MOBILE APP SECURITY ASSESSMENT USING STATIC ANALYSIS

MOBILE SECURITY FRAMEWORK (MobSF)

SHERYL TREFINA J

sheryltrefina04@gmail.com

TABLE OF CONTENTS:

6. CONCLUSION

6.1 Importance of Static Analysis in Mobile Security

6.2 Screenshots

1. INTRODUCTION:

1.1 PURPOSE OF THE REPORT

The purpose of this report is to provide a comprehensive analysis of mobile application security using static analysis techniques, specifically utilizing the Mobile Security Framework (MobSF). This report aims to identify potential vulnerabilities and insecure coding practices within mobile applications, offering insights and recommendations for improving security. By thoroughly examining the static components of mobile applications, this report seeks to highlight the importance of proactive security measures and the role of static analysis in the overall security assessment process.

1.2 SCOPE OF ANALYSIS

The scope of this analysis includes evaluating mobile applications for security vulnerabilities through static analysis using MobSF. This report covers:

- The setup and use of MobSF for scanning Android (APK) and iOS (IPA) applications.
- Analysis of the source code and binary files for identifying security weaknesses.
- Examination of application certificates for integrity and validity.
- Detection of potential malware within the application package.
- Providing detailed findings, interpretations, and recommendations for enhancing mobile application security based on the analysis results.

1.3 OVERVIEW OF MOBILE APPLICATION SECURITY

Mobile application security is a critical aspect of the software development lifecycle, aimed at protecting applications from threats and vulnerabilities that can lead to data breaches, unauthorized access, and other security incidents. With the increasing reliance on mobile apps for various personal and professional activities, ensuring their security is paramount. Key aspects of mobile application security include:

- **Confidentiality:** Protecting sensitive information from unauthorized access.

- **Integrity:** Ensuring that data is not altered or tampered with during storage or transmission.

- **Authentication:** Verifying the identity of users and ensuring secure access to the application.

- **Authorization:** Ensuring that users have appropriate permissions for accessing resources within the application.

- **Data Encryption:** Using encryption techniques to protect data at rest and in transit.

- **Secure Coding Practices:** Implementing best practices in coding to avoid common vulnerabilities such as SQL injection, cross-site scripting (XSS), and insecure data storage.

- **Regular Security Testing:** Conducting regular security assessments, including static and dynamic analysis, to identify and mitigate vulnerabilities early in the development process.

2. OVERVIEW OF MobSF

2.1 INTRODUCTION TO MobSF

The Mobile Security Framework (MobSF) is an open-source automated security testing tool specifically designed for mobile applications. It supports static and dynamic analysis of Android, iOS, and Windows apps, providing comprehensive insights into their security posture. MobSF is widely recognized for its ability to perform deep analysis on mobile application binaries (APK/IPA) and source code, detecting a wide range of security issues. By leveraging MobSF, developers and security professionals can identify vulnerabilities, insecure coding practices, and other potential threats within mobile applications early in the development lifecycle.

2.2 KEY FEATURES

MobSF offers a rich set of features that make it a powerful tool for mobile application security testing. Key features include:

- STATIC ANALYSIS:

- APK/IPA Scanning: Analyze Android and iOS application packages without requiring source code access.

- Code Analysis: Detect insecure coding practices, vulnerabilities, and other security issues within the source code.

- Binary Analysis: Assess the security of binary files to identify known vulnerabilities and potential risks.

- Certificate Analysis: Verify the integrity and validity of application certificates to ensure they are not compromised.

- Malware Analysis: Identify malicious components or behavior within the application package.

2.3 BENEFITS OF USING MobSF FOR STATIC ANALYSIS

Using MobSF for static analysis offers numerous benefits, making it an essential tool for enhancing mobile application security:

- Early Detection of Vulnerabilities: MobSF helps identify security issues early in the development process, allowing for timely remediation and reducing the risk of vulnerabilities being exploited in production.

- Comprehensive Analysis: MobSF provides a thorough examination of mobile applications, covering various aspects such as code, binary files, certificates, and potential malware, ensuring no stone is left unturned.

- Automated Testing: Automating the static analysis process with MobSF saves time and effort, enabling developers to focus on fixing issues rather than manually searching for them.

- Cost-Effective Security: As an open-source tool, MobSF provides a cost-effective solution for mobile application security testing, eliminating the need for expensive commercial tools.

- Improved Compliance: By incorporating checks for OWASP Mobile Top 10 vulnerabilities, MobSF helps ensure that applications comply with industry-standard security guidelines.

