

Names: Yasar Azimi

ID: #44158

Final Project: Mobile App Security Assessment

Dr. Ala Abdulaziz

Class: ITC 470

2023/11/20

**The Link to the video:** https://drive.google.com/file/d/1-MJUHnWpKna9Cm8o7OC5-oWE1DC10K-j/view?usp=sharing

**Executive Summary**

**Introduction:**

Before we start, we should be aware that the vulnerabilities found in this assessment should not be exploited as that goes against the code of penetration testing and ethical hacking. This security assessment report provides a comprehensive evaluation of the security aspects of the Android application, Disney+ App, developed by Walt Disney. The objective of this assessment is to identify potential vulnerabilities, assess the overall security posture, and make recommendations for enhancing the security of the application. The tools used for the assessment include "MobSF," "Genymotion,", "Oracle VM Virtual Box", and “OWASP ZAP”.

Mobile applications have become an integral part of modern life, granting users access to information and services directly on their mobile devices. While these apps offer convenience, they also handle sensitive user data, making their security crucial. Security vulnerabilities in mobile apps can lead to data breaches, privacy violations, financial losses, and reputational damage to both users and developers. The assessment's scope encompasses a thorough examination of the Disney+ app to uncover vulnerabilities and weaknesses in the application's code, architecture, and security controls.

**Methodology & Findings:**

The assessment of the Android application followed a structured methodology, incorporating both manual and automated testing techniques. The choice of tools and techniques was driven by the need to comprehensively evaluate security aspects. The Mobile Security Framework (MobSF) played a pivotal role in automating various aspects of the assessment, including static and dynamic analysis, API and server assessment, and third-party component analysis.

Other tools, such as "Genymotion", "Oracle VM Virtual Box", and “OWASP ZAP” were used for dynamic analysis. Genymotion served as the Android emulator for dynamic analysis, while the Virtual VM Box provided a controlled environment for monitoring application behavior and logs. While OWASP ZAP was used to attack the weaknesses we found in the assessment.

The assessment methodology included the following phases:

**Reconnaissance**: Gathering information about the application, including permissions, server security, and publicly available data, to understand the application's ecosystem.

**Static Analysis**: Reviewing the source code to identify vulnerabilities, insecure coding practices, data leakage, and improper permission handling.

**Dynamic Analysis**: Analysing the application's behavior in a controlled environment to detect runtime vulnerabilities, data storage issues, and vulnerabilities arising from user interactions.

**API and Server Assessment**: Evaluating the application's communication with external APIs and servers, focusing on authentication, data transmission, and validation checks.

**Penetration Testing**: Attempting to exploit identified vulnerabilities to assess their real-world impact.

**Vulnerability Identification & Potential Impacts:**

Two primary vulnerabilities were identified, each with its potential impact:

**Certificate Analysis:**

Description: The application was signed with the v1 signature scheme, making it vulnerable to the Janus vulnerability on Android 5.0-8.0 if signed solely with the v1 signature scheme. Additionally, applications running on Android 5.0-7.0 signed with v1 and v2/v3 schemes are also vulnerable.

Potential Impact: The use of the v1 signature scheme raises concerns about the app's security, particularly on older Android versions. Exploitation of this vulnerability could result in the modification of the application's code or behavior, leading to the installation of malicious software or unauthorized access to user data.

**Network Security:**

Description: Insecure network configuration allowed clear text traffic to all domains, creating a serious threat to data confidentiality and integrity.

Potential Impact: Attackers could intercept and manipulate data during transmission, potentially leading to data leakage, man-in-the-middle attacks, and the compromise of sensitive information. This poses a severe risk to user privacy and data security.

**Recommendations**:

**Network Security**: Strengthen network security to safeguard data transmission. Implement SSL/TLS encryption, certificate pinning, firewall rules, and security headers to enhance network security.

**Certificate Analysis**: Update the certificate and signature scheme to mitigate the Janus vulnerability. Migrate to v2 or v3 signature schemes and regularly renew the application's certificate. Implement strong code signing practices.

**General Security Recommendations**: Implement security training, code reviews, penetration testing, and privacy by design, regular updates, and an incident response plan to enhance overall security.

Implementing these recommendations and best practices will significantly enhance the security of the Disney+ mobile application, protect users from potential threats, and strengthen the application's overall integrity. Regular security assessments and continuous monitoring are essential to ensure ongoing security as new threats emerge.

This assessment report provides valuable insights and a roadmap for enhancing the security of the Disney+ app, demonstrating a commitment to user data protection and maintaining user trust. The security assessment of the Android application, Disney+ App, has provided a comprehensive overview of the application's overall security. The assessment aimed to identify potential vulnerabilities, assess the application's overall security, and make recommendations to enhance its protection. We hope that the report is taken seriously and the vulnerabilities are fixed as soon as possible.

**NOTE: This is the end of the summary and the start of the complete report.**

**Introduction:**

Before we start, we should be aware that the vulnerabilities found in this assessment should not be exploited as that goes against the code of penetration testing and ethical hacking. This security assessment report aims to provide a comprehensive evaluation of the security aspects of the Android application, Disney+ App, which was developed by Walt Disney. The objective of this assessment as was asked of us, is to identify potential vulnerabilities, assess the overall security posture, and make recommendations for enhancing the security of the application. The tools used include (“MobSF”, “Genymotion”, “Oracle VM Virtual Box”, and “OWASP ZAP”).

Nowadays Applications are an integral part of our modern life, which enables us (users) to access various kinds of information, and services in our mobile phone without having to go far from our living space or doing a specific chore. Thanks to these apps everything that we need is at the tip of our fingers, so we should be considering the importance of how secure these apps really are. Since these apps have access to our personal informations whether it’s our bank information, house information or personal information.

If there are any kind of security vulnerability in our mobile apps, they can lead to breaches of data, privacy violations, financial losses and reputational damage to both the user and the developer. The loss of such data can harm ones livelihood as well as put mental pressure on the person due to the risk of their personal info/images being released on the internet.

The scope of this assessment encompasses a thorough examination of Disney+ app in order to uncover vulnerabilities and weaknesses in the application's code, architecture, and security controls. The assessment was carried out through a combination of manual and automated testing, including static and dynamic analysis, API and server assessment, penetration testing, and a detailed review of user data handling.

By allowing us to conduct this assessment, Walt Disney has shown its stance and commitment to the security and privacy of its users. Since identifying and addressing security issues is a crucial step towards ensuring that the app not only functions as intended but also safeguard the users data all in order to maintain the trust of its user base.

The findings and recommendations presented in this report can serve as a valuable resource for Walt Disney to enhance the security of Disney+. Implementing the recommended security measures will not only protect users from potential threats but also strengthen the overall integrity of the application. We hope that our findings are taken seriously by Walt Disney, and that they allocate appropriate resources for the sake of its security. Methodology, Findings, Recommendations, and References sections of this report will provide a detailed account of the assessment process, identified vulnerabilities, and the steps to mitigate them.

**Methodology & findings**:

The assessment of the Android application was conducted following a structured methodology that has incorporated a combination of both manual and automated testing techniques. The choice of tools and techniques was driven by the need to comprehensively evaluate the security aspects of the application and identify potential vulnerabilities. One of the key tools employed in this assessment is the Mobile Security Framework (MobSF).

MobSF was used to automate several aspects of the assessment, including static and dynamic analysis, API and server assessment, and third-party component analysis. It assists in identifying potential vulnerabilities in the application and its dependencies. The automated scans provided by MobSF will gave us a valuable insights into potential security issues, helping the assessment process and identifying the vulnerabilities more efficiently.

Aside from MobSF we have also used “Genymotion” and “virtual VM box”, for the dynamic analysis. The Genymotion is used as an android emulator to use the dynamic analysis on, and the Virtual VM box is used for providing it a space to run and check the logs of what is happening and the codes.

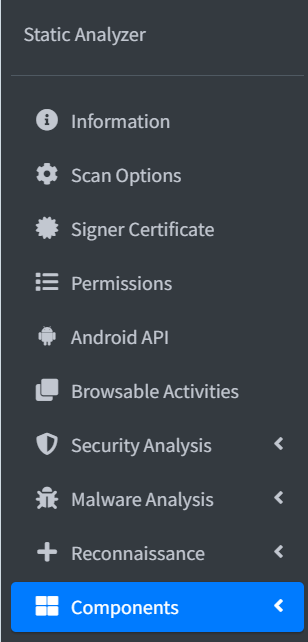
**1. Reconnaissance:**

The initial phase involved gathering different kinds of information about the application, such as its permissions, server security, and any publicly available information. This phase provided an understanding of the application's ecosystem and potential attack vectors.



**2. Static Analysis:**

Source code review was conducted to identify potential vulnerabilities, insecure coding practices, data leakage, and improper permission handling. This was used to reveal security issues within the code base that could be exploited by attackers. Static analysis helps uncover issues that may not be apparent during dynamic analysis.



