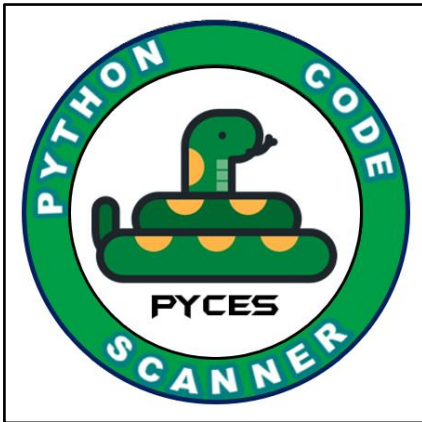


Automated Enhanced Capability Security Scanner

Contents

CONTENTS.....	1
INTRODUCTION	2
SETUP.....	4
REQUIREMENTS.....	4
INSTALLING DEPENDENCIES & RULESETS	5
RUNNING THE SCANNER.....	6
1. SCANNING MODES:	6
2. RUNNING THE SCANNER:	6
SPECIAL CASES	8
1. NETWORK BACKUP:	8
DEVELOPER.....	10

Introduction



PyCes (Python Code Scanner) is an automated enhanced capability security static analysis tool for Python based source code.

Devised with the sole purpose of boosting the security analysis capability by combining the expertise of Open-Source tools and internal rulesets/software to deliver the complete package.

PyCes gives a consolidated HTML report with the following:

- Type and line content of various types of code in the package.
- Any public third-party libraries that may be present in the code.
- List of potential Python vulnerabilities present in code.
- Dependency check report by OWASP DC.
- Grep based search for any vulnerable patterns which may be present in the code.

Currently Supported Frameworks:

- Django
- Web2Py
- TurboGears
- Flask
- CherryPy
- Splunk based Code

Currently Detectable Third-Party Libraries:

- The scanner currently supports the detection of 276 different public Python libraries such as Boto, PyX, pypspider etc.
- The consolidated list of all the libraries which can be detected by PyCes can be found here:

\\Network\Example Folder\Extras\Python Lib List

- Currently in a Beta state.

Grep Based Pattern Search

- PyCes enhances the scanning by filling the gaps where the static analysis tool cannot work.
- This helps to detect vulnerabilities which occurs when certain patterns occur in a sequence or multi stage fashion.

Setup

Requirements

- Python installed on the machine (version 3.5 or later).
- Java installed on the machine.
- Python PIP correctly installed with Env path set.
- It can be checked by simply typing “pip” in the command prompt

```
C:\Users\nitinram>pip

Usage:
  pip <command> [options]

Commands:
  install          Install packages.
  download         Download packages.
  uninstall        Uninstall packages.
  freeze           Output installed packages in requirements format.
  list             List installed packages.
  show             Show information about installed packages.
  check            Verify installed packages have compatible dependencies.
  search           Search PyPI for packages.
  wheel            Build wheels from your requirements.
  hash             Compute hashes of package archives.
  completion       A helper command used for command completion.
  help             Show help for commands.
```

- Note: Make sure you don't confuse Python PIP with Pearl's PIP.

Installing Dependencies & Rulesets

Dependencies:

- Dependencies can be installed by running the “First_Run.bat” file provided along with the tool.

Note: If PyCes is installed in windows environment. Please ensure top unzip the PyCes folder in a location with no spaces in the path. This is required as PyCes uses an active shell which interprets path spaces as arguments.

Incorrect:

```
C:\Example Folder\PyCes
```

Correct:

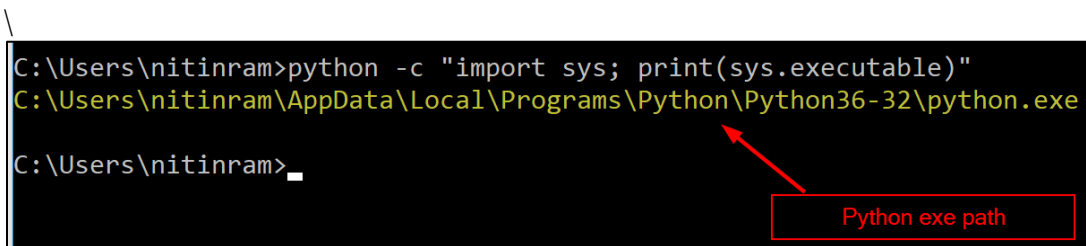
```
C:\Example_Folder\PyCes
```

Rulesets (Optional)

- Custom bandit rulesets must be placed where Bandit can pick it up to execute it.
- Step 1:** Identify the location where python is currently installed in the system.

Type the following command into the command prompt:

```
python -c "import sys; print(sys.executable)"
```



```
C:\Users\nitinram>python -c "import sys; print(sys.executable)"
C:\Users\nitinram\AppData\Local\Programs\Python\Python36-32\python.exe
C:\Users\nitinram>_
```

- Step 2:** Now that we located Python path:
Replace the call.py file found in the “First_Run” folder at the following relative path:

```
\Python\Python36-32\Lib\site-packages\bandit\blacklists
```
- That’s it, the scanner will automatically pick the file from here and run the ruleset.

