AKSHARA S
M.Sc Cyber Security, Sem 2
032200300002026

FRIDA

Frida is an open-source dynamic instrumentation toolkit primarily used for reverse engineering, debugging, and analyzing the behavior of software applications. Frida can be operated on multiple platforms, including Windows, Linux, Android and so on. Frida allows to inject custom code into running processes, enabling them to intercept and modify function calls, explore the memory of the target process, and monitor and manipulate various aspects of its execution. Frida is often used for bypassing security mechanisms, performing dynamic analysis of malware, analyzing and modifying the behavior of mobile apps, and developing patches or modifications for software.

Installation:

- Open command prompt on the computer.
- Ensure that we have Python installed on the system.
- Install Frida using pip, the Python package manager. Run the command :
  'pip install frida'.
- Wait for the installation to complete. Pip will download the necessary packages for Frida.
- Once the installation is finished, Frida will be successfully installed on the system.

How to use Frida:

To use Frida, follow these general steps:

- Identify the target: Determine the application or process we want to analyze or modify. This could be a mobile app, desktop application, or any other software.
- Set up the environment: Ensure to have Frida installed on the system. Also make sure to have the necessary dependencies and permissions to interact with the target application.
- Choose a Frida interface: Frida provides different interfaces depending on the needs. The two main options are:
  a.) Command-line interface (CLI): Frida provides command-line tools that allow injecting scripts into running processes. The primary CLI tools are frida-ps, frida-trace, and frida-discover.

b.) Frida Python API: Frida also provides a Python API that allows writing custom scripts in Python for more advanced interactions with the target application. This API can be used to inject code into processes, hook functions, read and write memory, and perform various other runtime manipulations.

- Write Frida script: Depending on the objectives, we need to write a Frida script using either the Frida CLI or the Python API. The script will define the behavior that we want to observe or modify in the target application.
- Run the Frida script: Execute Frida script by using the appropriate Frida command-line tool or by running the Python script. The script will be injected into the target process, and will start observing the desired behavior or modifications.
- Analyze and interact with the target: Once the Frida script is running, we can interact with the target application in real-time. We can inspect memory, intercept function calls, modify data, and perform other runtime manipulations to understand the application's behavior or achieve any specific goals.

Advantages:

Cross-platform support: Frida supports multiple platforms, including Windows, macOS, Linux and Android. This cross-platform compatibility allows us to analyze and manipulate applications on various operating systems.

Interoperability: Frida offers seamless interoperability with other tools and frameworks commonly used in the field of security research and reverse engineering. By combining Frida with tools like Burp Suite, IDA Pro, Radare2, and more, can enhance the analysis capabilities and integrate Frida into existing workflows.

JavaScript-based scripting: Frida provides a JavaScript API that allows to write scripts to interact with the target application.

Dynamic instrumentation: Frida enables us to perform runtime manipulation and dynamic instrumentation of running processes. We can inject custom code into processes, hook function calls, modify behavior on-the-fly, and analyze the application's execution in real-time.

Disadvantages:

Detection: As a dynamic instrumentation framework, Frida can potentially be detected by anti-debugging and anti-tampering techniques employed by the target application.

Limited visibility in obfuscated code: If the target application employs strong code obfuscation techniques, Frida may face challenges in effectively analyzing and manipulating the obfuscated code. Complex obfuscation can make it harder to identify relevant functions and data structures, potentially limiting the extent of dynamic analysis that can be performed.

Complexity: Working with Frida can require a certain level of technical expertise and familiarity with concepts like dynamic analysis, reverse engineering, and scripting. Users who are new to these domains may find a learning curve associated with understanding Frida's concepts, APIs, and usage.

Compatibility and stability: The effectiveness and stability of Frida can vary depending on the target application, operating system, and device. Not all applications may be fully compatible with Frida, and certain limitations may arise when working with specific platforms or versions.

