**Paper no:1**

**Automatically granted permissions in android apps**

**Authors;** Paolo Calciati, Konstantin Kuznetsov, Alessandra Gorla, Andreas Zelker

**Summary**

The meeting was held at MSR-ZOOM2 – security on 30th of July 2020 which abstract that the developers have to continuously have to update their android apps for security to keep competition in the market of play store and android although the such update do not bother the costumers or the users as android by default allows the permissions to the device unless there are major changes to the application which user have to grant permissions like get access to the gallery etc. however this thing might be difficult for the user to grant permissions which they do not want to grant because of privacy , some apps might grants permissions automatically to the device but it might be major problem for the privacy insecurity to the users this behavior got even worse by permission groups in the android permission model , as if the user by mistake gives a single permission the app can grant all the permissions silently to the application with each update without even asking the user in this paper the authors explains the threats that fiving permission is a major outbreak for the android users as there will be no privacy to the user team run a test named empirical study on 2,865,553 app released and that showed that more than 17% of the applications on the play store requests new permissions which might be hazard for the user without any approval team analysis shows that the apps actually gets over 55% of data without event getting permissions this show how abusive these apps are on android and it can get worse for the users as they can track your location your contact list your camera, mic, your phone call records and emails , even passwords without even the user knowing it .

**Paper no 2:**

Better Code, Better Sharing: On the Need of Analyzing Jupyter Notebooks

**Authors**; It was introduced by Fernando Pérez, Jiawei Wang who was from Monash University, Li Li from Monash University and Andreas Zeller from CISPA Helmholtz Center for Information Security.

**Introduction**; Jupyter notebooks is an software which is used for new or basic programmers who cannot code properly have to work on scientific field it makes code easy for the students and the scientist to work on their field and also store them on computer , jupyter is an open source environment where dozens of languages can be implemented it was made from I Python it is also awarded many times this type of notebook helped many students and new programmers to be advance in programming .following people who felt that there is a need for this kind of software and were chosen to introduced jupyter . jupyter can also be described as open source web application allowing users to write documents containing important text for them ,equations and visualizations , and also as well coding which is executable , Jupyter now has become very popular as it provides workbook for sharing information just like GitHub , now a days we can say it is compulsory notebook for scientist for data scientist to analyze although jupyter is also best for poor programming as it aims to get skills in programming

Summary

Jupyter notebook is an open-source web application which points towards the scientist and students to get advance in coding and making the scientific data analysis easier for the customer ,jupyter data is proceed by code which is converted into .py and then output is given by Jupiter although the code will not appear in python scripts , it’s called the Jupyter processing chain after that you can save the coding or your work in jupyter library as notebook it is referred as “library code” main idea of jupyter is to work easily on “statistics”. if Jupiter selected 1982 contains total of 202,332 lines of python code it explains that the code if distributed gives median and average lines of 62 and 102.5 lines, which contains 10 code cells

Furthermore, jupyter converts python code in pythons’ scripts it shows that ideally jupyter is made to practice on python as 1982 jupyter corresponds to 202,332 lines of python code if PEP8 is run than it gives 73,371 errors which shows that notebook is not well aligned with the recommended code practices as this table shows,

Table

Python remark Notebook remark

E501 line too long E231 missing whitespace after ,, ;, or: E231 missing whitespace after ,, ;, or: E501 line too long

W291 trailing whitespace W293 blank line contains whitespace W293 blank line contains whitespace W291 trailing whitespace

E111 indentation is not a multiple of four E225 missing whitespace around operator E201 whitespace after ( E251 unexpected spaces around keyword parameter equals

E265 block comment should start with # E703 statement ends with a semicolon

E302 expected 2 blank lines, found 0 E261 at least two spaces before inline comment

E225 missing whitespace around operator E265 block comment should start with # comment

E251 unexpected spaces around keyword / parameter equals E128 continuation line under-indented for visual indent

Methodology

Now comes the part for methodology jupyter illustrates the working process, it builds an abstract syntax tree (ATS) for the involving code

Example import NumPy as np

a = np.arange(15)

print(a)

print(a.shape)

this code with each variable is associated with special node indicating it is introduced into context or we can say it is referenced by the context and secondly jupyter performs transversal over the AST which separates all the variables that are associated with the context now jupyter application works as if a variable is stored but not loaded subsequently , it will be consider as unused variable by the flag . if it is recalled then now it will be considered as used. now this experiment shows that 803 notebooks out of 1982 contains code with unused variables

Now the team jupyter notebooks contain code which access the deprecated functions of the library as it if for the educational purposes but the team says that deprecated functions should be avoided .now in order to check jupyter team checks that if the deprecated functions are used or not they make them rely on ground truth of deprecated functions although the ground truth functions are not directly connected to the deprecated functions but the library maintainer is used to describe the changes if the deprecated functions , by simplifying these notes user will easily be able to collect a ground truth of deprecated functions.

Aim and result

Jupiter notebooks as mentioned above help students and Scientist to get advance in coding it is considered as educating application or we can say it is made only for beginners as it will harm the community if not changed but it is also mandatory for poor programming skills and technical debts.

