DroidDungeon

by ThreatNemesis

Whitepaper

Last modified on November 2024

Executive Summary

The Android operating system is the most widely used platform for mobile devices globally, with billions of active devices. Given its popularity, the Android platform has become a primary target for malicious actors, resulting in a significant increase in the prevalence of Android malware. The advancement of Android malware has outpaced the capabilities of conventional analysis tools, which are unable to keep pace with the increasingly sophisticated evasion techniques employed by modern malware.

At ThreatNemesis, we have developed DroidDungeon, a state-of-the-art Android malware analysis sandbox designed to address the shortcomings of existing solutions. DroidDungeon employs a unique approach that leverages kernel-level monitoring, providing unparalleled stealth and resilience against evasion techniques, which sets it apart from traditional sandboxes that rely on invasive user-space instrumentation. DroidDungeon is capable of monitoring every aspect of an app's execution, from decrypting network traffic to intercept exfiltrated data and reporting privacy leaks. This ensures comprehensive analysis for both automated batch workloads and manual investigations.

Contents

Executive Summary 1

1 Trends in Android Malware 2

2 Android Sandboxes and Their Limitations 5

3 DroidDungeon: *Sandboxing Android Malware "The Right Way"* 7

4 Conclusion 11

1 Trends in Android Malware

In the early days of Android malware, the threats were relatively simple and straightforward. Malicious actors often relied on basic techniques to achieve their goals, and these early forms of malware were easily detectable by traditional analysis tools. However, as the Android ecosystem has matured, so too has the sophistication of the malware targeting it. Today, Android malware is arguably as complex and advanced as its counterparts on more established platforms like Windows.

The evolution of Android malware can be attributed to several factors. First, the increasing financial incentives for cybercriminals have driven the development of more sophisticated malware. With the rise of mobile banking, mobile payments, and other financial services on Android devices, the potential rewards for successful attacks have grown exponentially. Second, the open nature of the Android platform, while a key factor in its widespread adoption, has also made it easier for attackers to develop and distribute malware.

The Defeat of Static Code Analysis

One of the most significant trends in Android malware is the widespread adoption of techniques designed to defeat static code analysis. Static analysis, which involves examining the code of an application without executing it, has traditionally been one of the primary methods for detecting malware. However, modern Android malware often employs strategies that render static analysis ineffective.

Multi-Stage Malware. A growing number of Android malware samples adopt a multi-stage approach, where the initial APK file serves only as a dropper or downloader [12]. The primary malicious payload is not included in the APK and is instead retrieved from a remote server controlled by the attacker. This approach effectively bypasses static analysis, as the initial APK appears benign until it connects to the server and downloads the actual malware. This technique is particularly effective also because it allows the attacker to dynamically change the payload based on the environment in which the malware is deployed.

Code Obfuscation. Another common technique used to defeat static analysis is code obfuscation [3]. Obfuscation involves altering the code in a way that makes it difficult to understand or analyze, without affecting its functionality. Android malware often uses various obfuscation techniques, such as renaming variables and methods to meaningless names, inserting junk code, encoding strings, encrypting portions of code, and even dynamic loading of code (which

is therefore not statically available). These techniques make it difficult for static analysis tools to identify the malware's intent, as the true nature of the code is hidden until it is executed. Additionally, obfuscation make understand the code challenging even for human analysts, further complicating the vetting process.

The combination of multi-stage malware and code obfuscation has made static analysis an increasingly unreliable method for detecting Android malware. A significant percentage of modern Android malware employs these techniques, rendering traditional static analysis tools ineffective on their own.

Takeaway #1

Static malware analysis alone is not viable for Android malware anymore.

Circumventing Dynamic Analysis

In response to the limitations of static analysis, many security professionals have turned to dynamic analysis, which involves executing the malware in a controlled environment and observing its behavior. Unfortunately, modern Android malware has also evolved to counter dynamic analysis techniques.

