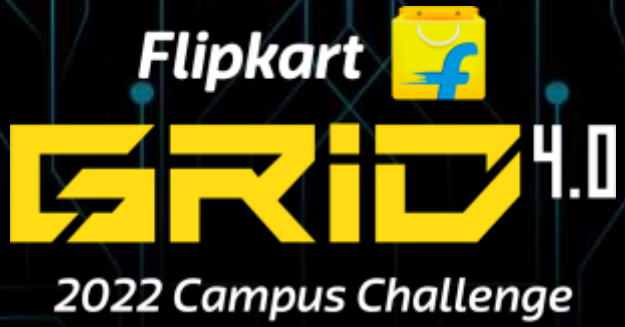


Infosec Challenge

Open Source Software (OSS) Security Inspector

Team Name: VI Team

Institute Name: Graphic Era (Deemed to be University)

Team members details

Team Name	**VI Team**		
Institute Name	**Graphic Era (Deemed to be University)**		
Team Members >	**1 (Leader)**	**2**	**3**
Name	Vinay Kumar	Km Indu	
Batch	2019-2023	2019-2023	

Glossary

Abbreviation	Expansion
package-info	Get information about NPM packages. Fetches the information directly from the registry.
fs	The **fs** module of **Node**.js provides useful functions to interact with the **file system**.
util	A **Node**.JS module, provides an object oriented wrapper **util** for the **NPM** API. It allow to automatically install modules for **Node**.js.
process	A simple utility to handle sub-**processes** and multi threading and tasking in a **node** driven application.
pypi-info	A simple tool to get information of **pypi** packages.
flask	**Flask** is a web framework, it's a **Python** module that lets you develop web applications easily.
naked	Naked is **a new Python command line application framework that is in development**.
requests	The **requests** module allows you to send HTTP **requests** using **Python**.
urllib	Urllib package is the **URL handling module for python**.

Use-cases

P0- Thorough check up of the files present in the repository.

P1- End to end verification of the user who have created the repository.

P2- Security check up of repository based on popularity.

Solution statement/ Proposed approach

Vulnerabilities introduced in OSS repos could be created by attackers to trick the users. We have created a tool which checks the repositories/packages thoroughly.

It performs:

1. thorough check up of the files present in the repositories, firstly extracting the code content, then matching types of files like C, C++, Java, Python, Php, JS, followed by matching with vulnerable functions that can steal data , executes certain commands, and takes control of the system and on this basis, it returns a vulnerability rating.
2. end to end security check up of the user who have created the repos on the basis of follower_count, following_count, bio, hireable, name, email,location, company, blog, twitter and it provides a user rating.
3. popularity check up of repositories with the help of stars_count, forks_count, watchers_count, issues_count and returns a popularity rating.

Lastly, our tool returns binary output in the form of Yes/No. It returns Yes when the user repo is malicious and vice-versa with the help of the ratings results obtained from the above 3 steps.

We are taking a total score of 100 and dividing the score into the ratio of 50, 30, and 20 for the three different use cases that are targeted.

- The score of 50 for the security check of code files present in the repositories depending on the number of files in the repository and the total of number of vulnerabilities(unsafe functions). We are getting a new vulnerability level by dividing calculated vulnerability level by number of files.

  We assumed, more the vulnerability level, less is the score. If vulnerability level is less than equal to 1 then 50 is added to the score, if vulnerability level is less than and equal to 2 then 40 is added to the score, if vulnerability level is less than and equal to 3 then 30 is added to the score, if vulnerability level is less than and equal to 4 then 20 is added to the score, if vulnerability level is less than and equal to 5 then 10 is added to the score, and if vulnerability level exceeds 5 then the score is 0. On this basis, we calculate a score out of 50 after a thorough check of the files present in the repository.

- The score of 30 for the security check of user profile. There are basically 10 factors for the validation of profile of a user, namely follower_count, following_count, bio, hireable, name, email, location, company, blog, twitter.

  We are giving a score of 12 for follower_count as it matters more as compared to other factors. More the followers, less the vulnerability, and high the score.

