(In)visible Privacy Indicator: Security Analysis of Privacy Indicator on Android Devices

Yurak Choe
wonpis6@g.skku.com
Sungkyunkwan University
Suwon, Gyeonggi-do, Korea

Hyungseok Yu
Samsung Electronics
Suwon, Gyeonggi-do
yelvis@samsung.com

Taeho Kim
Samsung Electronics
Suwon, Gyeonggi-do
taeho81.kim@samsung.com

Shinjae Lee
Samsung Electronics
Suwon, Gyeonggi-do
shinjae1.lee@samsung.com

Hojoon Lee*
Sungkyunkwan University
Suwon, Gyeonggi-do
hojoon.lee@skku.edu

Hyoungshick Kim*
Sungkyunkwan University
Suwon, Gyeonggi-do
hyoung@skku.edu

ABSTRACT

In Android 12, Google introduced a new security feature called the *privacy indicator* to protect users from spyware. The privacy indicator visually alerts users by displaying a green circle in the notification bar when an application accesses the camera. While this feature initially appears effective, our work has identified two possible attack scenarios that can undermine it. The first attack uses screen overlay techniques with a higher Z-order and deceptive status bar layouts to make it difficult to see the privacy indicator. In a user study involving 44 participants, only 13.6% of participants recognized the indicator under UI overlay attacks, compared to 63.6% in default Android 12 settings. The second attack exploits device configurations to disable the privacy indicator. Our findings were reported to the developers of the Android system UI at Samsung Electronics and the Google Issue Tracker, and we received acknowledgments from both parties. As countermeasures, we recommend ensuring the integrity of the privacy indicator using trusted execution facilities. We introduce a proof-of-concept solution called SEPI (**S**ecurity-**E**nhanced **P**rivacy **I**ndicator), which utilizes a secure hypervisor and ARM TrustZone. SEPI is designed to detect camera and microphone activities, subsequently displaying the relevant indicator with the highest Z-order in a securely isolated display buffer. Our experimental findings revealed only a minimal 3.3% reduction in benchmark scores compared to the device's default operational state. The SEPI privacy indicator is displayed with a negligible mean delay of 20.92 ms.

CCS CONCEPTS

• **Security and privacy → Privacy protections**; **Mobile platform security**; **Trusted computing**; **Privacy protections**.

KEYWORDS

Android privacy indicator, Mobile platform security, TrustZone

*Corresponding authors

ASIA CCS '24, July 1–5, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0482-6/24/07.
https://doi.org/10.1145/3634737.3645014

ACM Reference Format:
Yurak Choe, Hyungseok Yu, Taeho Kim, Shinjae Lee, Hojoon Lee, and Hyoungshick Kim. 2024. (In)visible Privacy Indicator: Security Analysis of Privacy Indicator on Android Devices. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '24), July 1–5, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3634737.3645014

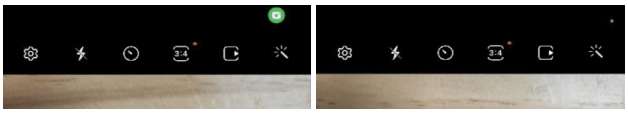
1 INTRODUCTION

Smartphone cameras have become essential in our daily lives, but they also have potential privacy risks. Unauthorized access to these cameras can lead to invasive surveillance and exposure of personal images or videos. Several vulnerabilities and spyware threats have been identified, such as Pegasus [30], which infects iPhones and Android devices, allowing operators to extract messages, photographs, and emails, record calls, and surreptitiously activate microphones and cameras without user consent. Checkmarx [13] revealed an Android camera app vulnerability (CVE-2019-2234), enabling attackers to covertly capture images and videos and intercept phone conversations [31, 41]. Recently, the Zimperium zLabs research team detected malware called PhoneSpy targeting smartphone users in South Korea, capable of accessing a device's camera to capture images and videos [46].

To address this issue, major smartphone operating systems, such as Apple iOS and Google Android, have recently incorporated a security feature that delivers visual cues to alert users when an app accesses the camera. This feature enhances user privacy by informing them about any potential unauthorized access to the camera resource. In Apple iOS 14, a small green dot appears at the top of the screen when an app uses the camera, as shown in Figure 1(a) [8]. In Android 12, a similar feature called the privacy indicator has been introduced. The privacy indicator feature exhibits camera and microphone icons during their usage before transitioning to a dot [20], as depicted in Figure 1(b). For example, when an application accesses the camera, a green dot is displayed at the top of the screen, indicating camera usage. This indicator animates prominently at the top right corner, enabling users to identify when the sensor is in use. By providing these visual cues, both Apple iOS and Google Android operating systems help users stay informed about camera or microphone usage.

In this paper, we present two attack scenarios that can render the Android privacy indicator ineffective during camera operation. The first attack uses overlay techniques from accessibility service with a higher Z-order and a deceptive status bar layout, making

(a) iPhone's privacy indicator feature: The orange indicator indicates that an app is using the microphone, while the green indicator indicates that an app is using the camera or both the camera and microphone at the same time.



(b) Android's privacy indicator feature: When an app is using the camera on an iPhone, the camera icon initially appears, followed by a transition to a green dot. The green dot remains visible until the app is either dismissed or closed.

Figure 1: iPhone's indicator vs. Android's indicator.

it difficult to recognize the indicator. The second attack involves reverse shell intrusions that exploit Android device configurations to disable the privacy indicator functionality. We confirmed the successful execution of both attacks on a Galaxy S22, Galaxy A52s 5G, Google Pixel 3, and Google Pixel 6. We conducted a user study with 44 participants to analyze their awareness of the Android privacy indicator under UI overlay attacks. Only 13.6% of participants identified the indicator when exposed to UI overlay attacks, while 63.6% recognized the indicator under default Android 12 settings.

To mitigate the threats of these attacks, we need to ensure the integrity of such security warnings. To achieve this goal, we implement a proof-of-concept privacy indicator protection called SEPI (**S**ecurity-**E**nhanced **P**rivacy **I**ndicator). SEPI leverages the secure hypervisor to collect the true device status in a trustworthy way and also protect a privacy indicator frame with a secure buffer used through TrustZone-based DRM. By ensuring that the privacy indicator is rendered with the highest Z-order through the operating system framebuffer, SEPI provides a trustworthy privacy indicator even under the compromise of the Android operating system.

To demonstrate the feasibility of SEPI, we implemented it on a Samsung Galaxy A52s 5G (equipped with an octa-core processor and 6GB RAM) and evaluated the performance overhead of SEPI on the device. We ran a Passmark benchmark app while recording videos in the background on both the vanilla device (i.e., the device after a factory reset) and the SEPI-implemented device. Our performance evaluation revealed that our SEPI implementation decreased the overall benchmark score by a mere 3.3% compared to the vanilla device. Additionally, we measured the execution latency overhead of TrustZone composition in SEPI by conducting 403 repeated tests from the surface, request, enqueue, to stop. We found that SEPI's privacy indicator via TrustZone was promptly displayed with a mean execution latency of 21.02 ms and a standard deviation of 2.86, indicating no substantial latency overhead when employing SEPI.

The key contributions of this paper are as follows:

- We provide a detailed analysis of novel attacks that can mask the privacy indicator on the device's status display interface during sensitive device usage (e.g., camera and microphone). We substantiate the effectiveness of the attacks through a user study in which only 3 out of 22 participants could identify the privacy indicator when it was cunningly obscured during camera usage.
- We report our findings to Samsung Electronics and Google. Samsung Electronics acknowledged the issues and confirmed that they have been resolved in OneUI 4.1.1, which is implemented in the latest Samsung smartphones. Google also acknowledged them as a potential misuse of the accessibility service but did not categorize it as a software bug vulnerability.
- We propose a proof-of-concept mitigation design called SEPI. SEPI leverages the existing hypervisor and TrustZone components that are widely available in modern Android mobile devices to preserve the visual integrity of privacy indicators even under full system compromise.

2 UI OVERLAY ATTACK

We have identified two potential attack scenarios that can undermine the effectiveness of the privacy indicator. The first involves screen overlay techniques with a higher Z-order and deceptive status bar layouts, designed to obscure the visibility of the privacy indicator. The second scenario involves manipulating device configurations to disable the privacy indicator. In this section, we will delve into the UI overlay attack scenario. The following section will discuss the device configuration tampering attack in further detail.

2.1 Threat Model

The threat model considered here involves a malicious application capable of performing UI overlay attacks. We envision a scenario where a victim's Android device is compromised through social engineering, leading to the installation of an application with accessibility service permission. Originally intended to assist users with disabilities by modifying the UI and overlaying graphics, this permission is examined for potential exploitation. Our paper investigates the misuse of Android's accessibility service permission by a malicious application to launch UI overlay attacks, which undermine the effectiveness of privacy indicator UI.

