Static Security Analyses

Alessandro Bocci

name.surname@phd.unipi.it

Advanced Software Engineering (Lab)

15/11/2023

What will you do?

- Use Bandit to analyse the code of (new) `microase2324`
- Examine the output of Bandit
- Resolve all the vulnerabilities

Software Prerequisites

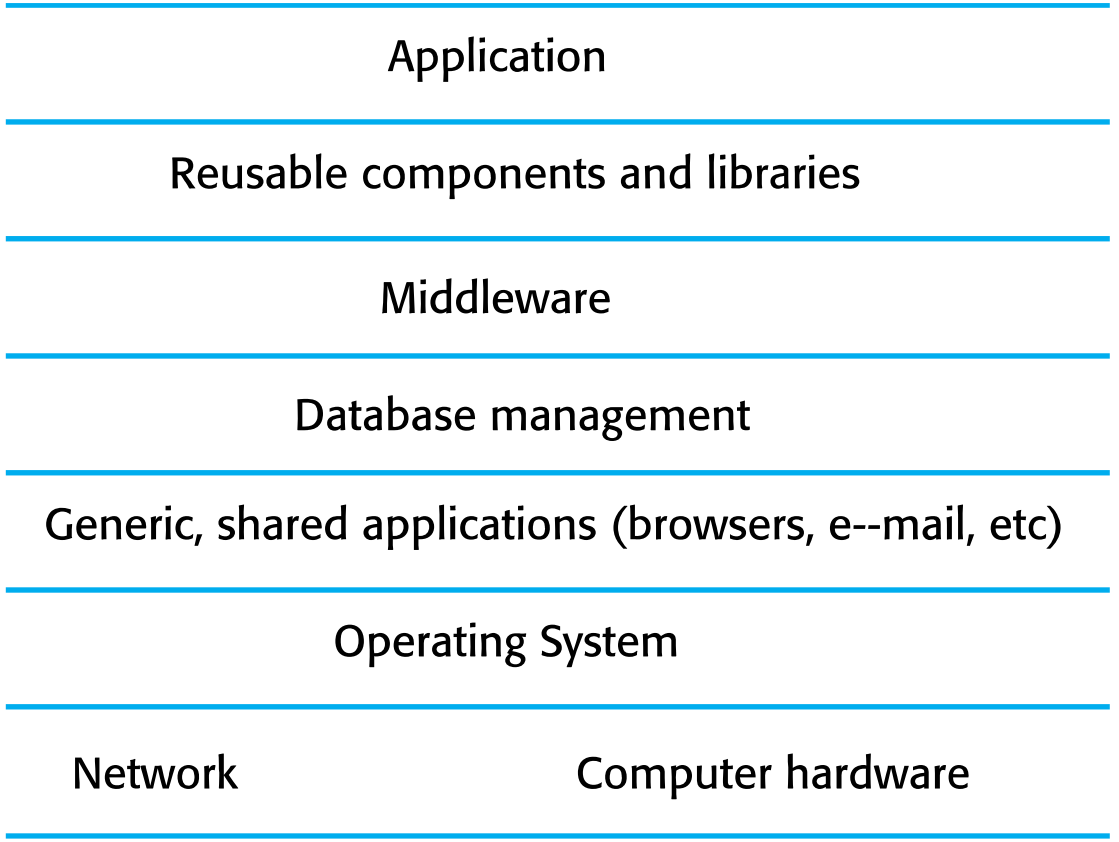
- Bandit
- Python code (new **`microase2324`** from Moodle)

Some new API examples:

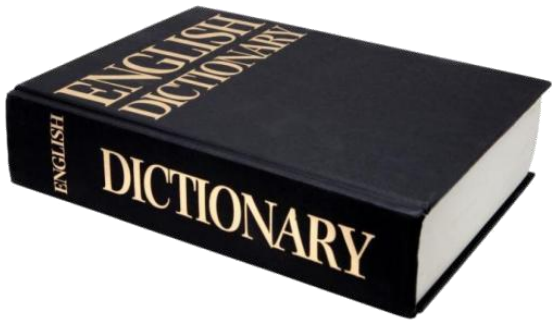
```
http://localhost:5000/ping/google.it                    #pings google.it
http://localhost:5000/log/count/math-service2:5000      #counts logs of math-service2 in db
http://localhost:5000/math/add?lst=[1,2,3,4]            #sums the content of a list
http://localhost:5000/math/secure_random?a=1&b=6        #returns a random number between 1 and 6
```

Security at all levels

- Security is a **holistic property** of a system, it should in principle be guaranteed at all levels:
  - **Infrastructure security**, maintaining the security of all systems and networks that support services.
  - **Application security**, securing individual application systems or related groups of systems
  - **Operational security**, secure operation and use of the organization's systems (e.g. users!)

