RESULTS TO VULPY

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/1.7.5/plugins/b113\_request\_without\_timeout.html

Location: ./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: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./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/1.7.5/plugins/b113\_request\_without\_timeout.html

Location: ./bad/api\_post.py:16:12

15

16 r = requests.post('http://127.0.1.1:5000/api/key', json={'username':username, ' password':password})

17

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/1.7.5/plugins/b113\_request\_without\_timeout.html

Location: ./bad/api\_post.py:30:8

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': api\_key})

31 print(r.text)

Issue: [B404:blacklist] Consider possible security implications associated with the subprocess module.

Severity: Low Confidence: High

CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/blacklists/blacklist\_imports.html#b404-import-subprocess

Location: ./bad/brute.py:3:0

2

3 import subprocess

4 import sys

Issue: [B603:subprocess\_without\_shell\_equals\_true] subprocess call - check for execution of untrusted input.

Severity: Low Confidence: High

CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b603\_subprocess\_without\_shell\_equals\_true.html

Location: ./bad/brute.py:21:13

20 for password in passwords:

21 result = subprocess.run([program, username, password], stdout=subprocess.DEVNULL)

22 if result.returncode == 0:

Issue: [B608:hardcoded\_sql\_expressions] Possible SQL injection vector through string-based query construction.

Severity: Medium Confidence: Medium

CWE: CWE-89 (https://cwe.mitre.org/data/definitions/89.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b608\_hardcoded\_sql\_expressions.html

Location: ./bad/db.py:19:18

18 for u,p in users:

19 c.execute("INSERT INTO users (user, password, failures) VALUES ('%s', '%s', '%d')" %(u, p, 0))

20

Issue: [B608:hardcoded\_sql\_expressions] Possible SQL injection vector through string-based query construction.

Severity: Medium Confidence: Medium

CWE: CWE-89 (https://cwe.mitre.org/data/definitions/89.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b608\_hardcoded\_sql\_expressions.html

Location: ./bad/db\_init.py:20:18

19 for u,p in users:

20 c.execute("INSERT INTO users (username, password, failures, mfa\_enabled, mfa\_secret) VALUES ('%s', '%s', '%d', '%d', '%s')" %(u, p, 0, 0, ''))

21

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./bad/libapi.py:16:18

15

16 for f in Path('/tmp/').glob('vulpy.apikey.' + username + '.\*'):

17 print('removing', f)

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./bad/libapi.py:20:14

19

20 keyfile = '/tmp/vulpy.apikey.{}.{}'.format(username, key)

21

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./bad/libapi.py:33:18

32

33 for f in Path('/tmp/').glob('vulpy.apikey.\*.' + key):

34 return f.name.split('.')[2]

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/1.7.5/plugins/b110\_try\_except\_pass.html

Location: ./bad/libsession.py:21:4

20 session = json.loads(base64.b64decode(cookie))

21 except Exception:

22 pass

23

Issue: [B608:hardcoded\_sql\_expressions] Possible SQL injection vector through string-based query construction.

Severity: Medium Confidence: Medium

CWE: CWE-89 (https://cwe.mitre.org/data/definitions/89.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b608\_hardcoded\_sql\_expressions.html

Location: ./bad/libuser.py:12:21

11

12 user = c.execute("SELECT \* FROM users WHERE username = '{}' and password = '{}'".format(username, password)).fetchone()

13

Issue: [B608:hardcoded\_sql\_expressions] Possible SQL injection vector through string-based query construction.

Severity: Medium Confidence: Medium

CWE: CWE-89 (https://cwe.mitre.org/data/definitions/89.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b608\_hardcoded\_sql\_expressions.html

Location: ./bad/libuser.py:25:14

24

25 c.execute("INSERT INTO users (username, password, failures, mfa\_enabled, mfa\_secret) VALUES ('%s', '%s', '%d', '%d', '%s')" %(username, password, 0, 0, ''))

26

Issue: [B608:hardcoded\_sql\_expressions] Possible SQL injection vector through string-based query construction.