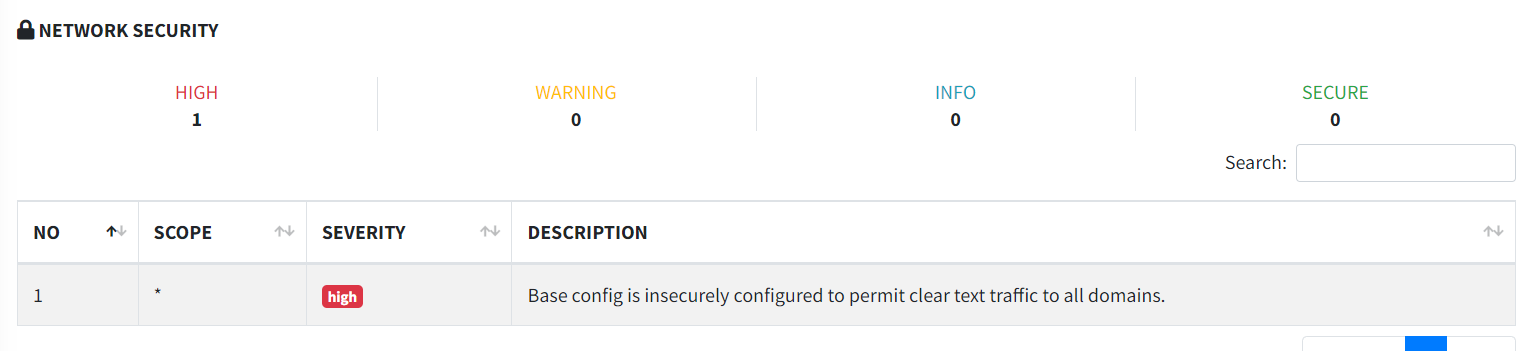




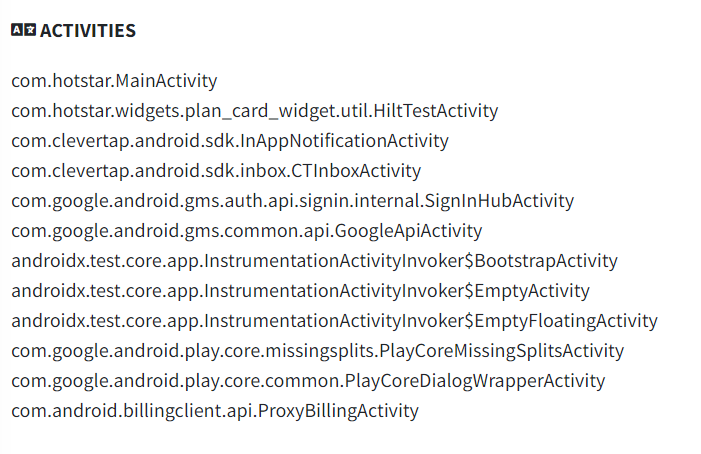
**Signer Certificate:**



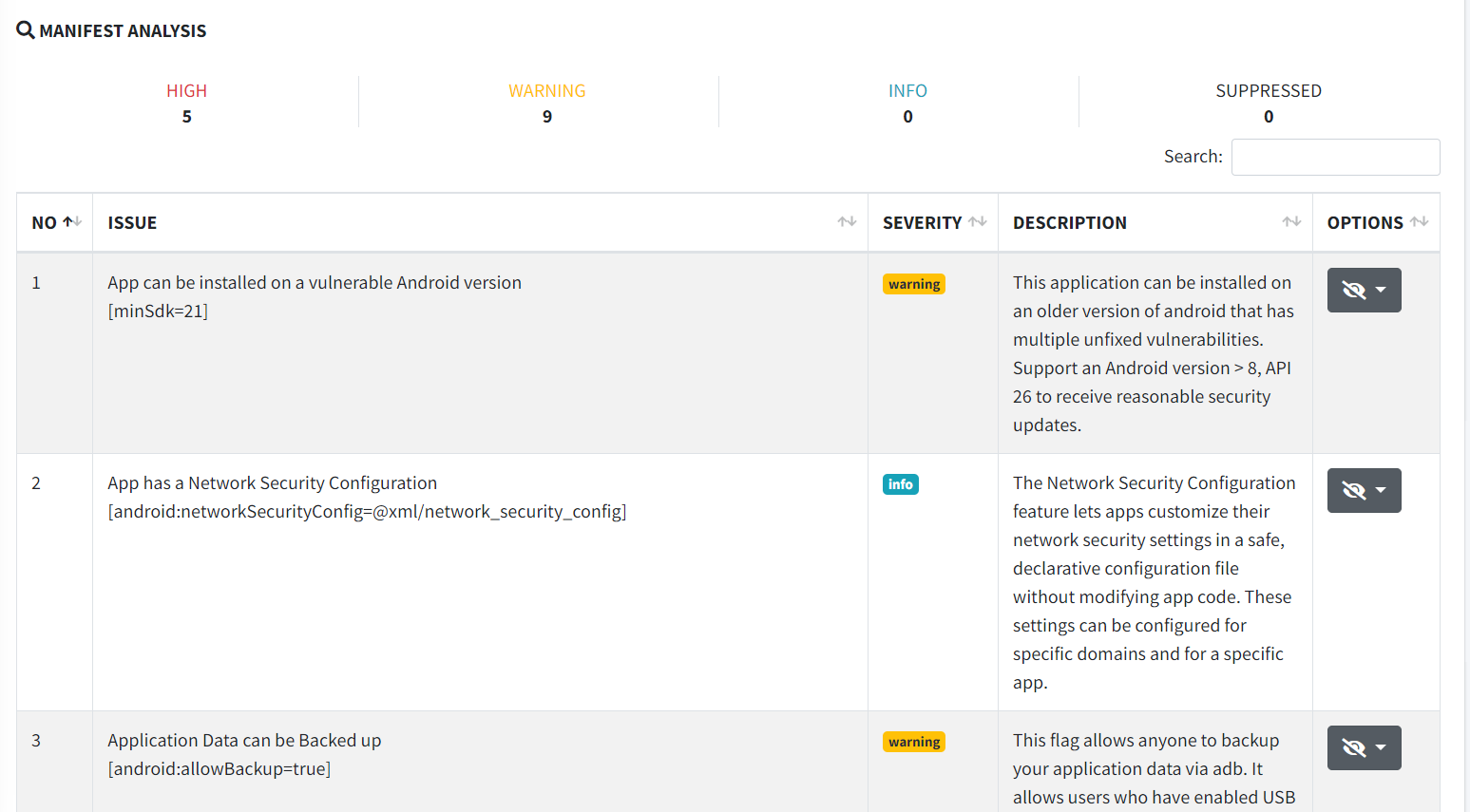
**Network Security**:



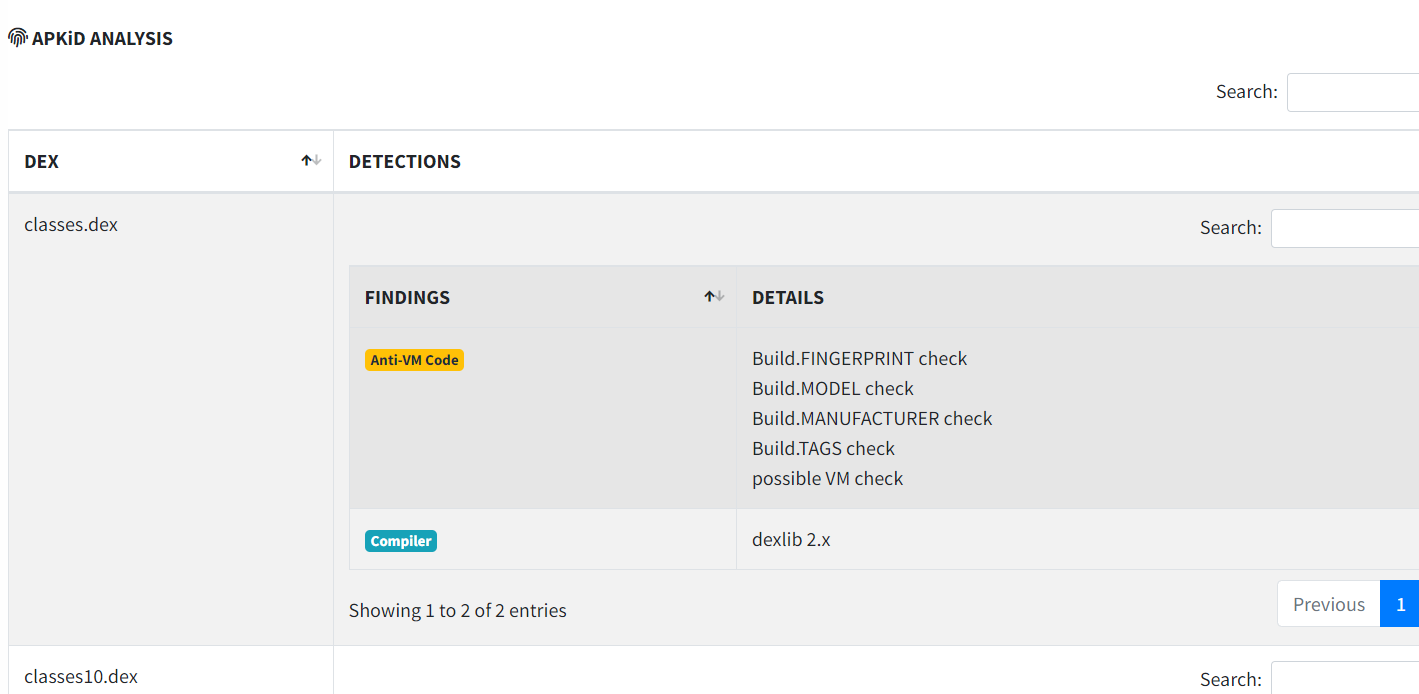
**Activities:**



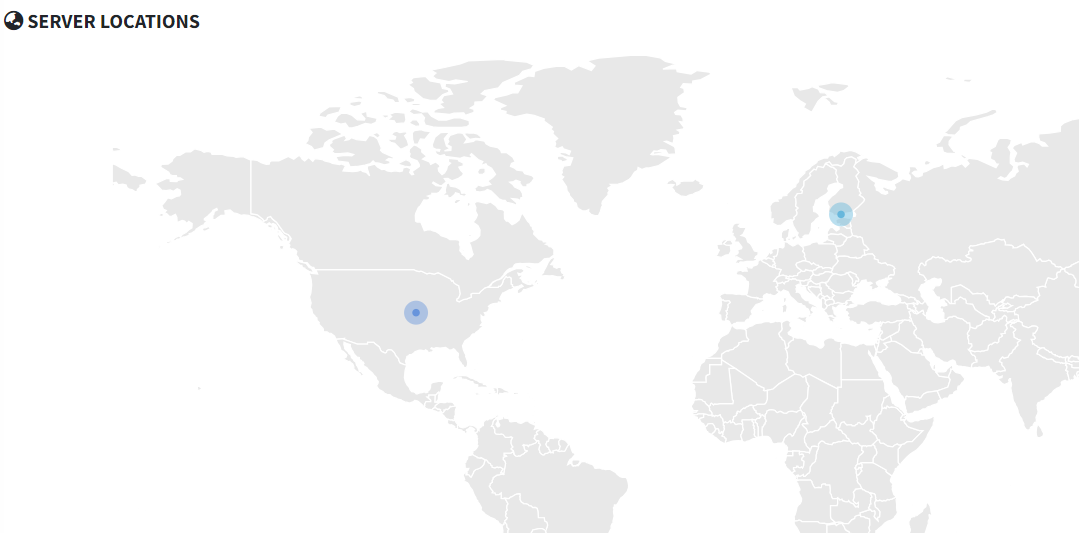
**Manifest Analysis:**



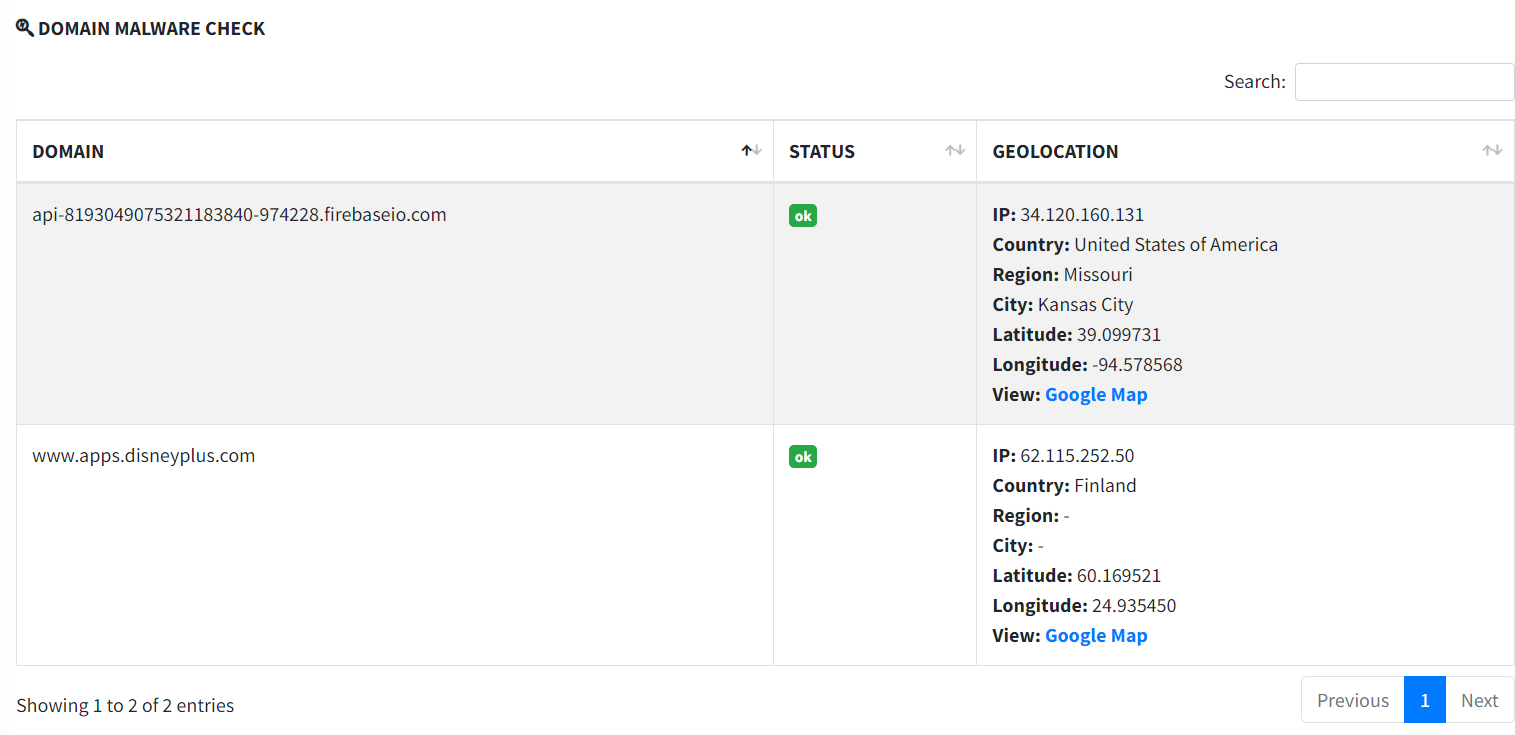
**APKID Analysis:**



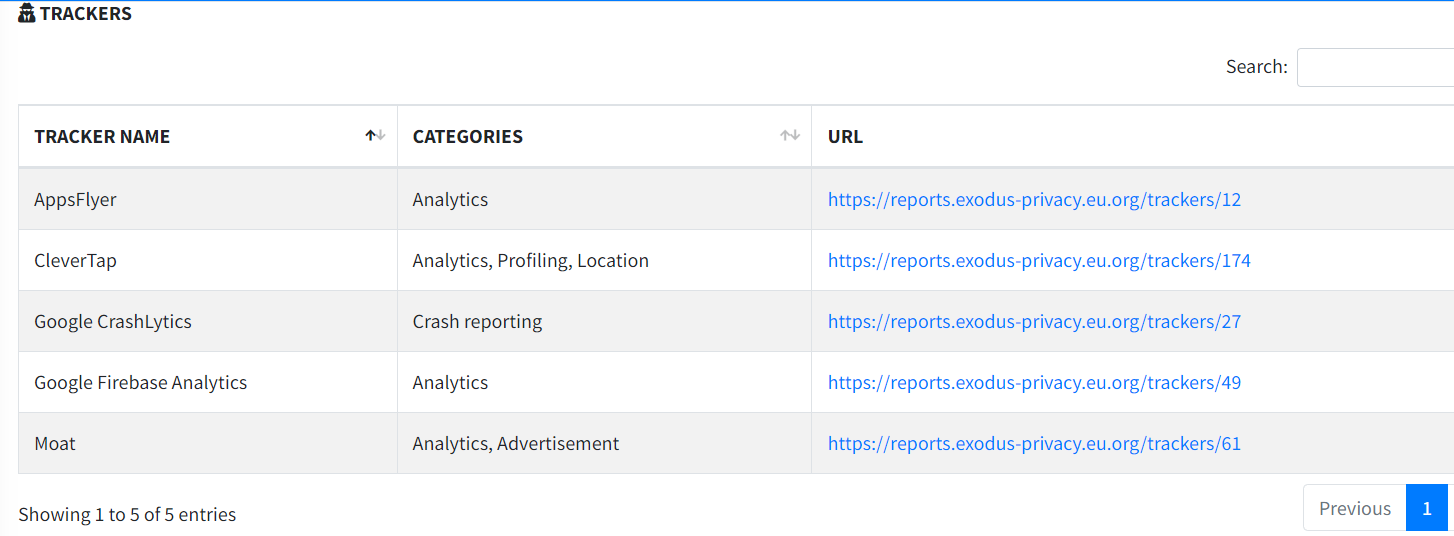
**Server Locations:**



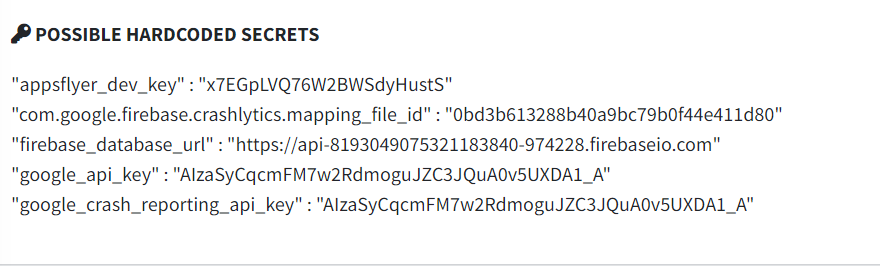
**Domain Malware Check:**



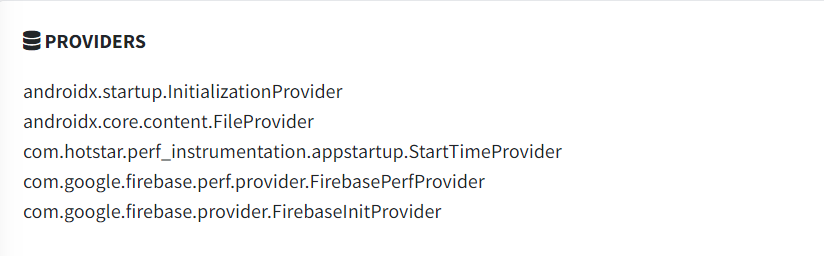
**Trackers:**

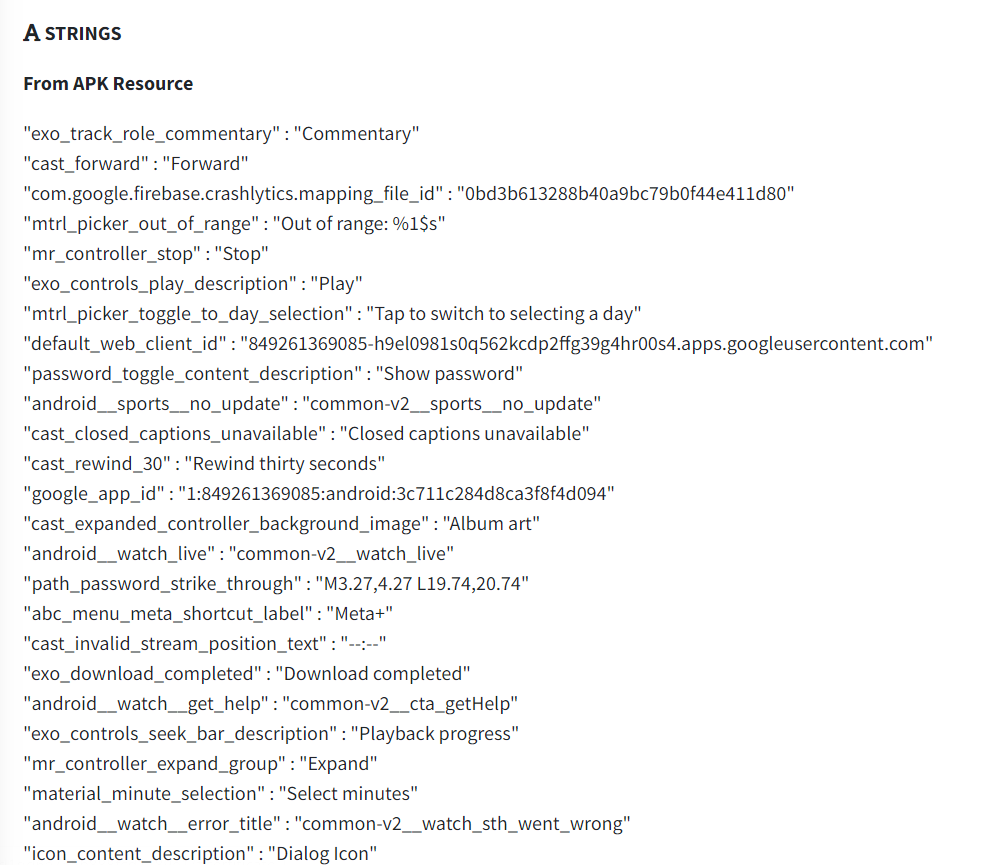


**Codes:**

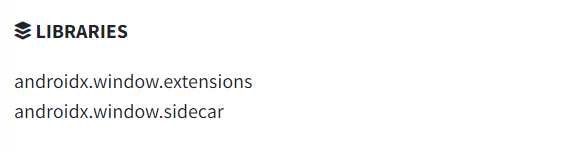


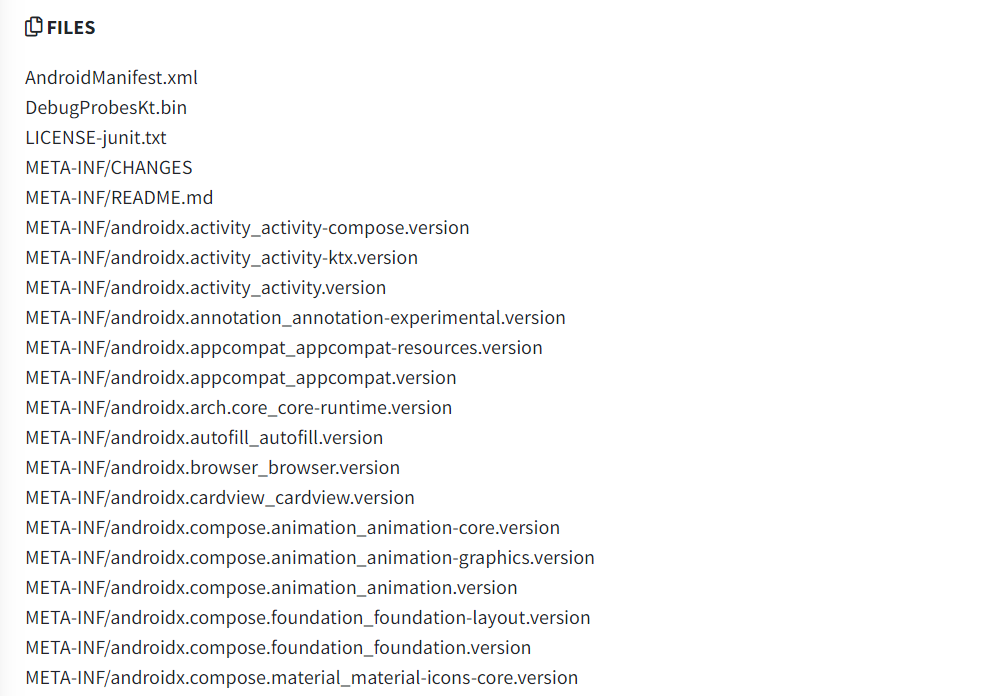
