

MOBILE APP TO ANALYSE APK FILES BASED ON YARA RULE

**MUHAMMAD IZHAM BIN NORHAMADI
B032020039**

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Table of Content

CHAPTER 1: INTRODUCTION.....	4
1.1 Introduction.....	4
1.2 Problem Statement.....	4
1.3 Project Question.....	4
1.4 Project Objective	5
1.5 Project Scope	5
1.5.1 Automatically generate YARA Rule using open-source tools.....	5
1.5.2 Develop mobile app.....	5
1.6 Project Contribution.....	5
1.7 Report Organization.....	6
1.8 Conclusion	7
CHAPTER 2: LITERATURE REVIEW	8
2.1 Introduction.....	8
2.2 Related Work/Previous Work.....	8
2.2.1 Malware Analysis	8
2.2.2 Malware Infection Cases	9
2.2.3 YARA in Malware Analysis	9
2.3 Critical review of current problem and justification.....	10
2.4 Proposed Solution/Further project.....	10
2.5 Conclusion	10
CHAPTER 3: PROJECT METHODOLOGY	11
3.1 Introduction.....	11

3.2	Methodology.....	11
3.3	Project Milestones	12
3.4	Conclusion	13
CHAPTER 4: ANALYSIS AND DESIGN		14
4.1	Introduction.....	14
4.2	Sample Malware	15
4.3	Yara Rule Structure	15
4.4	Yara Generation Tools Analysis.....	17
4.4.1	YaraGenerator	17
4.4.2	YarGen	17
4.4.3	Koodous.....	18
4.4.4	Androguard module for Yara	19
4.4.5	VirusTotal API	19
4.5	App Design	20
4.6	Conclusion	21

CHAPTER 1: INTRODUCTION

1.1 Introduction

Android Operating System is one of the most popular and widely used open-source mobile platforms and has the highest mobile market share until this day, making it the most widely used operating system in the world. This fact makes Android users the biggest target group for malware developers as trend analyses show large increase in mobile malware targeting the Android platform which leads to privacy thefts of Android users. The security and privacy of Android apps are very critical, especially that over 6000 apps are added to the Google Play Store every day. Thus, various mobile malware detection systems are proposed in the recent years to address this issue. The objective of this project is to develop a mobile application that can analyse APK files by utilizing YARA Rule. By the end of this project, it is expected to produce a functional mobile app that can analyse malware from APK files using certain YARA Rules.

1.2 Problem Statement

Android Operating System was found with increasing malware attacks in recent year. Often users are not aware of malware installed on their device which can come from the Google Play or other unknown third-party source.

PS	Problem Statement
PS ₁	Android devices are prime target for malware and often users are not aware of malicious applications that are going to be installed

Figure 1.2.1 Summary of Problem Statement

1.3 Project Question

Based on the problem statement above, a suitable mobile development framework and YARA Rules need to be developed for a functional mobile analysis tool. Further study on YARA Rule is needed to understand the tool and its usage.

PS	PQ	Problem Question
PS ₁	PQ ₁	How to create an effective malware analysis tool on Android platform?

Figure 1.3.1 Project Question

1.4 Project Objective

The project aims to investigate the application of YARA Rule on malware investigation and to develop a mobile app that utilizes YARA Rule using any suitable framework or tool available.

PS	PQ	PO	Project Objective
PS ₁	PQ ₁	PO ₁	To investigate YARA Rule and its application on malware investigation
		PO ₂	To develop a suitable YARA Rule mobile app
		PO ₃	To assess developed mobile app using sample APKs

Figure 1.4.1 Project Objective

1.5 Project Scope

1.5.1 Automatically generate YARA Rule using open-source tools

Writing YARA rules manually requires a highly specialized skill set in security, whereas using tools can help generate YARA rules automatically with relative ease. However, generated YARA rules are generally not optimized for operations and require post processing manually to reduce false positives and to increase its effectiveness. Therefore, an optimal YARA rule for mobile malware requires a generated YARA rule from a selected tool and manually processing the rule.

1.5.2 Develop mobile app

To produce a malware analysis app on Android, it is important to choose one of the mobile development frameworks such as React Native, Flutter and Kotlin each with their own pros and cons. Once developed, the software then can be installed through APK file to any Android devices.