Severity: Medium Confidence: Medium

CWE: CWE-89 (https://cwe.mitre.org/data/definitions/89.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b608\_hardcoded\_sql\_expressions.html

Location: ./bad/libuser.py:53:14

52

53 c.execute("UPDATE users SET password = '{}' WHERE username = '{}'".format(password, username))

54 conn.commit()

Issue: [B105:hardcoded\_password\_string] Possible hardcoded password: 'aaaaaaa'

Severity: Low Confidence: Medium

CWE: CWE-259 (https://cwe.mitre.org/data/definitions/259.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b105\_hardcoded\_password\_string.html

Location: ./bad/vulpy-ssl.py:13:11

12 app = Flask('vulpy')

13 app.config['SECRET\_KEY'] = 'aaaaaaa'

14

Issue: [B201:flask\_debug\_true] A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.

Severity: High Confidence: Medium

CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b201\_flask\_debug\_true.html

Location: ./bad/vulpy-ssl.py:29:0

28

29 app.run(debug=True, host='127.0.1.1', ssl\_context=('/tmp/acme.cert', '/tmp/acme.key'))

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./bad/vulpy-ssl.py:29:51

28

29 app.run(debug=True, host='127.0.1.1', ssl\_context=('/tmp/acme.cert', '/tmp/acme.key'))

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./bad/vulpy-ssl.py:29:69

28

29 app.run(debug=True, host='127.0.1.1', ssl\_context=('/tmp/acme.cert', '/tmp/acme.key'))

Issue: [B105:hardcoded\_password\_string] Possible hardcoded password: 'aaaaaaa'

Severity: Low Confidence: Medium

CWE: CWE-259 (https://cwe.mitre.org/data/definitions/259.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b105\_hardcoded\_password\_string.html

Location: ./bad/vulpy.py:16:11

15 app = Flask('vulpy')

16 app.config['SECRET\_KEY'] = 'aaaaaaa'

17

Issue: [B201:flask\_debug\_true] A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.

Severity: High Confidence: Medium

CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b201\_flask\_debug\_true.html

Location: ./bad/vulpy.py:55:0

54

55 app.run(debug=True, host='127.0.1.1', port=5000, extra\_files='csp.txt')

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./good/cutpasswd.py:3:10

2

3 with open('/tmp/darkweb2017-top10000.txt') as f:

4 for password in f.readlines():

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/1.7.5/plugins/b113\_request\_without\_timeout.html

Location: ./good/httpbrute.py:22:15

21 for password in passwords:

22 response = requests.post(URL, data = {'username': username, 'password': password})

23 if 'HOME' in response.text:

Issue: [B105:hardcoded\_password\_string] Possible hardcoded password: 'MYSUPERSECRETKEY'

Severity: Low Confidence: Medium

CWE: CWE-259 (https://cwe.mitre.org/data/definitions/259.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b105\_hardcoded\_password\_string.html

Location: ./good/libapi.py:10:9

9

10 secret = 'MYSUPERSECRETKEY'

11 not\_after = 60 # 1 minute

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/1.7.5/plugins/b110\_try\_except\_pass.html

Location: ./good/libsession.py:22:4

21 country = geo.country.iso\_code

22 except Exception:

23 pass

24

Issue: [B608:hardcoded\_sql\_expressions] Possible SQL injection vector through string-based query construction.

