

The background of the cover features a low-angle, black and white photograph of a modern building's structural elements, including beams and columns, creating a sense of depth and architectural complexity. A solid yellow horizontal bar is positioned at the very top of the page.

PENETRATION TESTING REPORT

STATIC CODE ANALYSIS

PREPARED BY
GOPI GOR

CONTENTS

Executive Summary

Page 3

Scope & Objectives

Page 4

Methodology

Page 5

Findings

Page 6

Recommendations

Page 8

Appendix

Page 9

References

Page 11

EXECUTIVE SUMMARY

This penetration testing report identifies and presents different kinds of findings and recommendations for static code analysis of 2 applications in Python. The test aimed to identify the vulnerabilities in the source code through Bandit and provide recommendations to make the application more secure.

There were multiple vulnerabilities identified in both applications, mainly due to the open ports that allow these vulnerabilities to be exploited. Each vulnerability has been assigned a severity level which can lead to data breaches, security challenges, and any kind of unauthorized access.

The identified weaknesses in the source code of the application could lead to major financial losses, regulatory issues and loss of customer trust. Apart from technical recommendations provided in detail below, it would be advised to ensure secure coding practices and employee training for better development of applications.

SCOPE & OBJECTIVES

The scope of this penetration testing report is to conduct a static code analysis on two applications, for which the source code and repositories have been provided. The overall scope is to complete tests on the applications and provide recommendations to improve the application.

The objectives for the test are listed below:

- Run a port scan script on a server to check open ports
- Install Bandit and clone repositories to conduct the test
- Do a complete static code analysis on 2 applications
- Identify vulnerabilities from results obtained
- Showcase findings for the report
- Provide recommendations for the remediation

METHODOLOGY

The methodology was used to identify potential vulnerabilities in the applications was done using Kali Linux, Metasploitable 2, Python, and Bandit analysis tools. The overall approach used to conduct the test was cloning, running repos and then analyzing each vulnerability for recommendations. The stepwise methodology has been explained below:

- First, the bandit repository was cloned and then the tool was installed in Kali.
- Next, we ran the portscan on Metasploit to understand the automation process in the Python scripts.
- Application 1 was cloned via the GitHub repository by using the command **git clone <repo-link>** and the command **bandit -r lets-be-bad-guys > app1.txt** was run. The **>** saved the output into a text file app1.txt for further analysis.
- Application 2 was similarly cloned and the and **bandit -r vulpy > app2.txt** was run.

The automated script for bandit -r as a recursive script was a use case that allowed a quick return to the bandit tool for further analysis.

FINDINGS

1. Application 1

When the application **lets-be-bad-guys** was cloned, we first found out that it had **780 objects** and **434 deltas**. After this, we ran the bandit tool to evaluate all the vulnerabilities. The total count was **6 issues**. These issues also show additional information like issue details, CVEs, and severity. Furthermore, at the end of the scan, the run metrics are sorted in severity and confidence levels. A few of the vulnerabilities discovered are listed below.

- B105 - possible hard-coded password
- B110 - try, except, pass detected
- B102 - use of exec detected

These vulnerabilities have repeated in certain conditions and the proof can be found in the appendix.

The severity and confidence levels depend on each of the cases, and while the confidence level may be low, the severity being high could be a sign to be more careful during development processes.

FINDINGS

2. Application 2

When the application **vulpy** was cloned, we first found out that it had **437 objects** and **221 deltas**. After this, we ran the bandit tool to evaluate all the vulnerabilities. The total count was **49 issues**. These issues also show additional information like issue details, CVEs, and severity. Furthermore, at the end of the scan, the run metrics are sorted in severity and confidence levels. A few of the vulnerabilities discovered are listed below.

- B113 - requests call without timeout
- B108 - insecure usage of temp file/directory
- B404 - security implications, subprocess module
- B603 - subprocess, untrusted input
- B608 - hardcoded sql expression, can be SQLi
- B110 - try, except, pass detected
- B105 - possible hard-coded password
- B201 - flask debugger allows arbitrary code to run

It was also noticed that the occurrence of B108 was very frequently discovered in the results. Severity was: **4 (High), 9 (Low), 36 (Medium)**.

RECOMMENDATIONS

As we have seen in the findings, there are vulnerabilities that can be harmful for the business if the applications are launched at this stage. Here are some recommendations that can be done to avoid that:

- The password storing mechanisms can be used securely and using of environment variables to avoid password issues.
- For the try, except, pass detections, the exceptions can be applied clearly in the code so that it is handled appropriately in each case.
- The exec functions allows arbitrary code to be run, so it better to avoid that and use functions instead and restrict file permissions.
- For the requests call timeouts, there can be a specific timeout defined based on the application.
- For the hardcoded tmp error, secure modules can be used for temporary files such as tempfile module.
- The subprocess module takes inputs, but these should be verified that they are taken from a secure and trusted source. They can also be cleaned as a first step.
- The query can be developed as per developer intent to avoid SQLi. This means it should not be in string format.
- Finally, the flash debugger can be disabled to avoid any arbitrary code running.

