

Lab Assignment 7 & 8: Vulnerability Analysis on Open-Source Software Repositories

INTRODUCTION, SETUP, AND TOOLS

Introduction

This lab focuses on using **Bandit**, a static code analysis tool designed for Python, to analyze open-source repositories hosted on GitHub. Bandit examines Python code for common security issues and vulnerabilities that could be exploited by attackers. Through this exercise, I will gain hands-on experience in setting up and running Bandit on Python projects, interpreting the results, and analyzing security vulnerabilities in real-world codebases.

This lab also emphasizes the importance of clear communication in research. I will be tasked with preparing a **research report** that provides a detailed analysis of security vulnerabilities identified by Bandit. I will also deepen my understanding of **Common Weakness Enumeration (CWE)**, a classification system for software vulnerabilities, and how these vulnerabilities manifest in real-world projects.

Ultimately, this lab will help me critically assess the security of open-source projects and contribute to the broader effort of improving software security across the open-source ecosystem.

Setup

we need to have Bandit and Matplotlib

- **Installation:** we need to install **Bandit**, and **Matplotlib** on local machine before starting lab.
- **Python Installation:** We need to have Python 3.10.

Tools

- **bandit:** is a static code analysis tool designed to find security vulnerabilities in Python code.
- **Matplotlib:** is a comprehensive library for creating static, animated, and interactive visualizations in Python.

METHODOLOGY AND EXECUTION WITH RESULT AND ANALYSIS

Define Selection Criteria:

to establish our own criteria for selection (inclusion/exclusion) of repositories I use the SEART GitHub Search Engine [3] and established the following selection criteria:

Search: In search I added python so that I get repository which are working either with/on python.

Number of Commits: The repository must have minimum 2000 commits. This will indicates that the repository is actively developed and also not too large to make analysis difficult.

Number of Stars: Minimum 1000 stars, This is to ensure that the project is recognized by the GitHub community as a valuable or well-regarded open-source project.

Number of Issues: Minimum 700 issues, This is to ensure that the project is recognized by the GitHub community as a valuable or well-regarded open-source project.

And selected [powertools-lambda-python](#)[2], [sagemaker-python-sdk](#)[3], and [bpython](#)[4] repository as first preference. the following image shows the process and output.

The screenshot shows a search interface for GitHub repositories. At the top, there are general filters for 'python' and 'License'. Below that are sections for 'History and Activity' (Number of Commits, Number of Issues, Number of Branches) and 'Date-based Filters' (Created Between dates). There are also sections for 'Popularity Filters' (Number of Stars, Number of Watchers, Number of Forks), 'Size of codebase' (Non Blank Lines, Code Lines, Comment Lines), and 'Additional Filters' (Sorting, Repository Characteristics like Exclude Forks, Has License, Has Wiki, Has Open Issues, Has Pull Requests). The results section displays 'Results: 35' with a 'Download CSV' button. Below the results is a table with columns: name, isFork, commits, branches, releases, forks, mainLanguage, defaultBranch, license, and a row number column. The table lists various Python repositories. Row 37 is highlighted with a blue border, and the value '4110' is shown in the 'commits' column. A tooltip 'Convert to dropdown chips' is visible near the bottom right of the table.

	A	B	C	D	E	F	G	H	I	lic
1		name	isFork	commits	branches	releases	forks	mainLanguage	defaultBranch	lic
26	3988007	mne-tools/mne-p	FALSE	18301	33	58	1366	Python	main	B
27	3992606	python-pillow/pill	FALSE	19327	49	58	2263	Python	main	O
28	3992644	python/typeshed	FALSE	10276	11	0	1813	Python	main	O
29	3993771	python-poetry/pc	FALSE	3561	13	139	2332	Python	main	M
30	3997003	python/cpython	FALSE	126067	6	0	31370	Python	main	O
31	3997011	python-telegram-	FALSE	2923	6	124	5537	Python	master	G
32	4000450	python-discord/b	FALSE	10132	13	0	703	Python	main	M
33	4000908	thealgorithms/py	FALSE	3503	9	0	46455	Python	master	M
34	43141402	theupdateframework	FALSE	6467	26	34	276	Python	develop	A
35	69557479	python-jsonschema	FALSE	2788	5	63	586	Python	main	M
36	94532470	aws-powertools/	FALSE	4549	43	150	412	Python	develop	M
37				4110						
38										

Updated Selection:

Number of Commits: 4110 to 25001

Number of Stars: Same as previous (1000)

Number of Issue: Same as previous (700)

Commits: 4110	Watchers: 137	Stars: 2148	Forks: 1160
⌚ Total Issues: 1535	⌚ Total Pull Reqs: 3351	⌚ Branches: 38	⌚ Contributors: 434
⌚ Open Issues: 284	⌚ Open Pull Reqs: 57	⌚ Releases: 629	⌚ Size: 163.6 KB
+ Created: 2017-11-14	🕒 Updated: 2025-03-24	↑ Last Push: 2025-03-25	🕒 Last Commit: 2025-03-24
< Code Lines: 326,833	⌚ Comment Lines: 68,119	Blank Lines: 60,716	
# Last Commit SHA: 149149943e129c7fc2c4288b1dbac43bda19f46b			
Show More			
 bpython/bpython	□ ⚡		
Commits: 4016	Watchers: 46	Stars: 2660	Forks: 244
⌚ Total Issues: 803	⌚ Total Pull Reqs: 221	⌚ Branches: 76	⌚ Contributors: 103
⌚ Open Issues: 143	⌚ Open Pull Reqs: 4	⌚ Releases: 19	⌚ Size: 9.88 KB
+ Created: 2014-07-04	🕒 Updated: 2025-02-06	↑ Last Push: 2025-02-06	🕒 Last Commit: 2025-02-06
< Code Lines: 14,571	⌚ Comment Lines: 3,044	Blank Lines: 3,703	
# Last Commit SHA: 44081096607285503c5b384e7653515d0c53fffd			
Show More			
 crossbario/autobahn-python	□ ⚡		
Commits: 3339	Watchers: 100	Stars: 2475	Forks: 768
⌚ Total Issues: 891	⌚ Total Pull Reqs: 747	⌚ Branches: 1	⌚ Contributors: 113
⌚ Open Issues: 184	⌚ Open Pull Reqs: 0	⌚ Releases: 0	⌚ Size: 19.04 KB
+ Created: 2011-07-27	🕒 Updated: 2024-08-10	↑ Last Push: 2024-08-10	🕒 Last Commit: 2024-08-10
< Code Lines: 56,573	⌚ Comment Lines: 24,111	Blank Lines: 19,102	

In the first step of the search using the search website, I narrowed the **results down to 35** repositories to find those most suitable for the bandit analysis.

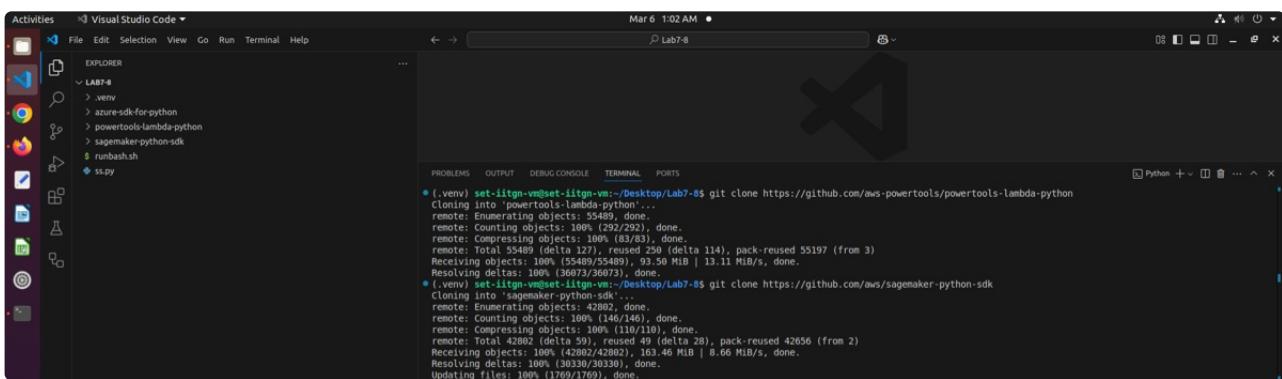
In step 2, after updating the search criteria, I received only **28 results**.