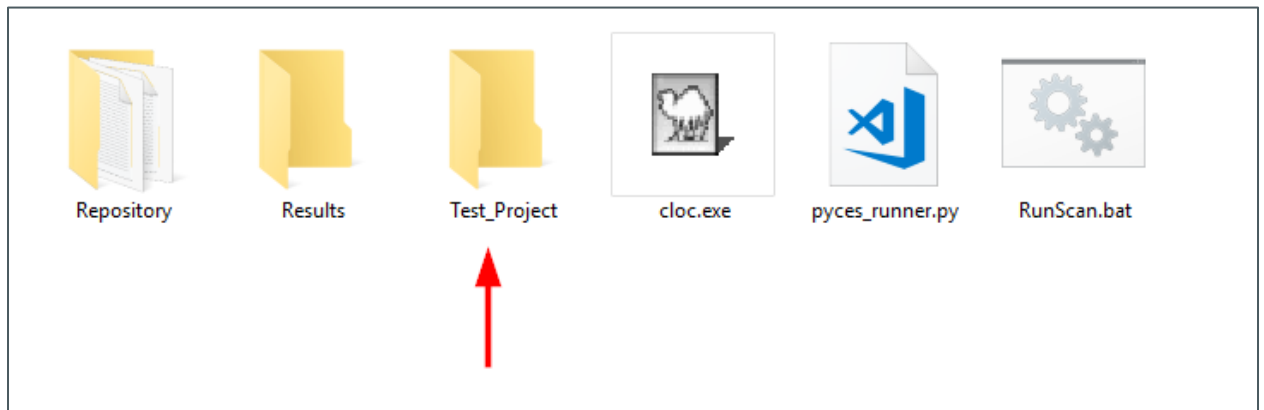
Running the Scanner

1. Scanning Modes:

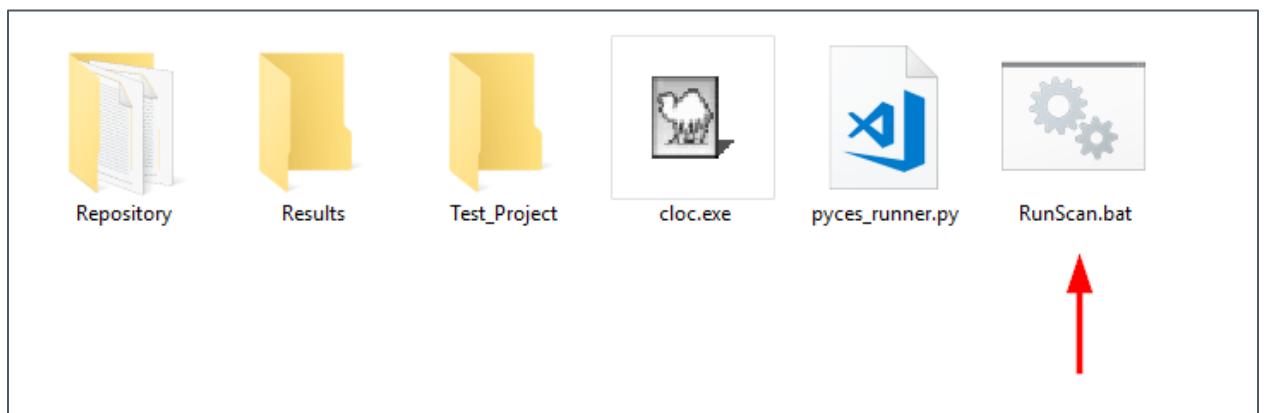
- **Normal:** Runs the scanner at normal severity. Recommended mode of setting.
- **High:** Runs with higher heuristic sensitivity. May lead to redundant findings or higher false positives

2. Running the Scanner:

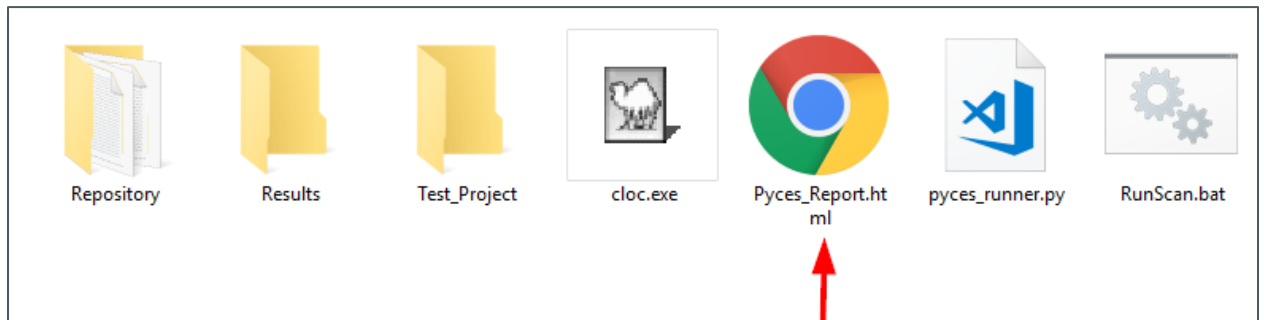
- Step 1: Place the Python Source code to be tested in the “Test Folder”.