APPENDIX

```
(user1@kali)~$ git clone https://github.com/PyCQA/bandit.git
Cloning into 'bandit'...
remote: Enumerating objects: 9104, done.
remote: Counting objects: 100% (1327/1327), done.
remote: Compressing objects: 100% (271/271), done.
remote: Total 9104 (delta 1158), reused 1125 (delta 1055), pack-reused 7777
Receiving objects: 100% (9104/9104), 3.45 MiB | 5.96 MiB/s, done.
Resolving deltas: 100% (6340/6340), done.

(user1@kali)~$ ls
Desktop  Downloads  Pictures  Public  Videos  bandit
Documents Music  PortScan.py  Templates  ab.py  name.py

(user1@kali)~$ cd bandit
$ git clone https://github.com/PyCQA/bandit.git
```

Fig: Bandit repo clone

```
(user1@kali)~$ sudo apt-get update
[sudo] password for user1:
Get:1 http://kali.download/kali kali-rolling InRelease [41.5 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [19
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (del
MB]
Get:4 http://kali.download/kali kali-rolling/non-free amd64 Packages
Get:5 http://kali.download/kali kali-rolling/non-free amd64 Contents
85 kB]
Get:6 http://kali.download/kali kali-rolling/non-free-firmware amd64
[33.0 kB]
Fetched 68.2 MB in 31s (2216 kB/s)
Reading package lists... Done
```

Fig: Bandit directory update

```
(user1@kali)~$ sudo apt install python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
python3-pip-whl
The following packages will be upgraded:
python3-pip python3-pip-whl
2 upgraded, 0 newly installed, 0 to remove and 1408 not upgraded.
Need to get 3117 kB of archives.
After this operation, 3072 B disk space will be freed.
Do you want to continue? [Y/n] y
Get:1 http://kali.org/kali kali-rolling/main amd64 python3-pip all 24.0+
dfsg-1 [1242 kB]
Get:2 http://kali.org/kali kali-rolling/main amd64 python3-pip-whl all 2
4.0+dfsg-1 [1774 kB]
Fetched 3117 kB in 2s (2073 kB/s)
(Reading database ... 399645 files and directories currently installed.)
Preparing to unpack .../python3-pip_24.0+dfsg-1_all.deb ...
Unpacking python3-pip (24.0+dfsg-1) over (23.3+dfsg-1) ...
Preparing to unpack .../python3-pip-whl_24.0+dfsg-1_all.deb ...
Unpacking python3-pip-whl (24.0+dfsg-1) over (23.3+dfsg-1) ...
Setting up python3-pip-whl (24.0+dfsg-1) ...
Setting up python3-pip (24.0+dfsg-1) ...
Processing triggers for man-db (2.12.0-1) ...
Processing triggers for kali-menu (2023.4.6) ...
```

Fig: Bandit directory python install

```
(user1@kali)~$ pip install pbr
Defaulting to user installation because normal site-packages is not writeable
Collecting pbr
  Downloading pbr-6.0.0-py2.py3-none-any.whl.metadata (1.3 kB)
  Downloading pbr-6.0.0-py2.py3-none-any.whl (107 kB)
    107.5/107.5 kB 646.3 kB/s eta 0:00:00
Installing collected packages: pbr
WARNING: The script pbr is installed in '/home/user1/.local/bin' which is not on PATH.
Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-w
rn-script-location.
Successfully installed pbr-6.0.0
```