2.2 Android Overlay

An overlay is a UI feature that has been supported since Android version 1.0. The overlay creates an additional view layer on top of the host view, allowing a mobile application to draw on top of other applications. The rationale for Android overlay capabilities is to improve the user experience when users interact with multiple applications simultaneously [43]. Unfortunately, this overlay feature is sometimes exploited for UI overlay attacks [17]. An attacker uses a transparent or opaque UI layer over a legitimate UI layer to interact with a malicious overlay before the user interacts with the legitimate UI layer or with a legitimate UI layer. Malicious overlays can be buttons, data entry fields, or other screens within the mobile application that mimic the actual user interface. As a result, hacker-controlled malware can obscure a victim's UI. The combination of malware and social engineering tactics with UI overlay attacks makes them more realistic and effective.

Previous research [17] utilized the `SYSTEM_ALERT_WINDOW` permission for UI overlay attacks. However, our findings show that combining `WindowManager.LayoutParams.TYPE_APPLICATION_OVERLAY` with `SYSTEM_ALERT_WINDOW` permission results in overlays with low Z-order values that are insufficient in concealing

the privacy indicator UI. Consequently, the privacy indicator remains visible during UI overlay attacks using these permissions. This underscores the unique necessity of the accessibility service permission to disable the privacy indicator UI–a novel vulnerability that we have identified in our paper.

2.3 Android Accessibility Service

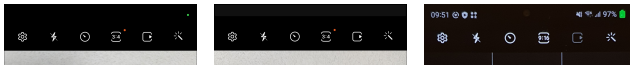
Accessibility services on Android, designed to assist visually impaired users, have potential vulnerabilities, as our study reveals. The Android WindowManager service provides the `TYPE_ACCESSIBILITY_OVERLAY` window type, allowing layouts created by accessibility services to overlay other views and possibly obscure them. The Z-order of a view in the Android window system is crucial for visibility, with higher Z-order values enabling a view to overlay those with lower values. Thus, an accessibility service using `TYPE_ACCESSIBILITY_OVERLAY` can create views that effectively conceal others beneath them. Consequently, apps using `TYPE_ACCESSIBILITY_OVERLAY` must adhere strictly to security guidelines to ensure legitimate use for accessibility purposes. Using accessibility services requires an app to request `BIND_ACCESSIBILITY_SERVICE` permission in its AndroidManifest.xml and obtain user consent.

2.4 UI Overlay Attack Method

The UI overlay attack conceals the privacy indicator when the camera is in use by overlaying a layout with a higher Z-order. Our implementation builds upon the third-party app 'Privacy Indicators,' which is open-source under the MIT license [18].

To execute a UI overlay attack, the application must have the `BIND_ACCESSIBILITY_SERVICE` permission to access the accessibility service and `TYPE_ACCESSIBILITY_OVERLAY` when initializing a layout from the WindowManager service. This allows the attacker to draw over the Android privacy indicator during camera operation. The `FLAG_LAYOUT_NO_LIMITS` and `FLAG_LAYOUT_IN_SCREEN` flags stretch the layout to the status bar. When the service is initiated, a layout is created that meets the above criteria. Its visibility is set to `View.GONE`. Consequently, it only appears when the camera or microphone is activated. When either of these sensors is activated, the visibility is changed to `View.VISIBLE`, overlaying the privacy indicator. The application must also register a callback from the CameraManager service to be alerted when the camera is in use. This callback is used to trigger the overlay of the layout when the camera is activated. The UI overlay attack can be divided into two categories:

- **Complete Obscuring Attack (COA):** This attack completely masks all privacy indicator UI components, including the camera animations, microphone icons, and the green dot, by overlaying a black background identical in size to the Android status bar. This occurs because, in Samsung's OneUI 4.1 version, the `PRIVATE_FLAG_IS_ROUNDED_CONERS_OVERLAY` flag for ensuring that the rounded corners of the privacy indicator are rendered with the highest Z-order.
- **Deceptive Obscuring Attack (DOA):** This attack uses a deceptive status bar with a green battery symbol to obscure the Android privacy indicator. The goal of this attack is to camouflage the green dot by superimposing it with the green battery icon or another system icon.



(a) Default setting (b) COA (c) DOA

Figure 2: UI overlay attacks on Samsung Galaxy S22.

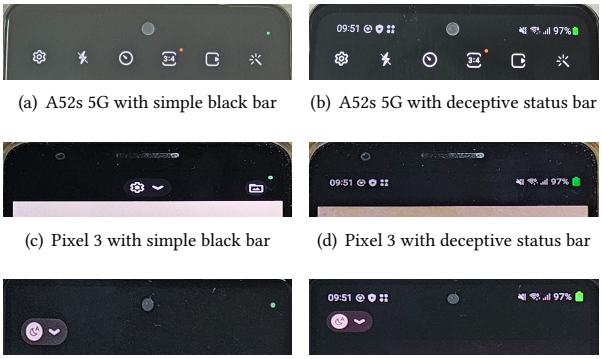
Figure 3: Effectiveness of UI overlay attacks with 10 third-party camera applications.

2.5 UI Overlay Attack Results

We demonstrated COA on a Samsung Galaxy S22 model running Android 12 (with OneUI 4.1). As shown in Figure 2(b), the malicious application generates a window layer that manifests as a long black bar obscuring the screen's top. This layer hides the device's status bar whenever the camera and microphone are activated.

We also tested COA with third-party camera applications. We confirmed that the Android privacy indicator was obscured by the long black bar in all 10 applications, as shown in Figure 3. A few camera applications made the black bar stand out too much by using a white background, which led to inconsistencies. Nevertheless, the black bar still effectively obscured the Android privacy indicator. Malicious applications can also avoid this inconsistency by using a long black bar with the same color as the background used by a specific application.

COA is viable only on devices where the `PRIVATE_FLAG_IS_ROUNDED_CORNERS_OVERLAY` flag is absent or misconfigured, thereby failing to guarantee the highest Z-order for the status bar. We chose the Samsung S22 as a representative model to demonstrate this vulnerability. However, we note that all OS versions running Samsung OneUI 4.1 are susceptible. In contrast, the COA was unsuccessful



(a) A52s 5G with simple black bar (b) A52s 5G with deceptive status bar

(c) Pixel 3 with simple black bar (d) Pixel 3 with deceptive status bar

(e) Pixel 6 with simple black bar (f) Pixel 6 with deceptive status bar

Figure 4: Comparison of the black bar and the deceptive status bar on the a52s 5G, Pixel 3, and Pixel 6 devices.

on both the Google Pixel 3 and Pixel 6 due to the proper implementation of the `PRIVATE_FLAG_IS_ROUNDED_CORNERS_OVERLAY` flag, ensuring the status bar's highest Z-order. Following our security report, this vulnerability was correctly fixed in the Samsung S22 with the release of OneUI 4.1.1.

DOA can be executed on any Android device running Android 12 or later that incorporates the Android privacy indicator feature. We demonstrated DOA on various models, including the Galaxy S22, Galaxy A52s 5G, Google Pixel 3, and Google Pixel 6, which serve as representative devices. Comprehensive demonstrations of the attack are available through video demos, including for the Galaxy A52s 5G illustrating DOA at https://www.youtube.com/shorts/1eiqnjZ8Foo, and for the Galaxy S22 demonstrating COA at https://www.youtube.com/shorts/CjTvgQcvOpM, highlighting the impact of UI overlay attacks.

The effects of applying a simple black bar layer versus a deceptive status bar through a malicious application are compared in Figure 4. With the simple black bar layer applied, the green dot representing the Android privacy indicator is clearly visible. However, when the deceptive status bar is applied, the green dot becomes difficult to discern without close inspection. This phenomenon is consistently observed across all the devices, including the Samsung Galaxy S22, Galaxy A52s 5G, Google Pixel 3, and Google Pixel 6, indicating that this type of attack would be a common concern in the Android ecosystem. Surprisingly, this behavior persisted even in Android 13, hinting that the UI overlay attack technique could be a major ongoing security challenge for Android devices.

We further carried out a test to demonstrate the effectiveness of the DOA attack without the user's awareness, involving the activation of the camera in the background. For this, we employed the Background Video Recorder (BVR Pro) application available on the Google Play Store. BVR Pro facilitates background video recording, which can be activated by pressing the volume-up button. In this scenario, we observed that the Android privacy indicator could be obscured, similar to what we found in the foreground process scenario.

2.6 User Study

Our user study aimed to demonstrate the practical effectiveness of DOA. We recruited 44 participants, spanning various age groups (23 in their 20s, 10 in their 30s, 11 in their 40s) and genders (35 male, 9 female). Recruitment was conducted through advertisements on online notice boards at a university and selective recruitment from an IT company. Participants were compensated with a Starbucks card valued at approximately 10 USD. The study was conducted using our prepared Google Pixel 6 devices, not the participants' personal devices. Before starting, participants were briefed and asked to sign a consent form. The study's purpose was not disclosed to participants beforehand to ensure ecological validity.