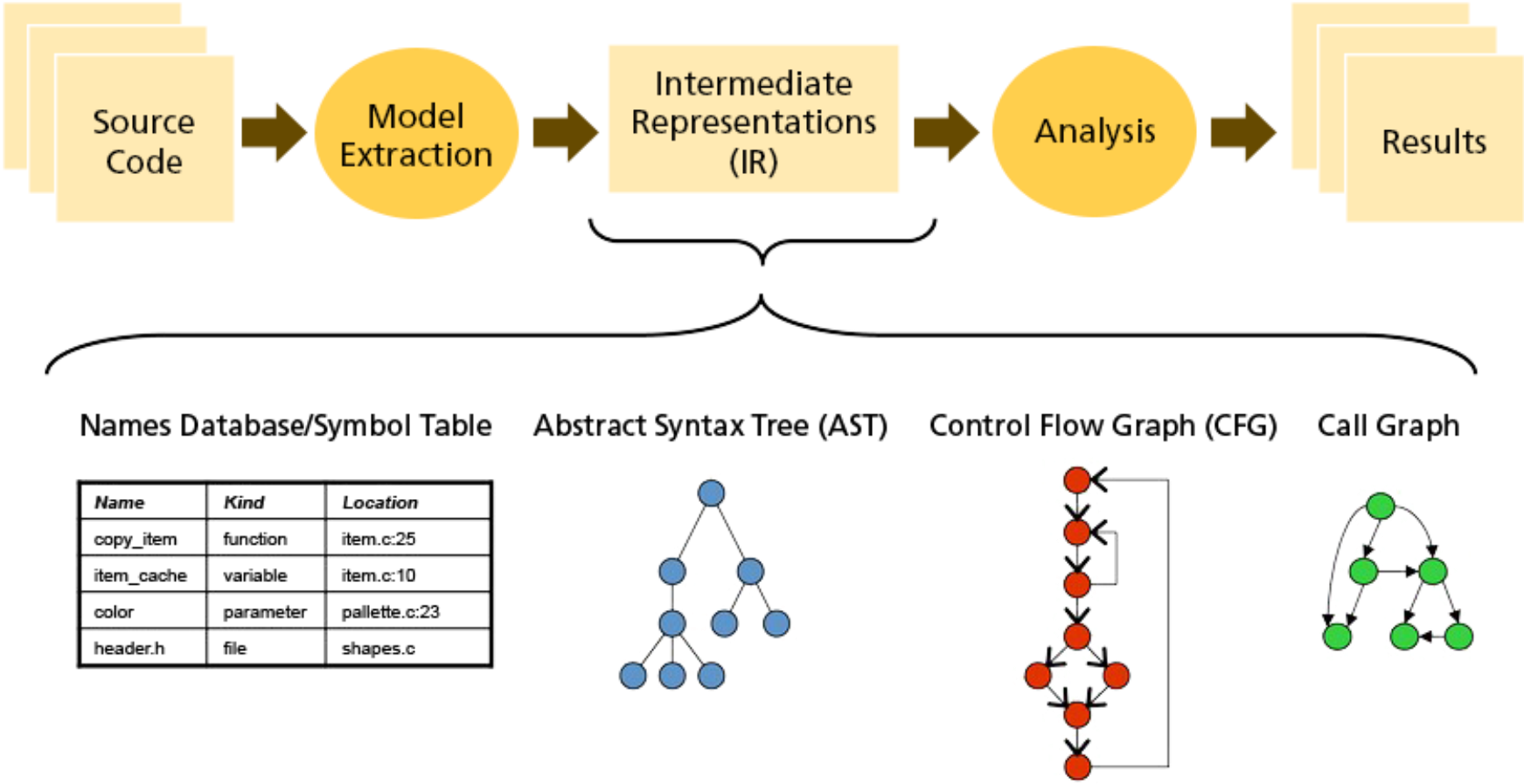


Terminology



Term	Definition
Asset	Something of value which has to be protected. The asset may be the software system itself or data used by that system.
Attack	An exploitation of a system's vulnerability. Generally, this is from outside the system and is a deliberate attempt to cause some damage.
Control	A protective measure that reduces a system's vulnerability. Encryption is an example of a control that reduces a vulnerability of a weak access control system
Exposure	Possible loss or harm to a computing system. This can be loss or damage to data, or can be a loss of time and effort if recovery is necessary after a security breach.
Threat	Circumstances that have potential to cause loss or harm. You can think of these as a system vulnerability that is subjected to an attack.
Vulnerability	A weakness in a computer-based system that may be exploited to cause loss or harm.