**Providers:**





**Libraries:**

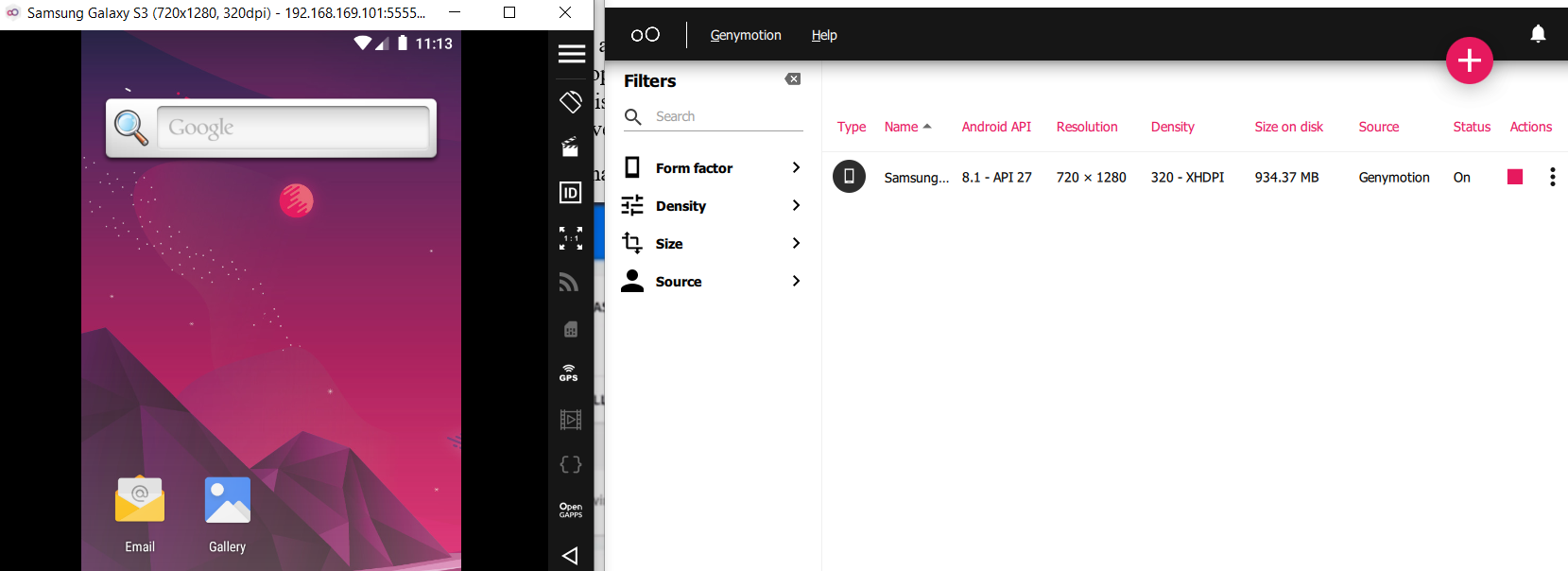


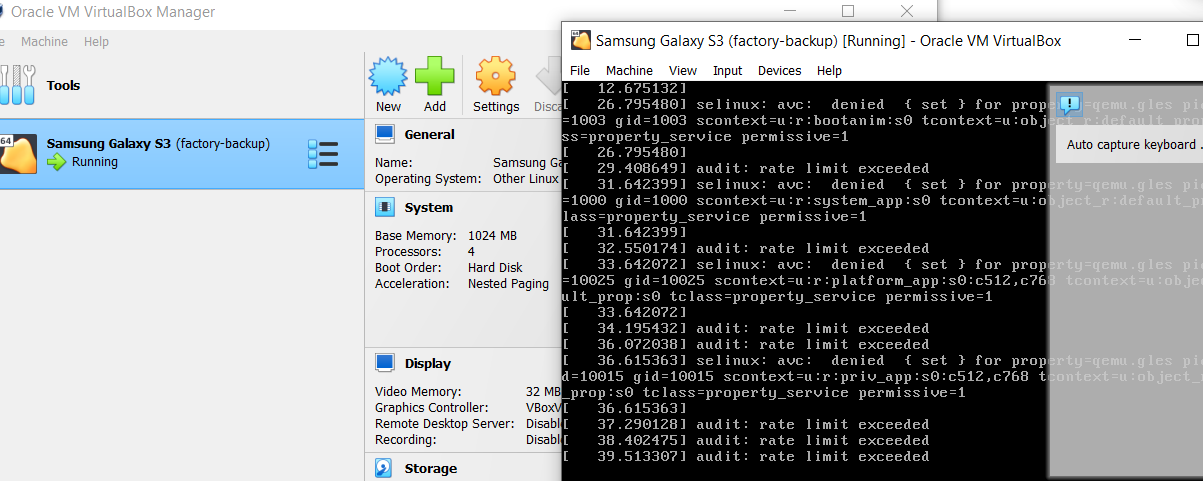


**3. Dynamic Analysis:**

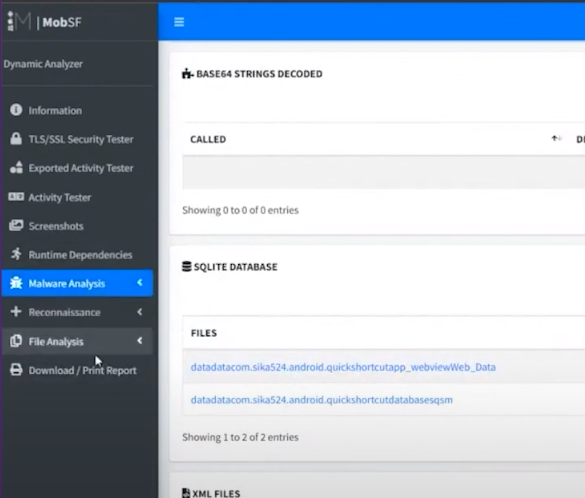
It’s the real-time analysis of the application in a controlled environment was performed to detect runtime vulnerabilities, including improper data storage, network communications, and vulnerabilities arising from user interactions. Dynamic analysis is used to show how the application behaves while being run, exposing any kind of issue that may have not been evident through static analysis alone.