Participants in our study were alternately assigned to two groups: Group A and Group B. Group A's age distribution included 12 participants aged 20-29, 5 aged 30-39, and 5 aged 40-49, while Group B had 11 in the 20-29 range, 5 in the 30-39 range, and 6 in the 40-49 range. Gender-wise, Group A had 19 males and 3 females, and Group B had 16 males and 6 females. Chi-square and Fisher's exact tests confirmed no significant demographic differences between the groups. Group A was given a standard device without any additional software. Group B received a device with our application installed, designed to execute the DOA attack. This application displayed a deceptive status bar with a green battery icon, but it was designed not to cause actual harm. All participants were allowed to use the camera application for three minutes. Subsequently, they were asked to complete a questionnaire to gather information about their demographics and ability to recognize privacy indicator UI components. The complete questionnaire can be found in Appendix A.

The study results, as presented in Appendix B, demonstrate a significant difference in the awareness of the Android privacy indicator between the two groups. In Group A, 14 out of 22 participants recognized the indicator, while in Group B, only 3 out of 22 participants recognized it. We conducted a Fisher's exact test to determine the statistical significance between the recognition success rates of the two groups, finding a significant difference ($p < 0.01$). This suggests that UI overlay attacks by malicious applications can significantly hinder the recognition of the Android privacy indicator. Additionally, we investigated whether the age of participants influenced the study's outcomes. However, no significant difference in recognition success rates was found across the three age groups.

Interestingly, we also observed that a considerable number of users (36.40%) in the default device group (Group A) did not recognize the privacy indicator. Most of these users admitted that they had not noticed its existence during their regular smartphone usage. However, a few users in Group B did notice the indicator under certain circumstances. For instance, one user noted that the battery icon turned green when the camera application was launched, prompting them to investigate further and thus recognize the privacy indicator. Similarly, another user noticed the green dot that appeared and disappeared as they switched between the front and back cameras, which was also when the deceptive status bar appeared and disappeared.

3 DEVICE CONFIGURATION TAMPERING (DCT) ATTACK

In this section, we present our second attack scenario involving device configuration manipulation to disable the privacy indicator.

3.1 Threat Model

The threat model we consider involves Android devices with a reverse shell, a type of connection where the target device initiates a connection to the attacker's machine. Once established, the attacker gains remote control over the target device, including the ability to disable the Android privacy indicator. We posit that the attacker installs a remote access trojan (RAT), a malware type that facilitates remote access by establishing a reverse shell. With this shell, the attacker gains control over the device, potentially spying on the user, stealing personal information, or commandeering the device.

This attack requires rooted devices. In a standard user-privileged application, attempts to execute commands to disable the Android privacy indicator using `Runtime.exec()` can be blocked by the SEAndroid policy and the system-privileged permission `WRITE_DEVICE_CONFIG`. However, a root process can bypass these protections. As the system's most privileged process, it can execute any command that alters device configuration. Root access is generally acquired via applications like Magisk [2] or by exploiting Linux kernel vulnerabilities, such as "dirty pipe" [5, 23]. Our study's reliance on this strong threat model (rooted devices) represents a notable limitation in assessing device configuration tampering attacks on standard Android devices in real-world scenarios.

3.2 Android DeviceConfig Service

The DeviceConfig service provides a way to modify the values of settings on Android devices from a remote server like the Android Debug Bridge (ADB) [7]. Developers usually utilize this service for testing purposes. They are able to view all currently running services, their corresponding flags, and specific values (e.g., timeouts and durations) by connecting to the device via the ADB shell and typing '`cmd -l`.'

Table 1 lists the potential operations that can be performed with the DeviceConfig service. Table 2 shows the default values associated with the keys in the privacy namespace for the Samsung Galaxy S22 (SM-S906U). The Android privacy indicator can be set using the flag `KEY:camera_mic_icons_enabled` within the `Namespace:privacy`. The default value for this key is `true`. The `Namespace:privacy` contains seven keys, each with a default value. The DeviceConfig service allows developers to change the values of these keys without modifying the corresponding source code or building the entire codebase. Regrettably, this functionality can be manipulated by attackers to disable the Android privacy feature.

Table 1: Supported DeviceConfig service commands on ADB.

Command	Description
get NAMESPACE KEY	Retrieve the current value of KEY from the given NAMESPACE.
Put NAMESPACE KEY VALUE [default]	Change the contents of KEY to VALUE for the given NAMESPACE {default} to set as the default value.
delete NAMESPACE KEY	Delete the entry for KEY for the given NAMESPACE.
list [NAMESPACE]	Print all keys and values defined, optionally for the given NAMESPACE.

3.3 DCT Attack Method

The device configuration tampering attack exploits rooted devices by leveraging a reverse shell with the Android DeviceConfig service.

Table 2: List of keys and values in privacy namespace with SAMSUNG Galaxy S22.

Key	Value
`camera_mic_icons_enabled`	`true`
`location_access_check_enabled`	`true`
`location_accuracy_enabled`	`true`
`location_indicators_enabled`	`false`
`permissions_hub_enabled`	`false`
`permissions_hub_subattribution_enabled`	`true`
`privacy_dashboard_7_day_toggle`	`false`

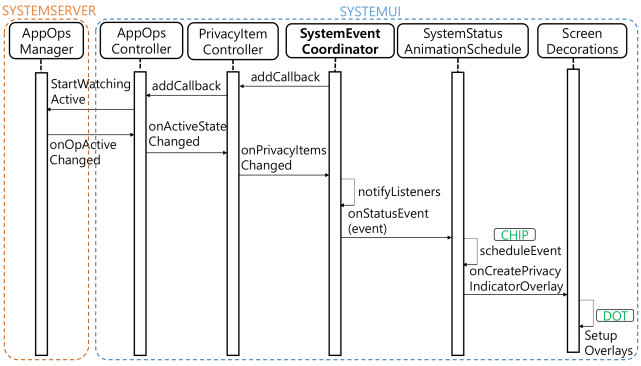


Figure 5: Sequence diagram of the Android privacy indicator workflow.

The attack employs publicly available exploit kits, such as Metasploit [34] and AndroidRAT [26]. Additionally, 'ngrok' is utilized to establish a secure tunneling service to the attacker's local host.

The Android privacy indicator can be toggled on or off via exploit tools that implement the device configuration tampering attack and through the Android-provided DeviceConfig service. The indicators can be manipulated using ADB shell commands as follows:

- Disable: `device_config` put `privacy camera_mic_icons_enabled false` default
- Enable: `device_config` put `privacy camera_mic_icons_enabled true` default

The system-privileged service "`com.android.shell`" possesses the `WRITE_DEVICE_CONFIG` permission and all essential SEAndroid policies to bind to the DeviceConfig service. Exploit kits can enable reverse shell sessions and spawn processes with user privileges on compromised devices. If a user-privileged application intends to execute an attack command through Runtime.exec(), the SEAndroid policy and the system-privileged permission `WRITE_DEVICE_CONFIG` could deny it. Therefore, a rooted device is necessary to execute DeviceConfig commands with elevated privileges, ensuring the success of the device configuration tampering attack.

The Android privacy indicator relies on the interplay between the System UI and the AppOpsManager service. The AppOpsManager service detects the use of the camera or microphone by an application, while the System UI is responsible for rendering the indicator icons and dots at a specified location on the device screen. Figure 5 illustrates the workflow of the privacy indicator functionality.

During boot-up, the SystemEventCoordinator invokes the addCallback() function, which in turn calls the startWatchingActive() function provided by the AppOpsManager service running under

(a) Before the attack (b) After the attack

Figure 6: Device configuration tampering attack on Samsung Galaxy S22.

systemserver. If a callback is registered through startWatchingActive() in the System UI service, the AppOpsManager service detects whether the privileged permission for the camera or microphone is in use and notifies the System UI service via onOpActiveChanged(). Upon receiving the notification, the System UI service displays the chip icon and dot on the device screen in a specific sequence. However, this sequence only operates correctly if the KEY "`camera_mic_icons_enabled`" is either `true` or `null`. If the KEY is `false`, the addCallback() from PrivacyItemController to AppOpsController is not invoked. Consequently, even if an application uses the camera or microphone, the privacy indicator is not displayed on the screen because the System UI service does not receive any notification from the AppOpsManager service.

This sequence of events occurs when the disable shell command is successfully transmitted through a reverse shell. By setting the "`camera_mic_icons_enabled`" KEY to `false`, the SystemEventCoordinator does not register a callback with AppOpsManager. Consequently, AppOpsManager does not notify the SystemUI service, even if the application uses the camera or microphone. The SystemEventCoordinator does not receive any callbacks and, therefore, does not take any further steps to display the visual component of the Android privacy indicator on the screen.

3.4 DCT Attack Results

We observed that manipulating the Android privacy indicator feature flag can potentially disable the privacy indicator feature introduced in Android 12 and subsequent Android devices.