1.6 Project Contribution

Using YARA tools and automation, the project will generate YARA Rule that will identify and classify malware families, in addition of producing a malware analysis tool for Android platform.

PS	PQ	PO	PC	Project Contribution
PS ₁	PQ ₁	PO ₁	PC ₁	Generation of YARA Rule
			PC ₂	Classification of malware
		PO ₂	PC ₃	Produces malware analysis tool on Android
		PO ₃		

1.7 Report Organization

Chapter 1: Introduction

Introduction chapter discusses about the overall picture of the project. This chapter also explains about the scope and explains the gist of the whole project.

Chapter 2: Literature Review

Literature Review will discuss about the problem statement in more detail. Next, this chapter will also discuss the findings found in research papers that are related to this topic. Things that are required to be inserted here are such as citation of the research papers.

Chapter 3: Project Methodology

Project Methodology will discuss the method completing this project. It will follow the project milestone given to make sure that each chapter are completed in time.

Chapter 4: Design

Design chapter models how the application will work which include the user interface, the framework used, the programming language and how the application communicate data online.

Chapter 5: Implementation

Implementation chapter is the development phase to build the application. It details the tools and requirement in the development and the creation of prototype.

Chapter 6: Testing and Analysis

Testing and Analysis chapter will test the functionality of the prototype and improving parts of the application to meet the requirements.

Chapter 7: Conclusion

Conclusion chapter concludes the result of the whole project. Starting from the design until testing and analysis chapter. This chapter will also conclude whether the project is a success or vice versa.

1.8 Conclusion

The introduction explains the rough idea about the mobile analysis project which state the project's problem statement, project question, project objective and project scope. It also shows the overview of the project report structure. The next chapter, literature review, will take an in-dept look on the success automatic YARA rule generation tools and its implementation on mobile platform.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

A literature review is a thorough overview of prior studies on a particular topic. The literature review examines scientific journals, books, and other references that applicable to specific research subject. Previous studies should be enumerated, defined, summarized, critically evaluated, and clarified in the analysis. It should provide a theoretical foundation for the study and assist the author in determining the scope of the study.

2.2 Related Work/Previous Work

2.2.1 Malware Analysis

Malware analysis is the process of determining the purpose and functionality of a given malware sample and determining its type such as a virus, Trojan horse, or ransomware. This process is a necessary step to be able to develop effective detection techniques for malicious code and its removal from an infected machine. Analyzing the program by inspecting it refers to as *static analysis*, while analyzing a program during execution is called *dynamic analysis*.

Static analysis techniques can be applied on different representations of a program. If the source code is available, static analysis tools can help find security flaws and issues during software development. Static analysis tools can also be used on the binary representation of a program although when the source code of a program compiled into binary executable, some information will be lost. Various techniques that are used for static malware analysis are file fingerprinting, extraction of hard coded strings, file format, disassembly, packer detection, and so on (Gadhiya, 2013) .

Since dynamic analysis is performed during runtime and malware unpacks itself, dynamic malware analysis evades the restrictions of static analysis (obfuscation issues). This way, it is easy to see the actual behavior of a program. Although the main drawback of this method is inability to analyses dormant code since dynamic analysis usually only monitors one execution path and thus suffers from incomplete code coverage. Two basic approaches for dynamic analysis are analyzing the difference between defined points and observing runtime-behavior (Gadhiya, 2013).

2.2.2 Malware Infection Cases

In 2016, an augmented mobile reality game Pokémon Go was first rolled out in Australia and New Zealand. In the frenzy that happened right after, gamers on Android devices who could not wait for the app to be officially rolled out to their regions decided to search for and download the game's APK to get the title into their smartphones ahead of the planned release in their country. Because of the method involved downloading app from a third party, there are risk involved as the app that users downloaded could be a malicious one. The infected version of Pokémon Go contains Droidjack, also known as SandroRAT, which is a malicious remote access tool that basically gives the attacker complete control over the devices of their victims (Mamiit, 2016).

In the year 2020, there has been rising of “covid” themed malware since COVID-19 pandemic has been one of the biggest news topics. The word “covid” in various combinations was typically used in the names of packages hiding spyware and banking Trojans, adware or Trojan droppers such as *covid.apk*, *covidMapv8.1.7.apk*, *tousanticovid.apk*, *covidMappia_v1.0.3.apk* and *coviddetect.apk*. These apps were placed on malicious websites, hyperlinks distributed through spam and so on (Chebyshev, 2021).