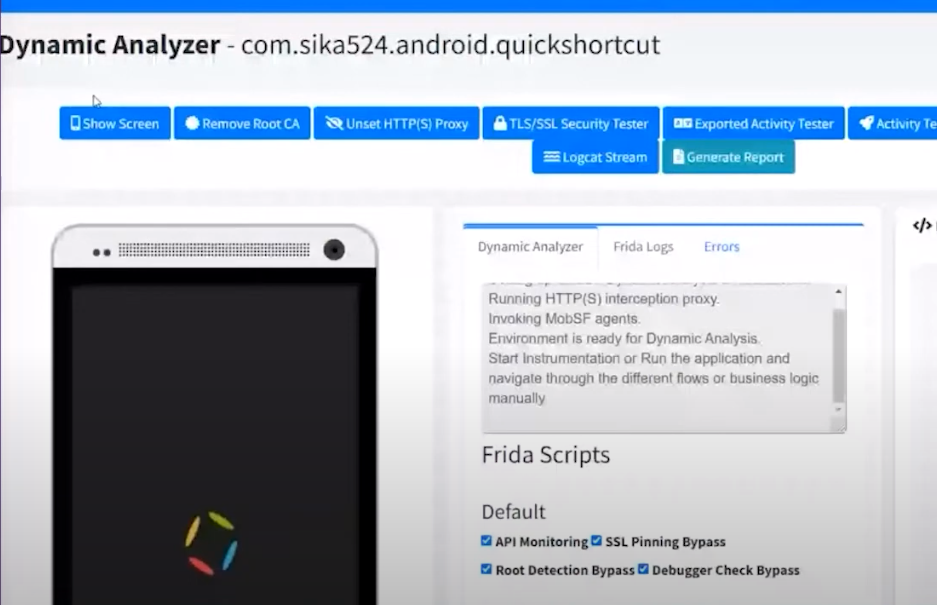
To start running the Dynamic analysis first we have to open Genymotion as our emulator and Virtual VM Box to check on the status of the application.





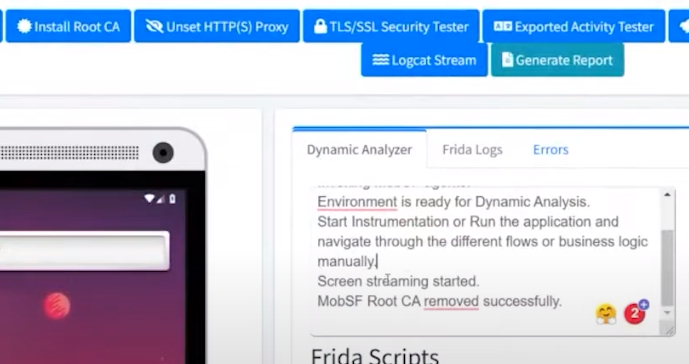
Then we open the MobSF Dynamic analyser and start our analysing of the application.



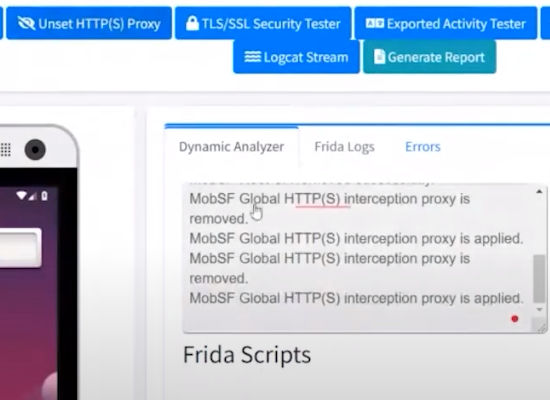


Now we have a few options in Dynamic Analysis which are as follow:

1. **Remove root CA:** Which is responsible for intercepting all the traffic from the device.

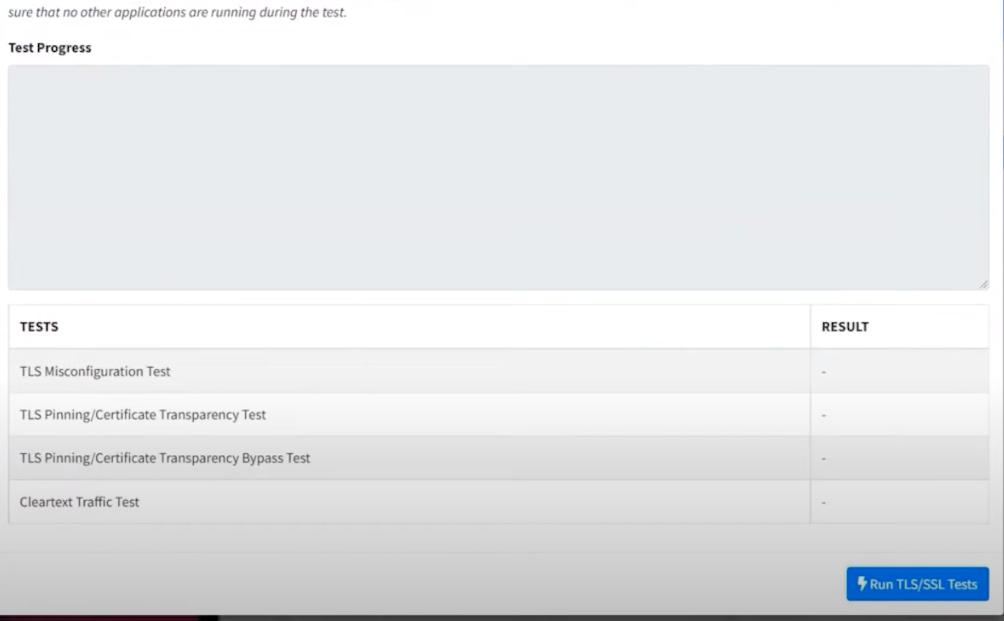


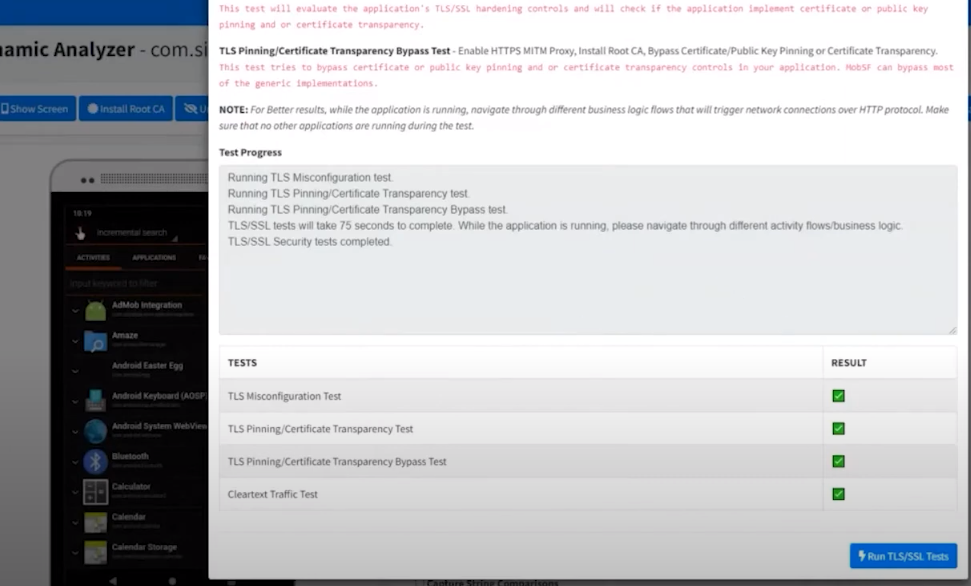
1. **Unset proxy:** is used to make sure our apps proxy server does not connect to another proxy server to share or retrieve its current resources.



1. **TLS/SSL Security tester**

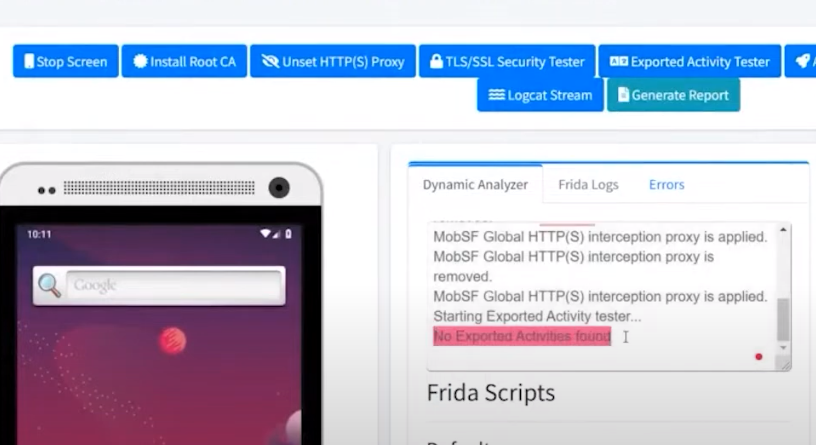
If you have a security service installed in your device, it’s used to tell if your security service is working correctly.