Severity: Medium Confidence: Medium

CWE: CWE-89 (https://cwe.mitre.org/data/definitions/89.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b608\_hardcoded\_sql\_expressions.html

Location: ./good/libuser.py:61:14

60 c = conn.cursor()

61 c.execute("INSERT INTO users (username, password, salt, failures, mfa\_enabled, mfa\_secret) VALUES ('%s', '%s', '%s', '%d', '%d', '%s')" %(username, '', '', 0, 0, ''))

62 conn.commit()

Issue: [B105:hardcoded\_password\_string] Possible hardcoded password: 'aaaaaaa'

Severity: Low Confidence: Medium

CWE: CWE-259 (https://cwe.mitre.org/data/definitions/259.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b105\_hardcoded\_password\_string.html

Location: ./good/vulpy-ssl.py:13:11

12 app = Flask('vulpy')

13 app.config['SECRET\_KEY'] = 'aaaaaaa'

14

Issue: [B201:flask\_debug\_true] A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.

Severity: High Confidence: Medium

CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b201\_flask\_debug\_true.html

Location: ./good/vulpy-ssl.py:29:0

28

29 app.run(debug=True, host='127.0.1.1', ssl\_context=('/tmp/acme.cert', '/tmp/acme.key'))

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./good/vulpy-ssl.py:29:51

28

29 app.run(debug=True, host='127.0.1.1', ssl\_context=('/tmp/acme.cert', '/tmp/acme.key'))

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./good/vulpy-ssl.py:29:69

28

29 app.run(debug=True, host='127.0.1.1', ssl\_context=('/tmp/acme.cert', '/tmp/acme.key'))

Issue: [B105:hardcoded\_password\_string] Possible hardcoded password: '123aa8a93bdde342c871564a62282af857bda14b3359fde95d0c5e4b321610c1'

Severity: Low Confidence: Medium

CWE: CWE-259 (https://cwe.mitre.org/data/definitions/259.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b105\_hardcoded\_password\_string.html

Location: ./good/vulpy.py:17:11

16 app = Flask('vulpy')

17 app.config['SECRET\_KEY'] = '123aa8a93bdde342c871564a62282af857bda14b3359fde95d0c5e4b321610c1'

18

Issue: [B201:flask\_debug\_true] A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.

Severity: High Confidence: Medium

CWE: CWE-94 (https://cwe.mitre.org/data/definitions/94.html)

More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b201\_flask\_debug\_true.html

Location: ./good/vulpy.py:53:0

52

53 app.run(debug=True, host='127.0.1.1', port=5001, extra\_files='csp.txt')

54

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./utils/ca-create.py:31:10

30

31 with open('/tmp/ca.key', 'wb') as out:

32 out.write(pem\_private)

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./utils/ca-create.py:34:10

33

34 with open('/tmp/ca.pub', 'wb') as out:

35 out.write(pem\_public)

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./utils/ca-create.py:58:10

57 # Write our certificate out to disk.

58 with open('/tmp/ca.cert', 'wb') as out:

59 out.write(cert.public\_bytes(serialization.Encoding.PEM))

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./utils/ca-csr-create.py:12:10

11

12 with open("/tmp/acme.key", "rb") as key\_file:

13 private\_key = serialization.load\_pem\_private\_key(

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./utils/ca-csr-create.py:35:10

34 # Write our CSR out to disk.

35 with open("/tmp/acme.csr", "wb") as out:

36 out.write(csr.public\_bytes(serialization.Encoding.PEM))

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./utils/ca-csr-load.py:13:10

12

13 with open("/tmp/ca.cert", "rb") as ca\_cert\_file:

14 ca\_cert = x509.load\_pem\_x509\_certificate(ca\_cert\_file.read(), default\_backend())

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./utils/ca-csr-load.py:16:10

15

16 with open("/tmp/acme.csr", "rb") as csr\_file:

17 csr = x509.load\_pem\_x509\_csr(csr\_file.read(), default\_backend())

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./utils/ca-csr-load.py:19:10

18

19 with open("/tmp/ca.key", "rb") as key\_file:

20 private\_key = serialization.load\_pem\_private\_key(

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./utils/ca-csr-load.py:35:10

34 # Write our certificate out to disk.