3. METHODOLOGY

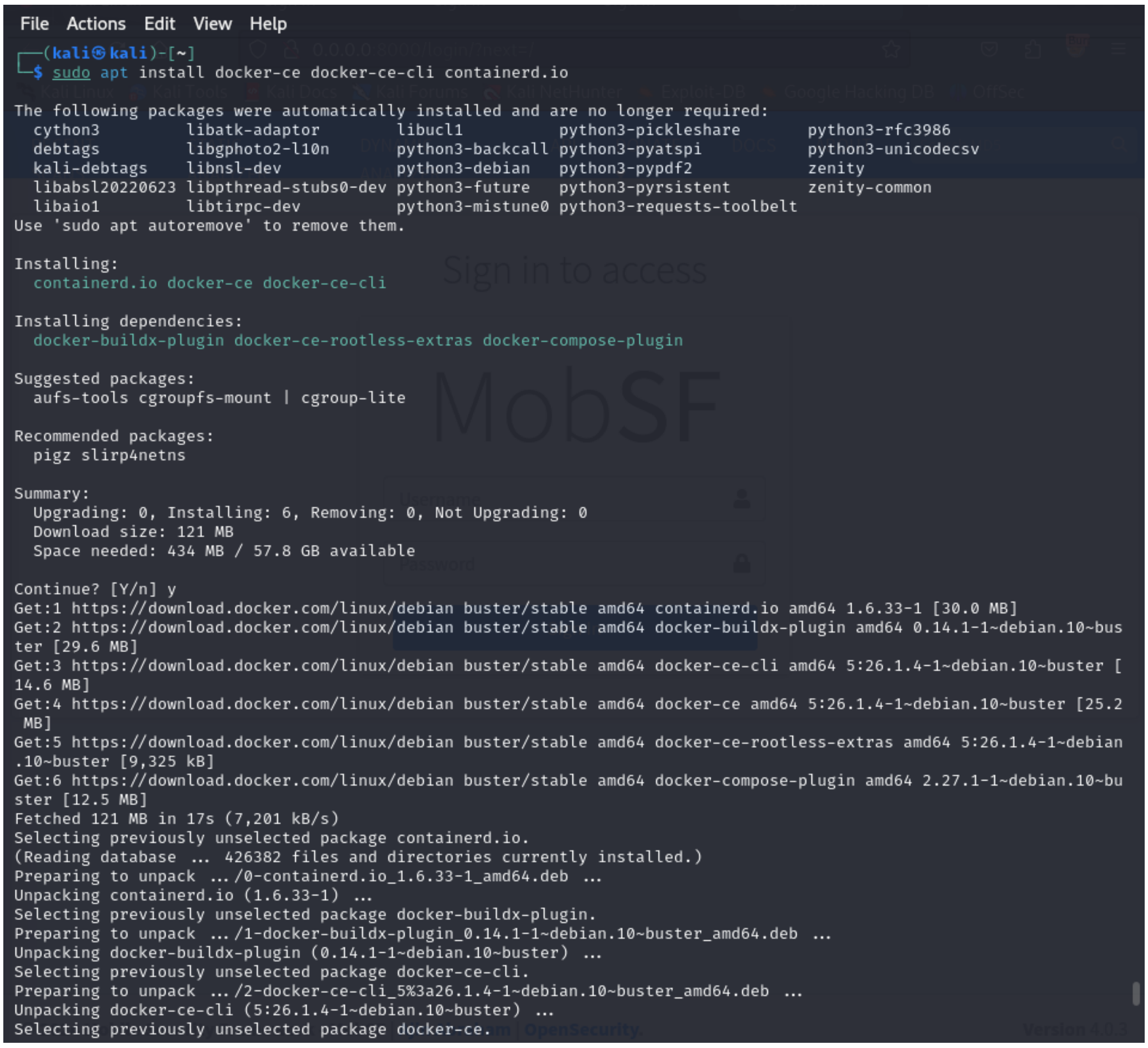
3.1 PREPARATION AND SETUP

To effectively utilize MobSF for static analysis, it is essential to follow a structured preparation and setup process:

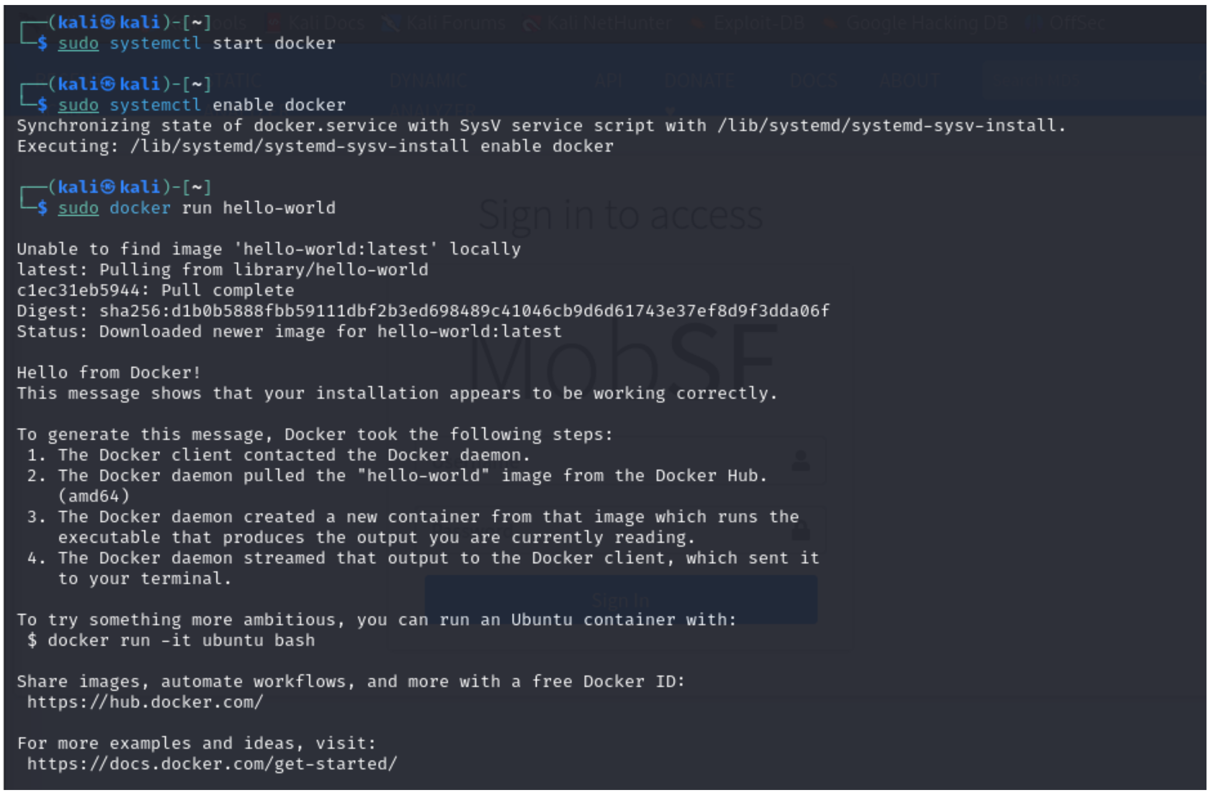
1. ENVIRONMENT SETUP:

- Ensure you have a stable environment for running MobSF. MobSF can be run on various platforms, including Windows, Linux, and macOS.