For our experiment, we set up a virtual victim by rooting a Samsung Galaxy S22 model using Magisk. We considered a situation where a malicious application was installed on the victim's device, crafted via Metasploit or AndroidRAT and supporting reverse shell. The test results for the Samsung Galaxy S22 are shown in Figure 6. Figure 6(a) presents the screen status with an active camera on a factory image before the attack, showcasing a visible green dot, a component of the Android privacy indicator. In contrast, Figure 6(b) shows the screen state after the attack, demonstrating that the Android privacy indicator feature was deactivated through a device configuration tampering attack. We performed additional testing on 10 popular third-party camera apps that are downloaded from the Google Play Store. Our tests showed that the Android privacy indicator was completely deactivated in all 10 third-party camera apps. We demonstrate that our proof of concept application can effectively disable the privacy indicator by employing the DeviceConfig service to modify the KEY value.

Post-exploitation entails actions after session initiation. A session is an active shell from a successful exploit or brute force attack, appearing as either a standard shell or Meterpreter. Actions in an open session can include collecting system information, running Meterpreter modules, and searching the file system. Post-exploitation assesses the capabilities and values of the target system.

4 EFFECTIVENESS OF ATTACKS

We have discovered that the Android privacy indicator feature is vulnerable to two distinct types of attacks: UI overlay attacks and device configuration tampering attacks.

UI overlay attacks exploit Android's overlay capabilities to create a UI layer that obscures the privacy indicator, making it invisible or difficult for users to recognize. One straightforward method involves a malicious application creating a layout with a higher Z-order, which overlays a black bar over the Android privacy indicator. This specific UI overlay attack is termed COA. Our testing on early Android 12 versions of the Samsung Galaxy S22 (OneUI 4.1) showed that COA was effective in concealing both the icon animations and dot components of the Android privacy indicator. However, only the icon animation was masked on Google Pixel 3 and Google Pixel 6 devices with Android 12 and 13, with the dot remaining visible. This difference is because these devices have a system-level flag that guarantees the highest Z-order for the status bar. After our report, the Samsung Galaxy S22 was updated to OneUI 4.1.1, which reinforces the highest Z-order for the status bar, thereby successfully preventing COA. This highlights the importance of maintaining the highest Z-order for the status bar to safeguard the integrity of the privacy indicator.

Another UI overlay attack, termed DOA, creates a deceptive status bar with a green battery icon, effectively camouflaging the Android privacy indicator dot. Due to its visual trickery, it is important to design the privacy indicator UI that is resistant to such deception in addition to ensuring the highest Z-order of the status bar.

In addition to overlaying another screen atop the privacy indicator, it is also possible to render it transparent. This can be achieved by changing the memory value where the transparency of the privacy indicator UI is stored, making it completely imperceptible.

Device configuration tampering attacks, requiring a rooted device, are typically executed via root exploit malware that alters the Android OS kernel for superuser access. A recent security vulnerability named "dirty pipe" (CVE-2022-0847) discovered in Linux kernels from version 5.8 and onwards exemplifies such attacks [27, 39]. This vulnerability enables unauthorized processes to inject code into root processes, resulting in privilege escalation and the ability to overwrite data in arbitrary read-only files. Even with patch updates, achieving root privileges remains a recurring vulnerability. The device configuration tampering attack was possible because a mechanism existed to turn the Android privacy indicator feature on/off with a simple shell command.

Moreover, the detection method in the Android privacy indicator could be exposed to a workaround to bypass the permission check. As illustrated in Figure 7, the Android privacy indicator feature executes detection and indicator display in EL0 (user mode). Since most threat models target Android's EL0 and EL1 regions, the privacy indicator is susceptible to malware attacks.

To address this, we propose a solution that uses isolated areas from Android's execution environment to alert users to the use of privacy-sensitive devices like cameras and microphones. In this

paper, we introduce a proof-of-concept architecture implementation that demonstrates the effectiveness of our proposed solution.

5 SEPI ARCHITECTURE

SEPI, a proof-of-concept design, ensures the integrity of Android system privacy indicators. It utilizes the hypervisor and TrustZone to maintain indicator trustworthiness, even under complete compromise. SEPI reliably detects camera or microphone status changes via the hypervisor. TrustZone's secure world collaborates with the hypervisor to enforce top Z-order display rendering of privacy indicators, ensuring on-screen indicators accurately reflect the camera's true state, impervious to malicious interference from user applications or the operating system.

5.1 Leveraging In-System Trust Anchors

SEPI proposes to introduce minimal and non-intrusive changes to the existing hypervisor [3] and TrustZone [36] support for system integrity monitoring. Many modern Android systems, in fact, not only ship with such trust anchors enabled but often feature basic system integrity monitoring. In our testing, all of Samsung's Galaxy S line of products have these trust anchors enabled, thereby motivating us to build SEPI on top of their security guarantees.

Before we proceed further with SEPI's design, we must elaborate on the system privilege organization in the Android system with the aforementioned trust anchors supported.

ARM TrustZone, introduced with the ARMv6 architecture, is a technology used in various devices like smartphones and tablets [9, 36]. It establishes a trustworthy ecosystem, leveraging the CPU as a programmable trusted platform module by creating a secure mode in the CPU, splitting the system into a protected 'secure world' and a 'normal world.' The secure world is protected and has access to designated RAM ranges and peripherals inaccessible to the normal world. Userspace or kernel code cannot access these protected resources even when the normal world is compromised.

Lightweight hypervisors are often deployed with ARM-based Android devices to serve as a trust anchor within the system. These hypervisors reside in a higher privileged layer (EL2) than the OS kernel (EL1), and can provide basic system integrity protection against possibly compromised kernel. Among such protection mechanisms, SEPI leverages the following for its design:

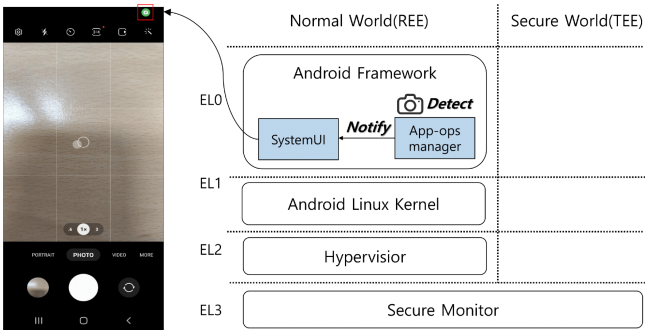


Figure 7: Android privacy indicator block diagram.

Address translation integrity. The hypervisor interposes on the normal world kernel's page table management to ensure address translation integrity on selected addresses. For instance, the in-memory kernel image cannot be relocated or modified through a duplicate RWX mapping. As explained in a previous work [11], this is achieved by eliminating a set of privileged instruction occurrences in the guest kernel. As such, page table updates can only be made through making explicit requests (i.e., through hypervisor calls) to the hypervisor.

By leveraging this security feature, SEPI can ensure the sensitive kernel mappings, such as the camera device's MMIO channel or frame buffer, can be reliably interposed. In other words, the adversary can not relocate or create an illegitimate duplicate mapping of these sensitive memory mappings.

Access control on devices. The hypervisor present in Samsung's mobile devices possesses a unique security layer known as Hypervisor-based Device Manager (HDM), which provides crucial hardware protection against threats [37, 38]. HDM can control the permissions of resources, such as subsystems, in an isolated safe space at a level higher than Android and Linux Kernel by using functions that manage virtual machines.

HDM controls the memory view within the virtual machine (VM). This allows the hypervisor to grant the VM control over memory-mapped system resources that are accessible to it, as well as determine where these resources are located within the VM's address space [10]. Leveraging the benefits of this high privilege of the hypervisor and stage 2 translation, we can govern access permissions for peripheral devices such as cameras and microphones, thereby ensuring the normal operation of devices.

We build SEPI on top of hypervisor-based system integrity guarantees that we explained thus far and inherited its security guarantees. Knox utilizes both a hypervisor (EL2) and TrustZone components to interpose and monitor normal world kernel operations to maintain the system's trustworthiness.

Knox and similar security platforms are designed to maintain their integrity and functionality even under full system compromise. These components undergo integrity checks performed by hardware-supported secure boot. They are also developed to form a small TCB in terms of complexity and code size [37]. SEPI depends on the trustworthiness of the hypervisor and TrustZone components, which are considered part of the TCB. The possibility of compromising these components falls outside the scope of our study.

5.2 Secure Device Status Updates

SEPI leverages a hypervisor-level page monitoring scheme to detect all camera and microphone hardware status changes against its powerful adversary model. By setting the MMIO pages that map the hardware control registers as *read-only*, SEPI's hypervisor is notified of every modification attempt. As a result, when an Android application engages these devices, a page fault is induced in EL1 (i.e., the corresponding device driver), and the execution is transitioned into the hypervisor's fault handler. Upon capturing such privacy-sensitive device status change, the hypervisor toggles the current state (i.e., off vs. in-use) of the device. Finally, the hypervisor emulates the faulting memory store operation before resuming execution.