Android malware often probes its execution environment for artifacts that give away the presence of a dynamic analysis system (*i.e.,* sandbox), altering their behavior if these "evasion checks" have positive outcomes. Some malware samples adopt a split-personality strategy: They refrain from performing malicious actions if they detect that they are being executed in a sandbox, while running unrestrained otherwise.

In our research, presented in 2024 at the renowned ACM Conference AsiaCCS, on a representative dataset[1] of Android malware [11], we found evidence of anti-dynamic analysis behavior in 68% of the analyzed samples. While already alarming in its own right, this figure represents only a lower bound of the share of evasive samples. The results of our investigation suggest that as many as 90% of the analyzed samples access data that can indirectly give away a dynamic analysis environment.

Targeting Specific Analysis Tools. Some of the evasion techniques Android malware employs aim at detecting entire classes of analysis environments (*e.g.,* those based on emulation or virtualization). Others specifically target the tools commonly employed for dynamic analysis. Prominent examples are Frida (a dynamic instrumentation framework) and Xposed (an app instrumentation framework) which are frequently used in Android sandboxes to capture malicious behavior. Android malware has been shown to search for the presence of Frida or Xposed in the process memory.

Takeaway #2

The vast majority of malware samples exhibits anti-dynamic analysis capabilities. Popular instrumentation frameworks are specifically targeted.

[1] Over 20 000 samples collected from the VirusTotal feed.

Encrypted Command and Control Communications

Malware authors have been documented employing command and control servers (C2) to orchestrate their campaign. C2 servers are used by attackers to remotely control infected devices, issuing commands and receiving data from the malware implants. In modern Android malware, the network traffic between the malware and its C2 is encrypted to prevent detection by intrusion detection systems (IDS) or sandboxes.

Encrypted Channels. More specifically, Android malware often relies HTTPS to encrypt its communications with C2 servers [6], effectively making any naive analysis based on passive network monitoring useless. In some cases, malware authors have gone even further, adding more encryption layers on top of HTTPS. These schemes are designed to prevent security tools from decrypting the traffic, even if the HTTPS channel gets compromised (*e.g.,* through TLS-aware man-in-the-middle proxies). For example, some malware samples [7, 14] use symmetric encryption algorithms like RC4, DES, or AES to encrypt their C2 communications, with the encryption keys being generated dynamically at runtime.

Takeaway #3

The use of encryption in C2 communications makes it difficult to gain insight into the commands issued by the attacker or the data exfiltrated by the malware.

2 Android Sandboxes and Their Limitations

As Android malware has become more sophisticated, the limitations of existing Android sandboxes have become increasingly apparent. These sandboxes often rely on technologies that are fundamentally unsuited for analyzing modern malware.

Pitfalls of Dynamic Analysis Technologies

Code Instrumentation: Xposed and Cuckoo-based Sandboxes. Several commercial Android sandboxes are based on the open-source Cuckoo sandbox framework, which makes use of the Xposed code instrumentation framework to monitor malware behavior. While Xposed is a powerful tool for dynamic analysis, it has significant limitations when used in a sandbox environment. By modifying the target's code, Xposed inevitably introduces artifacts that malware can spot by simply inspecting its own memory (*e.g.,* its call stack) at runtime.

Dynamic runtime instrumentation: Frida. Other sandboxes instrument the analyzed malware at runtime by means of the popular framework Frida. Similarly to Xposed, Frida is capable of monitoring and altering the execution of Android apps to detect evidence of malicious behavior. However, like Xposed, Frida leaves significant artifacts in the malware's memory, making it easy for the malware to detect its presence. In fact, Frida requires injecting an entire Javascript engine into the process being monitored, which is highly invasive and easily detectable by the malware. This makes Frida-based sandboxes vulnerable to evasion techniques and limits their effectiveness against sophisticated malware.

Notice how the limitations of these approaches are intrinsic to the technologies employed. In both cases, the monitoring instrumentation is placed within the malware's *security boundary*, *i.e.,* its running process, over which the malicious sample has total control. In other words, using either dynamic runtime or code instrumentation dooms Android sandboxes to fail against evasive malware.