- Install the necessary dependencies, such as Python, Docker (optional but recommended for easier setup), and other required libraries.



```
┌──(kali㉿kali)-[~]
└─$ sudo apt install docker-ce docker-ce-cli containerd.io

The following packages were automatically installed and are no longer required:
  cython3         libatk-adaptor        libucl1          python3-pickleshare      python3-rfc3986
  debtags         libgphoto2-l10n       python3-backcall python3-pyatspi          python3-unicodecsv
  kali-debtags    libnsl-dev            python3-debian   python3-pypdf2           zenity
  libabsl20220623 libpthread-stubs0-dev python3-future   python3-pyrsistent       zenity-common
  libaio1         libtirpc-dev          python3-mistune0 python3-requests-toolbelt
Use 'sudo apt autoremove' to remove them.

Installing:
  containerd.io docker-ce docker-ce-cli

Installing dependencies:
  docker-buildx-plugin docker-ce-rootless-extras docker-compose-plugin

Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite

Recommended packages:
  pigz slirp4netns

Summary:
  Upgrading: 0, Installing: 6, Removing: 0, Not Upgrading: 0
  Download size: 121 MB
  Space needed: 434 MB / 57.8 GB available

Continue? [Y/n] y
Get:1 https://download.docker.com/linux/debian buster/stable amd64 containerd.io amd64 1.6.33-1 [30.0 MB]
Get:2 https://download.docker.com/linux/debian buster/stable amd64 docker-buildx-plugin amd64 0.14.1-1~debian.10~bus
ter [29.6 MB]
Get:3 https://download.docker.com/linux/debian buster/stable amd64 docker-ce-cli amd64 5:26.1.4-1~debian.10~buster [
14.6 MB]
Get:4 https://download.docker.com/linux/debian buster/stable amd64 docker-ce amd64 5:26.1.4-1~debian.10~buster [25.2
 MB]
Get:5 https://download.docker.com/linux/debian buster/stable amd64 docker-ce-rootless-extras amd64 5:26.1.4-1~debian
.10~buster [9,325 kB]
Get:6 https://download.docker.com/linux/debian buster/stable amd64 docker-compose-plugin amd64 2.27.1-1~debian.10~bu
ster [12.5 MB]
Fetched 121 MB in 17s (7,201 kB/s)
Selecting previously unselected package containerd.io.
(Reading database ... 426382 files and directories currently installed.)
Preparing to unpack .../0-containerd.io_1.6.33-1_amd64.deb ...
Unpacking containerd.io (1.6.33-1) ...
Selecting previously unselected package docker-buildx-plugin.
Preparing to unpack .../1-docker-buildx-plugin_0.14.1-1~debian.10~buster_amd64.deb ...
Unpacking docker-buildx-plugin (0.14.1-1~debian.10~buster) ...
Selecting previously unselected package docker-ce-cli.
Preparing to unpack .../2-docker-ce-cli_5%3a26.1.4-1~debian.10~buster_amd64.deb ...
Unpacking docker-ce-cli (5:26.1.4-1~debian.10~buster) ...
Selecting previously unselected package docker-ce.
```



```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl start docker

┌──(kali㉿kali)-[~]
└─$ sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker

┌──(kali㉿kali)-[~]
└─$ sudo docker run hello-world

Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:d1b0b5888fbb59111dbf2b3ed698489c41046cb9d6d61743e37ef8d9f3dda06f
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

2. INSTALLING MOBSF:

- Download the latest version of MobSF from its [GitHub repository](https://github.com/MobSF/Mobile-Security-Framework-MobSF).

- Follow the installation instructions provided in the repository. This typically involves cloning the repository and running the setup scripts.

3. CONFIGURING MOBSF:

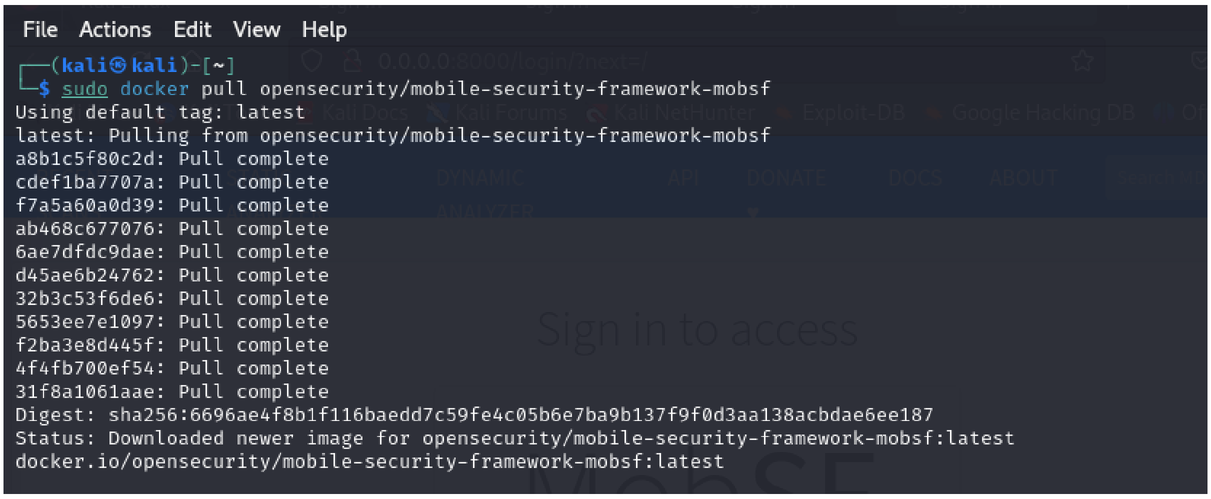
- Configure MobSF settings according to your needs. This includes setting up the database, configuring network settings, and adjusting analysis parameters.

- Ensure that MobSF has access to the necessary tools and SDKs for analyzing Android and iOS applications.

4. PREPARING THE MOBILE APPLICATIONS:

- Obtain the APK (Android) and IPA (iOS) files of the mobile applications you intend to analyze.

- Ensure that the applications are ready for analysis, with no obfuscation or encryption that might hinder the static analysis process.