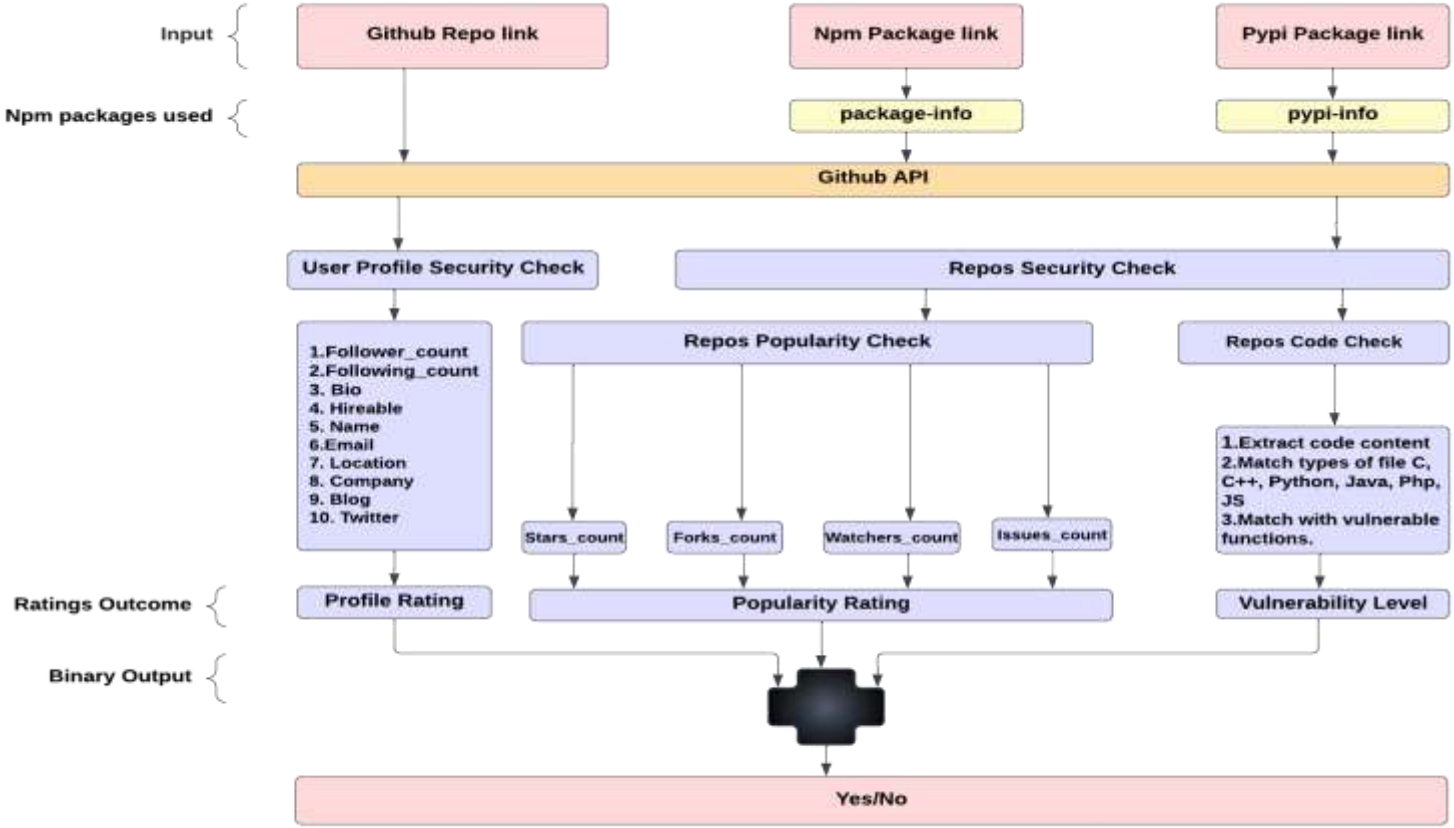
  If follower count is less than and equal to 5, we add 1 to the score, if follower count is less than and equal to 15, we add 2 to the score, if follower count is less than and equal to 25, we add 3 to the score, if follower count is less than and equal to 35, we add 4 to the score, if follower count is less than and equal to 45, we add 5 to the score, if follower count is less than and equal to 55, we add 6 to the score, if follower count is less than and equal to 65, we add 7 to the score, if follower count is less than and equal to 75, we add 8 to the score if follower count is less than and equal to 85, we add 9 to the score, if follower count is less than and equal to 95, we add 10 to the score, if follower count is less than and equal to 105, we add 11 to the score, if follower count exceeds 105 then we add 12 to the score.

  For rest of the 9 factors, we are giving a core of 2 for each, contributing a score of 16. On this basis, we calculate a score out of 30 after a thorough validation of user profile.