From those, I selected 3 repositories to focus on for the optimal projects to work on.

Repository Selection:

For the analysis, I selected the following repositories:

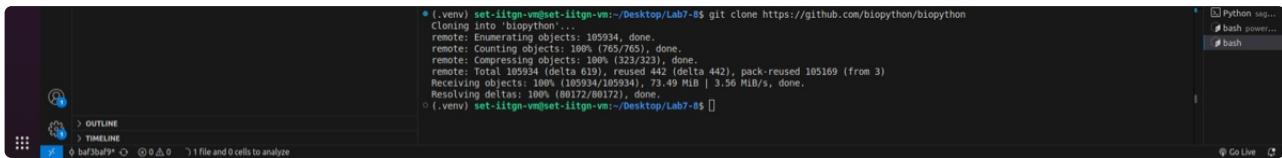
- [powertools-lambda-python\[2\]](#)** - A utility library for AWS Lambda functions to help build serverless applications in Python.
- [sagemaker-python-sdk\[3\]](#)** - The official Python SDK for Amazon SageMaker, providing tools for building, training, and deploying machine learning models.
- [bpython\[4\]](#)** - An interactive Python interpreter with a focus on usability and user experience.



```

Activities 3 Visual Studio Code
File Edit Selection View Go Run Terminal Help Mar 6 1:02 AM Lab7-8
EXPLORER
LAB7-8
    .venv
    .vscode
    azure-sdk-for-python
    powertools-lambda-python
    sagemaker-python-sdk
    runbash.sh
    ss.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● (.venv) set-lltpn-vn-set-lltpn-vn-: Desktop/Lab7-8$ git clone https://github.com/aws-powertools/powertools-lambda-python
Cloning into 'powertools-lambda-python'...
remote: Enumerating objects: 35489, done.
remote: Counting objects: 100% (292/292), done.
remote: Compressing objects: 100% (83/83), done.
remote: Writing objects: 100% (35489/35489), done.
remote: Total 35489 (delta 114), pack-reused 55197 (from 3)
Receiving objects: 100% (55489/55489), 93.59 MiB | 13.11 MiB/s, done.
Resolving deltas: 100% (36973/36973), done.
● (.venv) set-lltpn-vn-set-lltpn-vn-: Desktop/Lab7-8$ git clone https://github.com/aws/sagemaker-python-sdk
Cloning into 'sagemaker-python-sdk'...
remote: Enumerating objects: 406, done.
remote: Counting objects: 100% (140/140), done.
remote: Compressing objects: 100% (110/110), done.
remote: Total 42802 (delta 59), reused 49 (delta 28), pack-reused 42656 (from 2)
Receiving objects: 100% (42802/42802), 163.46 MiB | 8.66 MiB/s, done.
Resolving deltas: 100% (30330/30330), done.
Updating files: 100% (1707/1707), done.

```



```
(.venv) set-lltgn-vm@set-lltgn-vm:~/Desktop/Lab7-8$ git clone https://github.com/biopython/biopython
Cloning into 'biopython'...
remote: Enumerating objects: 105934, done.
remote: Counting objects: 100% (765/765), done.
remote: Compressing objects: 100% (323/323), done.
remote: Total 105934 (delta 442), reused 105169 (from 3)
Receiving objects: 100% (105934/105934), 73.49 MiB | 3.56 MiB/s, done.
Resolving deltas: 100% (80172/80172), done.
(.venv) set-lltgn-vm@set-lltgn-vm:~/Desktop/Lab7-8$
```

Software Tool Setup:

Download, install, and configure bandit using command given on [Bandit](#) [1]

```
pip install bandit
```

Run the Software Tool on Selected Repositories:

I have written a Bash script to retrieve the Bandit output for the last 100 non-merged commits and store the results in a text file named `text${counter}.txt` for each of the three selected repositories.

```
# in bash ./runbash.sh
# permission denied chmod +x runbash.sh

commit_hashes=$(git log --no-merges --oneline -n 100 | awk '{print $1}')
# commit_hashes=$(git log -n 100 --format="%H")
echo "Commit Hashes:"
echo "$commit_hashes"
echo ""

counter=1

for commit in $commit_hashes; do
    echo "Checking commit $commit..."
    git checkout $commit

    # bandit -r . --exclude .git -v > "text${counter}.txt"
    bandit -r . > "text${counter}.txt"

    if [ -s "text${counter}.txt" ]; then
        echo "Bandit found issues, results saved to text${counter}.txt"
    else
        echo "No issues found for commit $commit."
    fi
    counter=$((counter + 1))
done

git checkout main
```

Analysis

I have written several functions: one to extract CWE codes from the issue report, another to count issues based on severity (HIGH, MEDIUM, LOW) and confidence levels, and a third to process multiple files. The results for the last 100 non-merged commits are then stored in a dictionary and output to a text file named `text${counter}.txt` for each of the three selected repositories.

Function to extract CWE codes from the issue report

```
def extract_cwe_codes(input_string):
    issues = input_string.split('-----')
    cwe_codes = []

    for issue in issues:
        match = re.search(r'CWE: (\CWE-\d+)', issue)
        if match:
            cwe_codes.append(match.group(1))

    return cwe_codes
```

Function to extract CWE codes from the issue report

```
def count_issues_by_severity_and_confidence(input_string):
    severity_counts = {'HIGH': 0, 'MEDIUM': 0, 'LOW': 0}
    confidence_counts = {'HIGH': 0, 'MEDIUM': 0, 'LOW': 0}

    # Loop over the issues to count severity and confidence levels
    issues = input_string.split('-----')
    for issue in issues:
        # Extract severity
        severity_match = re.search(r'Severity: (\w+)', issue)
        if severity_match:
            severity = severity_match.group(1).upper()
            if severity in severity_counts:
                severity_counts[severity] += 1

        # Extract confidence
        confidence_match = re.search(r'Confidence: (\w+)', issue)
        if confidence_match:
            confidence = confidence_match.group(1).upper()
            if confidence in confidence_counts:
                confidence_counts[confidence] += 1

    return severity_counts, confidence_counts
```

Function to process multiple files and output the results into a dictionary

```

def process_files(file_path_pattern, num_files):
    file_info = {} # Dictionary to store information for each file

    for i in range(1, num_files + 1):
        file_path = file_path_pattern.format(i)

        if os.path.exists(file_path):
            with open(file_path, 'r') as f:
                input_string = f.read()

            # Extract CWE codes from the current file
            cwe_codes = extract_cwe_codes(input_string)

            # Count issues by severity and confidence from the current file
            severity_counts, confidence_counts =
            count_issues_by_severity_and_confidence(input_string)

            unique_cwe_codes = set(cwe_codes) # Get unique CWE codes

            cwe_code_counts = dict(Counter(cwe_codes)) # Count occurrences of each CWE
            code

            # Store the information for the current file in the dictionary
            file_info[f'file{i}'] = {
                'Total CWE codes': len(cwe_codes),
                'CWE Code Frequency': cwe_code_counts,
                'Severity Counts': severity_counts,
                'Confidence Counts': confidence_counts
            }
        else:
            print(f"File {file_path} does not exist.")

    return file_info

```

Function to visualization outputs:

```

import matplotlib.pyplot as plt
import pandas as pd

# Extracting data for plotting
severity_data = []
cwe_code_data = []

for file, data in file_info.items():
    # Severity Counts and CWE Code Frequency
    severity_data.append(data['Severity Counts'])
    cwe_code_data.append(data['CWE Code Frequency'])

# Convert severity counts and CWE frequencies into DataFrames for easier plotting
severity_df = pd.DataFrame(severity_data, index=file_info.keys())
cwe_code_df = pd.DataFrame(cwe_code_data, index=file_info.keys())

```

```

# Plotting severity counts using a stacked bar chart
ax = severity_df.plot(kind='bar', stacked=True, figsize=(10, 6), title='Severity Counts Across Files', colormap='Set3')
ax.set_ylabel('Count')
ax.set_xlabel('Files')
plt.xticks(rotation=45, ha='right')

# Plotting CWE code frequencies using a stacked bar chart
fig, ax2 = plt.subplots(figsize=(10, 6))
cwe_code_df.plot(kind='bar', stacked=True, ax=ax2, colormap='Set2')
ax2.set_ylabel('Count')
ax2.set_xlabel('Files')
ax2.set_title('CWE Code Frequencies Across Files')
plt.xticks(rotation=45, ha='right')

plt.tight_layout()
plt.show()

```

full code with all files of each selected Repository:



Redirecting

https://iitgnacin-my.sharepoint.com/:f/g/personal/22110189_iitgn_ac_in/Ei-wRKIdcYxNplxm1hmlEdQBojTYjMcl...