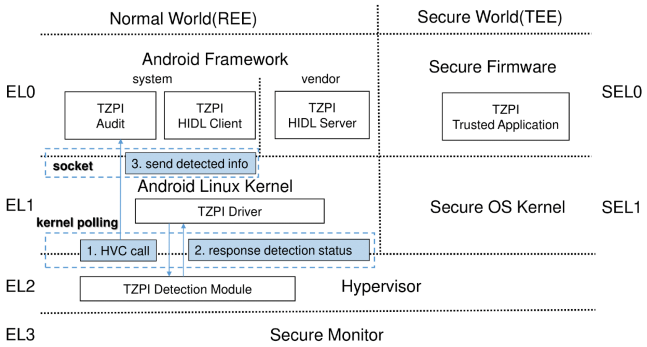


Figure 8: Hypervisor detection block diagram.

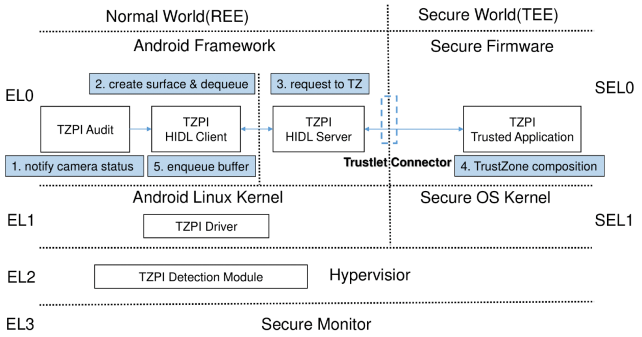


Figure 9: Trustzone composition block diagram.

The device status change record maintained by the hypervisor is periodically queried by the SEPI driver in the EL1; a dedicated kernel thread executes the SEPI driver's polling function with a one-second interval using hypercalls (hvc) as shown in Figure 8.

Status updates sent upwards to the normal world user space notify the SEPI user components (SEPI Audit, HIDL Client, and HIDL Server) to render privacy indicator with the highest Z-order to visually override any fraudulent attempt to mask the indicators. This design allows SEPI to reuse the existing software layers for rendering while still guaranteeing the display integrity of the indicators. The hypervisor and TrustZone secure world software stack designed for implementing sensitive low-level tasks, however, are ill-suited for high-level tasks such as constructing and rendering images, not to mention that the display device is not directly exposed to the secure world in most cases. In the following subsections, we further explain how SEPI render the indicator with TrustZone composition approach.

5.3 Rendering Privacy Indicators

As shown in Figure 9, SEPI Audit monitoring device status receives device status update information from the SEPI driver, and passes it to the SEPI *HAL Interface Description Language (HIDL)* client.

Protected graphics buffer (Secure buffer). SEPI combines the Android OS *Digital Rights Management (DRM)* infrastructure and the hypervisor's address integrity guarantee to implement a protected graphics buffer for privacy indicator rendering. SEPI utilizes Qualcomm's customized implementation of Gralloc. Gralloc is an allocator that allocates device buffers that can be directly accessed by the devices. The customization of the allocator provides hypervisor involvement in the protection of the buffer. As such, not only the content of the buffer is protected, but also the address mapping integrity is guaranteed. This prevents the kernel can arbitrarily modify the address mappings for the buffer. During the allocation of the SEPI's protected graphics buffer, the hypervisor engages and allocates the buffer in a pre-defined physical address range that can only be accessed by the TrustZone secure world.

Privacy indicator layer. HIDL Client is responsible for generating a privacy indicator frame to be loaded into the secure buffer and also passing the frame to the TrustZone which holds exclusive access to the secure buffer. HIDL Client creates a surface with the required width and height for the indicator, along with the `PIXEL_FORMAT_RGBA_8888` attribute. The `PIXEL_FORMAT_RGBA_8888` attribute is used because the alpha channel can represent transparency, allowing the area for drawing the indicator to be transparent except for the indicator itself. Furthermore, the `setLayer` function ensures that the newly created SEPI layer is rendered at the top. In Android, `setLayer` is a method in the View class that adjusts the rendering order and Z-order of a View. By passing a maximum value to the `setLayer` function, the View can be set as the one with the highest Z-order and be rendered at the top.

Finally, the SEPI HIDL system utilizes `hidl_handle` [4] to transmit information about the secure memory allocated by the SEPI HIDL client to the SEPI HIDL server. The implementation is on top of the A52s 5G (A528) device with Qualcomm chipset SM7325, and Qualcomm-provided API was used to pass the memory buffer information from the SEPI HIDL server to TrustZone. The data header passed from SEPI HIDL to TrustZone includes the integrity information of the top Z-order setting from the `setLayer` function along with the indication of the `BufferUsage::PROTECTED` flag activation. The integrity information contains the combination of magic numbers from the top-ordering function and secure buffer flag to the SEPI surface layer.

5.4 Display Integrity Enforcement

SEPI is necessary to confirm whether the graphic buffer received from Gralloc is a secure buffer and whether a surface Z-order is maintained as a top to enforce the display integrity. There are two validation elements for display integrity; (i) Magic number verify the setLayer function set to top Z-order to the frame. (ii) UsageFlags verify the graphic buffer is from a secure buffer. User components hand over the combination of two elements contained in a header when requesting indicator rendering to TrustZone. The SEPI trusted application (TA) already has the information about the validation element and verifies the corresponding value to check the integrity of the graphic buffer.

The TA conducts an initial integrity check on the header information. If the integrity check confirms the authenticity, the TA proceeds with composing the indicator to the received buffer with a physical address. However, if the integrity check fails, indicating potential data tampering, the TA presents an integrity alert image. Additionally, if the Knox warrant fuse is set to 0x1, the TA displays the integrity alert image. Once the composition is completed, a notification is sent back to the SEPI HIDL client, and the composed buffer from TrustZone is enqueued for display on the device.

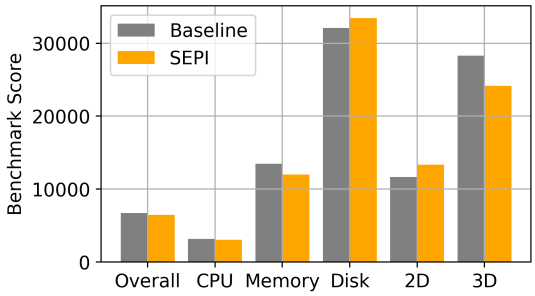
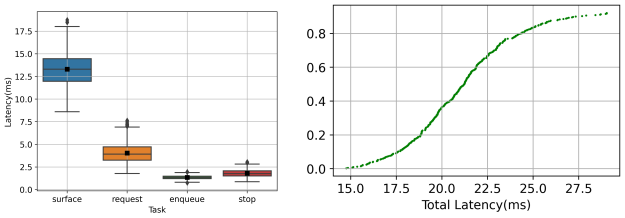


Figure 10: PassMark benchmark score comparing Baseline (factory image) and SEPI detection with one-second kernel polling enabled binary.



(a) Latency for each task.

(b) Cumulative Distribution Function for total (surface + request + enqueue + stop) latency.

Figure 11: Execution latency of tasks and Total latency with TrustZone Composition.

6 EVALUATION

We evaluated SEPI in three aspects for the hypervisor detection and TrustZone composition: (1) performance, (2) security, and (3) Trusted Computing Base (TCB) size. Our experiments were conducted on the Samsung Galaxy A52s 5G model, which is equipped with a Qualcomm SM7325 processor and 6GB memory. The normal world was running Android version 13 and the Linux kernel version 5.4.210, while the secure world was based on the Qualcomm Trusted Execution Environment (QTEE), a hardware-based security solution developed by Qualcomm. QTEE runs in a separate partition on the processor, isolated from the Android operating system and other software on the device.

6.1 Performance Overhead

In this section, we present the impact of SEPI on the performance of hypervisor detection and TrustZone composition.

6.1.1 ***Hypervisor Detection.*** SEPI driver's has a dedicated kernel thread polling function with a one-second interval using hypercalls to detect device status as explained in Section 5.2 We evaluate the impact of kernel thread polling utilizing an Android benchmark application called PassMark, which is available on Google Play. PassMark includes 26 benchmarks in five categories: CPU, memory, disk, 2D graphics, and 3D graphics, and provides an overall system score. We conduct evaluations with two Samsung Galaxy A52s 5G devices, one with a factory binary (Baseline) and the other with a SEPI-enabled binary (SEPI). In the case of the SEPI-enabled binary, the Android Privacy Indicator feature is disabled through the shell command introduced in the Section 3.3 to exclude interference in operation.

