ABSTRACT

In recent years, the proliferation of drones has significantly expanded their applications across various domains, from recreational photography to critical infrastructure surveillance. However, this rapid evolution has brought about new challenges, particularly in ensuring the security and integrity of drone software. One of the key strategies employed to protect drone applications from reverse engineering and tampering is obfuscation. Obfuscation techniques aim to obscure the codebase, making it difficult for adversaries to understand the underlying logic and extract sensitive information. While obfuscation serves as a fundamental defence mechanism, it also poses challenges for developers and security analysts attempting to analyse and verify the behaviour of drone applications. In response to the growing need for understanding the intricacies of obfuscated drone software, this report delves into the realm of deobfuscation and obfuscation techniques specific to Android-based drone applications. By unravelling the layers of obfuscation employed in these applications, we aim to shed light on the underlying functionalities and potential vulnerabilities , As utilization of obfuscation techniques in Android applications poses a significant challenge in identifying tampered code introduced by attackers or malware. As a consequence of obfuscation, the code becomes obscured, complicating the process of identifying potential risks or malicious modifications. Therefore, it becomes imperative to employ deobfuscation methods to revert the code to its original form, enabling thorough analysis and scrutiny for any nefarious alterations. This necessitates a meticulous approach to ensure the security and integrity of Android applications in the face of evolving threats. We intend to raise awareness about the critical role of deobfuscation in mitigating potential attacks on drone Android applications. By bridging the gap between theory and practice in drone software security, we aspire to foster a safer and more resilient ecosystem for the integration of drones across diverse applications. Through this endeavor, developers, security analysts, and policymakers can be better equipped to mitigate risks and ensure the integrity of drone technologies in contemporary society.

The dataset comprises a total of 60 applications, meticulously gathered and vetted to ensure a comprehensive representation of the landscape.

- 3D survey pilot.apk
- aeroGCS green.apk
- air control.apk
- AirHub.apk
- Dji fly.apk
- Dji go.apk
- Dji smartfarm.apk
- Dmo pilot.apk
- Drone buddy.apk
- Drone surveyor.apk
- Dronecast.apk
- Dronedeploy.apk
- Dronelink.apk
- Droniq.apk
- PIX4Dcapture pro.apk
- Plex pilot.apk
- Rainbow.apk
- Smart flight.apk
- Uav forecast.apk
- Airhub-smartcontroller-660
- Android_vision_guanwang
- App-arm
- airhub-sc.apk
- Aptoide.apk
- AR.freeflight
- Drone App.apk
- DroneDepoly.apk
- Dronelink.apk
- DRONI.apk
- drony.apk
- flyingdronecamera.apk
- GoFlyD.J.IDrone.apk
- gosky.apk
- Litchi.apk
- Mi_Drone.apk
- P2Braz.apk
- SYMA-FPV.apk
- Vision+s11.apk
- Wifi-UAV.apk
- Tello.apk
- Shox-chorma.apk
- Ronin.apk
- quadcopter-drone-rc-all-drones-3-1.apk
- PrecisionFlight-2.0.apk
- Mi_Drone_vV_1.3.11.20601_www.9apps.com_.apk
- Luma-AI-v2.1.1.apk
- io-kittyhawk-1.apk
- hover-drone-and-uav-pilot-app.apk
- Fyd-uav.apk
- ELF_VRDrone-5.28.apk
- drone-harmony-for-dji-drones.apk
- djipilot-pe.apk
- DJI-Mimo.apk
- DJI-Go.ap

These links contain the above mentioned drone applications.-
https://drive.google.com/drive/folders/1-1uHfFo4fbLDIxOQCWjiscCiDWgflwSs?usp=sharing

https://drive.google.com/drive/folders/1z_C1muDeuGMq7WyOe-l6H-GzK1Yx-1kW?usp=sharing

https://drive.google.com/drive/folders/1YRvX0w_7XRV8YgfieJGmOLWlhM0EJAUI?usp=sharing

The Tools We Used In This Project Are–

Deguard– http://apk-deguard.com/
deoptfuscator–https://github.com/Gyoonus/deoptfuscator
Katalina–https://github.com/huuck/Katalina
siMBA–https://github.com/DenuvoSoftwareSolutions/SiMBA
Macneto -https://github.com/Programming-Systems-Lab/macneto_release

CONCLUSION

In conclusion, the examination of various deobfuscation techniques, including DeGuard, Deoptfuscator, SiMBA, Katalina, and MACNETO, presents a nuanced understanding of their strengths and limitations in deciphering obfuscated code within drone Android applications.

DeGuard exhibits commendable proficiency in reversing ProGuard obfuscation, particularly in predicting third-party libraries with high precision and recall rates, thus facilitating improved comprehension of application structures and functionalities. However, challenges persist with sophisticated obfuscation techniques and scalability concerns.

Deoptfuscator demonstrates effectiveness in handling control-flow obfuscation, contributing to code structure restoration and optimizing app performance through ReDex. Nevertheless, its limited applicability to other obfuscation techniques and the potential risk of errors require careful implementation consideration.

SiMBA introduces an efficient approach to linear MBA computation, boasting competitive runtimes and adaptability. Despite its efficiency, challenges arise with handling complexity and verifying expression equivalence, impacting its scalability for larger datasets.

Katalina serves as a valuable free and open-source alternative for malware analysis, efficiently deobfuscating strings and accelerating reverse engineering processes. However, its lack of multidex support and potential performance overheads warrant attention, especially with complex malware.

MACNETO showcases high precision rates in deobfuscating obfuscated programs, exhibiting versatility against various obfuscation techniques. Yet, dependencies on callgraph and challenges with certain obfuscation techniques necessitate further refinement for broader effectiveness.

In conclusion, it's evident that no single deobfuscation method stands as universally superior, with each technique showcasing specific strengths and weaknesses tailored to different types of obfuscation. Effective deobfuscation often demands the integration of multiple methods to comprehensively address various obfuscation strategies. Furthermore, not every application opts for obfuscation; while some prioritize transparency and ease of debugging, others employ heavy

obfuscation to safeguard proprietary algorithms or sensitive data, depending on developers' objectives and security concerns. Crucially, deobfuscators play a critical role in detecting malware within drone applications by reversing obfuscation techniques and revealing the original code. This allows analysts to identify and analyze malicious components effectively, ensuring the safety and security of drone operations by uncovering suspicious code patterns and behaviors.