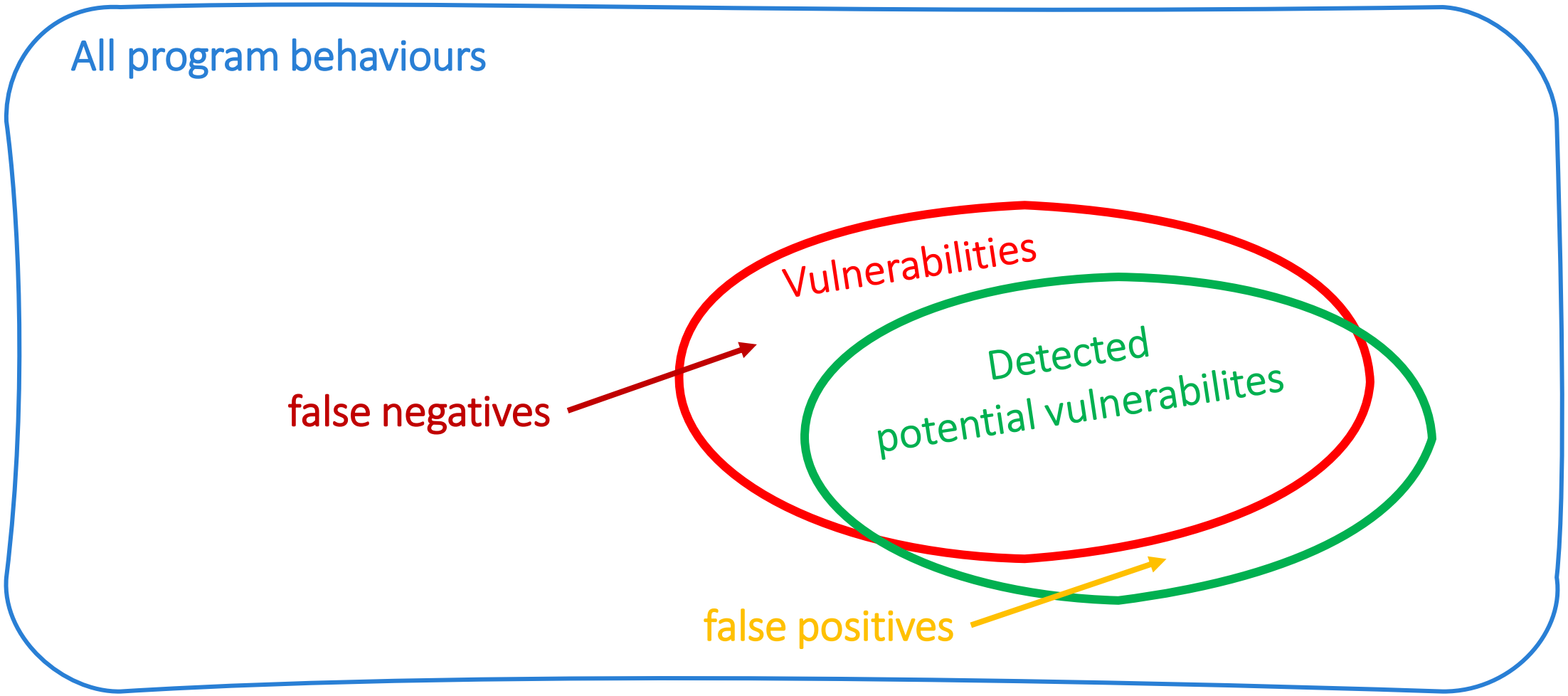
Vulnerability Avoidance with Static Analysis



Name	Kind	Location
copy_item	function	item.c:25
item_cache	variable	item.c:10
color	parameter	pallette.c:23
header.h	file	shapes.c

Static Analysis = Analyse the system (source code or its representation) to check some property without running it.

A note on Static Analysis



Bandit => Secure Programming

- Bandit is a static analysis tool designed to find common security issues in Python code, by exploiting known patterns (plugins).
- Bandit was originally developed within the OpenStack Security Project and later re-homed to PyCQA.
- It recognizes 70 vulnerabilities out-of-the-box.


```
pip install bandit
```

```
bandit -r <path to code>
```

