Week 3: Empirical Study of Android Static Analysis

Nate Bachmeier

TIM-7140:Software Engineering

March 14, 2021

Northcentral University

# Empirical Study of Android Static Analysis

Development teams require automation to find issues within their Android code. Historically, SonarQube is the default choice, but that could change with Amazon’s new static analysis service, Code Guru.

## Problem Statement

Few inventions are more personal than the mobile phone. Some studies estimate that the average consumer touches their phone over 2600 times per day across 73.8 hours per month (Reizer, 2019; Winnick & Zolna, 2016). Those users interact with dozens of specialized apps that broker their lives with unfettered access to their privacy. Anyone with a valid email address and twenty-five-dollars can publish content into Google’s Playstore (Microsoft, 2018). This low entry barrier for both developers and consumers fueled its growth to 86.2% of the market (MobileApps, 2021).

Within this vast community lies an ocean of security, performance, and quality defects. It is challenging to educate everyone within such a diverse population. Instead, automation needs to exist for proactively discovering these issues. While static analysis tooling exists for this scenario, selecting a suitable analyzer is more art-than-science.

## Purpose Statement

Traditionally, Android developers turn to SonarQube for their static analysis needs. SonarQube is a common choice because it is open-source, intuitively designed, and offers many integrations (SonarQube, 2021). Recently released Amazon CodeGuru supports analysis across Java and Python sources (Vaidya, 2020). DevOps teams require a mechanism to assess these tools against one another.

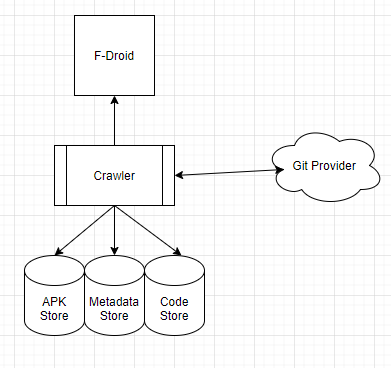
While it could be tempting to compare one or two source repositories, this could produce misleading results (e.g., selection-bias and poor representation). Instead, both tools need to analyze a standard baseline, like F-Droid (Krutz et al., 2015). F-Droid is an installable catalog of FOSS (Free and Open Source) apps for the Android platform (F-Droid, 2021). There are roughly four thousand applications within their collection. Each project’s entry contains downloadable links to source repositories and official milestone releases. While these efforts shine light into the problem space, they are not complete. Even thoroughly analyzed, F-Droid represents 0.1% of the size of Google Play Store’s three million apps (Statista, 2021).

# Data Collection Process

## Download Process

Crawling the F-Droid catalog is relatively simple because it is not very big. The process’ implementation uses Amazon cloud services to centralize APK files into object storage (S3), metadata into relational tables (Relational Data Store), and source code into local repositories (CodeCommit). Downloading these artifacts took 35 minutes and cumulatively consumed 100 GiB storage. Next, random sampling chose 1000 projects from the set. This filter was necessary due to soft limits within the AWS environment.

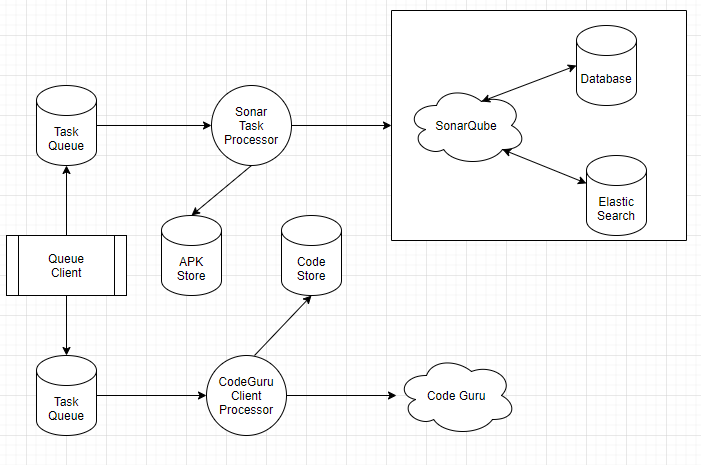
Figure 1: Crawler Process



## Producing Recommendations

There are multiple components to a SonarQube cluster, including the deployment of an Amazon RDS PostgreSQL database and ElasticSearch dependencies. A distributed queue (Simple Queuing Services) holds a pointer to all pending analysis jobs, which feeds into a custom container wrapping the sonar-scan tool. After completing, the results upload into SonarQube for further analysis. Configuring the CodeGuru path was less effort because of its fully managed design. The most challenging aspect involves tuning the processing rate to avoid service throttling errors.