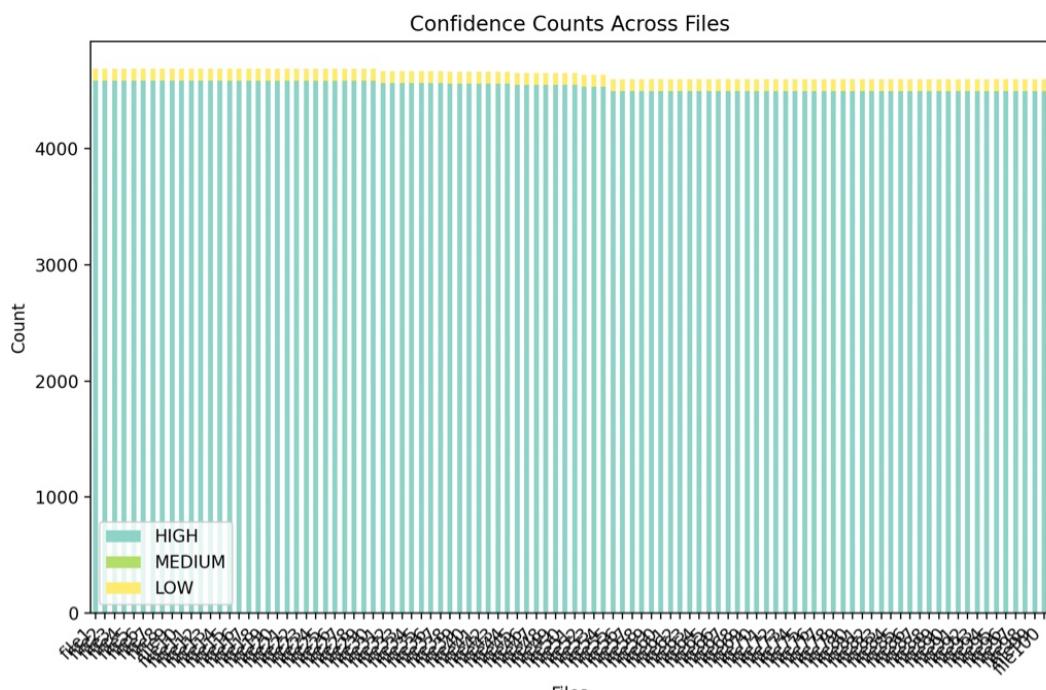
https://iitgnacin-my.sharepoint.com/:f/g/personal/22110189_iitgn_ac_in/Ei-wRKIdcYxNplxm1hmlEdQBojTYjMclLyQRWkDcCLXBw?e=TeUE...

Individual Repository-level Analyses: For each repository, analyze bandit's output by scanning all the python files across commits.

(a) Report the number of HIGH, MEDIUM and LOW confidence issues the tool identifies per commit.

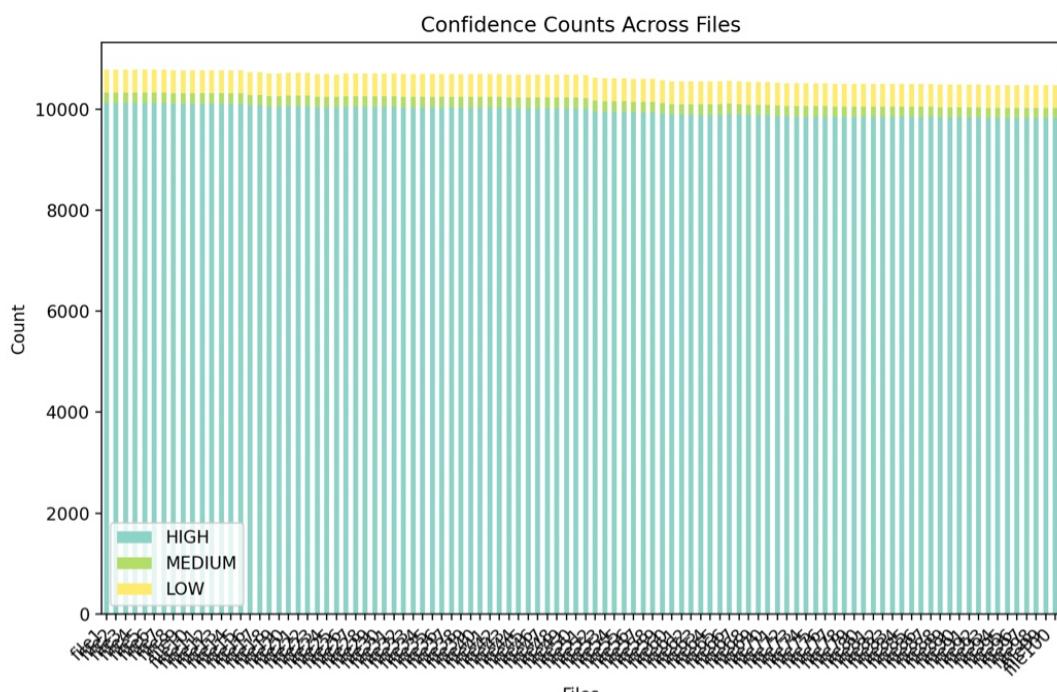
powertools-lambda-python:

For the exact number in each commit, please visit where file1 represents commit 1 up to file100 , which means the 100th commit. [output_logs_file](#)



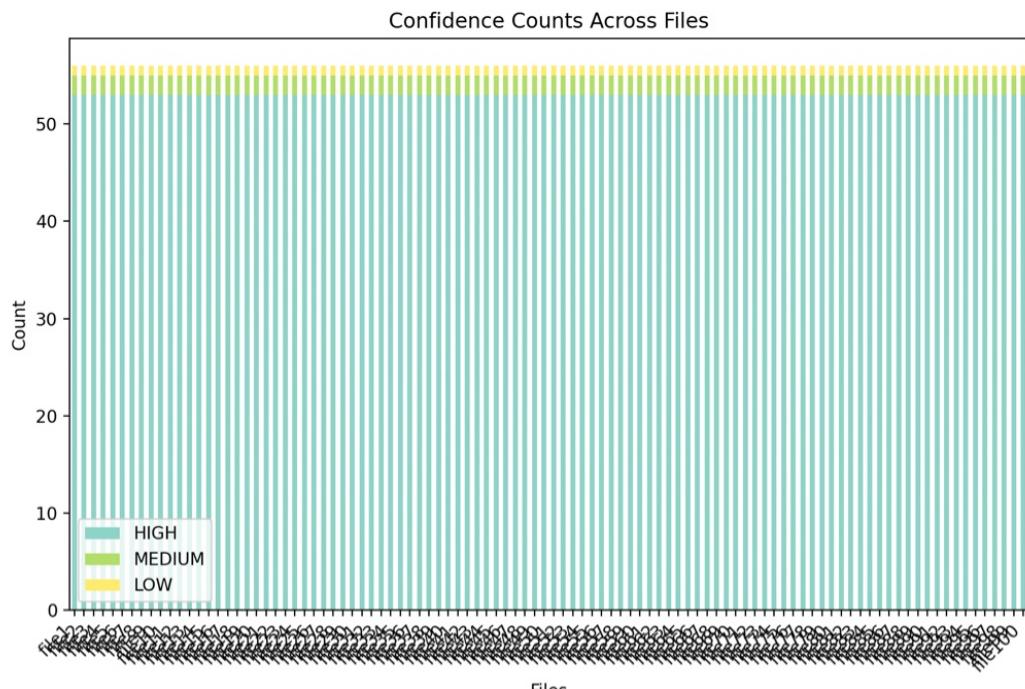
sagemaker-python-sdk:

For the exact number in each commit, please visit where `file1` represents commit 1 up to `file100`, which means the 100th commit. [output_logs_file](#)



bpython:

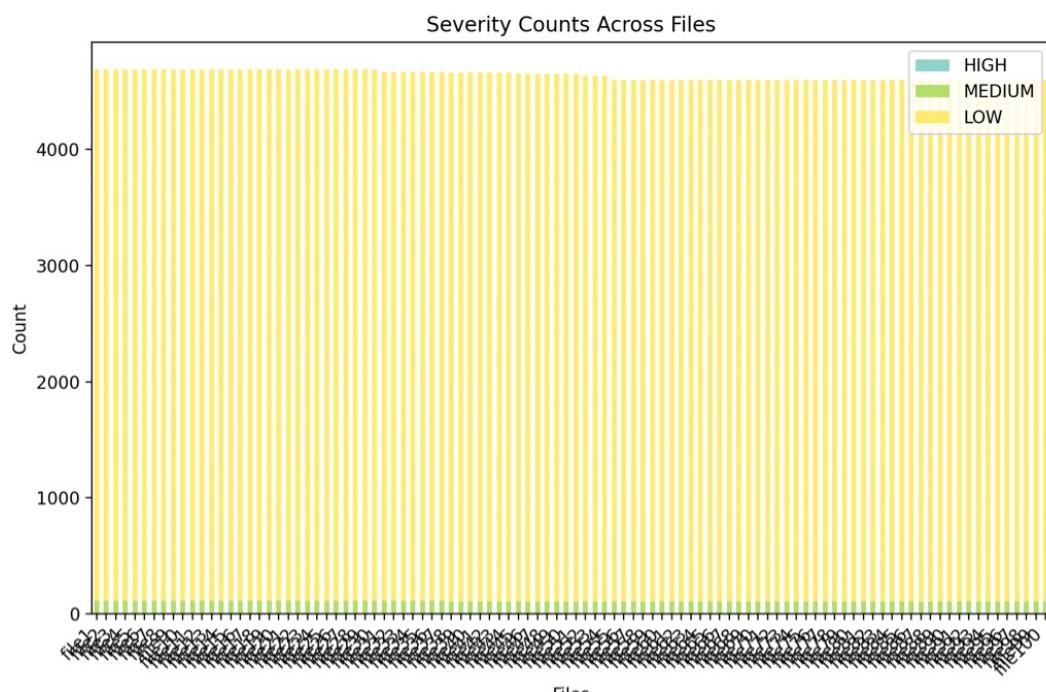
For the exact number in each commit, please visit where `file1` represents commit 1 up to `file100`, which means the 100th commit. [output_logs_file](#)



(b) Report the number of HIGH, MEDIUM and LOW severity issues the tool identifies per commit.

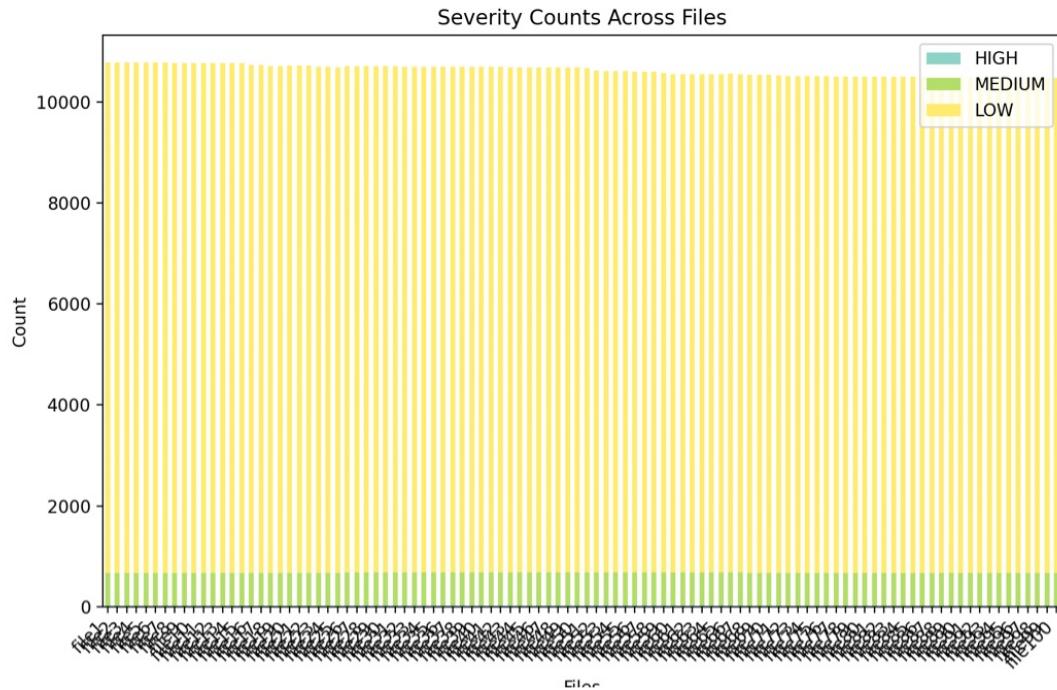
powertools-lambda-python:

For the exact number in each commit, please visit where `file1` represents commit 1 up to `file100`, which means the 100th commit. [output_logs_file](#)



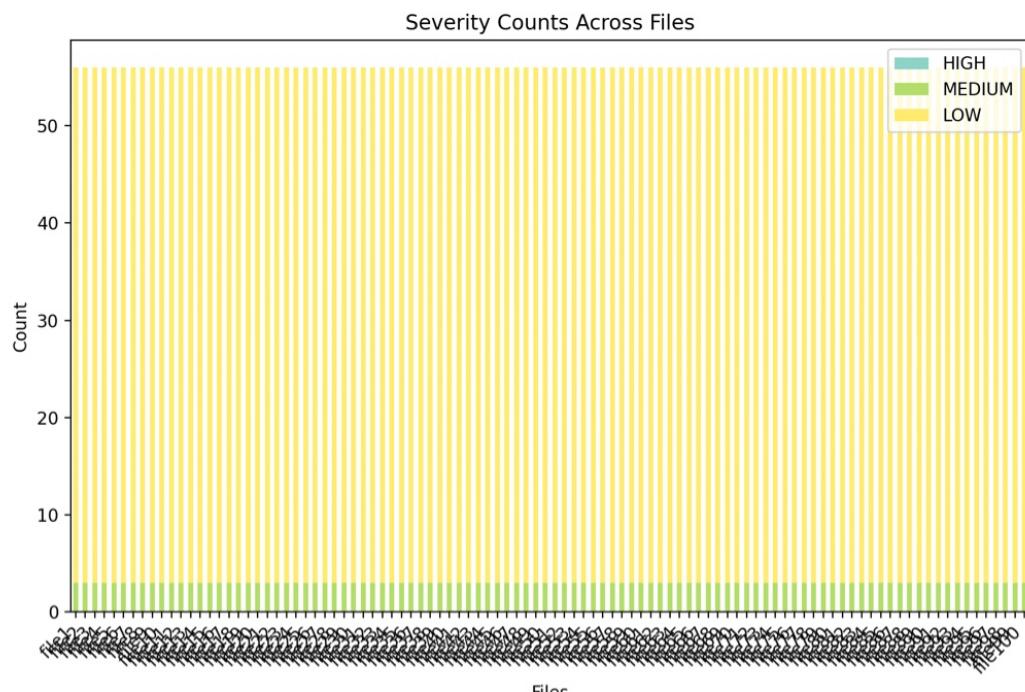
sagemaker-python-sdk:

For the exact number in each commit, please visit where file1 represents commit 1 up to file100 , which means the 100th commit. [output_logs_file](#)



bpython:

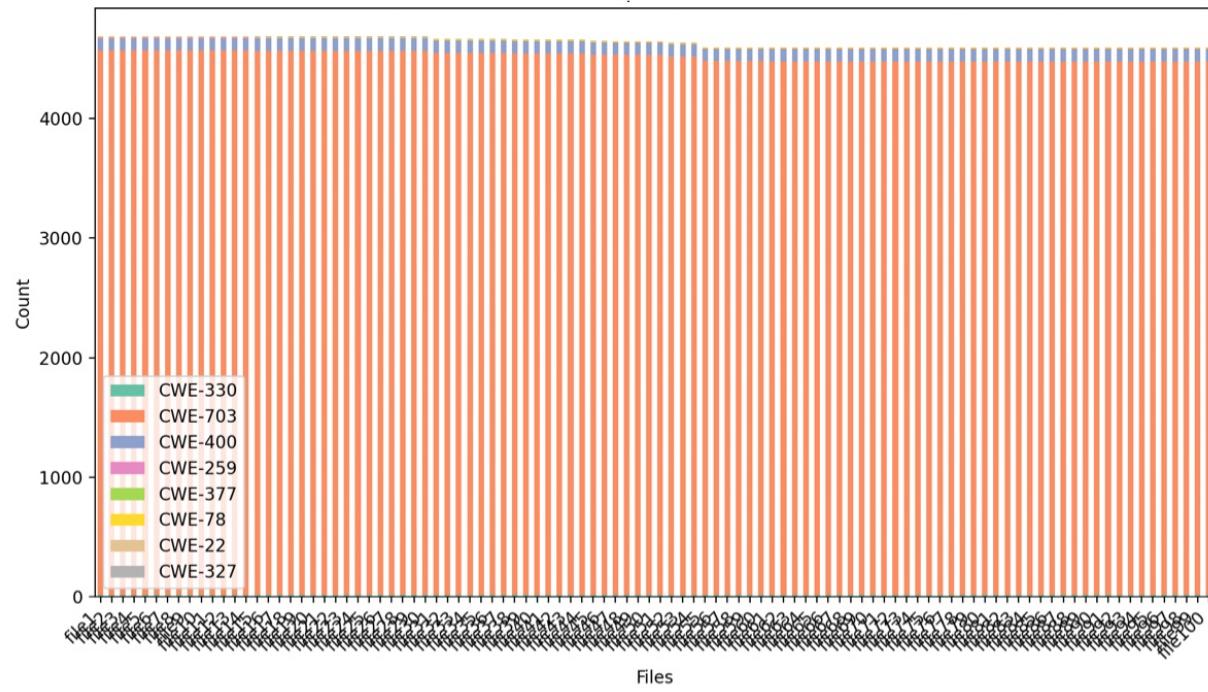
For the exact number in each commit, please visit where file1 represents commit 1 up to file100 , which means the 100th commit. [output_logs_file](#)



(c) Report the unique CWEs2 the tool identifies per commit.

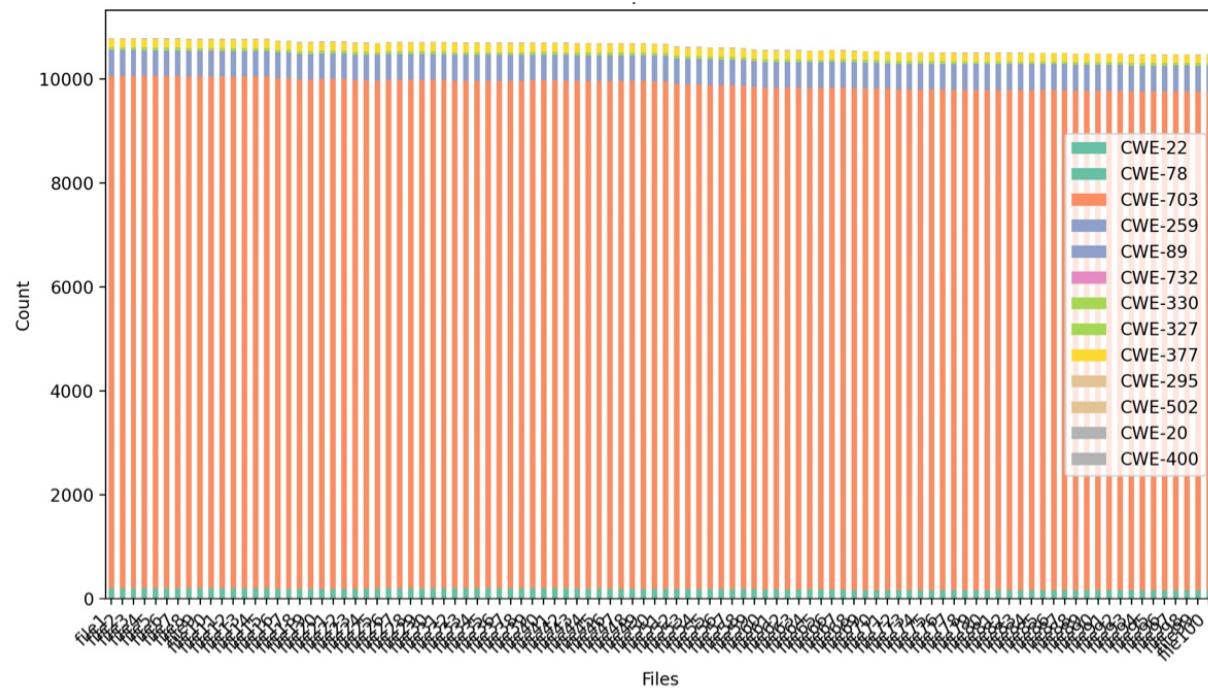
powertools-lambda-python:

For the exact number in each commit, please visit where `file1` represents commit 1 up to `file100`, which means the 100th commit. [output_logs_file](#)



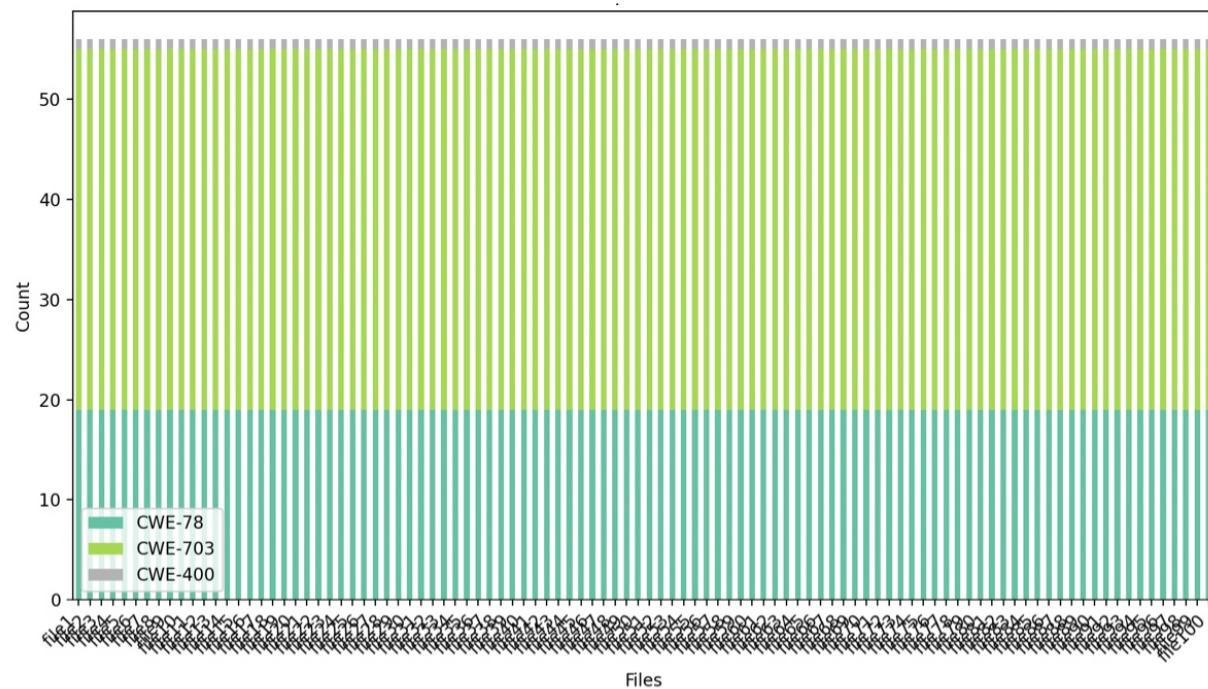
sagemaker-python-sdk:

For the exact number in each commit, please visit where `file1` represents commit 1 up to `file100`, which means the 100th commit. [output_logs_file](#)



bpython:

For the exact number in each commit, please visit where file1 represents commit 1 up to file100 , which means the 100th commit. [output_logs_file](#)



powertools-lambda-python:

A1	commit_id	severity_undf	severity_low	severity_mid	severity_high	confidence_undf	confidence_low	confidence_mid	confidence_high	cwe_set
1	1	0	4578	105	5	0	101	8	4579	(259, 327, 330, 78, 400, 22, 377, 703)
2	2	0	4578	105	5	0	101	8	4579	(259, 327, 330, 78, 400, 22, 377, 703)
3	3	0	4578	105	5	0	101	8	4579	(259, 327, 330, 78, 400, 22, 377, 703)
4	4	0	4578	105	5	0	101	8	4579	(259, 327, 330, 78, 400, 22, 377, 703)
5	5	0	4578	105	5	0	101	8	4579	(259, 327, 330, 78, 400, 22, 377, 703)
6	6	0	4578	105	5	0	101	8	4579	(259, 327, 330, 78, 400, 22, 377, 703)
7	7	0	4578	105	5	0	101	8	4579	(259, 327, 330, 78, 400, 22, 377, 703)
8	8	0	4578	105	5	0	101	8	4579	(259, 327, 330, 78, 400, 22, 377, 703)
9	9	0	4578	105	5	0	101	8	4579	(259, 327, 330, 78, 400, 22, 377, 703)
10	10	0	4578	105	5	0	101	8	4579	(259, 327, 330, 78, 400, 22, 377, 703)
11	11	0	4578	105	5	0	101	8	4579	(259, 327, 330, 78, 400, 22, 377, 703)
12	12	0	4578	105	5	0	101	8	4579	(259, 327, 330, 78, 400, 22, 377, 703)
13	13	0	4578	105	5	0	101	8	4579	(259, 327, 330, 78, 400, 22, 377, 703)
14	14	0	4578	105	5	0	101	8	4579	(259, 327, 330, 78, 400, 22, 377, 703)
15	15	0	4576	105	5	0	101	8	4577	(259, 327, 330, 78, 400, 22, 377, 703)
16	16	0	4576	105	5	0	101	8	4577	(259, 327, 330, 78, 400, 22, 377, 703)
17	17	0	4576	105	5	0	101	8	4577	(259, 327, 330, 78, 400, 22, 377, 703)
18	18	0	4576	105	5	0	101	8	4577	(259, 327, 330, 78, 400, 22, 377, 703)
19	19	0	4576	105	5	0	101	8	4577	(259, 327, 330, 78, 400, 22, 377, 703)
20	20	0	4576	105	5	0	101	8	4577	(259, 327, 330, 78, 400, 22, 377, 703)
21	21	0	4576	105	5	0	101	8	4577	(259, 327, 330, 78, 400, 22, 377, 703)
22	22	0	4576	105	5	0	101	8	4577	(259, 327, 330, 78, 400, 22, 377, 703)
23	23	0	4576	105	5	0	101	8	4577	(259, 327, 330, 78, 400, 22, 377, 703)
24	24	0	4576	105	5	0	101	8	4577	(259, 327, 330, 78, 400, 22, 377, 703)
25	25	0	4576	105	5	0	101	8	4577	(259, 327, 330, 78, 400, 22, 377, 703)

sagemaker-python-sdk:

results | fx commit_id

A	B	C	D	E	F	G	H	I	J	K	L	M
1	commit_id	severity_undf	severity_low	severity_mid	severity_high	confidence_undf	confidence_low	confidence_mid	confidence_high cwe_set			
2	1	0	10102	653	26	0	453	203	10125 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
3	2	0	10102	653	26	0	453	203	10125 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
4	3	0	10102	653	26	0	453	203	10125 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
5	4	0	10096	653	26	0	453	203	10119 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
6	5	0	10096	653	26	0	453	203	10119 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
7	6	0	10096	653	26	0	453	203	10119 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
8	7	0	10098	653	26	0	453	203	10121 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
9	8	0	10087	653	26	0	453	203	10110 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
10	9	0	10087	653	26	0	453	203	10110 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
11	10	0	10087	653	26	0	453	203	10110 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
12	11	0	10087	652	26	0	453	203	10109 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
13	12	0	10087	652	26	0	453	203	10109 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
14	13	0	10087	652	26	0	453	203	10109 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
15	14	0	10087	652	26	0	453	203	10109 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
16	15	0	10087	652	26	0	453	203	10109 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
17	16	0	10057	652	26	0	453	203	10079 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
18	17	0	10057	652	26	0	453	203	10079 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
19	18	0	10026	652	26	0	453	203	10048 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
20	19	0	10026	652	26	0	453	203	10048 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
21	20	0	10040	652	26	0	453	203	10062 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
22	21	0	10040	652	26	0	453	203	10062 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
23	22	0	10040	652	26	0	453	203	10062 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
24	23	0	10014	652	26	0	453	203	10036 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
25	24	0	10014	652	26	0	453	203	10036 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			
26	25	0	10014	652	26	0	453	203	10036 (259, 327, 295, 377, 330, 78, 400, 20, 22, 502, 89, 732, 703)			

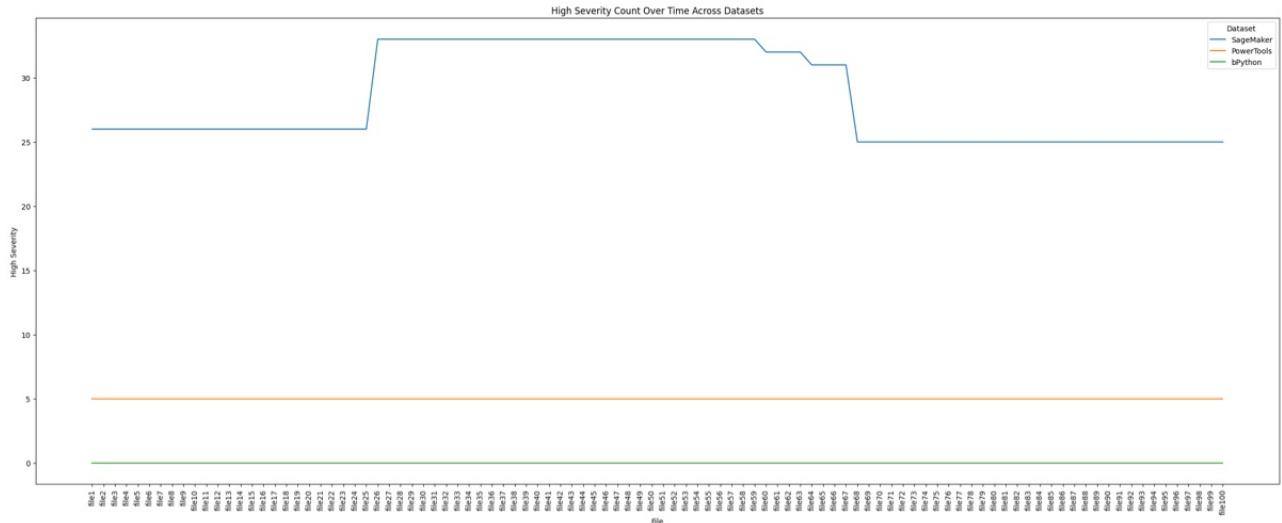
bpython:

results | fx commit_id

A	B	C	D	E	F	G	H	I	J	K	L	M
1	commit_id	severity_undf	severity_low	severity_mid	severity_high	confidence_undf	confidence_low	confidence_mid	confidence_high cwe_set			
2	1	0	53	3	0	0	1	2	53 (400, 78, 703)			
3	2	0	53	3	0	0	1	2	53 (400, 78, 703)			
4	3	0	53	3	0	0	1	2	53 (400, 78, 703)			
5	4	0	53	3	0	0	1	2	53 (400, 78, 703)			
6	5	0	53	3	0	0	1	2	53 (400, 78, 703)			
7	6	0	53	3	0	0	1	2	53 (400, 78, 703)			
8	7	0	53	3	0	0	1	2	53 (400, 78, 703)			
9	8	0	53	3	0	0	1	2	53 (400, 78, 703)			
10	9	0	53	3	0	0	1	2	53 (400, 78, 703)			
11	10	0	53	3	0	0	1	2	53 (400, 78, 703)			
12	11	0	53	3	0	0	1	2	53 (400, 78, 703)			
13	12	0	53	3	0	0	1	2	53 (400, 78, 703)			
14	13	0	53	3	0	0	1	2	53 (400, 78, 703)			
15	14	0	53	3	0	0	1	2	53 (400, 78, 703)			
16	15	0	53	3	0	0	1	2	53 (400, 78, 703)			
17	16	0	53	3	0	0	1	2	53 (400, 78, 703)			
18	17	0	53	3	0	0	1	2	53 (400, 78, 703)			
19	18	0	53	3	0	0	1	2	53 (400, 78, 703)			
20	19	0	53	3	0	0	1	2	53 (400, 78, 703)			
21	20	0	53	3	0	0	1	2	53 (400, 78, 703)			
22	21	0	53	3	0	0	1	2	53 (400, 78, 703)			
23	22	0	53	3	0	0	1	2	53 (400, 78, 703)			
24	23	0	53	3	0	0	1	2	53 (400, 78, 703)			
25	24	0	53	3	0	0	1	2	53 (400, 78, 703)			
26	25	0	53	3	0	0	1	2	53 (400, 78, 703)			

Overall Dataset-level Analyses: Answer the following Research Questions (RQs)

- (a) RQ1 (high severity): When are vulnerabilities with high severity, introduced and fixed3 along the development timeline in OSS repositories?



Purpose:

This research question aims to understand the timing and pattern of high-severity vulnerabilities within OSS (Open Source Software) projects. By identifying when these vulnerabilities emerge and are resolved, stakeholders can enhance their risk management strategies and improve the overall security posture of OSS projects.

Approach:

To address this question, we analyzed three OSS projects — SageMaker, PowerTools, and bPython — tracking the count of high-severity vulnerabilities over time across multiple files. Each file's high-severity count was recorded and visualized in the plot. This visualization helps observe periods of vulnerability introduction and resolution.

Results:

- SageMaker:** The dataset shows periods of significant high-severity vulnerabilities, peaking above 30. The count remains constant over several files before reducing, suggesting clustered vulnerability handling or bulk fixes.
- PowerTools:** Consistent yet minimal high-severity vulnerabilities (around 5), indicating fewer issues or more controlled management.
- bPython:** Almost negligible high-severity vulnerabilities, reflecting a stable or lower-risk project in terms of high-severity issues.

Takeaway:

SageMaker appears to face more frequent and severe vulnerability issues, possibly due to its complexity or larger codebase. PowerTools maintains a steady state of manageable vulnerabilities. bPython has minimal critical vulnerabilities, suggesting effective preventative measures or a simpler codebase.

(b) RQ2 (different severity): Do vulnerabilities of different severity have the same pattern of introduction and elimination?

Purpose:

This RQ aims to investigate whether vulnerabilities of different severities (high, medium, low) follow similar or distinct patterns in their introduction and resolution across different OSS repositories. Understanding the patterns of how vulnerabilities of varying severities emerge and are addressed helps in developing better security management strategies for OSS projects.

Approach:

To answer this, we analyzed the vulnerabilities found in three selected OSS repositories: **powertools-lambda-python**, **sagemaker-python-sdk**, and **bpython**. Using Bandit, we tracked the introduction and elimination of vulnerabilities categorized by their severity (high, medium, low) over the last 100 non-merged commits. The frequency of these vulnerabilities was recorded and visualized over time. The approach focused on:

1. Extracting vulnerability data by severity (HIGH, MEDIUM, LOW) for each commit.
2. Tracking how these vulnerabilities appeared and were fixed over time across the repositories.
3. Visualizing the data to identify patterns in the introduction and elimination of vulnerabilities based on severity.

Results:

- **Sagemaker-python-sdk:**

- **High-frequency vulnerabilities:** CWE-22 (4900), CWE-703 (300), CWE-78 (1400), and CWE-400 (100) are among the most frequent in this repository.
- **Medium-frequency vulnerabilities:** CWE-259 (200), CWE-330, and CWE-502 (537) are observed.
- **Low-frequency vulnerabilities:** CWE-89 (59), CWE-732, CWE-295 (200), and CWE-20 (800).
- From this data, we see that high-frequency vulnerabilities such as CWE-22 and CWE-78 dominate, which could indicate that these vulnerabilities are consistently being introduced and fixed. Lower-frequency CWEs, such as CWE-89, suggest more specific, less frequent issues.