35 with open('/tmp/acme.cert', 'wb') as out:

36 out.write(cert.public\_bytes(serialization.Encoding.PEM))

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/1.7.5/plugins/b113\_request\_without\_timeout.html

Location: ./utils/generate\_bad\_passwords.py:21:9

20 click.echo('Downloading password file...', nl=False, err=True)

21 with requests.get(url, stream=True) as r:

22 r.raise\_for\_status()

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/1.7.5/plugins/b113\_request\_without\_timeout.html

Location: ./utils/httpbrute.py:25:19

24 for password in passwords:

25 response = requests.post(url, data = {'username': username, 'password': password})

26 logging.info('{} {} {}'.format(username, password, response.status\_code))

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./utils/rsa-decrypt.py:14:10

13

14 with open("/tmp/acme.key", "rb") as key\_file:

15 private\_key = serialization.load\_pem\_private\_key(

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./utils/rsa-encrypt.py:14:10

13

14 with open("/tmp/acme.pub", "rb") as key\_file:

15 public\_key = serialization.load\_pem\_public\_key(

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./utils/rsa-keygen.py:26:10

25

26 with open('/tmp/acme.key', 'wb') as out:

27 out.write(pem\_private)

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./utils/rsa-keygen.py:29:10

28

29 with open('/tmp/acme.pub', 'wb') as out:

30 out.write(pem\_public)

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./utils/rsa-sign.py:15:10

14

15 with open("/tmp/acme.key", "rb") as key\_file:

16 private\_key = serialization.load\_pem\_private\_key(

Issue: [B108: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/1.7.5/plugins/b108\_hardcoded\_tmp\_directory.html

Location: ./utils/rsa-verify.py:16:10

15

16 with open("/tmp/acme.pub", "rb") as key\_file:

17 public\_key = serialization.load\_pem\_public\_key(

**RESULTS TO LETS-BE-BAD-GUYS**

Test results:

Issue: [B110:try\_except\_pass] Try, Except, Pass detected.

Severity: Low Confidence: High

**CWE: CWE-703 (https://cwe.mitre.org/data/definitions/703.html)**

Location: ./badguys/vulnerable/views.py:65:8

More Info: https://bandit.readthedocs.io/en/1.7.4/plugins/b110\_try\_except\_pass.html

64 os.unlink('p0wned.txt')

65 except:

66 pass

Issue: [B102:exec\_used] Use of exec detected.

Severity: Medium Confidence: High

**CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)**

Location: ./badguys/vulnerable/views.py:72:12

More Info: https://bandit.readthedocs.io/en/1.7.4/plugins/b102\_exec\_used.html

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)**

Location: ./badguys/vulnerable/views.py:76:16

More Info: https://bandit.readthedocs.io/en/1.7.4/plugins/b102\_exec\_used.html

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)**

Location: ./badguys/vulnerable/views.py:77:12

More Info: https://bandit.readthedocs.io/en/1.7.4/plugins/b110\_try\_except\_pass.html

76 exec(base64.decodestring(first\_name))

77 except:

78 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)**

Location: ./badguys/vulnerable/views.py:79:8

More Info: https://bandit.readthedocs.io/en/1.7.4/plugins/b110\_try\_except\_pass.html

78 pass

79 except:

80 pass

|  |  |  |
| --- | --- | --- |
|  | CWE | Definition |
| 6 | 400 | The product does not properly control the allocation and maintenance of a limited resource, thereby enabling an actor to influence the amount of resources consumed, eventually leading to the exhaustion of available resources.  Limited resources include memory, file system storage, database connection pool entries, and CPU. If an attacker can trigger the allocation of these limited resources, but the number or size of the resources is not controlled, then the attacker could cause a denial of service that consumes all available resources. This would prevent valid users from accessing the product, and it could potentially have an impact on the surrounding environment. For example, a memory exhaustion attack against an application could slow down the application as well as its host operating system.  There are at least three distinct scenarios which can commonly lead to resource exhaustion:   * Lack of throttling for the number of allocated resources * Losing all references to a resource before reaching the shutdown stage * Not closing/returning a resource after processing   Resource exhaustion problems are often result due to an incorrect implementation of the following situations:   * Error conditions and other exceptional circumstances. * Confusion over which part of the program is responsible for releasing the resource. |
| 25 | 377 | Creating and using insecure temporary files can leave application and system data vulnerable to attack. |
| 2 | 78 | The product constructs all or part of an OS command using externally influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended OS command when it is sent to a downstream component.  This could allow attackers to execute unexpected, dangerous commands directly on the operating system. This weakness can lead to a vulnerability in environments in which the attacker does not have direct access to the operating system, such as in web applications. Alternately, if the weakness occurs in a privileged program, it could allow the attacker to specify commands that normally would not be accessible, or to call alternate commands with privileges that the attacker does not have. The problem is exacerbated if the compromised process does not follow the principle of least privilege, because the attacker-controlled commands may run with special system privileges that increases the amount of damage. |
| 6 | 89 | The product constructs all or part of an SQL command using externally influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component.  Without sufficient removal or quoting of SQL syntax in user-controllable inputs, the generated SQL query can cause those inputs to be interpreted as SQL instead of ordinary user data. This can be used to alter query logic to bypass security checks, or to insert additional statements that modify the back-end database, possibly including execution of system commands.  SQL injection has become a common issue with database-driven web sites. The flaw is easily detected, and easily exploited, and as such, any site or product package with even a minimal user base is likely to be subject to an attempted attack of this kind. This flaw depends on the fact that SQL makes no real distinction between the control and data planes. |
| 2 | 703 | The product does not properly anticipate or handle exceptional conditions that rarely occur during normal operation of the product. |
| 5 | 259 | The product contains a hard-coded password, which it uses for its own inbound authentication or for outbound communication to external components.  A hard-coded password typically leads to a significant authentication failure that can be difficult for the system administrator to detect. Once detected, it can be difficult to fix, so the administrator may be forced into disabling the product entirely. There are two main variations:  Inbound: the product contains an authentication mechanism that checks for a hard-coded password.  Outbound: the product connects to another system or component, and it contains hard-coded password for connecting to that component.  In the Inbound variant, a default administration account is created, and a simple password is hard coded into the product and associated with that account. This hard-coded password is the same for each installation of the product, and it usually cannot be changed or disabled by system administrators without manually modifying the program, or otherwise patching the product. If the password is ever discovered or published (a common occurrence on the Internet), then anybody with knowledge of this password can access the product. Finally, since all installations of the product will have the same password, even across different organizations, this enables massive attacks such as worms to take place.  The Outbound variant applies to front-end systems that authenticate with a back-end service. The back-end service may require a fixed password which can be easily discovered. The programmer may simply hard-code those back-end credentials into the front-end product. Any user of that program may be able to extract the password. Client-side systems with hard-coded passwords pose even more of a threat since the extraction of a password from a binary is usually very simple. |
| 4 | 94 | The product constructs all or part of a code segment using externally influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the syntax or behavior of the intended code segment.  When a product allows a user's input to contain code syntax, it might be possible for an attacker to craft the code in such a way that it will alter the intended control flow of the product. Such an alteration could lead to arbitrary code execution.  Injection problems encompass a wide variety of issues -- all mitigated in very different ways. For this reason, the most effective way to discuss these weaknesses is to note the distinct features which classify them as injection weaknesses. The most important issue to note is that all injection problems share one thing in common -- i.e., they allow for the injection of control plane data into the user-controlled data plane. This means that the execution of the process may be altered by sending code in through legitimate data channels, using no other mechanism. While buffer overflows, and many other flaws, involve the use of some further issue to gain execution, injection problems need only for the data to be parsed. The most classic instantiations of this category of weakness are SQL injection and format string vulnerabilities. |