Furthermore, Xposed, Frida and similar code instrumentation frameworks require super-user (*i.e.,* root) privileges, which introduce other artifacts[1] that malware can leverage for detecting the analysis sandbox.

Takeaway #4

Common technologies used in currently available Android sandboxes are fundamentally vulnerable to evasion.

[1]For example, the *su* binary is usually not available on stock Android devices.

Testing Android Sandboxes at Scale

For our research [11], we tested many currently available Android sandboxes. To this end, we crafted a custom Android application implementing all known evasion techniques we surveyd from a variety of sources (*e.g.,* blogposts, conference talks, academic papers) and analyzed it through VirusTotal (therefore all the sandboxes that automatically analyze its feed), JoeSandox, and Tria.ge. The results of this experiment were concerning: *all* tested sandboxes are susceptible to at least one known evasion technique.

The vast majority of sandbox solutions can be fingerprinted due to root-related artifacts. Notable example is RecordedFuture's Tria.ge for which our test app detected the presence of Magisk, a tool often used to equip Android system with super-user capabilities. Our educated guess based on the company business history[2] is that this sandbox is based on Cuckoo and makes use code instrumentation.

The JoeSandbox' Android dynamic analysis system also proved vulnerable to known evasion techniques. In particular, our evasive app found out that the system maps several suspicious libraries in the analyzed app's memory. This approach is similar to Frida's, suggesting that JoeSandbox for Android shares the same capabilities and weaknesses found in this framework.

Takeaway #5

None of the tested sandbox (commercial, open source, and private alike) was resilient to known evasion strategies.

[2] In 2022, RecordedFuture acquired Hatching.io [10], the company behing the Cuckoo sandbox, whose open source development has then stopped.

3 DroidDungeon

Sandboxing Android Malware "The Right Way"

The Innovative Approach of DroidDungeon

DroidDungeon is an Android sandbox developed with a clear understanding of the limitations of traditional sandboxes and the need for a fundamentally different approach to malware analysis. By avoiding the mistakes of the past and adopting innovative techniques, DroidDungeon offers a robust and resilient solution for analyzing even sophisticated Android malware.

Unparalleled Stealth

One of the key innovations of DroidDungeon is its decision to eliminate user-space instrumentation. As we saw previously, traditional sandboxes often rely on tools like Frida and Xposed or similar technologies, whose artifacts are easily detectable by malware, making these analysis systems vulnerable to evasion.

In contrast, DroidDungeon implements its monitoring logic in kernel space, *i.e.,* beyond the reach of most malware. This approach drastically reduces the attack surface that malware can use to detect the analysis system. In fact, malware would need to escalate to kernel-level privilege in order to access the monitoring logic. Given their very high market price [1], such attacks are rarely found in Android malware distributed at scale, and are typically used only against specific high-profile targets. This makes DroidDungeon highly resilient against the vast majority of samples in the wild.

Additionally, unlike other sandboxes, DroidDungeon does not require root privileges to operate, thus it cannot be fingerprinted by means of root-related artifacts (*su* binary, Magisk, etc.). This further widens the gap between DroidDungeon's and its competitors' stealth capabilities.

[1] For example, Zerodium is a renowned American information security company specializing in the acquisition of zero-day vulnerabilities with functional exploits from security researchers and organizations. At the time of writing, the company's acquisition program offers a substantial reward of up to 200,000$ for the successful exploitation of a local privilege escalation to Kernel/Root in Android/iOS. [16]

Advanced Network and Cryptographic Monitoring

DroidDungeon also excels at monitoring network traffic and cryptographic operations, providing analysts with comprehensive visibility into the malware's network communications and its exchanges with C2 servers.