PassMark scores while Background Video Recorder Pro (BVR Pro) application is running in the background recording the video. The video recording continuously triggers the kernel thread polling of the SEPI and Android privacy indicator operations for each test case. For the Equivalent test environment, the devices are maintained as follows; (i) keep the battery level at 100%, (ii) record the video in the background through the BVR Pro application, (iii) delete the recorded video file every trial to rule out the impact on the DISK score, (iv) have a five-minute term cooling down the device to rule out the impact of overheating. We measure the average value after 30 times tests for each Baseline and SEPI device.

Figure 10 shows a comparison of benchmark scores between the Baseline and SEPI device. The overall benchmark score of the baseline device is 8,182 points, while the device with SEPI applied shows a minor performance decrease of about 3.3% compared to baseline, with a score of 7,912 points.

6.1.2 ***TrustZone Composition.*** SEPI's TrustZone composition involves executing four primary tasks sequentially: (1) Surface, (2) Request, (3) Enqueue, and (4) Stop.

- Surface involves creating and configuring a surface, as well as dequeuing a buffer for the surface.
- Request involves transferring the dequeued buffer to TrustZone and requesting that the indicator image be composed on the buffer.
- Enqueue involves displaying the image composed on the buffer by TrustZone on the screen.
- Stop involves removing the surface so that the indicator can be removed from the screen.

We conducted a total of 403 repeated tests for each of the four tasks to measure the execution latency when using SEPI. To thoroughly evaluate the data's distribution, we performed the Shapiro-Wilk test to assess data set normality. This statistical test confirmed that the task completion times are not normally distributed. Consequently, we present the median results of our trials along with box plot representations.

Figure 11(a) shows a box plot of the execution latency for each task. To enhance visualization, we excluded certain outliers from these plots. For the surface task, the median latency is 13.298 ms, although a few outliers (2/403) extended up to 70 ms. The median latency for the request task is 3.917 ms; however, one trial (1/403) took as long as 107.147 ms. The enqueue task exhibited the lowest latency among the four, with a median of 1.351 ms, consistently processed within 10 ms. Similarly, the stop task had a median of 1.77 ms, showing no significant load, similar to the enqueue task, with only one trial (1/403) taking up to 14 ms.

Figure 11(b) presents a cumulative distribution function graph for the total latency of the surface, request, enqueue, and stop tasks. In about 3 out of 403 trials, task completion exceeded 100 ms, with one instance reaching a total latency of 185.475 ms. However, the median of the total execution latency (surface + request + enqueue + stop) was 20.92 ms, with an interquartile range of 4.12, suggesting that using SEPI does not incur substantial latency overhead.

6.2 Security Analysis

In this section, we evaluate and describe the security benefits of SEPI's hypervisor detection and TrustZone composition.

6.2.1 ***Hypervisor Detection.*** Android provides a privacy indicator that allows users to detect when the camera and microphone are being used at runtime. However, this privacy indicator is implemented at the EL0 (user mode) level, where detection is performed by AppOpsManager and indicator display is handled by SystemUI. Additionally, the Android privacy indicator can be deactivated using shell commands, making it less secure. In contrast, SEPI's Hypervisor detection takes advantage of the higher privilege level of the Hypervisor (EL2), which can directly access and control hardware peripheral resources such as the camera and microphone. The Hypervisor is isolated from the Android OS (EL1 and EL0 levels), thereby protecting the system from malicious attacks and malware targeting user and kernel spaces.

As a result, SEPI places itself in a secure and isolated environment in which the hardware peripheral resources are accessed directly to confirm the true status of the camera and microphone. In order for the SEPI to distinguish whether this exception is the operation of the camera and microphone, the SEPI is based on each significant predefined value when the camera and microphone operate and does not operate. Since this value is based on the datasheet for the actual operation of the camera and microphone hardware, it cannot be compromised and is unique.

SEPI integrates the hypervisor into the secure boot chain, beginning with the boot ROM. The hash of the root cert for secure booting is fixed to the CPU's eFuse hardware at the manufacturing stage, ensuring the hypervisor's role as a TCB. This setup eliminates the possibility of malicious modifications to the hypervisor and TrustZone images, including SEPI's trusted components. Device status changes are monitored at the hypervisor level due to its exclusive access to Stage 2 page tables and MMIO mappings, which are essential for device communication (e.g., memory-mapped registers).

6.2.2 ***TrustZone Composition.*** SEPI involves a security buffer to provide the following security advantages. First, indicator buffer memory can be protected using a security buffer since the secure memory region is isolated from general graphic memory. The secure buffer protects the rendered indicator images from being accessed or modified by unauthorized applications. Thus, the security of sensitive image data can be ensured.

More specifically, SEPI thwarts the following two malicious operations in user applications that were possible with unprotected privacy indicator rendering in conventional Android OS configuration:

- Losing indicator functionality by accessing the indicator buffer memory in a malicious application and adjusting the icon pixel value to make it transparent. Since the surface of the SEPI is made with the `PIXEL_FORMAT_RGBA_8888` attribute, it becomes transparent when adjusting the alpha value to the max value.
- Malicious applications use screenshots or mirroring to expose images such as indicator images and status bars of target devices, UI overlay attacks using deceptive status bars can be facilitated. The secure buffer can make the indicator frame buffer isolated from the Android OS so that it can be safely protected from the above threats.

(a) Test application unable to render green in the top-most area, designated for the privacy indicator.

(b) Screenshot showcasing the activated privacy indicator (represented by a red dot) during a UI overlay attack.

Figure 12: Screenshot with secure buffer and SEPI with UI overlay attack (DOA).

Figure 12 shows the effects of SEPI's display integrity enforcement mechanism. Figure 12(a) is designed to demonstrate that under SEPI, the untrusted Android OS is restricted from displaying privacy indicators. It highlights the Android OS's limitation in overriding the privacy indicator, which is rendered at a higher Z-order within a secure display buffer. The screenshot from our test application, attempting to display a green overlay across the entire screen, is unable to affect the topmost area reserved for the privacy indicator. This limitation of the Android OS in manipulating content in this specific region is a key focus of our study. Figure 12(b) captures a moment during a UI overlay attack (DOA), where the attacker attempts to display a fraudulent status bar while the activated privacy indicator (represented by a red dot) remains visible on the screen.

6.3 TCB Size

TCB refers to the part of a system that is considered trustworthy. A smaller TCB can result in fewer security vulnerabilities, as it is easier to verify, audit, maintain, and update the code. In the case of SEPI, 0.34K LoC was added for implementing hypervisor detection and 0.29K LoC were added for implementing TrustZone composition. This LoC count only includes pure code and excludes files that were used by the SConScript build system.

7 DISCUSSION

Responsible Disclosure and Ethical Considerations. During our user study, we maintained high ethical standards, prioritizing vulnerability disclosure, harm prevention, and data privacy.

Our responsible disclosure efforts involved active engagement with the security teams at Google and Samsung, focusing on the identified security vulnerabilities. We reported our findings to Samsung Electronics developers responsible for Android system UI and Google's Issue Tracker team. Samsung Electronics acknowledged the issues and promptly addressed them in OneUI 4.1.1, which is shipped in the latest Samsung smartphones. Google recognized our report as a potential abuse of the accessibility service but not as a software bug vulnerability. Due to the stance of Google's Android security team on UI overlay attacks and the accessibility team's emphasis on the legitimate use of overlays, it appears challenging to fundamentally restrict the use of the accessibility service in the Android operating system. Therefore, we have decided not to publish our attack implementation code.

The ethical considerations extended to our user study. We employed a non-invasive approach, utilizing our specially prepared

Pixel smartphones instead of participants' devices. The study focused on interactions with the camera application and privacy indicator awareness. No personal data from participants were collected. Demographic data collection was limited to gender and age range to ensure participant anonymity and privacy. We had contacted our institution's IRB prior to our user study. Our internal review of the institution's IRB terms indicated that our experiment is not subject to an IRB review, and we indeed received an informal confirmation to proceed with the study.

Proposals for trusted component changes. We have demonstrated how attackers can launch deceitful privacy indicator attacks. Motivated by the serious consequences of attacker control of the camera and microphone, and the reported cases of such control being perpetuated through compromised system components, we proposed a prototype defense called SEPI. SEPI's TCB resides in the hypervisor and TrustZone to withstand a complete system compromise and maintain the trustworthiness of the privacy indicators.

However, there are limitations in the design due to the restrictive development environment of the vendor-provided hypervisor and TrustZone implementations. Specifically, the design challenge in establishing SEPI's security model is the necessity of DoS mitigation compared to conventional DRM security solutions. This is because the involvement of untrusted EL1 and EL0 components in SEPI's data flow may hinder the security guarantees since the untrusted components may not comply with privacy indicator rendering requests.

We propose two trivial but imperative changes to Samsung and Qualcomm's secure hypervisor and TrustZone implementations. First, the hypervisor must be able to directly invoke the TrustZone, which holds exclusive rights to the secure buffer. Second, the contents of the graphics framebuffer directly consumed by the display device must also be available in TrustZone's address space. This can be achieved by having the hypervisor create a duplicate memory mapping of the region into the TrustZone memory space.