Today's Lab

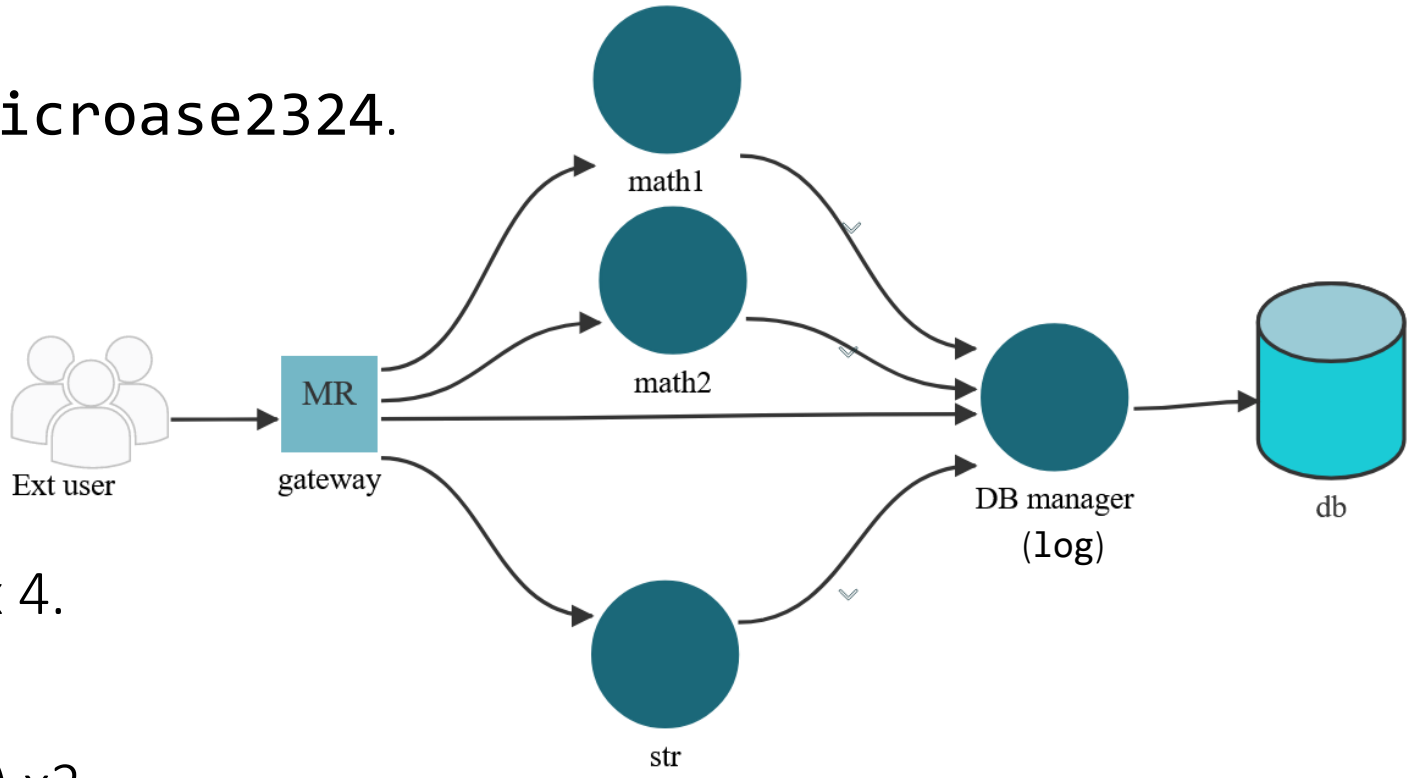
Download the new version of `microase2324` from the Moodle

LAB TODO

1. Run Bandit in the root folder of `microase2324`.
2. Examine the output.
3. Resolve the vulnerabilites.

VULNERABILITIES

1. B602 (run subprocess with shell).
2. B113 (request without timeout) x 4.
3. B105 (hardcoded password).
4. B608 (hardcoded SQL expression) x2.
5. B307 (used function eval).
6. B311 (pseudo random generator for security/cryptograpy).



Example – B324: hashlib, use of weak MD5

```
>> Issue: [B324:hashlib] Use of weak MD5 hash for security. Consider usedforsecurity=False
   Severity: High   Confidence: High
   CWE: CWE-327 (https://cwe.mitre.org/data/definitions/327.html)
   More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b324_hashlib.html
   Location: ASE/lab6/test.py:2:9
1       import hashlib
2       result = hashlib.md5(b'ASE ASE ASE')
3       print("The byte equivalent of hash is : ", end ="")
```

Example – B324: hashlib, use of weak MD5

```
>> Issue: [B324:hashlib] Use of weak MD5 hash for security. Consider usedforsecurity=False
   Severity: High   Confidence: High
   CWE: CWE-327 (https://cwe.mitre.org/data/definitions/327.html)
   More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b324_hashlib.html
   Location: ASE/lab6/test.py:2:9
1          import hashlib
2          result = hashlib.md5(b'ASE ASE ASE')
3          print("The byte equivalent of hash is : ", end ="")
```

Issue -> Use of MD5 (cryptographic hash function).