2.2.3 YARA in Malware Analysis

YARA rules discover malware based on a string-matching technique which can be customized depending on specific requirements to uncover security threats. It achieves this by creating descriptions of malware families based on textual or binary patterns. YARA rule conditions are Boolean expressions which are mostly focused on the binary outcome of the malware analysis. Both quality and quantity of YARA rules are crucial for an effective performance of malware analysis thus rule optimization process are important to generate an effective YARA rule and avoid false positives.

Due to how flexible and customizable nature YARA rules for malware analysis, there are several YARA rules generator tools that were created to automate the time-intensive process of generating the rules manually such as *yarGen*, *yaraGenerator*, and *yarbin* (Naik, 2020). *yarGen* is a Python-based tool utilized to generate YARA rules using intelligent techniques such as fuzzy regular expression, Naïve Bayes classifier and Gibberish Detector (Roth, 2017). *yaraGenerator* is a Python-based tool used for the generation of YARA rules

with completely different signature for different types of files such as EXEs, PDFs and Emails utilizing string prioritization logic and code refactoring (Clark, 2013). *yarbin* is another Python-based tool that generates YARA rule by finding rare functions in a certain malware samples or families by checking function prologues which define the start of functions (Chrisdoman, 2016).

2.3 Critical review of current problem and justification

Despite YARA being widely accepted technique for malware analysis, few tools exist and relatively little work has been done to automate the generation of YARA rules for specific malware families (Raff, 2020), many of the automatic generation tools were barely maintained and written with outdated Python libraries. Thus, they were deemed unfit to be implanted to the current project. Besides that, APKs are zip archives, it is not ideal to build rules using APK information such as manifest and certificate as decompressing zip and extracting the information into a single module would add a lot of unnecessary complexity and dependencies. Most analysts resort to building YARA rules for the dex file only without external help. *Koodous* platform provides an API to parse the APK file into a JSON report to be used with *androguard-yara* but the usage limit for analysis report for free user is 5/day which is not ideal.

2.4 Proposed Solution/Further project

In order to develop the project, Flutter was chosen as a suitable android development framework due to its compatibility with Android OS and extensiveness of Flutter library that allows rapid development of application. This project will make full use of VirusTotal API to parse APK file and to extract YARA rules from VirusTotal's crowdsourced YARA rules that matches sample APKs. Next, the project will make use of Koodous to extract APK information into JSON report, then use Androguard module for YARA to integrate static APK analysis with custom YARA rules.

2.5 Conclusion

In conclusion, this project will develop a static analysis app based on YARA rules using Flutter framework. The next chapter will discuss on the methodologies that will be done throughout this project.

CHAPTER 3: PROJECT METHODOLOGY

3.1 Introduction

In this project, several libraries and API will be used to integrate YARA functionality and automatic generation from within the application. The project will mostly follow the standard Software Development Life Cycle (SDLC) with a few changes to suit the project requirement.