```
┌──(kali㉿kali)-[~]
└─$ sudo docker pull opensecurity/mobile-security-framework-mobsf
Using default tag: latest
latest: Pulling from opensecurity/mobile-security-framework-mobsf
a8b1c5f80c2d: Pull complete
cdef1ba7707a: Pull complete
f7a5a60a0d39: Pull complete
ab468c677076: Pull complete
6ae7dfdc9dae: Pull complete
d45ae6b24762: Pull complete
32b3c53f6de6: Pull complete
5653ee7e1097: Pull complete
f2ba3e8d445f: Pull complete
4f4fb700ef54: Pull complete
31f8a1061aae: Pull complete
Digest: sha256:6696ae4f8b1f116baedd7c59fe4c05b6e7ba9b137f9f0d3aa138acbdae6ee187
Status: Downloaded newer image for opensecurity/mobile-security-framework-mobsf:latest
docker.io/opensecurity/mobile-security-framework-mobsf:latest
```

3.2 STATIC ANALYSIS PROCESS

The static analysis process using MobSF involves several key steps:

1. UPLOADING THE APPLICATION:

- Start the MobSF server and access the MobSF web interface.

- Upload the APK or IPA file to the MobSF interface. MobSF will automatically start the analysis process once the file is uploaded.

2. ANALYZING THE APPLICATION:

- MobSF performs a series of automated checks and scans on the uploaded application. This includes decompiling the APK/IPA, extracting the source code, and analyzing the code for security vulnerabilities.

- The analysis covers various aspects, such as code quality, insecure coding practices, binary file integrity, certificate validity, and potential malware.

3. GENERATING REPORTS:

- Once the analysis is complete, MobSF generates detailed reports summarizing the findings.

- The reports include identified vulnerabilities, their severity levels, and recommendations for remediation.

- Additional insights, such as API usage, permission analysis, and potential data leaks, are also included in the reports.

4. REVIEWING AND INTERPRETING RESULTS:

- Review the generated reports to understand the identified issues and their potential impact on the application's security.

- Prioritize the remediation efforts based on the severity of the vulnerabilities and the criticality of the application.

4. STATIC ANALYSIS PROCESS USING MOBSF

4.1 APK/IPA SCANNING

MobSF efficiently scans APK (Android) and IPA (iOS) files to start the static analysis process. When an application file is uploaded, MobSF decompiles it to access its underlying structure and content, extracting manifest files, resource files, and other components necessary for thorough analysis.