CWE-400

* This result is from a static code analysis tool called Bandit. It indicates that in the file api\_post.py, a requests.post call is made without a timeout parameter, which could potentially cause the program to hang or become unresponsive if the server does not respond in a timely manner.
* The severity of the issue is labeled as "Medium", which means that it could potentially cause a security vulnerability or impact system stability. The confidence level of the tool's analysis is "Low", which means that further investigation may be needed to confirm the issue.
* The issue is mapped to a specific Common Weakness Enumeration (CWE) number, which provides a standardized way of identifying and categorizing software weaknesses. In this case, the CWE number is 400, which is related to the lack of robustness in error handling or failure to gracefully handle unexpected conditions.
* The output provides a link to the Bandit documentation, which provides more information about the specific issue (B113) and how it can be resolved. To address this issue, the requests.post call should be modified to include a timeout parameter, which specifies the maximum amount of time to wait for a response from the server before raising an exception or returning an error. This helps ensure that the program remains responsive and does not hang or become unresponsive if the server does not respond in a timely manner.

CWE-377

* This result means that the code is probably using a hardcoded temporary file directory (/tmp/) for storing files related to the vulpy.apikey and the directory permissions might not be set up securely. It's important to note that /tmp/ is a common target for attackers as it is usually world-readable and writable, and thus provides an easy way for attackers to manipulate files and execute code on the system.
* The glob() function call in the code sample suggests that the code is iterating over the files matching a certain pattern in the /tmp/ directory and removing them. If the username value comes from an untrusted source, an attacker may be able to perform directory traversal attacks by crafting a malicious input that manipulates the username value and accesses arbitrary files outside of the expected directory. To avoid this, it's recommended to use a more secure temporary file directory with restricted permissions or to generate a unique, unpredictable temporary file name.

CWE-78

* This result is a warning that the subprocess module is being imported in a file named brute.py. The warning suggests that you should consider the possible security implications associated with using the subprocess module, as it can allow for the execution of arbitrary system commands. The subprocess module is often used to spawn new processes, which can execute system commands. If not used carefully, it can introduce vulnerabilities, such as command injection attacks, which could allow an attacker to execute malicious commands on the system. The warning is just a reminder to ensure that the use of the subprocess module is secure and that the user input is validated before it is passed to the subprocess module.

CWE-89

* This result is a warning about a possible SQL injection vulnerability in the code. The code appears to be constructing SQL queries using string concatenation rather than using parameterized queries, which can make the code vulnerable to SQL injection attacks. An attacker could exploit this vulnerability to execute arbitrary SQL code on the underlying database, potentially leading to data theft or destruction. The severity of this issue is Medium, which means that while it is not as serious as some other security issues, it still needs to be addressed. The confidence level is Medium, which indicates that the analysis is not completely certain that this is a vulnerability, but it is highly likely. The CWE associated with this issue is CWE-89, which is a commonly used identifier for SQL injection vulnerabilities.
* This Bandit result is warning about the potential for SQL injection vulnerability in the code. Specifically, the result is labeled as [B608:hardcoded\_sql\_expressions] Possible SQL injection vector through string-based query construction.
* The code snippet in question is using string concatenation to construct an SQL query, which includes variables for the username, password, and a value for a "failures" column. However, this approach is vulnerable to SQL injection attacks, where an attacker can manipulate the data being inserted into the database and potentially execute malicious SQL statements.
* To avoid this risk, it is recommended to use parameterized queries instead of string concatenation when building SQL queries. Parameterized queries use placeholders for values that are then bound to parameters, separating the data from the query itself and preventing SQL injection attacks.
* In the case of the code snippet, a better approach would be to use a parameterized query using placeholders for the values, and then pass the values as parameters to the query. For example, using the execute method of a cursor object in Python, the query could be written like this: c.execute("INSERT INTO users (user, password, failures) VALUES (%s, %s, %s)", (u, p, 0))
* Here, the %s placeholders indicate where the parameters should be inserted, and the second argument to execute provides a tuple of values that will be substituted in place of the placeholders. This approach avoids the risk of SQL injection vulnerabilities.