DroidDungeon does this by intercepting the key generation routines used for encryption, allowing it to capture and decrypt HTTPS network communications without breaking end-to-end encryption. This approach is particularly effective against techniques such as certificate pinning, which is often used by malware to prevent traffic interception, and SSL-aware man-in-the-middle proxies, which are often used in competitors' solutions. Moreover, by monitoring the use of cryptographic APIs such as RC4, DES, and AES, DroidDungeon can also detect portions of hidden malicious code and further protect its network activity.

This comprehensive network and cryptographic APIs monitoring capability allows analysts to gain deep insights into the malware's behavior, including the commands being issued by the attacker, the data being exfiltrated, and the encryption methods being used. This level of visibility is essential for understanding and mitigating modern Android threats.

360° Behavioral Tracing With Privacy Leaks Monitoring

DroidDungeon captures many aspects of malware's runtime operations to find evidence of potentially harmful behaviors. In particular, the sandbox monitors a wide set of Android APIs and syscalls that malware often abuses to leak sensitive data, make phishing attempts more credible, and even performing remote device takeovers. This unprecedented level of behavioral monitoring allows analysts to fully characterize the malware's TTPs and objective.

Moreover, whenever it analyzes a new sample, DroidDungeon randomizes privacy-sensitive information, such as the emulated device's phone number, IMEI, and model. At the end of execution, DroidDungeon reports potential privacy leaks by checking if any of these randomized identifiers have been transmitted over the network.

Customizable Detection Rules

Our malware analysis platform includes a signature matching system that allows analysts to automatically tag malware whose reported activity match certain pattern. Specifically, our system supports YARA [13] signatures based on the report generated by DroidDungeon and Zeek [15] scripts processing the malware's decrypted traffic.

Alongside a set of rules mainteined by ThreatNemesis, analysts can upload and use their own custom rules. This capability is helpful for grouping malware of the same family or with similar characteristics and can prove crucial for threat attribution and tracking malware campaigns.

Flexible Analysis Environment

While DroidDungeon's automatic pipeline can fully analyze the vast majority of Android malware samples in the wild, in certain cases, the analyst might still need to manually intervene in

some capacity. For example, certain malware samples execute their malicious operations only after the user performs some operations (e.g., clearing the first level of a fake game). To assist the analyst in such scenarios, DroidDungeon supports *manual UI interaction*, giving the analyst remote access to the virtual device's user interface.

For more complex use cases requiring to actively tamper with the malware runtime (*e.g.,* defusing timebombs [4]), DroidDungeon can also grant remote access to the virtual device via ADB-over-ssh. This gives analysts full flexibility and control over the dynamic analysis system. We also support the usage of custom Frida scripts, but it is worth emphasizing that it may jeopardize DroidDungeon's anti-evasion guarantees.

Artificial Intelligence-Assisted Malicious Activity Scoring

While providing undoubtedly valuable insight into malware's behavior, the large amount of data provided by DroidDungeon's integrated analysis system might be too detailed in some scenarios. This is particularly true for large scale, fully automated pipelines that needs to deliver concise and actionable information about the risks associated with malware samples.

To address this and other use cases that value concision over nuance, DoidDungeon is equipped with an Artificial Intelligence system that sums up the data collected by the analysis system into a malicious score. More precisely, it is a Machine Learning model trained following the best practices highlighted in recent research [1, 5] on a large dataset of known malware samples, properly balanced for families, and famous Android applications from the Google Play Store. We evaluated it, and DroidDungeon's malware classification performance is on par with that of state-of-the-art models [2, 9, 8].

Comparative Analysis: A Superior Solution

To evaluate DroidDungeon's benefit as a commercial malware analysis solution, we have conducted a detailed comparison with two of the leading market competitors: Recorded Future's Tria.ge and JoeSandbox for Android. The following table highlights the key differences between these tools and DroidDungeon:

	JoeSandbox	Tria.ge	**DroidDungeon**
Stealth	✗	✗	✓
Network & crypto	⚠	⚠	✓
Behavior Tracing	⚠	⚠	✓
Custom Rules	✓	✓	✓
Flexibility	✗	⚠	✓
Malicious Score	⚠	⚠	✓

Table 3.1: Competitors: Feature Comparison

This comparison clearly shows that DroidDungeon offers several advantages over the examined competitors. For instance, neither JoeSandbox nor Tria.ge report the use of cryptographic

primitives and the related decrypted payloads. Their behavioral tracing is also lagging behind DroidDungeon's, which captures Android API invocations from kernel-space.