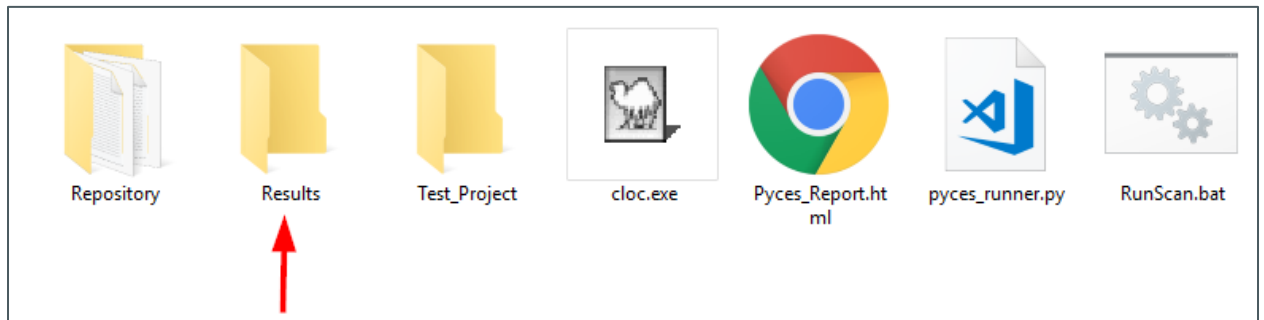
- Step 2: Double click the runner.bat file.



- Collect the “Pyces_Report.html” from the same folder.



- Backups of previous PyCes scans run can be found at the “Results” folder. The older reports are ordered by the date the scan was run on.



Note: On the 1st run, the tool will connect over network to update its databases. This will increase scanning time for that scan.

Special Cases

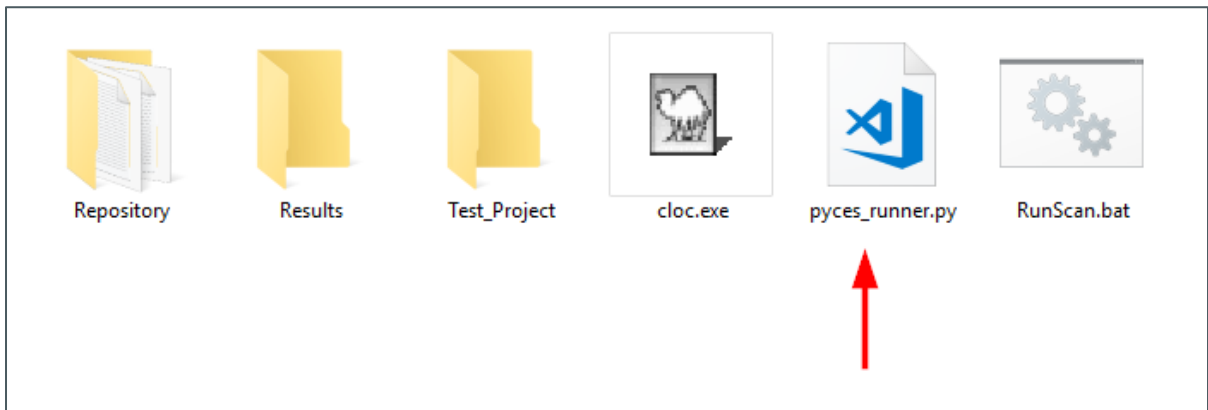
1. Network Backup:

In case there is requirement to save the scan results on a network drive for the purposes of backup. The following steps are to be done:

Step 1: Select the path of the network drive where you want to store a backup of the scan files.

```
\\Network\Example_Folder\Extras\Backup
```

Step 2: Open the pyces_runner.py file.



Step 3: Uncomment the line 532 and put the path of the network drive in line 531 within the quotes (while maintaining python alignment, of course 😊) :

```
529
530     #Activate network share here
531     ntw_path = "Put Path of network drive here"
532     #network_backup(ntw_path)
533
```

Note: When putting paths in a windows system, make sure the metacharacters are escaped.

Suppose the path you wish to provide is:

```
\\Network\Example_Folder\Extras\Backup
```

The path should be provided on line 531 with the '\' escaped with another '\' as follows:

```
\\\\Network\\Example_Folder\\Extras\\Backup
```

Example:

```
529
530     #Activate network share here
531     ntw_path = "\\\\\\Example_Folder\\Extras\\Backup"
532     network_backup(ntw_path)
533
```

Developer

Nitin Ramesh

Ethical Hacker/Python Developer with flair for Hacking and a Sloth for a spirit animal.

Website: nitinramesh.wordpress.com

Email: nitin.n.ramesh@gmail.com