Example – B324: hashlib, use of weak MD5

```
>> Issue: [B324:hashlib] Use of weak MD5 hash for security. Consider usedforsecurity=False
   Severity: High   Confidence: High
   CWE: CWE-327 (https://cwe.mitre.org/data/definitions/327.html)
   More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b324_hashlib.html
   Location: ASE/lab6/test.py:2:9
1          import hashlib
2          result = hashlib.md5(b'ASE ASE ASE')
3          print("The byte equivalent of hash is : ", end ="")
```

Severity of the Issue:

- Bandit classifies issues in Low, Medium and High.
- It gives a level of confidence for every issue, also Low, Medium and High.

Example – B324: hashlib, use of weak MD5

```
>> Issue: [B324:hashlib] Use of weak MD5 hash for security. Consider usedforsecurity=False
   Severity: High   Confidence: High
   CWE: CWE-327 (https://cwe.mitre.org/data/definitions/327.html)
   More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b324_hashlib.html
   Location: ASE/lab6/test.py:2:9
1          import hashlib
2          result = hashlib.md5(b'ASE ASE ASE')
3          print("The byte equivalent of hash is : ", end ="")
```

Definition of *why* it is an issue.

Use it for understand how to resolve it!

Example – B324: hashlib, use of weak MD5

```
>> Issue: [B324:hashlib] Use of weak MD5 hash for security. Consider usedforsecurity=False
   Severity: High   Confidence: High
   CWE: CWE-327 (https://cwe.mitre.org/data/definitions/327.html)
   More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b324_hashlib.html
   Location: ASE/lab6/test.py:2:9
1          import hashlib
2          result = hashlib.md5(b'ASE ASE ASE')
3          print("The byte equivalent of hash is : ", end ="")
```

Bandit's info about the issue.

Example – B324: hashlib, use of weak MD5

```
>> Issue: [B324:hashlib] Use of weak MD5 hash for security. Consider usedforsecurity=False
   Severity: High   Confidence: High
   CWE: CWE-327 (https://cwe.mitre.org/data/definitions/327.html)
   More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b324_hashlib.html
   Location: ASE/lab6/test.py:2:9
1          import hashlib
2          result = hashlib.md5(b'ASE ASE ASE')
3          print("The byte equivalent of hash is : ", end ="")
```

Location of the issue. In this example:

- The path for the file having the issue is `ASE/lab6/`
- The file with the issue is `test.py`

Example – B324: hashlib, use of weak MD5

```
>> Issue: [B324:hashlib] Use of weak MD5 hash for security. Consider usedforsecurity=False
   Severity: High   Confidence: High
   CWE: CWE-327 (https://cwe.mitre.org/data/definitions/327.html)
   More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b324_hashlib.html
   Location: ASE/lab6/test.py:2:9
1          import hashlib
2          result = hashlib.md5(b'ASE ASE ASE')
3          print("The byte equivalent of hash is : ", end ="")
```

Location of the issue. In this example:

- The path for the file having the issue is `ASE/lab6/`
- The file with the issue is `test.py`
- The line of the issue is 2 (and column 9).

Example – B324: hashlib, use of weak MD5

How to resolve it?

Change the hash function with a secure one!

From:

```
result = hashlib.md5(b'ASE ASE ASE’)
```

To:

```
result = hashlib.sha256(b'ASE ASE ASE')
```

Example – B324: hashlib, use of weak MD5

Remember:

- Do not change the behaviour of the code!!!

  (in the example the hash of the string is done but with SHA256).

- Do not follow Bandit's suggestions blindly!!!

  (in the example the suggestion was to put `usedforsecurity=False`, ask yourself when it is correct).

BONUS STAGE!


CONGRATULATIONS!
Sanwa Bank
BEAMS
OPEL
SUMITOMO BANK
GROUP
SPECIAL MATCH
OPEL
FUJIFILM
BEAMS
adidas
CREDIT 0
INSERT COIN

Bonus stage

Hack `microase2324` by exploiting the vulnerabilities before your patches!

- Use of `eval` and `subprocess` -> try to read a file of your system.
- Harcoded SQL -> drop al the info in the database.
- Pseudo random generator -> guess the next generated number.
  (remember: 2 math-services, 2 pseudo-random sequences)

Lab take away

- ❑ Familiarise with a static analysis tool for security.
- ❑ Understand vulnerabilities in python code.
- ❑ Patch vulnerabilities.