3.2 Methodology

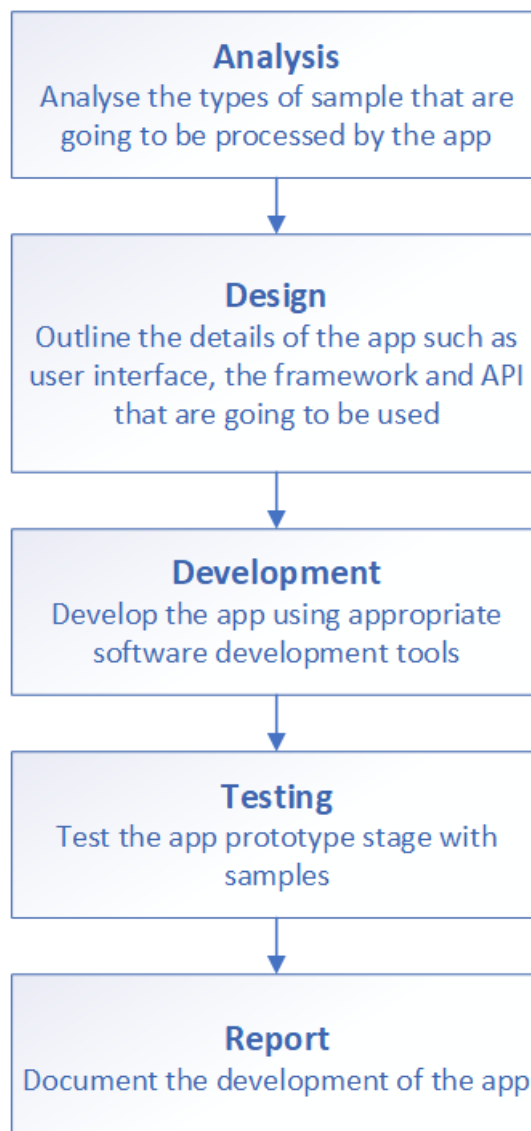


Figure 3.2.1 Methodology Framework

The first step of this project is to study the files that are going to be analyzed, which are sample APKs. There are several components inside an APK file such as META-INF, AndroidManifest.xml, classes.dex, and resources that are required by the app. Next comes the design phase, the prototype design of the app will be created used Visio. Next, the application will be developed using Flutter Framework because of its rich library to quickly develop a functioning prototype. Then, the app will be tested by sample APKs take note of the results. Then we record and document project findings and the performance of the app in a report.

3.3 Project Milestones

A project milestone is a scheduling method for identifying a specific stage in a project's timeline. It's a useful for determining schedule goals and ensuring the research's success.

Process/Phases	Activities	Completion Date
Analysis	<ul style="list-style-type: none"> Discover previous work on the application of malware investigation with YARA Find and evaluates the tools that are going to be used 	22 April 2022
Design	<ul style="list-style-type: none"> Use of templates to picture the rough design of the application Creation of design prototype 	9 May 2022
Development	<ul style="list-style-type: none"> Coding process of the application Implementation of YARA tools into the application 	24 May 2022
Testing	<ul style="list-style-type: none"> Evaluation of the application to make sure it is functioning properly Finding and fixing errors within the application 	28 May 2022
Report	<ul style="list-style-type: none"> Documentation of the development process and user guide on application usage 	6 June 2022

Table 3.3.1 Project Milestones Table



Figure 3.3.1 Gantt Chart

3.4 Conclusion

This chapter specifies the development approach utilized in this project. This will follow Software Development Life Cycle for on-time and smoother development. Prototype will also be developed to identify if the software functionalities are working as intended. This chapter also included a timetable for the project. The analysis and design of this project will be discussed in the next chapter.

CHAPTER 4: ANALYSIS AND DESIGN

4.1 Introduction

This chapter explains the details of analysis and design process of this project that will development of malware analysis tool. The figures follow will show the graphical representation of the software phases.

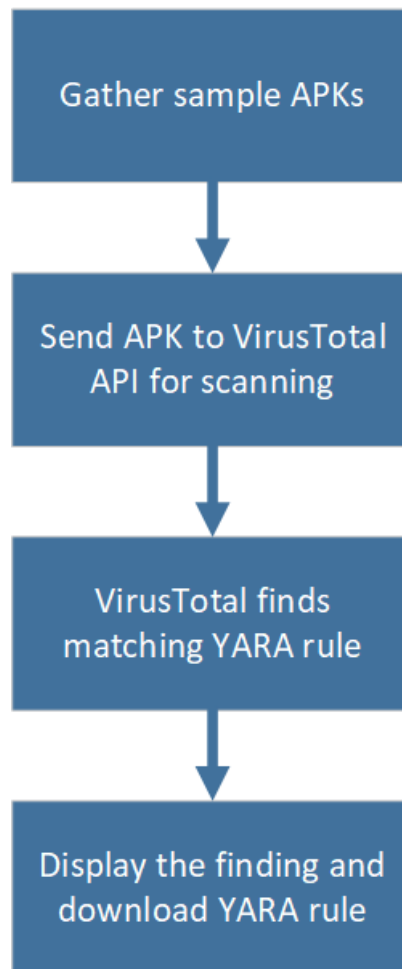


Figure 4.1.1 Yara Generation Phases

4.2 Sample Malware

Sample of malwares are handpicked from the year 2019 to 2021 to be tested by this project. The source of the malwares can be found in Koodous and github repositories. The table below show the list of malware samples that are used.