For what concerns flexible analysis, Tria.ge supports an interactive mode similar to DroidDungeon's manual analysis. On the other hand, JoeSandbox does not provide any alternative mode other than automatic analysis. In comparison, DroidDungeon provides both a UI-interactive mode and a fully manual mode, empowering the analyst with complete control over the analysis environment.

While both competitors ship a report summary in the form of a malicious score, we noticed that in both cases such score seems to be calculated from the number of detection rules matching the analyzed sample. This system is more prone to false-negatives, especially since both JoeSandbox and Tria.ge do not provide a sufficient level of stealth. In fact, evasive malware implementing split-personality behavior triggers less detection rules, thus making the scoring system fail.

What really set aside DroidDungeon, however, is its unprecendented stealth which makes it the best in class in terms of number of samples it can analyze successfully. The following table presents a projection of the number of samples in our malware dataset that each tested sandbox can detonate. In the table we included Tria.ge, JoeSandbox, and all the other Android sandboxes attached to VirusTotal.

Product	**Vendor**	**Malware detonation (%)**	**Technology**
VTSandbox 1	Unknown	$< 50\%$	
VTSandbox 2	Unknown	$< 50\%$	
VTSandbox 3	Unknown	$< 55\%$	
VTSandbox 4	Unknown	$< 50\%$	
VTSandbox 5	Unknown	$< 50\%$	
VTSandbox 6	Unknown	$< 75\%$	
VTSandbox 7	Unknown	$< 50\%$	
VTSandbox 8	Unknown	$< 55\%$	
JoeSandbox	JoeSecurity	$< 55\%$	Userspace instrumentation, strace
Tria.ge	RecordedFuture	$< 55\%$	Cuckoo-based, Magisk
DroidDungeon	ThreatNemesis	98%	Kernelspace monitoring

Table 3.2: Competitors: Detonation rates

With only the exception of one sandbox (which we assume is an in-house non-commercial solution), all the tested systems cannot achieve a detonation rate abouve 55%. For methodological reasons, the figures reported in the table are strict upper bounds of the actual detonation rates for third party sandboxes. We expect the actual rate to be significantly lower than what we reported. In comparison DroidDungeon's *measured* detonation rate approaches 98%[2].

[2]The 2% failure rate is intrinsic to the nature of our dataset. In fact, the VirusTotal feed, from which the dataset was built, often contains corrupted samples. We manually investigated several samples that DroidDungeon could not detonate automatically. In all the investigated cases, the samples either did not present any exported components (thus, they lack a proper entrypoint) or were malformed (invalid APK bundles that Android could not install).

4 Conclusion

The landscape of Android malware has changed dramatically in recent years, with modern threats becoming increasingly sophisticated and difficult to analyze. Traditional malware analysis tools, which rely on static and dynamic analysis techniques, are struggling to keep pace with the advanced evasion techniques employed by today's malware authors.

This whitepaper has explored the current trends in Android malware, highlighting the challenges posed by techniques such as multi-stage payloads, code obfuscation, anti-dynamic analysis behaviors, and encrypted command and control communications. These trends have exposed the limitations of existing Android sandboxes, which often rely on invasive user-space instrumentation and other technologies that are easily detectable by sophisticated malware.

In response to these challenges, we have introduced DroidDungeon, a next-generation Android malware analysis sandbox that sets a new standard for the industry. By adopting a fundamentally different approach to malware analysis, DroidDungeon addresses the shortcomings of traditional sandboxes and provides a robust and resilient solution for analyzing even the most complex Android threats.

