# Literature review

## 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.

## Related Work/Previous Work

### 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).

### 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 form 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).

### 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).

## 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.

## 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.

## 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.