Package name	SHA256 hash	Year in the wild
com.saver.batterymobi	64ebe9b975de022b888f17db429af3a93d3db95db5af274e3eefd3ca7f24e350	2019
operatore.italia	0f5f1409b1ebbee4aa837d20479732e11399d37f05b47b5359dc53a4001314e5	2019
com.gelini	48618153df1b2b5be3f83e6e1fa6aa5f517b173b10f3f6e925d1598a22b459e1	2019
com.tencent.mm (flubot)	30937927e8891f8c0fd2c7b6be5fbc5a05011c34a7375e91aad384b82b9e6a67	2021
kijxlnbftwdhbbet.eaafsym.fziuffcjyjetmqxsmcd (aug_banking.apk)	fe2e8b115b3ffc2f3ab668c08c67b21afa6761426cef1c6a99f6cb9074d8076f	2020

4.3 Yara Rule Structure

Each Yara rule has to start with the word rule, followed by the name or identifier. Rules are composed of several sections, the meta, strings, and condition. The condition section is the only one that is required. It contains a Boolean expression that determines the result whether it is true for the object (file). The strings section is where the strings that will be looked for in a file will be defined. Things that can be looked for in strings are hexadecimals, text strings, and regular expressions. The meta can be added to help identify the files that were picked up by a certain rule. The only purpose for it is to store additional information about the rule.

```

1 import "androguard"
2 import "file"
3 import "cuckoo"
4
5 rule PromonShield
6 {
7     meta:
8         author = "Govind Sharma & Eduardo Novella"
9         date = "2022/06/01"
10
11     strings:
12         $lib1      = /lib[a-z]{10,12}\.so/
13         $lib2      = "libshield.so"
14         $config    = /[a-z]{10,12}\.dat/
15         $export1   = "sigsetjmp"
16         $export2   = "siglongjmp"
17         $export3   = "wcsnrtombs"
18         $export4   = "strtod"
19         $export5   = "strxfrm"
20         $export6   = "sigdelset"
21         $export7   = "wcslen"
22         $export8   = "localtime"
23         $section1  = /\.ncu/
24         $section2  = /\.ncc/
25         $section3  = /\.ncd/
26
27
28     condition:
29         ($lib1 or $lib2) and $config and 3 of ($export*) and all of ($section*)
30 }

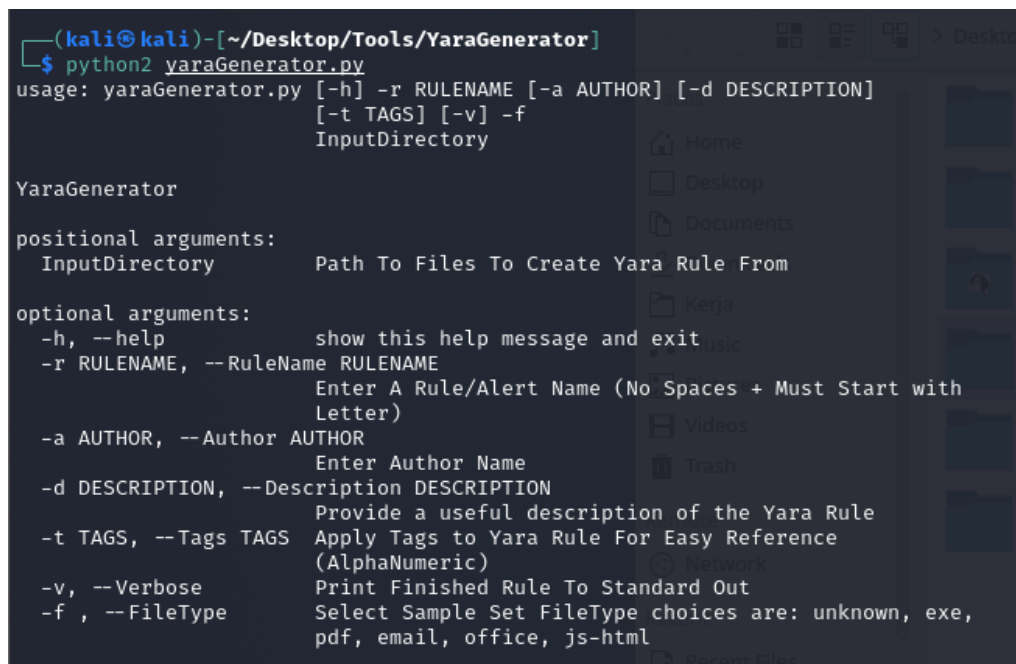
```