With these two changes, SEPI's design can incorporate an additional DoS attack mitigation as follows: The hypervisor detects device status changes and requests the EL0 and EL1 components to render the indicator with TrustZone's involvement with the secure buffer, as done in our current design.

The hypervisor then directly invokes the TrustZone afterward to validate the active framebuffer to inspect the EL0 and EL1 components that have correctly serviced the hypervisor's request. We plan to cooperate with the vendors of trusted components for DoS-resistant DRM design with the above-mentioned proposals. **Android software stack.** The SEPI architecture via the hypervisor relies on communication paths between the Android software stack (EL1 and EL0). The TrustZone composition also involves EL0 in allocating and enqueuing a secure buffer. While the current SEPI solution is generally effective, it still faces the risk of OS compromise since it cannot completely exclude the Android surface. To enhance the security against OS compromise, additional measures such as utilizing TrustZone and hypervisor can be considered.

One such measure is to utilize the Trusted Boot Chain (TBC). The TBC is a series of security checks performed during device boot-up to ensure system integrity and protect against security threats. Verification is performed on all components, including the bootloader, operating system, system services, and applications. If TBC verification fails in TrustZone, the hypervisor can utilize its high privilege to block hardware-level access to the camera or microphone from the Android OS, preventing any malicious use of these devices and ensuring a secure environment. However, the best research direction for achieving stronger security would be to find an Indicator solution that completely excludes the Android surface.

One promising approach is to use TrustZone for a secure image display solution on mobile devices. Researchers have proposed a system that utilizes TrustZone and memory protection mechanisms to prevent image content from being leaked through untrusted components. The system provides a secure frame buffer for image display and implements secure image rendering algorithms to prevent information leakage through side-channel attacks. To achieve this, they utilize the Image Processing Unit (IPU), a hardware feature that supports multiple display channels [33]. Implementing the privacy indicator using IPU on commercial devices would require significant collaboration and engineering efforts with chip vendors. However, this could be a promising alternative for future research to reduce the Android surface.

Indicator design. Adversaries can obscure the Android privacy indicator (green dot) by overlaying it with a visually similar green battery icon, as shown in the UI overlay attack demonstration (available at https://www.youtube.com/shorts/1eiqnjZ8Foo). The green battery icon's striking and captivating appearance makes it harder for users to notice the green privacy indicator. SEPI enhances protection against UI overlay attacks by ensuring the highest Z-order, superseding the deceptive status bar created by the UI overlay attacks discussed in this research. However, even with SEPI, attackers can still impede user recognition of the privacy indicator through UI overlay attacks like DOA. Thus, it is crucial to not only prioritize the indicator's position at the top of the Z-order but also to employ an icon design that resists seamless blending with its surroundings. Previous studies, such as those by Amrutkar et al. [6] and Zhang et al. [44], highlight the essential role of privacy indicators in mobile applications. However, considerations regarding UI design are beyond the scope of this paper. Consequently, we did not conduct a user study and underscore the necessity for further comprehensive evaluations to accurately assess the effectiveness of SEPI.

Various image formats. The current implementation of SEPI utilizes a 32-bit RGBA format, assigning 8 bits (ranging from 0 to 255) for each color channel to represent the color information of every pixel. This is a standard practice in digital image processing. In order to support more complex UI components, drawing images in formats such as PNG or JPEG onto the surface created with the `PIXEL_FORMAT_RGBA_8888` attribute may be beneficial. However, before drawing image data onto the surface, the corresponding image must first be loaded into memory and decoded [19]. Open-source libraries such as [1] and libjpeg [22] can handle PNG and JPEG image file formats, respectively. These libraries are compatible with TrustZone, which means that SEPI can be extended to support more image formats. This will ensure the integrity of the application's generic UI components, in addition to the privacy indicator.

8 RELATED WORK

Spyware. Spyware is malicious software that collects users' data and personal information on mobile devices, often for malicious purposes such as identity theft and financial gain. A study analyzed the actions of various spyware applications (e.g., Spyhuman [14], TruthSpy [42], mSPy [24, 28], Cerberus [35], Flexispy [16], Mobistealth [25], Spyfone [32], Retina-X [40, 47]) and found that they pose a significant threat to user privacy. Most of these applications support invisible camera and microphone access, which allows them to record users without their knowledge or consent. To avoid detection, spyware applications often use methods such as creating an invisible preview, accessing the camera's output without displaying a preview or using an invisible WebView to stream live videos. They may also use native code to circumvent Android's limitations on audio recording or employ various workarounds to capture downlink audio from phone calls and third-party applications like WhatsApp. Therefore, providing security measures such as the Android privacy indicator against such spyware is essential.

Transplantation attacks. Researchers have presented a method called transplantation attack [45] that can be used to collect users' personal information through the Android camera service. The camera service is a system service that provides access to the camera hardware for camera applications. To perform a transplantation attack, a malicious application would first create a new camera service with the same name as the legitimate one. This new camera service would then be granted the same system permissions as the legitimate one. The malicious application could then use the new camera service to take pictures or record videos without the user's knowledge or consent. In addition to creating a new camera service, a malicious application could modify an existing camera application. This would allow the malicious application to take pictures or record videos using the modified camera application, even if the user has not granted the application permission to do so. Transplantation attacks are a serious threat to user privacy. They allow malicious applications to take pictures or record videos without the user's knowledge or consent. This can track users' movements, collect sensitive information such as passwords and credit card numbers, or even blackmail users.

Disabling webcam indicator. Most laptops with built-in cameras have an important privacy feature: an LED light that turns on whenever the camera is in use. This light is meant to serve as a warning to users that their camera is active and could be recording them. However, a previous study [12] demonstrated that it is possible to disable these warning lights on MacBooks and iMacs. The researchers reprogrammed the microcontrollers inside the camera, which are responsible for controlling its functions. By reprogramming the microcontrollers, the researchers were able to turn on the camera without turning on the LED light. This finding raises serious concerns about the security of laptop cameras. If someone can disable the warning light, they could secretly record users without their knowledge or consent. Recently, iOS and Android have introduced software-based indicators called privacy indicator [8, 20] for privacy enhancement. These functions are designed to notify users when an application is accessing their camera. However, there is no prior work to investigate the security issues of these privacy indicator features. In this paper, we show that the Android privacy indicator can be obfuscated by the UI overlay attacks, making it unrecognizable, and disabled by the device configuration tampering attack.

UI overlay attacks and misuse of accessibility API. UI overlay attacks are not a novel concept. Fratantonio et al. [17] demonstrated how the combination of `SYSTEM_ALERT_WINDOW` and `BIND_ACCESSIBILITY_SERVICE` permissions could be exploited to manipulate the UI feedback loop. However, as presented in Section 2.3, their techniques result in overlays with Z-order values too low to conceal the privacy indicator UI, which is our target. Diao et al. [15] systematically analyzed the misuse of Android's accessibility service API, revealing significant design flaws and instances of API misappropriation. Our work extends these studies, introducing novel attack scenarios like COA and DOA, specifically targeting the privacy indicator feature in Android 12. Unlike previous studies, we use the `TYPE_ACCESSIBILITY_OVERLAY` parameter in WindowManager combined with `BIND_ACCESSIBILITY_SERVICE` permission to create windows with higher Z-order, enabling the execution of COA and DOA attacks. Huang et al. [21] investigated the potential misuse of accessibility features in 95 apps, proposing an enhanced accessibility framework to improve user privacy while maintaining assistive functionality. However, their framework does not encompass the UI overlay attacks like DOA and COA, as identified in our study.

TrustZone research. Previous research has implemented a mobile peripheral control system using the ARM TrustZone architecture and hardware components provided by the SoC to control the secure modes of the CPU. The system uses a custom kernel designed to run in the TrustZone security mode and utilizes ARM security configuration registers (SCR), the TrustZone Address Space Controller (TZASC), and the Central Security Unit (CSU) to enable secure mobile peripheral control [29]. SEPI controls peripheral devices by changing the access permissions of IO addresses at the hardware level from the hypervisor. This approach has some common characteristics of prior research [29] in operating separately from the Android OS. However, a key distinction lies in SEPI's reliance on Knox HDM [37], a tool used to ensure secure and accurate communication between the Android OS and the hardware. The utilization of Knox HDM enables SEPI to securely control peripheral devices such as the camera and microphone without necessitating significant modifications to the existing system architecture. This ensures not only an added level of security but also maintains the compatibility and user experience of the original system.

9 CONCLUSION

In this paper, we presented two types of attacks: the UI overlay attack and the device configuration tampering attack. Both can disable the Android privacy indicator feature, which is designed to notify users when applications access their camera or microphone. We tested the feasibility of these attacks on various Android devices and applications. Subsequently, we communicated our findings to Samsung Electronics and Google, from whom we received acknowledgments. To counter these security weaknesses, we developed a hardware-based security solution, SEPI. We conducted thorough tests of SEPI on the Samsung Galaxy A52s 5G, where it showed robust protection with only a minimal impact on performance. Since SEPI operates independently from the Android OS, it can promptly restore the privacy indicator in case of tampering.