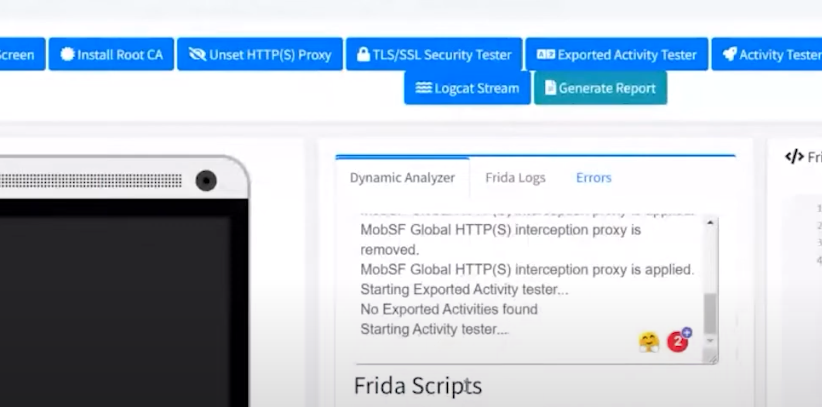
1. **Exported Activity Tester**

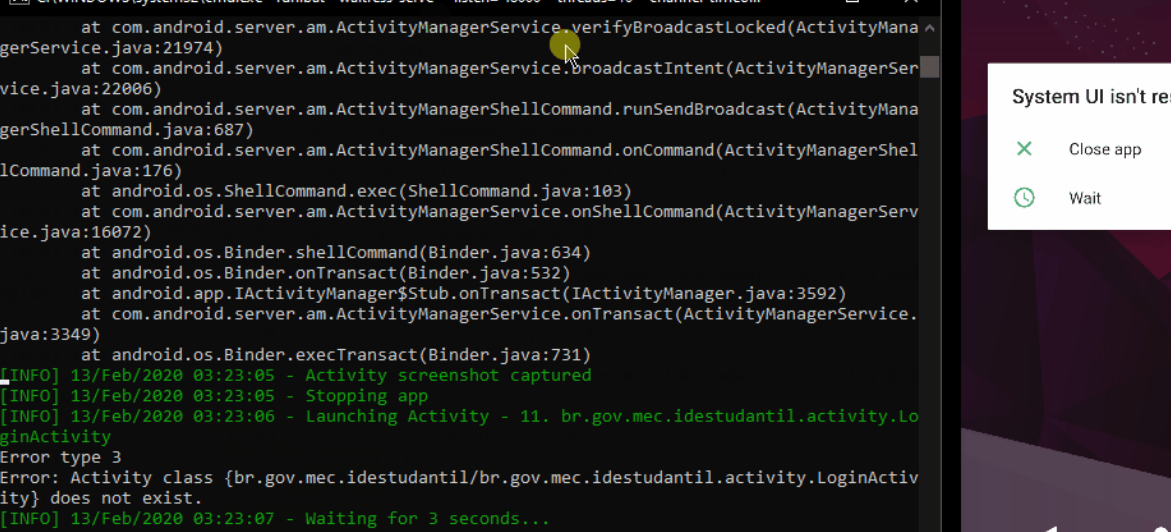
It’s used to make sure that a component of an application is not launched by a component of another application.



1. **Activity Tester:**

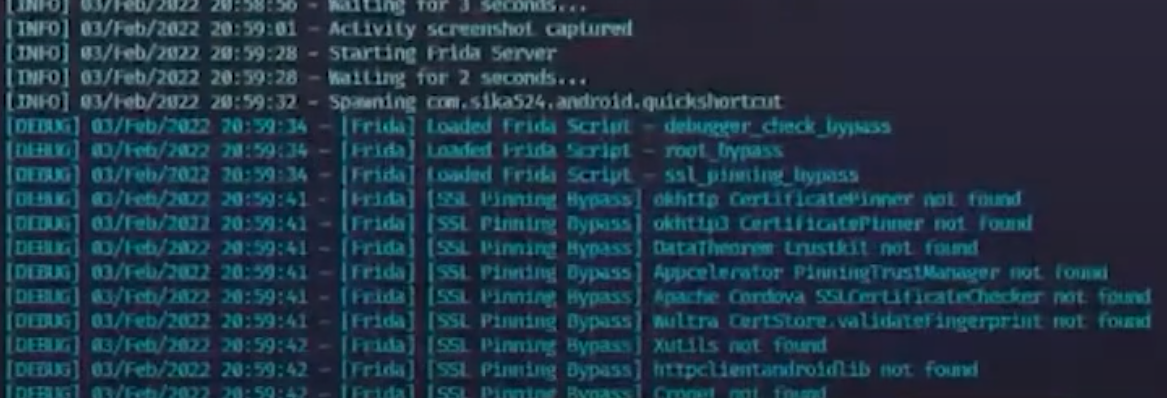
Is used to test the non-expodite activities.



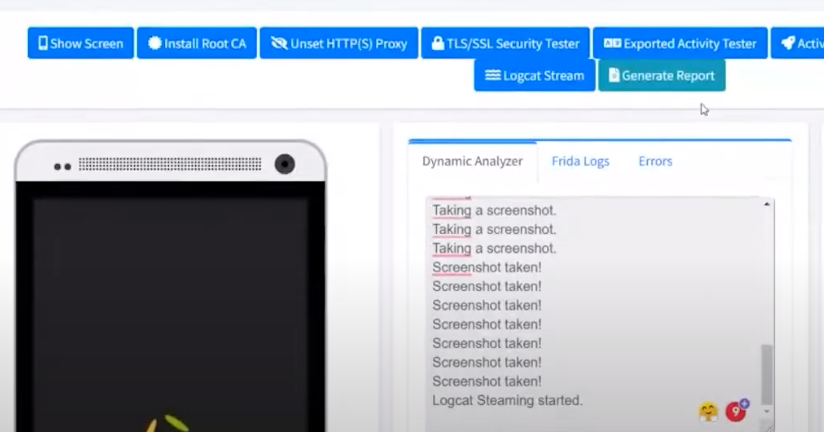


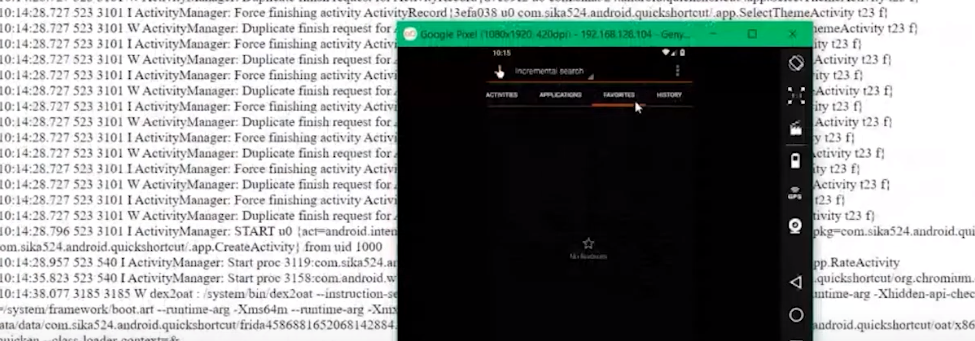
1. **Get Dependencies:**

To see if there is any problem in testing, test data, test environment, execution and the outcome of the test.



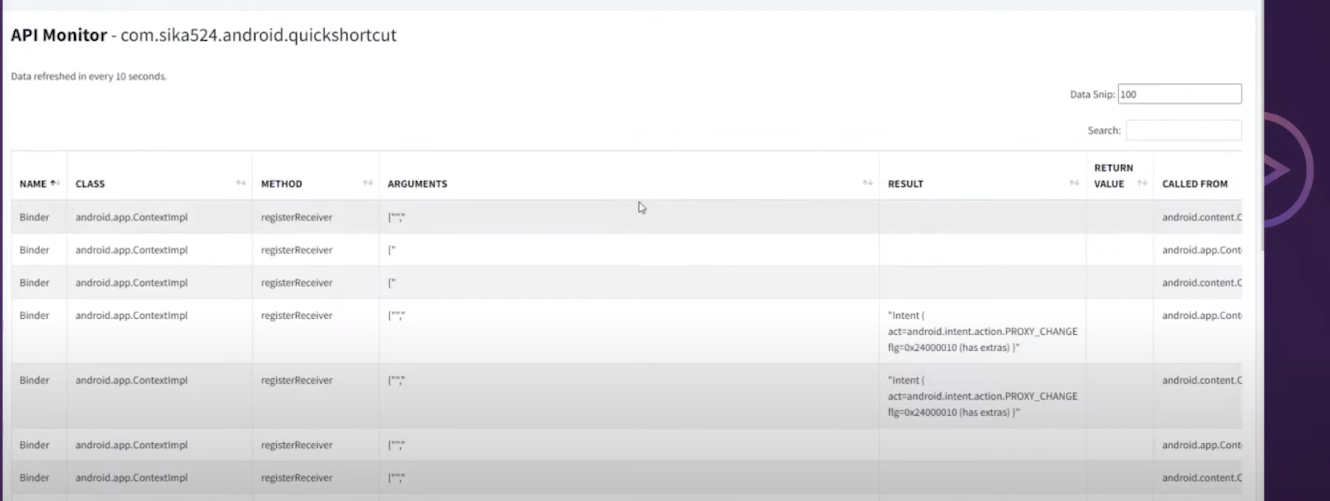
1. **Logcat Stream:** Used to show all the logs of the application in real time





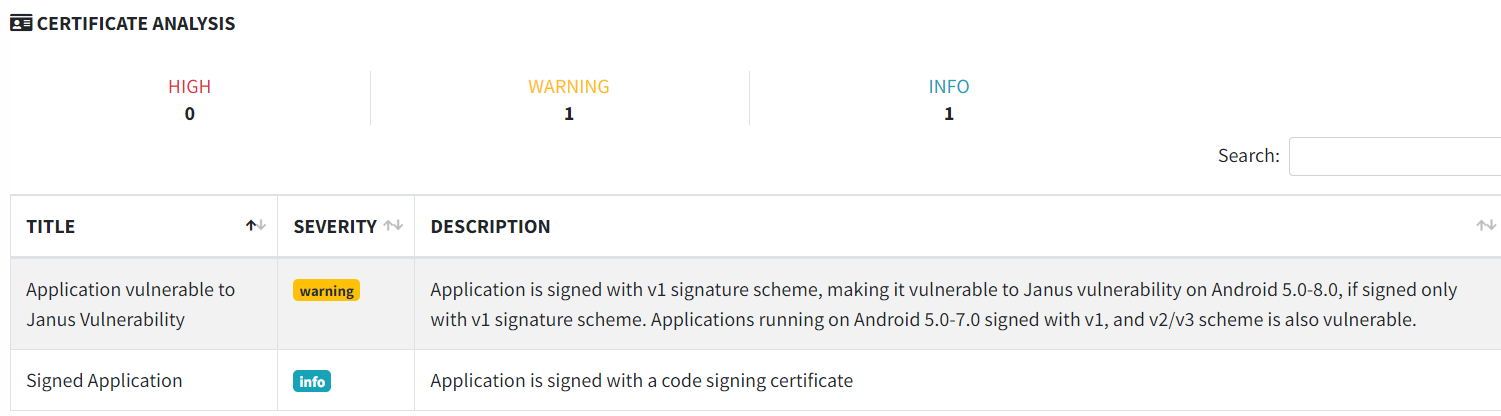
**4. API and Server Assessment using “Live API Monitor”**

Review of the application's communication with external APIs and servers, focusing on authentication, data transmission, and validation checks. Which helps in identifying potential vulnerabilities in data exchange and server interactions, which can be exploited by attackers.