Figure 4.3.1 PromonShield Rule

4.4 Yara Generation Tools Analysis

4.4.1 YaraGenerator

YaraGenerator is a tool to attempt to allow for quick, simple, and effective yara rule creation to isolate malware families and other malicious objects of interest.



```
(kali㉿kali)-[~/Desktop/Tools/YaraGenerator]
$ python2 yaraGenerator.py
usage: yaraGenerator.py [-h] -r RULENAME [-a AUTHOR] [-d DESCRIPTION]
                        [-t TAGS] [-v] -f
                        InputDirectory

YaraGenerator

positional arguments:
  InputDirectory          Path To Files To Create Yara Rule From

optional arguments:
  -h, --help              show this help message and exit
  -r RULENAME, --RuleName RULENAME
                        Enter A Rule/Alert Name (No Spaces + Must Start with
                        Letter)
  -a AUTHOR, --Author AUTHOR
                        Enter Author Name
  -d DESCRIPTION, --Description DESCRIPTION
                        Provide a useful description of the Yara Rule
  -t TAGS, --Tags TAGS    Apply Tags to Yara Rule For Easy Reference
                        (AlphaNumeric)
  -v, --Verbose           Print Finished Rule To Standard Out
  -f, --FileType           Select Sample Set FileType choices are: unknown, exe,
                        pdf, email, office, js-html
```

Figure 4.4.1 yaraGenerator.py

The file types that are accepted are exe, pdf and emails. However, it does not support APK file and the project is outdated since its last update is on 2013.

4.4.2 YarGen

YarGen is also a Yara creation tool using strings found in malware files while removing all strings that also appear in goodware files.

```
(kali@kali)~[~/Desktop/Tools/yarGen]
$ python3 yarGen.py

Yara Rule Generator
Florian Roth, July 2020, Version 0.23.3

Note: Rules have to be post-processed
See this post for details: https://medium.com/@cyb3rops/121d29322282

usage: yarGen.py [-h] [-m M] [-y min-size] [-z min-score] [-x high-scoring] [-w superrule-overlap] [-s max-size] [-rc maxstrings]
               [--excludegood] [-o output_rule_file] [-e output_dir_strings] [-a author] [-r ref] [-l lic] [-p prefix]
               [-b identifier] [--score] [--strings] [--nosimple] [--nomagic] [--nofilesizes] [-fm FM] [--globalrule]
               [--nosuper] [--update] [-g G] [-u] [-c] [-i I] [--dropzone] [--nr] [--oe] [-fs size-in-MB] [--noextras]
               [--debug] [--trace] [--opcodes] [-n opcode-num]


yarGen
```

Figure 4.4.2 yarGen.py

It's repository also details on how to write and post-process Yara rules generated by YarGen. It does not recognize APK file or extracted APK file however.


4.4.3 Koodous

Koodous is a collaborative web platform for research on Android malware using malware analysis tools and Yara rule matching. Koodous will use it's own tools to extract information from APK files into a JSON report to be matched with rules.



DroidGuardVM
SafetyNet
Public

Author

 Govindsharma

Status

Enabled

Visibility

No social

General

Comments

```
5 rule DroidGuardVM
6 {
7     meta:
8         author = "Govind Sharma"
9         date = "2022/06/04"
10
11     strings:
12         $dvm1 = "libdroidguard.so" nocase
13         $dvm2 = "droidguard" nocase
14         $dvm3 = "library.txt" nocase
15
16
17     condition:
18         all of them
19 }
```

Figure 4.4.3.1 DroidGuardVM ruleset on Koodous

Despite having a large collection of community written Yara rules, recent changes to API usages have made malware analysis somewhat limited. Malware analysis requests and downloading malware sample are only made exclusive for paid API keys.