Fig: Bandit pbr to install for missing dependencies

```
(user1@kali)~$ sudo pip3 install bandit
Collecting bandit
  Downloading bandit-1.7.7-py3-none-any.whl.metadata (5.9 kB)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/lib/python3/dist-packages (from bandit) (6.0.1)
Collecting stevedore>=1.20.0 (from bandit)
  Downloading stevedore-5.2.0-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: rich in /usr/lib/python3/dist-packages (from bandit) (13.3.1)
Collecting pbr<=2.1.0, >=2.0.0 (from stevedore>=1.20.0->bandit)
  Downloading pbr-6.0.0-py2.py3-none-any.whl.metadata (1.3 kB)
Collecting markdown-it-py<3.0.0, >=2.1.0 (from rich->bandit)
  Downloading markdown_it_py-2.2.0-py3-none-any.whl.metadata (6.8 kB)
Requirement already satisfied: pygments<3.0.0, >=2.14.0 in /usr/lib/python3/dist-packages (from rich->bandit) (2.15.1)
Requirement already satisfied: mdurl<=0.1 in /usr/lib/python3/dist-packages (from markdown-it-py<3.0.0, >=2.1.0->rich->bandit) (0.1.2)
Downloading bandit-1.7.7-py3-none-any.whl (124 kB)
   124.2/124.2 kB 2.5 MB/s eta 0:00:00
Downloading stevedore-5.2.0-py3-none-any.whl (49 kB)
   49.7/49.7 kB 1.3 MB/s eta 0:00:00
Downloading markdown_it_py-2.2.0-py3-none-any.whl (84 kB)
   84.5/84.5 kB 2.0 MB/s eta 0:00:00
Downloading pbr-6.0.0-py2.py3-none-any.whl (107 kB)
   107.5/107.5 kB 2.0 MB/s eta 0:00:00
Installing collected packages: pbr, markdown-it-py, stevedore, bandit
Attempting uninstall: markdown-it-py
```

Fig: Installing bandit tool

```
Enter a remote host to scan: 10.0.2.5

Please wait, scanning remote host 10.0.2.5

Port 21:      Open
Port 22:      Open
Port 23:      Open
Port 25:      Open
Port 53:      Open
Port 80:      Open
Port 111:     Open
Port 139:     Open
Port 445:     Open
Port 512:     Open
Port 513:     Open
Port 514:     Open
Scanning Completed in: 0:00:11.037485
```

Fig: PortScan.py to check open ports

APPENDIX

```

(user1@kali)-[~/bandit]
$ git clone https://github.com/mpirnat/lets-be-bad-guys
Cloning into 'lets-be-bad-guys'...
remote: Enumerating objects: 780, done.
remote: Total 780 (delta 0), reused 0 (delta 0), pack-reused 780
Receiving objects: 100% (780/780), 638.54 KiB | 4.23 MiB/s, done.
Resolving deltas: 100% (434/434), done.

```

Fig: Application 1 clone

```

(user1@kali)-[~/bandit]
$ git clone https://github.com/fportantier/vulpy
Cloning into 'vulpy'...
remote: Enumerating objects: 437, done.
remote: Counting objects: 100% (76/76), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 437 (delta 54), reused 54 (delta 54), pack-reused 361
Receiving objects: 100% (437/437), 2.90 MiB | 4.71 MiB/s, done.
Resolving deltas: 100% (221/221), done.

```

Fig: Application 2 clone

```

(user1@kali)-[~/bandit]
$ bandit -r lets-be-bad-guys
[main] INFO profile include tests: None
[main] INFO profile exclude tests: None
[main] INFO cli include tests: None
[main] INFO cli exclude tests: None
[main] INFO running on Python 3.11.6
Run started:2024-02-23 01:35:36.007564

Test results:
>>> Issue: [B105:hardcoded_password_string] Possible hardcoded password: 'h++jszpe1l@kay_b-cpM'
Location: lets-be-bad-guys/badguys/settings.py:90:13
Severity: Low Confidence: Medium
CWE: CWE-259 (https://cwe.mitre.org/data/definitions/259.html)
More Info: https://bandit.readthedocs.io/en/1.7.7/plugins/b105_hardcoded_password_string.html

90 # Make this unique, and don't share it with anybody.
SECRET_KEY = 'h++jszpe1l@kay_b-cpM'

91

>>> Issue: [B110:try_except_pass] Try, Except, Pass detected.
Severity: Low Confidence: High
CWE: CWE-703 (https://cwe.mitre.org/data/definitions/703.html)
More Info: https://bandit.readthedocs.io/en/1.7.7/plugins/b110_try_except_pass.html
Location: lets-be-bad-guys/badguys/vulnerable/views.py:65:8
64 os.unlink('powned.txt')
65 except:
66     pass
67

>>> Issue: [B102:exec_used] Use of exec detected.
Severity: Medium Confidence: High
CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
More Info: https://bandit.readthedocs.io/en/1.7.7/plugins/b102_exec_used.html
Location: lets-be-bad-guys/badguys/vulnerable/views.py:72:12
71 # Try it the Python 3 way...
72 exec(base64.decodestring(bytes(first_name, 'ascii')))
73 except TypeError:

>>> Issue: [B102:exec_used] Use of exec detected.
Severity: Medium Confidence: High
CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
More Info: https://bandit.readthedocs.io/en/1.7.7/plugins/b102_exec_used.html
Location: lets-be-bad-guys/badguys/vulnerable/views.py:76:16
75 try:
76     exec(base64.decodestring(first_name))
77 except:

>>> Issue: [B110:try_except_pass] Try, Except, Pass detected.
Severity: Low Confidence: High
CWE: CWE-703 (https://cwe.mitre.org/data/definitions/703.html)
More Info: https://bandit.readthedocs.io/en/1.7.7/plugins/b110_try_except_pass.html
Location: lets-be-bad-guys/badguys/vulnerable/views.py:77:12
76 except:
77     pass
78 except:
79     pass

>>> Issue: [B110:try_except_pass] Try, Except, Pass detected.
Severity: Low Confidence: High
CWE: CWE-703 (https://cwe.mitre.org/data/definitions/703.html)
More Info: https://bandit.readthedocs.io/en/1.7.7/plugins/b110_try_except_pass.html
Location: lets-be-bad-guys/badguys/vulnerable/views.py:79:8
78 except:
79     pass
80     pass
81

Code scanned:
Total lines of code: 337
Total lines skipped (#nosec): 0

Run metrics:
Total issues (by severity):
Undefined: 0
Low: 4
Medium: 2
High: 0
Total issues (by confidence):
Undefined: 0
Low: 0
Medium: 1
High: 5