4.2 CODE ANALYSIS

In the code analysis phase, MobSF inspects the decompiled source code for security vulnerabilities and insecure coding practices. This includes:

- Identifying hardcoded secrets, such as API keys and passwords.

- Detecting the use of insecure or deprecated APIs.

- Examining code for common security flaws like SQL injection and cross-site scripting (XSS).

- Reviewing the application's permission requests for security implications.

4.3 CERTIFICATE ANALYSIS

Certificate analysis in MobSF verifies the integrity and validity of the application's digital certificates by:

- Checking the certificate chain for proper signing and trust.

- Verifying expiration dates to prevent the use of outdated certificates.

- Ensuring certificates are not self-signed unless necessary.

- Detecting certificate pinning to prevent man-in-the-middle attacks.

4.4 MALWARE ANALYSIS

MobSF performs malware analysis to detect malicious components or behaviors within the application by:

- Scanning for known malware signatures using integrated databases.

- Analyzing the application's behavior for suspicious activities, such as unauthorized data access or communication with malicious servers.

- Checking for embedded payloads, backdoors, or other malicious code.

5. INTERPRETATION OF RESULTS

5.1 IMPACT OF IDENTIFIED VULNERABILITIES

Identifying vulnerabilities in mobile applications through tools like MobSF can have significant impacts on security:

- Data Breaches: Vulnerabilities may expose sensitive user data to unauthorized access.

- Financial Loss: Exploited vulnerabilities can lead to financial losses through fraud or theft.

- Reputation Damage: Security breaches can damage the app's reputation and erode user trust.

- Legal and Compliance Issues: Non-compliance with data protection regulations may result in legal consequences.

5.2 RECOMMENDATIONS FOR MITIGATION

To mitigate identified vulnerabilities, follow these recommendations:

- **Patch Management:** Regularly update the application with security patches and fixes.

- **Secure Coding Practices:** Adhere to secure coding guidelines to prevent common vulnerabilities.