Characteristic	Free API	Fan API	Research API	Corporate API
Apks search	15 / day	25 / day	50 / hour	Unlimited
Apks detail	4 / min	50 / min	100 / min	Unlimited
Apks downloads	0 / day	15 / day	300 / day	5000 / day
Analysis requests	0 / day	25 / day	300 / day	Unlimited
Analysis reports	5 / day	25 / day	100 / day	Unlimited
Private YARA rules	0	5	50	200
Monthly price	€0	€60	€300	Contact info@koodous.com

Figure 4.4.4.2 Koodous API usage limit

4.4.4 Androguard module for Yara

Androguard is part of Koodous project that integrates static APK analysis with Yara. This can be used to find APKs by package name, permissions, or API level. This module is ready to use with Koodous reports using python script to get it automatically. Unfortunately, the project is no longer maintained as Koodous rejects old API key.

4.4.5 VirusTotal API

VirusTotal's API lets you upload and scan files, submit and scan URLs, access finished scan reports and make automatic comments on URLs and samples without the need of using the HTML website interface which allows for simple scripts to access the information generated by VirusTotal.

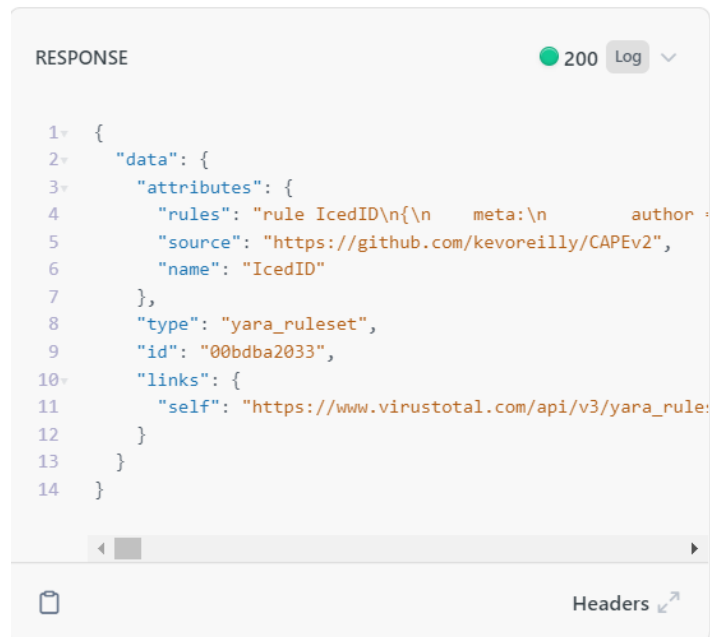


Figure 4.4.5 VirusTotal Yara ruleset API response

Using this API we can implement malware analysis in app.

4.5 App Design



Figure 4.5.1 App main page

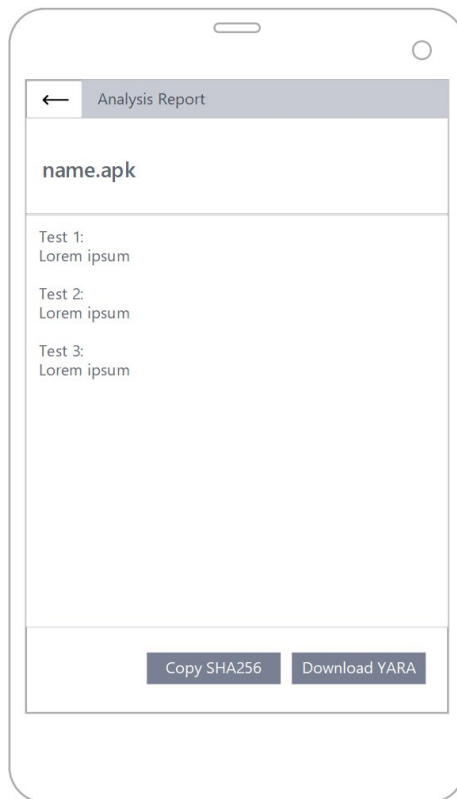


Figure 4.5.2 App analysis report page

4.6 Conclusion

This chapter explores the Yara generation phases along with the tools that will be used. This is followed by the design of the app that's going to be developed. The next chapter will go over the project's implementation.