```

Fig: Application 1 results after bandit

```

(user1@kali)-[~/bandit]
$ bandit -r vulpy
[main] INFO profile include tests: None
[main] INFO profile exclude tests: None
[main] INFO cli include tests: None
[main] INFO cli exclude tests: None
[main] INFO running on Python 3.11.6
Run started:2024-02-23 01:56:33.318805

Test results:
>>> Issue: [B113:request_without_timeout] Requests call without timeout
Severity: Medium Confidence: Low
CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
More Info: https://bandit.readthedocs.io/en/1.7.7/plugins/b113_request_without_timeout.html
Location: vulpy/bad/api_list.py:10:8
9
10 r = requests.get('http://127.0.1.1:5000/api/post/{}'.format(username))
11 if r.status_code != 200:

>>> Issue: [B100:hardcoded_tmp_directory] Probable insecure usage of temp file/directory.
Severity: Medium Confidence: Medium
CWE: CWE-377 (https://cwe.mitre.org/data/definitions/377.html)
More Info: https://bandit.readthedocs.io/en/1.7.7/plugins/b100_hardcoded_tmp_directory.html
Location: vulpy/bad/api_post.py:6:20
5
6 api_key_file = Path('/tmp/supersecret.txt')
7

>>> Issue: [B113:request_without_timeout] Requests call without timeout
Severity: Medium Confidence: Low
CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
More Info: https://bandit.readthedocs.io/en/1.7.7/plugins/b113_request_without_timeout.html
Location: vulpy/bad/api_post.py:16:12
15
16 r = requests.post('http://127.0.1.1:5000/api/key', json={'username':username, 'password':passw
17 and})

>>> Issue: [B113:request_without_timeout] Requests call without timeout
Severity: Medium Confidence: Low
CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
More Info: https://bandit.readthedocs.io/en/1.7.7/plugins/b113_request_without_timeout.html
Location: vulpy/bad/api_post.py:38:8
29 api_key = api_key_file.open().read()
30 r = requests.post('http://127.0.1.1:5000/api/post', json={'text':message}, headers={'X-APIKEY': ap
31

>>> Issue: [B100:hardcoded_tmp_directory] Probable insecure usage of temp file/directory.
Severity: Medium Confidence: Medium
CWE: CWE-377 (https://cwe.mitre.org/data/definitions/377.html)
More Info: https://bandit.readthedocs.io/en/1.7.7/plugins/b100_hardcoded_tmp_directory.html
Location: vulpy/utlis/rsa-verify.py:16:10
15
16 with open('/tmp/acme.pub', 'rb') as key_file:
17     public_key = serialization.load_pem_public_key(

Code scanned:
Total lines of code: 1556
Total lines skipped (#nosec): 0

Run metrics:
Total issues (by severity):
Undefined: 0
Low: 9
Medium: 36
High: 4
Total issues (by confidence):
Undefined: 0
Low: 6
Medium: 39
High: 4
Files skipped (0):

```

Fig: Application 2 results after bandit

REFERENCES

1. <https://realpython.com/prevent-python-sql-injection/>
2. <https://offensive360.com/how-to-prevent-hardcoded-passwords/>
3. <https://learn.snyk.io/lesson/insecure-temporary-file/>
4. <https://www.codiga.io/blog/python-subprocess-security/>
5. <https://github.com/ACandeias/PenTestingScripts.git>
6. <https://github.com/PyCQA/bandit.git>
7. <https://github.com/mpirnat/lets-be-bad-guys>
8. <https://github.com/fportantier/vulpy>