- **Penetration Testing:** Conduct regular penetration testing to identify and address security weaknesses.

- **Encryption:** Encrypt sensitive data in transit and at rest to protect against unauthorized access.

- **User Education:** Educate users about security best practices and potential risks.

5.3 BEST PRACTICES FOR SECURE MOBILE APP DEVELOPMENT

Implement these best practices to enhance mobile app security:

- **Use Secure Libraries and APIs:** Use trusted libraries and APIs with strong security measures.

- **Authentication and Authorization:** Implement robust authentication and authorization mechanisms.

- **Data Minimization:** Collect only necessary data and minimize storage of sensitive information.

- **Code Reviews:** Conduct regular code reviews to identify and fix security issues early.

- **Privacy by Design:** Incorporate privacy protections throughout the app development lifecycle.

- **Monitoring and Logging:** Implement logging and monitoring to detect and respond to security incidents.

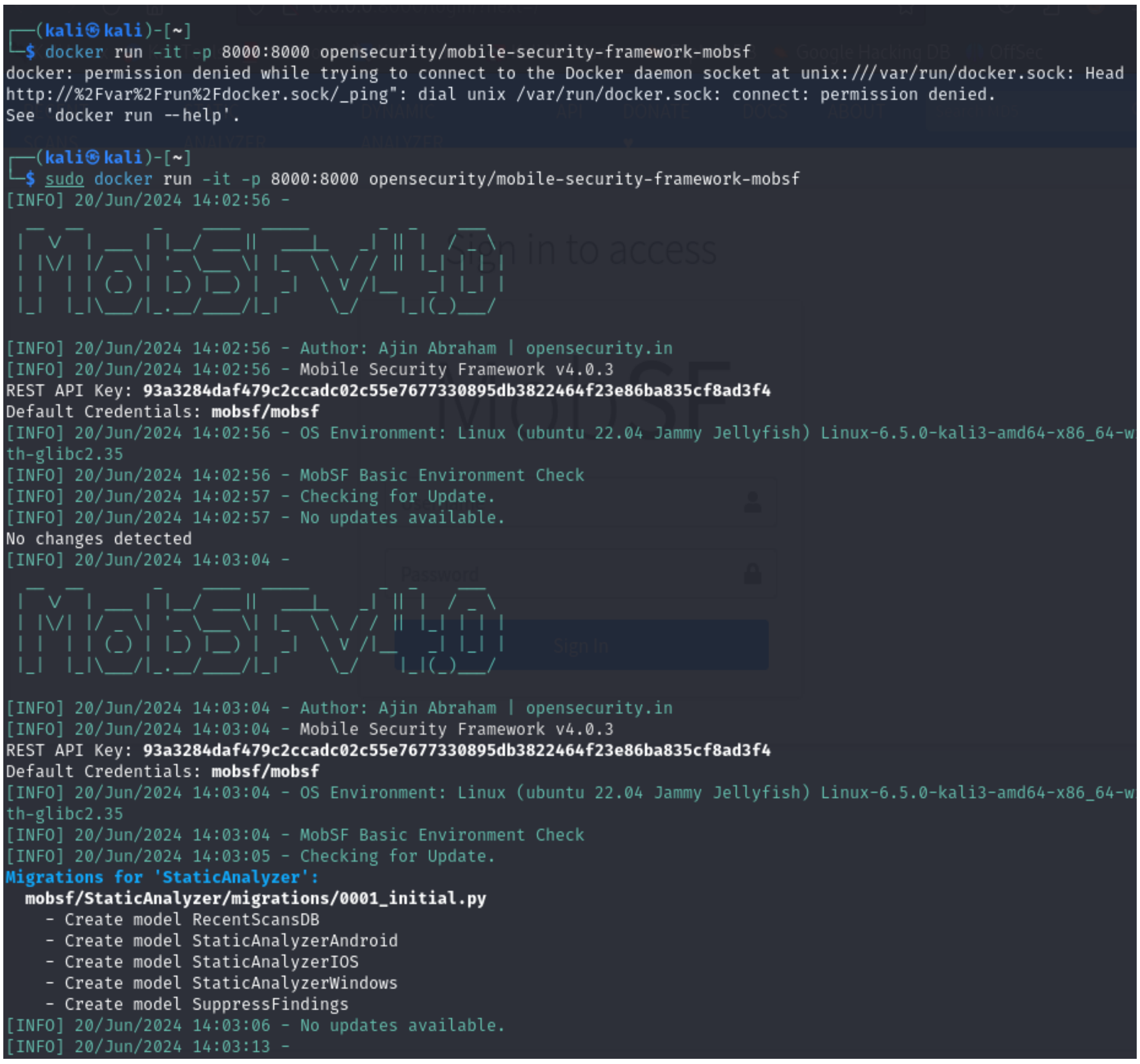
6. CONCLUSION

6.1 IMPORTANCE OF STATIC ANALYSIS IN MOBILE SECURITY

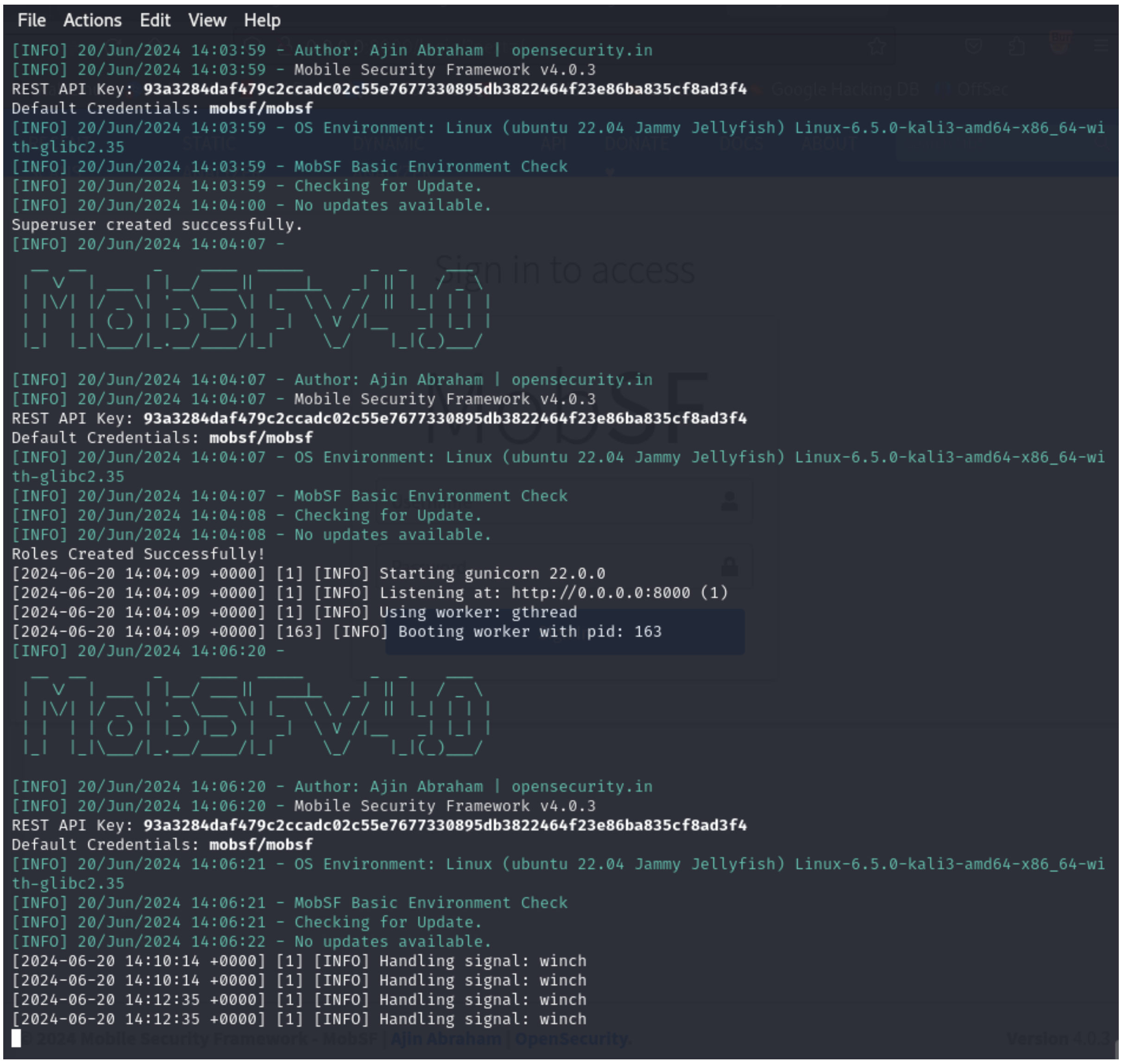
Static analysis plays a crucial role in ensuring the security and reliability of mobile applications throughout their development lifecycle. By examining the application's code and binaries without execution, static analysis helps identify potential security vulnerabilities, design flaws, and coding errors that could compromise the application's security. This proactive approach allows developers and security professionals to detect and address issues early, reducing the likelihood of vulnerabilities being exploited in production environments.

Furthermore, static analysis contributes to maintaining compliance with security standards and regulations by identifying deviations from best practices and security guidelines. It provides a foundation for implementing secure coding practices, ensuring that mobile applications are built with security in mind from the outset. This approach not only protects sensitive user data but also safeguards the reputation of the organization by minimizing the risk of security breaches and data leaks.

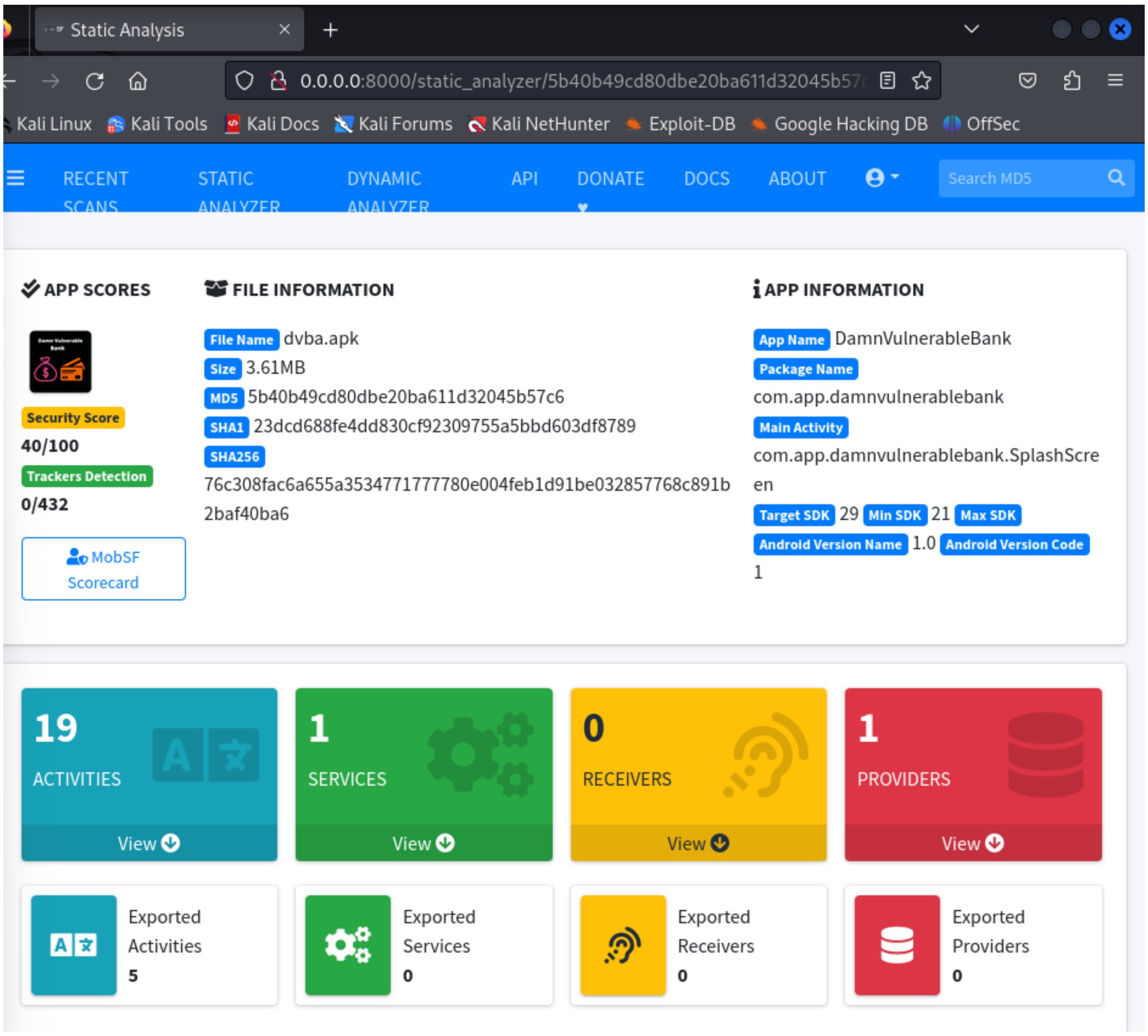
6.2 SCREENSHOTS