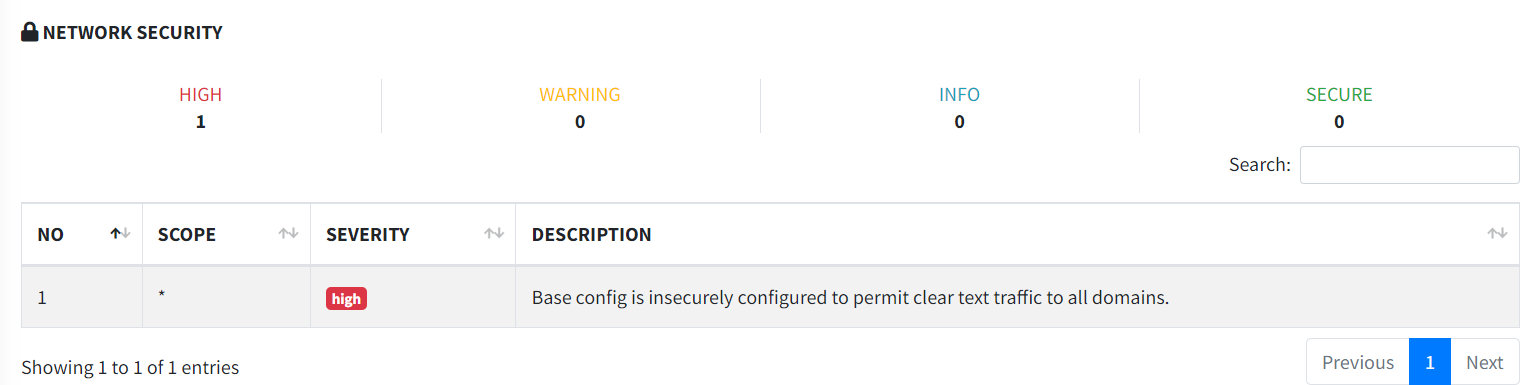
**5- Vulnerability Identification & Potential Impacts**:

1. **Certificate Analysis**: During Static Analysis “MobSF” scanned the application's certificate security (SSL/TLS) analysis revealed that it is signed with the v1 signature scheme, making it vulnerable to the Janus vulnerability on Android 5.0-8.0 if signed only with the v1 signature scheme. Additionally, applications running on Android 5.0-7.0 signed with v1, and v2/v3 scheme are also vulnerable.



**Potential Impact**: The use of the v1 signature scheme for the application's certificate raises concerns about the security of the app, especially on Android versions 5.0-8.0. Applications signed solely with v1 signature schemes on these versions are vulnerable to the Janus vulnerability. If exploited, this vulnerability could allow attackers to modify the application's code or behavior, potentially leading to the installation of malicious software or unauthorized access to user data. This represents a critical risk to the integrity of the application and the safety of users on affected Android versions.

1. **Network Security**: During Static Analysis MobSF, scanned the source code to identify network communication points. It highlighted potential issues related to unencrypted data transmission and identified insecure base configurations that allowed clear text traffic to all domains. And During dynamic analysis, MobSF detected unencrypted network communications and identified the insecure base configuration, indicating a lack of network security and improper traffic permissions.

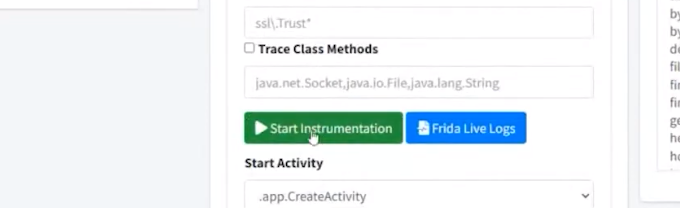


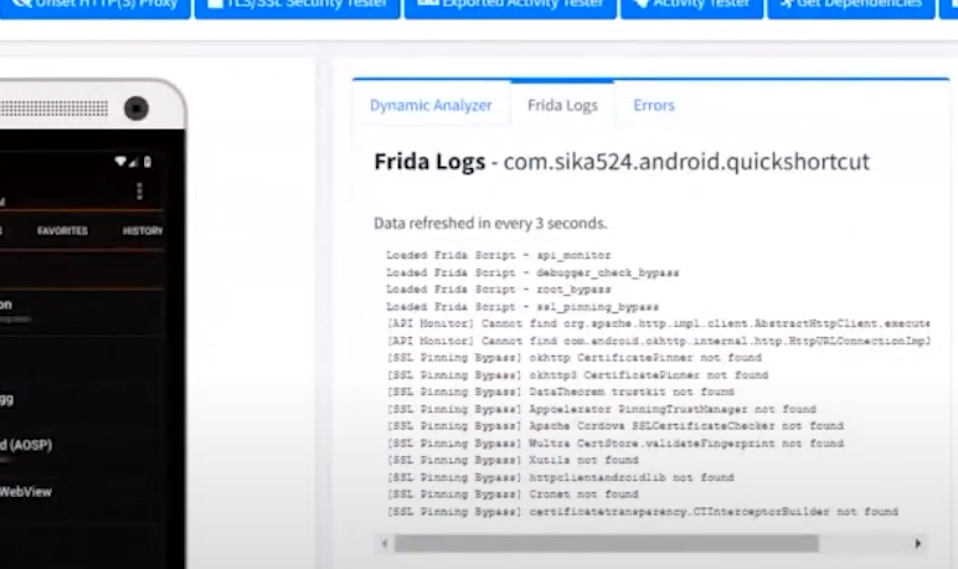
**Potential Impact**: The network security vulnerability, coupled with the insecure configuration permitting clear text traffic to all domains, creates a serious threat to the confidentiality and integrity of data transmitted over the network. As a consequence, attackers can intercept and manipulate data during transmission, potentially leading to data leakage, man-in-the-middle attacks, and the compromise of sensitive information. This poses a severe risk to user privacy and the security of data exchanged with the application.

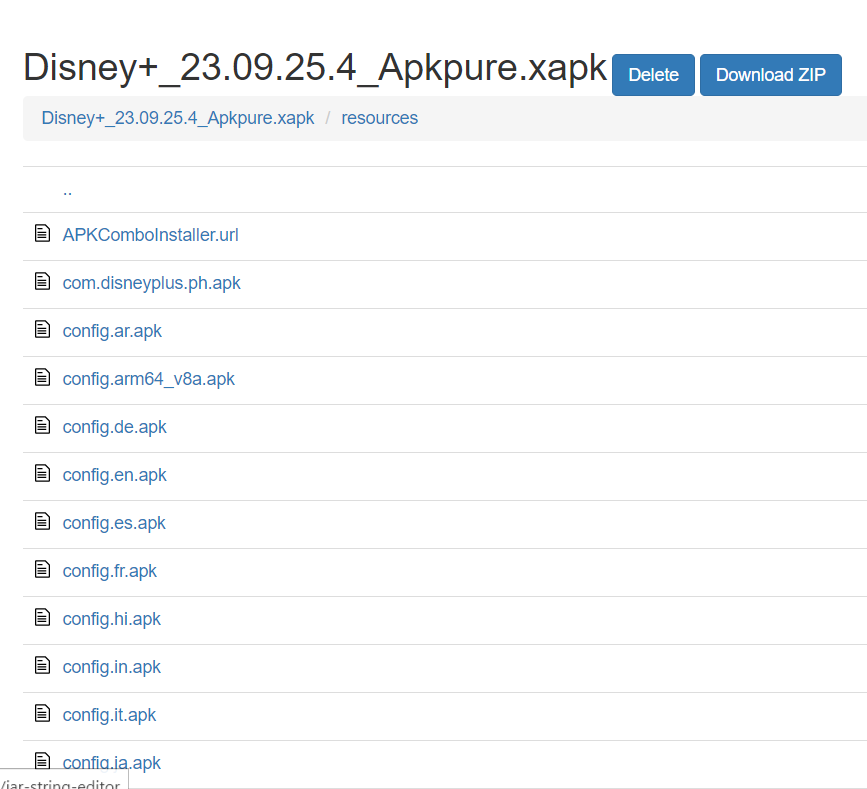
These vulnerabilities were identified through a combination of static analysis and dynamic analysis using MobSF, which systematically analysed the application's source code and runtime behavior. MobSF's automated scans and analysis capabilities were instrumental in flagging potential security issues, allowing for their further investigation and validation. The combination of these techniques provided a comprehensive view of the application's vulnerabilities.

**6- Penetration Testing & Exploitation:**

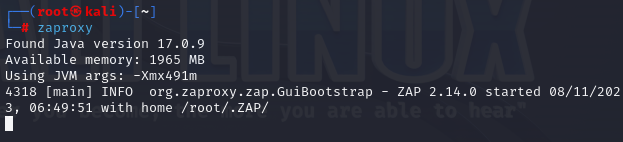
Attempted exploitation of identified vulnerabilities to assess their real-world impact and confirm their potential for harm. Now this will show us if there is any problem in the codes and the logs using dynamic analysis, to check for any loopholes that we may find for our use in penetration testing.

