiles` in graphtaint.py.

LESSONS LEARNED

1. Bandit is very picky about white space and does not like blsn.json at all because it is not formatted according to its standards.
2. I had to push files independently to make sure they were up to Bandit's standards.
3. Some of the checking functions in scanner.py worked better than others, as they wouldn't allow many fraudulent inputs go through as valid.