```
┌──(kali㉿kali)-[~]
└─$ docker run -it -p 8000:8000 opensecurity/mobile-security-framework-mobsf
docker: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Head
http://%2Fvar%2Frun%2Fdocker.sock/_ping": dial unix /var/run/docker.sock: connect: permission denied.
See 'docker run --help'.

┌──(kali㉿kali)-[~]
└─$ sudo docker run -it -p 8000:8000 opensecurity/mobile-security-framework-mobsf
[INFO] 20/Jun/2024 14:02:56 -
  __  __       _    ____  _____       _  _    ___  
 |  \/  | ___ | |__/ ___||  ___|_   _| || |  / _ \ 
 | |\/| |/ _ \| '_ \___ \| |_  \ \ / / || |_| | | |
 | |  | | (_) | |_) |__) |  _|  \ V /|__   _| |_| |
 |_|  |_|\___/|_.__/____/|_|     \_/    |_|(_)___/ 

[INFO] 20/Jun/2024 14:02:56 - Author: Ajin Abraham | opensecurity.in
[INFO] 20/Jun/2024 14:02:56 - Mobile Security Framework v4.0.3
REST API Key: 93a3284daf479c2ccadc02c55e7677330895db3822464f23e86ba835cf8ad3f4
Default Credentials: mobsf/mobsf
[INFO] 20/Jun/2024 14:02:56 - OS Environment: Linux (ubuntu 22.04 Jammy Jellyfish) Linux-6.5.0-kali3-amd64-x86_64-w
th-glibc2.35
[INFO] 20/Jun/2024 14:02:56 - MobSF Basic Environment Check
[INFO] 20/Jun/2024 14:02:57 - Checking for Update.
[INFO] 20/Jun/2024 14:02:57 - No updates available.
No changes detected
[INFO] 20/Jun/2024 14:03:04 -
  __  __       _    ____  _____       _  _    ___  
 |  \/  | ___ | |__/ ___||  ___|_   _| || |  / _ \ 
 | |\/| |/ _ \| '_ \___ \| |_  \ \ / / || |_| | | |
 | |  | | (_) | |_) |__) |  _|  \ V /|__   _| |_| |
 |_|  |_|\___/|_.__/____/|_|     \_/    |_|(_)___/ 

[INFO] 20/Jun/2024 14:03:04 - Author: Ajin Abraham | opensecurity.in
[INFO] 20/Jun/2024 14:03:04 - Mobile Security Framework v4.0.3
REST API Key: 93a3284daf479c2ccadc02c55e7677330895db3822464f23e86ba835cf8ad3f4
Default Credentials: mobsf/mobsf
[INFO] 20/Jun/2024 14:03:04 - OS Environment: Linux (ubuntu 22.04 Jammy Jellyfish) Linux-6.5.0-kali3-amd64-x86_64-w
th-glibc2.35
[INFO] 20/Jun/2024 14:03:04 - MobSF Basic Environment Check
[INFO] 20/Jun/2024 14:03:05 - Checking for Update.
Migrations for 'StaticAnalyzer':
  mobsf/StaticAnalyzer/migrations/0001_initial.py
    - Create model RecentScansDB
    - Create model StaticAnalyzerAndroid
    - Create model StaticAnalyzerIOS
    - Create model StaticAnalyzerWindows
    - Create model SuppressFindings
[INFO] 20/Jun/2024 14:03:06 - No updates available.
[INFO] 20/Jun/2024 14:03:13 -
```



```
[INFO] 20/Jun/2024 14:03:59 - Author: Ajin Abraham | opensecurity.in
[INFO] 20/Jun/2024 14:03:59 - Mobile Security Framework v4.0.3
REST API Key: 93a3284daf479c2ccadc02c55e7677330895db3822464f23e86ba835cf8ad3f4
Default Credentials: mobsf/mobsf
[INFO] 20/Jun/2024 14:03:59 - OS Environment: Linux (ubuntu 22.04 Jammy Jellyfish) Linux-6.5.0-kali3-amd64-x86_64-wi
th-glibc2.35
[INFO] 20/Jun/2024 14:03:59 - MobSF Basic Environment Check
[INFO] 20/Jun/2024 14:03:59 - Checking for Update.
[INFO] 20/Jun/2024 14:04:00 - No updates available.
Superuser created successfully.
[INFO] 20/Jun/2024 14:04:07 -
  __  __       _      _____ ______        _  _    ___
 |  \/  |     | |    / ____|  ____|      | || |  / _ \
 | \  / | ___ | |__ | (___ | |__  __   __| || |_| | | |
 | |\/| |/ _ \| '_ \ \___ \|  __| \ \ / /|__   _| | | |
 | |  | | (_) | |_) |____) | |     \ V /    | |_| |_| |
 |_|  |_|\___/|_.__/|_____/|_|      \_/     |_(_)\___/

[INFO] 20/Jun/2024 14:04:07 - Author: Ajin Abraham | opensecurity.in
[INFO] 20/Jun/2024 14:04:07 - Mobile Security Framework v4.0.3
REST API Key: 93a3284daf479c2ccadc02c55e7677330895db3822464f23e86ba835cf8ad3f4
Default Credentials: mobsf/mobsf
[INFO] 20/Jun/2024 14:04:07 - OS Environment: Linux (ubuntu 22.04 Jammy Jellyfish) Linux-6.5.0-kali3-amd64-x86_64-wi
th-glibc2.35
[INFO] 20/Jun/2024 14:04:07 - MobSF Basic Environment Check
[INFO] 20/Jun/2024 14:04:08 - Checking for Update.
[INFO] 20/Jun/2024 14:04:08 - No updates available.
Roles Created Successfully!
[2024-06-20 14:04:09 +0000] [1] [INFO] Starting gunicorn 22.0.0
[2024-06-20 14:04:09 +0000] [1] [INFO] Listening at: http://0.0.0.0:8000 (1)
[2024-06-20 14:04:09 +0000] [1] [INFO] Using worker: gthread
[2024-06-20 14:04:09 +0000] [163] [INFO] Booting worker with pid: 163
[INFO] 20/Jun/2024 14:06:20 -
  __  __       _      _____ ______        _  _    ___
 |  \/  |     | |    / ____|  ____|      | || |  / _ \
 | \  / | ___ | |__ | (___ | |__  __   __| || |_| | | |
 | |\/| |/ _ \| '_ \ \___ \|  __| \ \ / /|__   _| | | |
 | |  | | (_) | |_) |____) | |     \ V /    | |_| |_| |
 |_|  |_|\___/|_.__/|_____/|_|      \_/     |_(_)\___/