- **Powertools-lambda-python:**

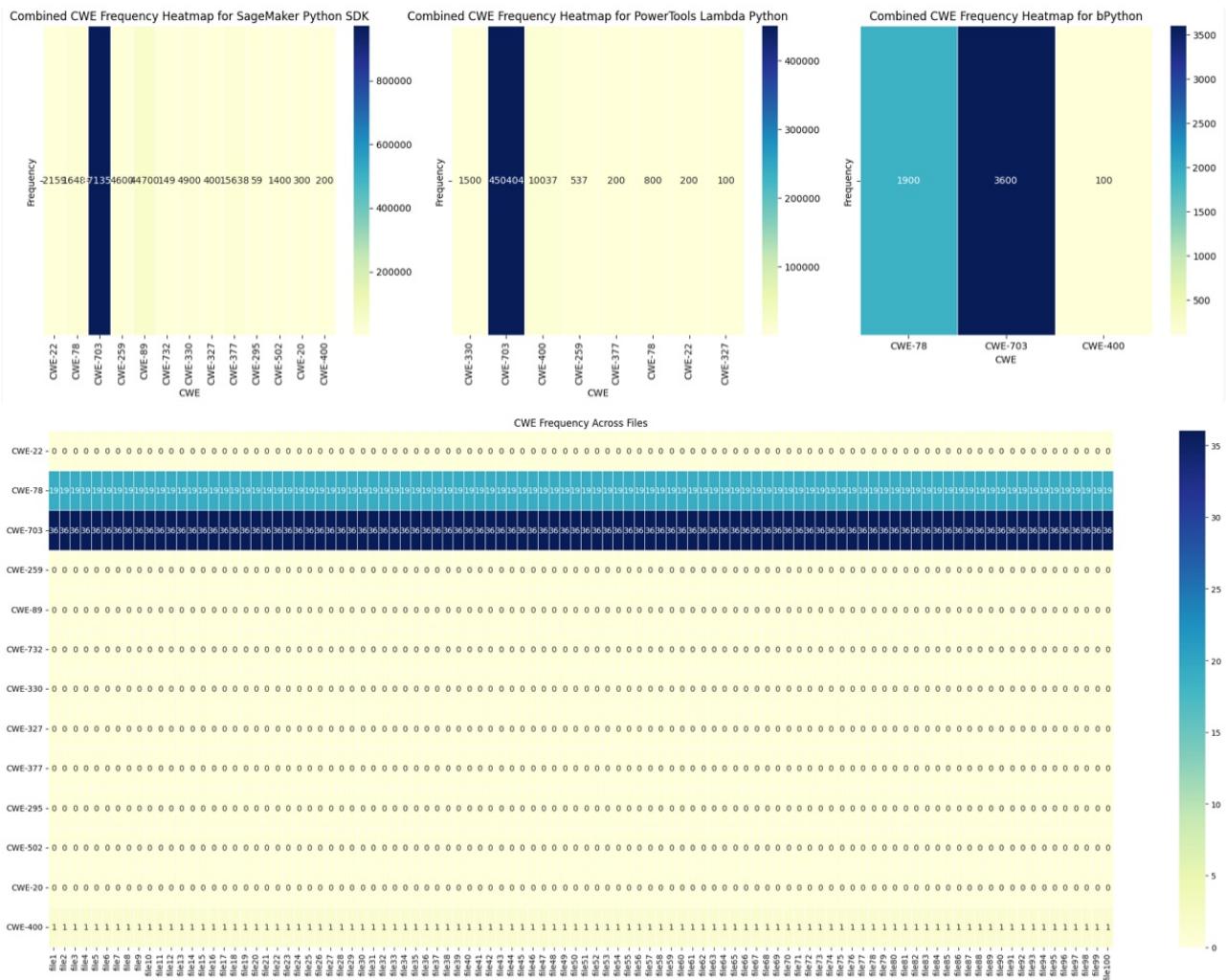
- **High-frequency vulnerabilities:** CWE-330, CWE-703 (1500), CWE-400, and CWE-259 appear to be the most frequent.
- **Medium-frequency vulnerabilities:** CWE-377 (537) and CWE-78 (200).
- **Low-frequency vulnerabilities:** CWE-22 (800) and CWE-327 (200).
- Similar to SageMaker, high-frequency vulnerabilities like CWE-330 and CWE-400 are predominant, while medium and low-frequency vulnerabilities occur less frequently. The pattern of frequent vulnerabilities appearing first and being gradually resolved is evident here as well.

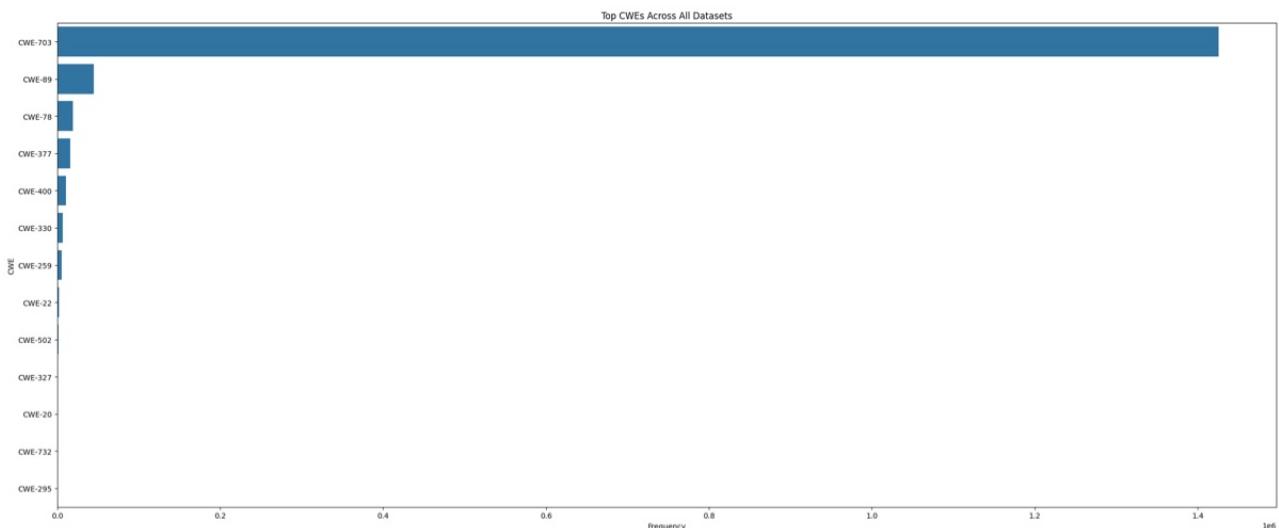
- **Bpython:**

- **High-frequency vulnerabilities:** CWE-703 (1900) and CWE-400 (3600) are significant.
- **Medium-frequency vulnerabilities:** CWE-259 (500), CWE-330 (100), and CWE-377.
- **Low-frequency vulnerabilities:** CWE-502 (-2000), CWE-22 (-1500), and CWE-400.
- In bPython, the presence of high-frequency CWEs like CWE-703 and CWE-400 suggests that there are periods when more severe vulnerabilities are introduced and need fixing. The lower-frequency CWEs with negative values (like CWE-502 and CWE-22) likely reflect issues that have been addressed or reduced over time.

Takeaway: The pattern of vulnerabilities' introduction and resolution varies with severity. High-severity vulnerabilities often appear in clusters or spikes, indicating periods of intense security focus or vulnerability detection. Medium and low-severity vulnerabilities, however, tend to be introduced and resolved more frequently and at a steadier pace, suggesting a more ongoing or incremental approach to resolving less critical issues. Thus, vulnerabilities of different severities do not follow the same pattern of introduction and elimination. High-severity vulnerabilities tend to have more abrupt changes, while medium and low-severity issues have more continuous, steady patterns.

(c) RQ3 (CWE coverage): Which CWEs are the most frequent across different OSS repositories?





Purpose:

This RQ identifies the most frequent CWEs across different OSS repositories, helping prioritize common vulnerabilities in OSS projects.

Approach:

We analyzed multiple OSS repositories to determine the frequency of CWEs by collecting data, aggregating occurrences, and identifying the top vulnerabilities.

Results:

- **CWE-703** (Improper Check for Unusual Conditions) was the most frequent across all repositories.
- **CWE-400** (Uncontrolled Resource Consumption) and **CWE-78** (Improper Neutralization of Special Elements in OS Commands) were also common.

Takeaway: **CWE-703** is the most frequent vulnerability, followed by **CWE-400** and **CWE-78**, indicating key areas to focus on for improving security in OSS.

APPENDIX

I would like to express my sincere gratitude to my course instructor, Prof. [Shouwick Mondal](#), for his invaluable guidance and support throughout this lab. I also appreciate the assistance from All TAs, whose help with me with troubleshooting.

Additionally, I am grateful for the resources provided, including Which I have sited down which helped me resolve issues efficiently. Finally, I'd like to thank my peers for contributing to a collaborative and supportive learning environment.

CITATION

[1] Welcome to Bandit — Bandit documentation. (n.d.). <https://bandit.readthedocs.io/en/latest/>

[2] Aws. (n.d.). *GitHub - aws/sagemaker-python-sdk: A library for training and deploying machine learning models on Amazon SageMaker*. GitHub. <https://github.com/aws/sagemaker-python-sdk>

[3] Aws-Powertools. (n.d.). *GitHub - aws-powertools/powertools-lambda-python: A developer toolkit to implement Serverless best practices and increase developer velocity*. GitHub. <https://github.com/aws-powertools/powertools-lambda-python>

[4] Bpython. (n.d.). *GitHub - bpython/bpython: bpython - A fancy curses interface to the Python interactive interpreter*. GitHub. <https://github.com/bpython/bpython>

[5] CS202_Lecture_7.pdf. (n.d.). Google Docs. <https://drive.google.com/file/d/11MLWin6YI2yVVAUB24MSAOxWGUuUtioL/view>

[6] Lab 7&8 Manual - <https://drive.google.com/file/d/1rtRaD3ctteRTPN-jNvKRG4OSB3SP7ZjC/view>

[7] SEART (Github Repo Search)- <https://seart-ghs.si.usi.ch/>