Jupyter is used for the educational purposes as it helps the learners

1. Enforcing good coding styles.  
   jupyter teams made this application to help learners get some better skills in coding and points out mistakes which might help to make better programming skills and styles
2. Improving code quality and reliability  
   automated tools made within the application helps to point out poor coding style which makes the code more reliable and it also recommend to fix the error and also indicates the duplicated codes and inefficient algorithm and much more

Jupyter application team further aims to first investigate the necessities of the application by researching and views by the community.

**Paper no:3**

**Comprehension of challenges at the level of software ecosystem and global software engineering**

Authors: basically, software ecosystem was given by “David G. Messerschmitt” and “Clemens Szyperski” which explains or defines to make a functioning unit for shared market and make a relationship with business and software. On the other hand, global software engineering means same. But this page contains the ecosystems of Facebook which was given by or written by Ralf lammel on July 2020.

**Introduction.** This page which mentions the ecosystem of software at Facebookwhich includes the infrastructure, Data infrastructure and AI Infrastructure and how Facebook makes its business in software marks and how they solve their challenges as their total business is on social media.

Summary

Software engineering and ecosystem at Facebook and its business includes the infrastructures includes version control, CI, language services, testing automation, data infrastructure includes storage engines, query engines, pipelines, metastores and AI infrastructure includes ML workflows, feature stores, online and offline prediction on GitHub

Now the page tells us about what really is a software ecosystem it’s actually a collection of software projects which are developed together in same environment like same platform as environment can be physically or virtual as projects are part of open-source community.

Now comes the challenging part for the Facebook as they have to continuously develop app, services internal tool, bug management and tracking, engineering in different time zones different employee and their degree types.

Comprehension challenges in developer workflow, ownership management, Code review automation

Scenarios of work-item prediction I/II

The ‘Incident Response’ Scenario:

• Work item: Alert for suboptimal performance

• Question: The workflow steps to follow in response

• Automation: Record steps in past instances

• Challenge: To know when someone is responding

Scenarios of work-item prediction II/II

The ‘Aggregate Performance’ Scenario:

• Work item: A diff (a system change)

• Question: Time spent on diff

• Automation: Record all activities on diff

• Challenge: To know when someone is working on the diff

These scenario shows that how scenarios are responded which are incidentally given by the employees are worked by the Alert suboptimal performance and questions asked by the costumers are responded by the workflow steps, automation records step in the past instances and the challenges which are responded by the employees 2nd scenario tells that if the work item is on change system which means on Linux or windows questions replied on different time zones record all activities on diff database and challenges are solved by different teams that’s called Aggregate performance scenario

Now the team talks about the dark natter which means the data which is not in use they efficiently try to manage it by a way they called dark matter work flow analysis, they do it by making the data go into query DB interactively and then send it to commit a version locally now the team reads the data if the data is reusable they leave it but if the data is not in use they send it to a different company or publisher and publish it from another company if the data is integrity is better than they review it or they leave it.

In this page they also mentioned that why do they have dark matter probable reasons given by the teams are that Facebook tools don`t track work items consistently and the tools aren`t fully integrated although the American government tracks all the details which Facebook is not mentioning also they tell us that the logging in is not designed with workflow analysis in mind as the developer workflow don`t have proper structured also the developers are constantly in engage in a lot of context Switching now the Facebook team consider the aspects by tools, tools which are added obsoleted and are easily removed, tools which are functionally add and removed which means new versioning, integrating with others might also help Facebook team, main problem of logging dark matter they resolve it by schema or semantics evolves in the end practice make it better

On the other page the author Ralf talks about how Facebook switches their context in a table which explains that at the end of the year different are created then by other team diff is shipped by the other team who sends to diff developers to abandoned and diff is reverted if needed and of the year the work is still in progress the next year diff is published by the developers version is created for the new version and developers mod is introduced to get review from the costumers if the diff is accepted then changes are requested and are made although the Facebooks says that they might need more time in high confidence events .on the next page Ralf showed a table that shows a system for diff prediction which explains that logging foundation is managed by the integrate all avail logs which are version control continuous integration cli, internal web-based tools . Time windows into dark matter in which team uses 10 minutes or more to test the probability of the employee on diff. candidates work items includes anything may have possible working on a diff in the end Ralf tells us about the Comprehension Challenges in Ownership Management which relates that their work in Ownesty, now what really is ownership management, Facebooks says that each asset has the most accountable owner at all the time. Basic challenges in ownership management are mentioned as Challenge Details Ownership Decay How to know whether to trust owners on file? Asset sub classing How to identify and handle specific subsets of assets? Team-level ownership How to assign teams as owners with individual signal? Ranking owner candidates What ranking to use to recommend one or more candidates? Whole/part asset relationships How to obey those relationships with recommendations? Monotonic features How to make sure that "more" means "more likely owner"? Explainable recommendations How to explain recommendations to use so that they accept? In the end Ralf writes about the entities involved in code review at Facebook which are Diff • Diff summary • Diff test plan • Commit • CI signal Miscellaneous • Task (bug or feature) • Alert • Incident • Root causing diff they are improved by knowledge graph change impact analysis traceability recovery and summaries I want to include that the author likes the cats too much .