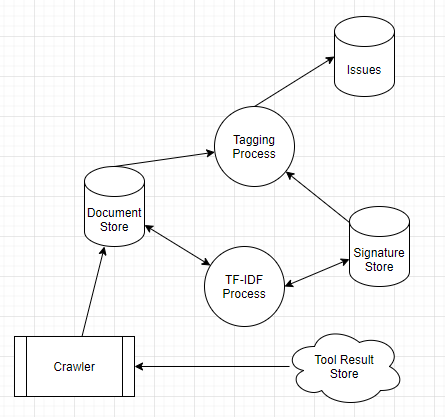
Figure 2: Transformation



## Normalize Results

A mechanism for directly comparing results between SonarQube and CodeGuru does not exist. There are also subtle differences between SonarQube using compiled APK files versus CodeGuru relies on source code. It would also be misleading to compare counts since it is better to find one critical risk than one hundred low-priority issues.

Instead, a custom crawler downloads the static analysis results into a document store (Elastic FileSystem). Each document contains a list of tool recommendations as unstructured text. Next, a Term Frequency and Inverse Document Frequency (TF-IDF) process convert the free-form text into signatures. TF-IDF is a statistical process that weights the importance of a word within a document corpus. Then an Issues database persists a mapping between projects and signature counts. This entire process (see Figure 5) repeats in isolation for both CodeGuru and SonarQube.

Figure 3: Convert Unstructured Text into Buckets  


Code Guru reported 56,562 recommendations across 1000 repositories (only 682 had results). After completing this process with a 1000 keyword limit, results in 318 buckets. With more time and effort, the count would likely decrease by another 35%. Due to time constraints, implementing the SonarQube crawler did not happen. However, from a cursory investigation, it should encounter favorable results.

Figure 4: Generate the TF-IDF

## 

## Linking the Issue Buckets

The previous step consolidates tens of thousands of static analysis recommendations into a manageable number of buckets. Using light automation and manual analysis needs to build a related findings hierarchy (see Figure 5).

For instance, one SonarQube warning states, “*object leak of ‘foo’ at line 123*,” versus CodeGuru reports “*missing disposal of resource ‘foo’ at line 123*.” Both issues are describing the same problem (a memory leak) and now comparable. Similar verbiage differences exist for numerous performance, security, and error handling situations. Finally, the normalized result trees are comparable between CodeGuru and SonarQube. Researchers can explore the different levels to understand how many critical issues either tool detected. Using direct count at this stage is also acceptable because each leaf is an apples-to-apples comparison.

Figure 5: Hierchial Tree

# Conclusions

Tight integration exists between mobile devices and modern-day life, with significant engagement occurring within the Android ecosystem. A critical component of its success comes from eliminating many entry barriers encouraging developers of all skill levels. Those engineers need automation to discover common mistakes across security, reliability, and performance. Historically, SonarQube is the default choice, though more recently, Amazon CodeGuru offers similar capabilities. Configuring and managing CodeGuru is significantly more straightforward, and that warrants evaluating the tool.

Both SonarQube and CodeGuru provide unstructured recommendations that are not directly comparable. This effort mitigates those challenges through a TF-IDF process that creates signatures for each message and buckets those issues. Mapping those buckets into a hierarchical structure enables consistently assessing the results across the project corpus. Depending on the organization’s needs, this will direct them toward the best tool for the job.

# References

F-Droid. (2021, March 7). *What is F-Droid*? Retrieved from F-Droid Organization: https://f-droid.org/en/about/

Krutz, D., Mirakhorli, M., Malachowsky, A., Ruiz, a., Peterson, J., Filipski, A., & Smith, J. (2015). A dataset of open-source Android applications. *12th Working Conference on Mining Software Repositories* (pp. 522-525). IEEE Press. doi:10.5555/2820518.2820603

Microsoft. (2018, February 16). *Publish to Google Play*. Retrieved from Xamarin Documentation: https://docs.microsoft.com/en-us/xamarin/android/deploy-test/publishing/publishing-to-google-play/

MobileApps. (2021, February 16). *Android vs. iOS Market Share 2020: Stats and Facts*. Retrieved from MobileApps: https://www.mobileapps.com/blog/android-vs-ios-market-share

Reizer, J. (2019, January 19). *How Often do you Touch Your Smartphone in One Day?* Retrieved from No Fake News: https://nofakenews.net/2019/01/19/how-often-do-you-touch-your-smartphone-in-one-day/

SonarQube. (2021, March 14). *Try out SonarQube*. Retrieved from SonarQube: https://docs.sonarqube.org/latest/setup/get-started-2-minutes/

Statista. (2021, March 7). *The number of available applications in the Google play store*. Retrieved from Statista: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

Vaidya, N. (2020, February 27). *Automating code reviews and application profiling with Amazon CodeGuru*. Retrieved from AWS DevOps Blog: https://aws.amazon.com/blogs/devops/automating-code-reviews-and-application-profiling-with-amazon-codeguru/

Winnick, M., & Zolna, R. (2016, June 16). *Putting the finger on Our Phone Obsession*. Retrieved from dscout: https://blog.dscout.com/mobile-touches