[INFO] 20/Jun/2024 14:06:20 - Author: Ajin Abraham | opensecurity.in
[INFO] 20/Jun/2024 14:06:20 - Mobile Security Framework v4.0.3
REST API Key: 93a3284daf479c2ccadc02c55e7677330895db3822464f23e86ba835cf8ad3f4
Default Credentials: mobsf/mobsf
[INFO] 20/Jun/2024 14:06:21 - OS Environment: Linux (ubuntu 22.04 Jammy Jellyfish) Linux-6.5.0-kali3-amd64-x86_64-wi
th-glibc2.35
[INFO] 20/Jun/2024 14:06:21 - MobSF Basic Environment Check
[INFO] 20/Jun/2024 14:06:21 - Checking for Update.
[INFO] 20/Jun/2024 14:06:22 - No updates available.
[2024-06-20 14:10:14 +0000] [1] [INFO] Handling signal: winch
[2024-06-20 14:10:14 +0000] [1] [INFO] Handling signal: winch
[2024-06-20 14:12:35 +0000] [1] [INFO] Handling signal: winch
[2024-06-20 14:12:35 +0000] [1] [INFO] Handling signal: winch
```



Static Analysis
0.0.0.0:8000/static_analyzer/5b40b49cd80dbe20ba611d32045b57
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
RECENT SCANS STATIC ANALYZER DYNAMIC ANALYZER API DONATE ♥ DOCS ABOUT
Search MD5

APP SCORES

Security Score
40/100

Trackers Detection
0/432

MobSF Scorecard

FILE INFORMATION

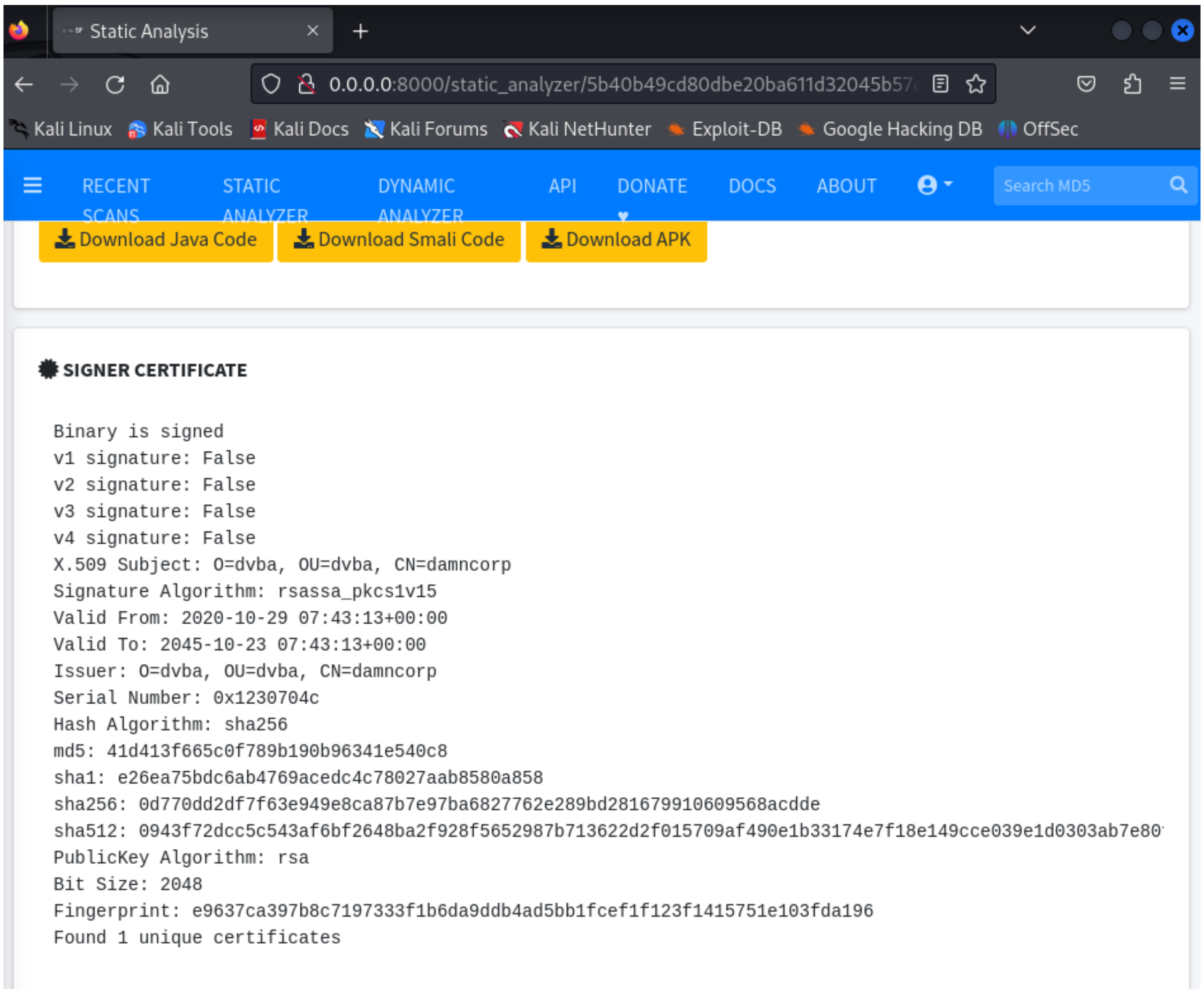
File Name dvba.apk
Size 3.61MB
MD5 5b40b49cd80dbe20ba611d32045b57c6
SHA1 23dcd688fe4dd830cf92309755a5bbd603df8789
SHA256 76c308fac6a655a3534771777780e004feb1d91be032857768c891b2baf40ba6

APP INFORMATION

App Name DamnVulnerableBank
Package Name com.app.damnvulnerablebank
Main Activity com.app.damnvulnerablebank.SplashScreen
Target SDK 29 Min SDK 21 Max SDK
Android Version Name 1.0 Android Version Code 1

19 ACTIVITIES View
1 SERVICES View
0 RECEIVERS View
1 PROVIDERS View

Exported Activities 5
Exported Services 0
Exported Receivers 0
Exported Providers 0



SIGNER CERTIFICATE

```
Binary is signed
v1 signature: False
v2 signature: False
v3 signature: False
v4 signature: False
X.509 Subject: O=dvba, OU=dvba, CN=damncorp
Signature Algorithm: rsassa_pkcs1v15
Valid From: 2020-10-29 07:43:13+00:00
Valid To: 2045-10-23 07:43:13+00:00
Issuer: O=dvba, OU=dvba, CN=damncorp
Serial Number: 0x1230704c
Hash Algorithm: sha256
md5: 41d413f665c0f789b190b96341e540c8
sha1: e26ea75bdc6ab4769acedc4c78027aab8580a858
sha256: 0d770dd2df7f63e949e8ca87b7e97ba6827762e289bd281679910609568acdde
sha512: 0943f72dcc5c543af6bf2648ba2f928f5652987b713622d2f015709af490e1b33174e7f18e149cce039e1d0303ab7e80
PublicKey Algorithm: rsa
Bit Size: 2048
Fingerprint: e9637ca397b8c7197333f1b6da9ddb4ad5bb1fcef1f123f1415751e103fda196
Found 1 unique certificates
```