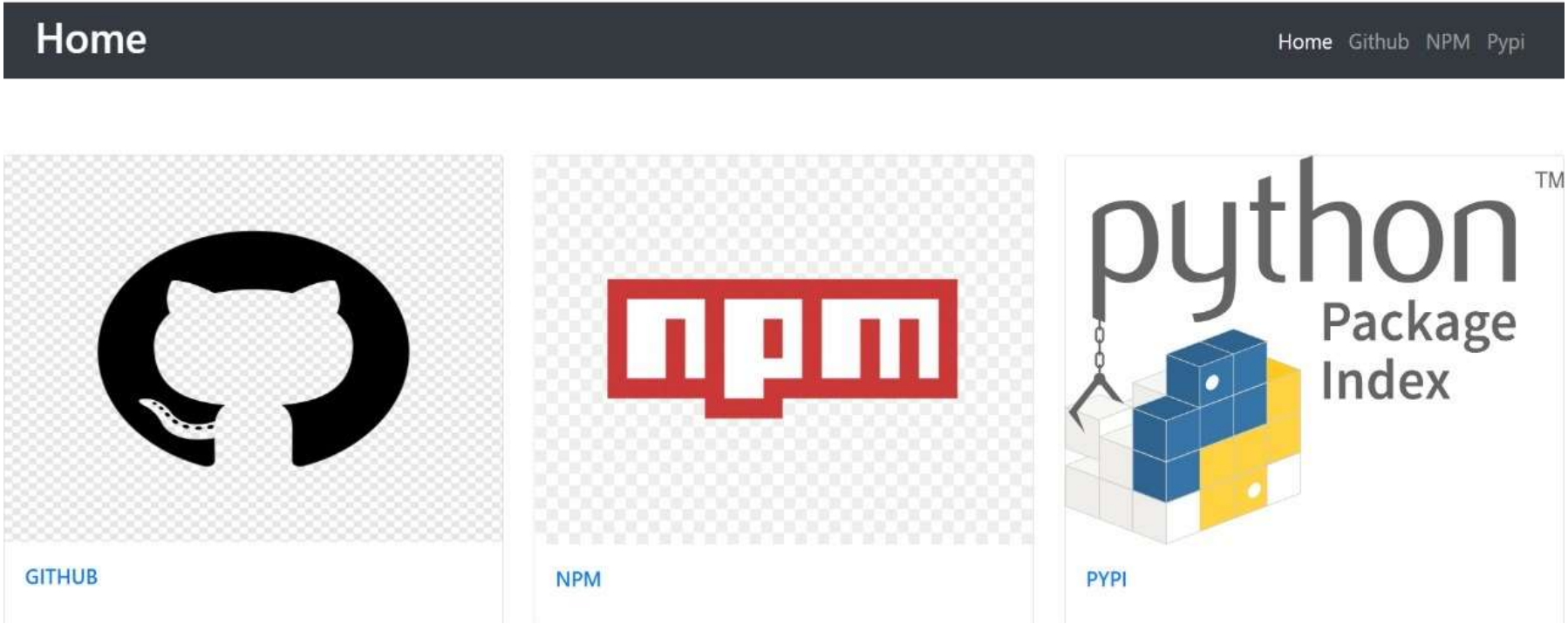
- The score of 20 is provided for the popularity check of repository. We have included 4 factors for getting popularity rating, namely stars_count, forks_count, watchers_count and issues_count. The score is the addition of stars, forks_count and watchers_count. Also, twice the count of issues is subtracted to the addition of stars, forks_count and watchers_count to give the final score of repo popularity.

Lastly, the final score of 100 depends on the score of above three cases. If the final score of the repository is less than and equal to 70 then we return a negative response towards the repository/package and if the final score is greater than 70 then we give a positive response for the repository/package.

Block diagram



Web UI


Home
Home Github NPM Pypi
GITHUB
NPM
python
Package
Index
TM
PYPI

Visualization of result

Github Profile Score
3

Repository Popularity Score
5 - not popular

User Code Vulnerability Score
0

Total Score
8

Repositories Vulnerability Levels	
30daysofdev	0
Algorithms	57
Algorithms1	0
Alien-Invasion	0
Btcoin-Price-Prediction-Project	0
career-guidance-website	616

Repositories Vulnerable Functions
gets
gets
gets
strlen
strlen
scanf
scanf
scanf
scanf

Tools and Technologies used

Operating System:
Windows

Source Code Editor:
Visual Studio Code

Languages:
Python, HTML, CSS, Node.js

Packages of NPM:

1. user-info
2. fs
3. util
4. process
5. pypi-info

Packages of Pypi:

1. flask
2. Naked
3. requests
4. urllib

Limitations

- **GitHub API**:

We have used GitHub personal access tokens here, while using this requests to the API is restricted to a maximum of 10 requests per second per user.
This is a limitation of our project which is using GitHub API to extract all the data related to a user.
An Enterprise tier of GitHub could work here since it will charge on the basis of requests taken and we can have unlimited requests.

- **Needs constant internet connection:**

Since the application is making constant calls to the GitHub API, it needs constant and fast internet connection to work properly.

Future Scope

Since the amount of data and content in open-source repos is increasing day by day, there is a huge need of vulnerability and identity scanner, now and in the future. All OSS repositories may not be maintained properly, because of which, vulnerabilities may get introduced with time. Whereas some OSS repos could be created by attackers themselves to trick the users. Our project fulfils this requirement. It checks the repositories/packages thoroughly. It performs end to end security check up of the user who have created the repos, the repositories, and the code present in the source repositories. And returns an output which says if the repository/package is a malicious or not.

Suggestions

1. We have already built a tool for the most popular OSS repos like github, npm and pypi. We can use similar approach and logic for other repositories also. For e.g., Java Repositories. Only the approach for extracting the code will be little different.

2. The processing could be very fast if we have good internet connectivity because with fast internet, we can clone the repositories speedily rather than working on API which has limited request access.