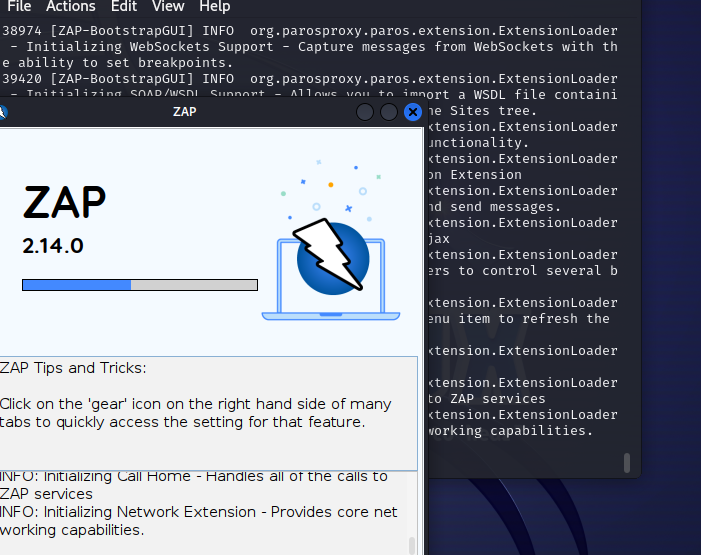


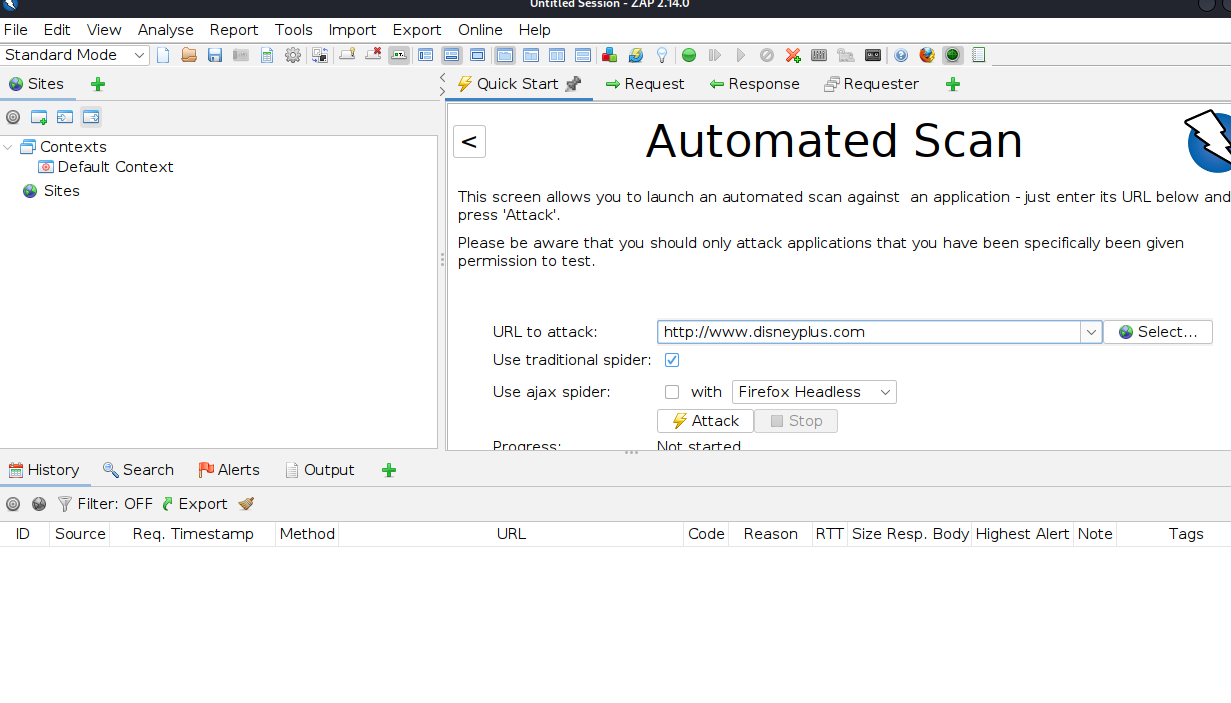


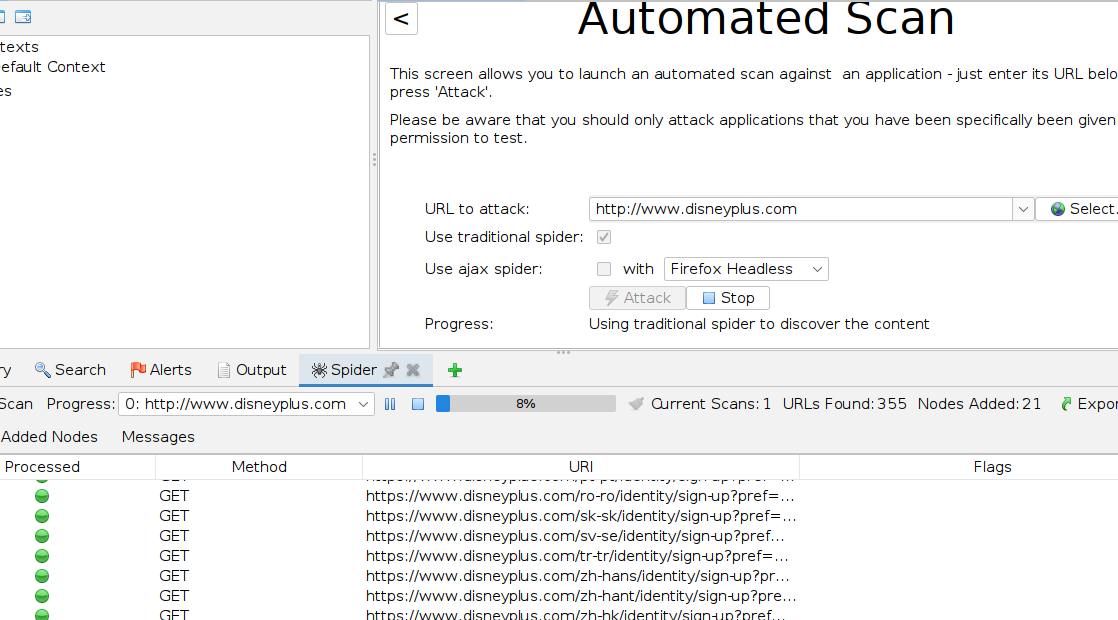
One of the ways we can get exploit the Janus vulnerabilities is decompiling the app and adding an extra APK code of our own into the app, since Janus vulnerability comes from the possibility to add extra bytes to APK files and to DEX files. In here we can tamper with the existing APK, insert our code and manipulate the data inside the code.

Now the way to exploit the network vulnerability we found, is by intercepting the data, spoofing the application data tampering and doing Man-in-the-middle attack. One such an attack is using ZAP and the way to do that is by using Zaproxy, which will automatically attack the target server using the hole in the security as shown bellow:









**7- Recommendations:**

**Network Security:**

**Recommendation**: Strengthen network security to safeguard data transmission. And the best way to do that is:

1. By implementing and enforcing the use of SSL/TLS encryption for network communications to prevent data interception.
2. By implementing certificate pinning to ensure the authenticity of the server and prevent man-in-the-middle attacks.
3. By Applying appropriate firewall rules to control network traffic and restrict unauthorized access.
4. Using security headers, such as HTTP Strict Transport Security (HSTS), to enhance the security of web traffic.

**Certificate Analysis**:

**Recommendation**: Update the certificate and signature scheme to mitigate the Janus vulnerability. Aside from that we can use:

1. Use of the more secure v2 or v3 signature schemes to avoid Janus vulnerability on older Android versions.
2. Periodically renew and update the application's certificate to ensure the latest security measures are in place.
3. Implement strong code signing practices to protect the integrity of the application's code.

**General Security Recommendations:**

We also have some General Security Recommendations aside from the vulnerabilities that can strengthen the overall security of the application.

1. **Security Training:** Ensure that development and security teams receive training on secure coding practices and stay updated with the latest security threats and mitigation strategies.
2. **Code Reviews**: Conduct regular code reviews to identify and fix vulnerabilities early in the development process. This includes reviewing data storage, network communication, and certificate handling code.
3. **Penetration Testing:** Periodically perform penetration testing to proactively identify and address security weaknesses, as well as to validate the effectiveness of security measures.
4. **Privacy by Design:** Integrate privacy considerations into the app's design and development to protect user data and comply with relevant data protection regulations, such as GDPR.
5. **Regular Updates**: Keep the mobile application and its components, including third-party libraries, up to date to patch known vulnerabilities.
6. **Incident Response Plan:** Develop and maintain an incident response plan to address potential security incidents promptly and effectively.

Implementing these recommendations and adhering to best practices will significantly enhance the security of the mobile application and reduce the risks associated with the identified vulnerabilities. Regular security assessments and continuous monitoring are also essential to ensure that the application remains secure as new threats emerge.

**Conclusion:**

In conclusion, the security assessment of the Android application, Disney+ App, has provided a comprehensive overview of the application's overall security. The assessment aimed to identify potential vulnerabilities, assess the application's overall security, and make recommendations to enhance its protection. Through a combination of manual and automated testing techniques, we uncovered crucial info regarding the security of the App.

The findings (vulnerabilities) and recommendations presented in this report can serve as a significant resource for improving the security of the mobile application. Security in mobile applications is really important, given their access to sensitive user data, and any vulnerabilities can have serious consequences, including data breaches, financial losses, and damage to reputation.

We hope that the findings and recommendations provided in this report are taken seriously and that appropriate resources are allocated to ensure the security and privacy of Disney+ App users. Walt Disney's commitment to enhancing the application's security will undoubtedly contribute to a safer and more secure digital environment for all its user and will bring even more user toward it as well.

**The Link to the video: The video is 5 minutes (600 MB) so it can only be watch from google drive.**

**https://drive.google.com/file/d/1-MJUHnWpKna9Cm8o7OC5-oWE1DC10K-j/view?usp=sharing**

**References:**

1. Anwar, C., Hady, S., Rahayu, N., & Kraugusteeliana, K. (2023). The Application of Mobile Security Framework (MOBSF) and Mobile Application Security Testing Guide to Ensure the Security in Mobile Commerce Applications. *Jurnal Sistim Informasi dan Teknologi*, 97-102.
2. Alanda, A., Satria, D., Mooduto, H. A., & Kurniawan, B. (2020, May). Mobile application security penetration testing based on OWASP. In *IOP Conference Series: Materials Science and Engineering* (Vol. 846, No. 1, p. 012036). IOP Publishing.
3. Anwar, C., Hady, S., Rahayu, N., & Kraugusteeliana, K. (2023). The Application of Mobile Security Framework (MOBSF) and Mobile Application Security Testing Guide to Ensure the Security in Mobile Commerce Applications. *Jurnal Sistim Informasi dan Teknologi*, 97-102.
4. Bergadano, F., Boetti, M., Cogno, F., Costamagna, V., Leone, M., & Evangelisti, M. (2020). A modular framework for mobile security analysis. *Information Security Journal: A Global Perspective*, *29*(5), 220-243.