CWE-703

* This result is from a static code analysis tool called Bandit. It indicates that a try/except block with a pass statement was detected in the file libsession.py on line 21, column 4.
* The try/except block is used to handle exceptions that may occur while executing the code within the block. However, in this case, the pass statement within the except block effectively does nothing and ignores any exception that might be raised. This can hide potential issues in the code and make it more difficult to identify and resolve problems.
* The severity of the issue is labeled as "Low", which means that the issue may not necessarily pose a significant security risk, but it may still impact the performance or functionality of the system. The confidence level of the tool's analysis is "High", which means that the issue is almost certainly present and can be easily confirmed.
* The issue is mapped to a specific Common Weakness Enumeration (CWE) number, which provides a standardized way of identifying and categorizing software weaknesses. In this case, the CWE number is 703, which is related to the use of improper exception handling.
* Finally, the output provides a link to the Bandit documentation, which provides more information about the specific issue (B110) and how it can be resolved. To resolve this issue, the try/except block should be updated to handle the exception in a more appropriate manner, such as logging the error or providing a more informative error message. Alternatively, if the exception is not important for the application's functionality, the try/except block can be removed altogether.

CWE-259

* This result is from a static code analysis tool called Bandit. It indicates that a possible hardcoded password string was found in a file called vulpy-ssl.py on line 13, column 11. The value of the hardcoded password string is aaaaaaa.
* The severity of the issue is labeled as "Low" and the confidence level of the tool's analysis is "Medium". This means that while the issue has the potential to be a security vulnerability, it may not necessarily be exploitable in practice. Further investigation is needed to determine the impact of the issue on the security of the system.
* The issue is mapped to a specific Common Weakness Enumeration (CWE) number, which provides a standardized way of identifying and categorizing security vulnerabilities. In this case, the CWE number is 259, which is related to the use of hard-coded credentials.
* Finally, the output provides a link to the Bandit documentation, which provides more information about the specific issue (B105) and how it can be resolved. To resolve this issue, the hardcoded password should be removed or replaced with a more secure solution such as a randomly generated secret or an environment variable.
* The problem with a hardcoded password string, such as the one identified by Bandit in this result, is that it is a security vulnerability that could allow an attacker to gain unauthorized access to the system. Hardcoded passwords are easy to discover and exploit, and they can be a common attack vector for malicious actors.
* If an attacker gains access to the hardcoded password, they could use it to impersonate a legitimate user and gain unauthorized access to sensitive data or system resources. Additionally, if the password is used in multiple places, an attacker could use the same password to gain access to other parts of the system.
* To mitigate this vulnerability, the hardcoded password should be removed or replaced with a more secure solution such as a randomly generated secret or an environment variable. This makes it much harder for an attacker to discover the password and gain unauthorized access to the system.

CWE-94

* This result is from a static code analysis tool called Bandit. It indicates that the Flask application in the file vulpy.py appears to be running with the debug=True setting, which exposes the Werkzeug debugger and allows the execution of arbitrary code.
* When Flask is run with debug mode enabled, Werkzeug provides a built-in debugger that allows developers to interactively debug their applications. However, this mode is not intended for use in production environments and can expose sensitive information and vulnerabilities.
* The severity of the issue is labeled as "High", which means that the issue poses a significant security risk that could be exploited by attackers. The confidence level of the tool's analysis is "Medium", which means that while the issue is likely present, further investigation may be needed to confirm it.
* The issue is mapped to a specific Common Weakness Enumeration (CWE) number, which provides a standardized way of identifying and categorizing software weaknesses. In this case, the CWE number is 94, which is related to allowing an attacker to execute arbitrary code.
* Finally, the output provides a link to the Bandit documentation, which provides more information about the specific issue (B201) and how it can be resolved. To resolve this issue, the debug parameter should be set to False before deploying the Flask application in a production environment.