QARK

Qark stands for "Quick Android Review Kit" and is an open-source tool designed for Android application security testing and static analysis. Qark aims to assist developers and security researchers in identifying potential security vulnerabilities and privacy issues in Android apps.

Qark performs static analysis on the APK (Android Application Package) files of Android apps to detect security vulnerabilities and provide a comprehensive report of potential risks. It combines multiple security scanning techniques and checks against known security best practices to identify common security issues that can impact the privacy and security of Android applications.

Some of the key features and checks performed by Qark include:

Permission-related issues: Qark examines the permissions requested by the app and identifies any potential overprivileged or underprivileged permissions. It also checks for dangerous permissions that may pose a risk to user privacy or security.

Exported components: Qark examines exported components like activities, services, and broadcast receivers, and identifies any potential security risks associated with them, such as unauthorized access or misuse.

Code-related vulnerabilities: Qark analyzes the app's code to identify common security vulnerabilities, such as insecure data storage, input validation issues, insecure communication (HTTP instead of HTTPS), improper use of cryptographic functions, and more.

Intent-related vulnerabilities: Qark checks for potential security issues related to the usage of Android Intents, such as insecure intent handling, intent spoofing, or intent injection vulnerabilities.

WebView vulnerabilities: Qark inspects the usage of WebView within the app and looks for potential vulnerabilities, such as JavaScript injection, insecure WebView settings, or potential vulnerabilities in the WebView implementation.

Third-party libraries: Qark scans for the usage of known vulnerable third-party libraries within the app and provides information about any associated security risks.

Installation:

- Ensure that we have Python installed on the system.
- Open a terminal or command prompt on your computer.
- Install Qark using pip, the Python package manager. Run the command: 'pip install qark'.
- Wait for the installation to complete. Pip will download and install the necessary packages for Qark.

How to use Qark:

To use Qark, follow these steps:

- Ensure Qark is installed on the system.
- Open command prompt.
- Navigate to the directory where APK file is located. Use the cd command to change directories in the terminal.
- Run Qark by executing the command: 'qark --apk (path)'.
- Qark will start analyzing the APK file and perform various security checks.
- Once the analysis is complete, Qark will generate a report detailing any security vulnerabilities, privacy issues, or best practice violations it detected in the Android application.
- Review the Qark report to understand the identified issues. The report will provide information on the specific vulnerabilities, their severity, and recommendations for mitigation.

Advantages:

Automated scanning: Qark automates the process of scanning and analyzing APK files, making it efficient and convenient for security testing. By running Qark on an APK, we can quickly identify potential security risks without the need for manual code inspection or reverse engineering.

Comprehensive analysis: Qark performs static analysis on APK files and checks for a wide range of security vulnerabilities, privacy issues, and best practice violations. It covers various aspects of app security, including permissions, code vulnerabilities, intent handling, WebView usage, and more.

Severity assessment: Qark assigns severity levels to the identified security vulnerabilities, helping prioritize the remediation efforts. This allows developers and security professionals to address critical issues first, enhancing the overall security posture of the application.

Detailed reporting: Qark generates comprehensive reports that provide detailed information about the identified security vulnerabilities. The reports include descriptions of the issues, their impact, and recommendations for mitigation, enabling developers to understand the risks and take appropriate remedial actions.

Disadvantages:

Lack of real-time or dynamic analysis: Qark focuses on static analysis, meaning it analyzes the APK file without considering the dynamic runtime behavior of the application. Some vulnerabilities, such as those related to runtime environment interactions or server-side issues, may require dynamic analysis techniques that Qark does not provide.

False positives and false negatives: As with any automated security analysis tool, Qark may generate false positives or false negatives. It is crucial to manually review and validate the findings to ensure accurate results.

Limited coverage: Qark primarily focuses on static analysis of APK files and may not cover all possible security vulnerabilities or privacy issues. It is essential to use Qark as part of a comprehensive security testing approach that includes other tools and techniques.