DroidDungeon's key innovations, including its kernel-space monitoring, advanced network and cryptographic analysis capabilities, and scalable cloud deployment model, make it an essential tool for organizations looking to stay ahead of the rapidly evolving threat landscape. As Android malware continues to evolve, DroidDungeon will remain at the forefront, providing analysts with the tools they need to effectively detect, analyze, and mitigate modern Android threats.

In conclusion, DroidDungeon is not just another Android malware analysis tool—it is a comprehensive platform designed to meet the challenges of today's threat environment and to future-proof your organization's malware analysis capabilities. By investing in DroidDungeon, organizations can ensure that they are equipped with the most advanced and effective tools available to protect their Android devices and data from malicious actors.

Bibliography

[1] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. Dos and don'ts of machine learning in computer security. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3971–3988, 2022.

[2] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss*, volume 14, pages 23–26, 2014.

[3] Fortinet & Virus Bulletin. Obfuscation in android malware, and how to fight back. `https://www.virusbulletin.com/virusbulletin/2014/07/obfuscation-android-malware-and-how-fight-back`, 2014. Accessed: 2024/11/25.

[4] CheckPoint. Braintest – a new level of sophistication in mobile malware. `https://www.globalsecuritymag.fr/BrainTest-a-new-level-of,20150921,56019.html`, 2015. Accessed: 2024/11/25.

[5] Savino Dambra, Yufei Han, Simone Aonzo, Platon Kotzias, Antonino Vitale, Juan Caballero, Davide Balzarotti, and Leyla Bilge. Decoding the secrets of machine learning in malware classification: A deep dive into datasets, feature extraction, and model performance. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 60–74, 2023.

[6] ESET. Arid viper poisons android apps with aridspy. `https://www.welivesecurity.com/en/eset-research/arid-viper-poisons-android-apps-with-aridspy/`, 2024. Accessed: 2024/11/25.

[7] FoxIT. Android malware vultur expands its wingspan. `https://blog.fox-it.com/2024/03/28/android-malware-vultur-expands-its-wingspan/`, 2024. Accessed: 2024/11/25.

[8] Han Gao, Shaoyin Cheng, and Weiming Zhang. Gdroid: Android malware detection and classification with graph convolutional network. *Computers & Security*, 106:102264, 2021.

[9] Junyang Qiu, Jun Zhang, Wei Luo, Lei Pan, Surya Nepal, and Yang Xiang. A survey of android malware detection with deep neural models. *ACM Computing Surveys (CSUR)*, 53(6):1–36, 2020.

[10] RecordedFuture. Recorded future acquires hatching to extend intelligence cloud coverage with malware analysis. https://www.recordedfuture.com/press-releases/20220708, 2022. Accessed: 2024/11/25.

[11] Antonio Ruggia, Dario Nisi, Savino Dambra, Alessio Merlo, Davide Balzarotti, and Simone Aonzo. Unmasking the veiled: A comprehensive analysis of android evasive malware. In *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, ASIA CCS '24, page 383–398, New York, NY, USA, 2024. Association for Computing Machinery.

[12] ThreatFabric. Anatsa trojan returns: Targeting europe and expanding its reach. https://www.threatfabric.com/blogs/anatsa-trojan-returns-targeting-europe-and-expanding-its-reach, 2024. Accessed: 2024/11/25.

[13] VirusTotal. Yara – the pattern matching swiss knife for malware researchers. https://virustotal.github.io/yara/, 2014. Accessed: 2024/11/25.

[14] Qianxin XLab. Playing possum: What's the wpeeper backdoor up to? https://blog.xlab.qianxin.com/playing-possum-whats-the-wpeeper-backdoor-up-to/, 2024. Accessed: 2024/11/25.

[15] Zeek. Zeek – an open source network security monitoring tool. https://zeek.org/, 2015. Accessed: 2024/11/25.

[16] Zerodium. Zerodium exploit acquisition program. https://zerodium.com/program.html, 2024. Accessed: 2024/11/25.