ACKNOWLEDGMENTS

We deeply appreciate our shepherd and the anonymous reviewers for their constructive comments and feedback. This work was supported by the Korean government: the National Research Foundation of Korea (NRF) grant (NRF-2022R1C1C1010494), the Institute for Information & communication Technology Planning & Evaluation (IITP) grants (No. 2018-0-00532, Development of High-Assurance (>=EAL6) Secure Microkernel (50%), No. 2022-0-01199, Graduate School of Convergence Security, and No. 2022-0-00688, AI Platform to Fully Adapt and Reflect Privacy-Policy Changes).

REFERENCES

[1] [n. d.]. libpng. http://www.libpng.org/pub/png/libpng.html. Accessed: May 4, 2023.
[2] [n. d.]. Magisk GitHub Repository. https://github.com/topjohnwu/Magisk.
[3] 2023. Android Virtualization Framework (AVF) overview. Retrieved June 7, 2023 from https://source.android.com/docs/core/virtualization
[4] 2023. HIDL C++ Types. https://source.android.com/docs/core/architecture/hidl-cpp/types. Accessed: June 29, 2023.
[5] Alexis Ahmed. 2022. CVE-2022-0847 Dirty Pipe Exploits. https://github.com/AlexisAhmed/CVE-2022-0847-DirtyPipe-Exploits
[6] Chaitrali Amrutkar, Patrick Traynor, and Paul C Van Oorschot. 2013. An empirical evaluation of security indicators in mobile web browsers. *IEEE Transactions on Mobile Computing* 14, 5 (2013), 889–903.
[7] Android Developers. 2021. DeviceConfig. https://developer.android.com/reference/kotlin/androidx/wear/watchface/client/DeviceConfig. Accessed on 2023-05-02.
[8] Apple. 2021. *If the camera or flash on your iPhone, iPad, or iPod touch isn't working.* iOS. https://support.apple.com/en-us/HT211808
[9] ARM. 2022. SoC and CPU System-Wide Approach to Security. ARM. https://www.arm.com/technologies/trustzone-for-cortex-a
[10] ARM. 2023. *Stage 2 Translation.*
[11] Ahmed M Azab, Peng Ning, Jitesh Shah, Quan Chen, Rohan Bhutkar, Guruprasad Ganesh, Jia Ma, and Wenbo Shen. 2014. Hypervision across worlds: Real-time kernel protection from the arm trustzone secure world. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 90–102.
[12] Mathew Brocker and Stephen Checkoway. 2014. IseeYou: disabling the MacBook webcam indicator LED. In *23rd USENIX Security Symposium (USENIX Security 14)*. USENIX Association, 387–402.
[13] Checkmarx. 2006. Checkmarx. https://checkmarx.com/ Accessed on August 17, 2023.
[14] Joseph Cox. 2022. *Hacker steals customers' text messages from android spyware company.* https://www.vice.com/en/article/qvm44m/hacker-steals-text-messages-android-spyware-company-spyhuman
[15] Wenrui Diao, Yue Zhang, Li Zhang, Zhou Li, Fenghao Xu, Xiaorui Pan, Xiangyu Liu, Jian Weng, Kehuan Zhang, and XiaoFeng Wang. 2019. Kindness is a Risky Business: On the Usage of the Accessibility {APIs} in Android. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019)*. 261–275.
[16] Lorenzo Franceschi-Bicchierai. 2022. *Stalkerware company flexispy calls catastrophic hack 'just some false news'.* https://www.vice.com/en/article/xyjwpw/flexispy-calls-catastrophic-hack-just-some-false-news
[17] Yanick Fratantonio, Chenxiong Qian, Simon P Chung, and Wenke Lee. 2017. Cloak and dagger: from two permissions to complete control of the UI feedback loop. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1041–1057.
[18] Nitish Gadangi. 2021. Privacy Indicator App. https://github.com/NitishGadangi/Privacy-Indicator-App. Accessed: 2023-05-05.
[19] Google. 2020. Skia: a 2D graphics library for accelerating the user interface. *Google Developers* (2020).
[20] Google. 2021. Android 12 Features and APIs. https://developer.android.com/about/versions/12/features. Accessed on 2023-05-01.
[21] Jie Huang, Michael Backes, and Sven Bugiel. 2021. A11y and Privacy don't have to be mutually exclusive: Constraining Accessibility Service Misuse on Android. In *30th USENIX Security Symposium (USENIX Security 21)*. 3631–3648.
[22] Independent JPEG Group. 2021. libjpeg. http://libjpeg.sourceforge.net/.
[23] Fedor Indutny and Michael Zalewski. 2022. Dirty Pipe: Reading and Writing to Any Memory Location on Linux. *https://dirtypipe.cm4all.com/* (2022).
[24] Cyber Insurance. 2022. *mspy - cyberinsurance.com.* https://www.cyberinsurance.com/breaches/mspy/
[25] Joseph Cox. 2022. *Hacker strikes 'stalkerware' companies, stealing alleged texts and GPS locations of customers.* https://www.vice.com/en/article/7x77ex/hacker-strikes-stalkerware-companies-stealing-alleged-texts-and-gps-locations-of-customers
[26] Karma9874. [n. d.]. AndroRAT. https://github.com/karma9874/AndroRAT. Accessed: May 5, 2023.
[27] Max Kellermann. 2022. The Dirty Pipe Vulnerability. M4ALL. https://dirtypipe.cm4all.com/
[28] Brian Krebs. 2022. *mspy breach krebs on security.* https://krebsonsecurity.com/tag/mspy-breach/
[29] Matthew Lentz, Rijurekha Sen, Peter Druschel, and Bobby Bhattacharjee. 2018. Secloak: Arm trustzone-based mobile peripheral control. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. 1–13.
[30] Bill Marczak, John Scott-Railton, Sarah McKune, Bahr Abdul Razzak, and Ron Deibert. 2018. *Hide and seek: Tracking NSO group's Pegasus spyware to operations in 45 countries.* Technical Report.
[31] MITRE. 2019. CVE-2019-2234. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-2234. Accessed: 2023-05-03.
[32] Charlie Osborne. 2022. *Spyware firm spyfone leaves customer data, recordings exposed online.* https://www.zdnet.com/article/spyware-firm-spyfone-leaves-customer-data-recordings-exposed-online/
[33] Chang Min Park, Donghwi Kim, Deepesh Veersen Sidhwani, Andrew Fuchs, Arnob Paul, Sung-Ju Lee, Karthik Dantu, and Steven Y Ko. 2021. Rushmore: securely displaying static and animated images using TrustZone. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*. 122–135.
[34] Rapid7. 2021. Metasploit. https://www.metasploit.com/. Accessed: 2021-09-29.
[35] Rithvik. 2022. *Cerberus acknowledges data breach, states some usernames and encrypted passwords stolen.* https://www.droidlife.com/2014/03/26/cerberus-data-breach/
[36] Nezer Jacob Zaidenberg Ron Stajnrod, Raz Ben Yehuda. 2021. Attacking TrustZone on devices lacking memory protection. In *Journal of Computer Virology and Hacking Techniques*.
[37] Samsung. 2020. Samsung Knox - HdmManager. https://docs.samsungknox.com/devref/knox-sdk/reference/com/samsung/android/knox/hdm/HdmManager.html.
[38] Samsung Insights. 2022. Defense in Depth: How Samsung Knox Defeats Mobile Malware. https://insights.samsung.com/2022/08/14/defense-in-depth-how-samsung-knox-defeats-mobile-malware-2/.
[39] Saurav Tanwar and Hee Wan Kim. 2022. A study on Dirty Pipe Linux vulnerability. *International Journal of Internet, Broadcasting and Communication* 14, 3 (2022), 17–21.
[40] Lisa Vaas. 2018. Hacker claims spyware maker retina-x has been breached, again. https://nakedsecurity.sophos.com/2018/02/23/hacker-claims-spyware-maker-retina-x-has-been-breached-again/.
[41] Preethi Vennam, Pramod TC, Thippeswamy BM, Yong-Guk Kim, and Pavan Kumar BN. 2021. Attacks and preventive measures on video surveillance systems: A review. *Applied Sciences* 11, 12 (2021), 5571.
[42] Waqas. 2022. *Company that sells spyware to domestic abusers hacked.* https://www.hackread.com/company-that-sells-spywareto-domestic-abusers-hacked/
[43] Yuxuan Yan, Zhenhua Li, Qi Alfred Chen, Christo Wilson, Tianyin Xu, Ennan Zhai, Yong Li, and Yunhao Liu. 2019. Understanding and detecting overlay-based android malware at market scales. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*. 168–179.
[44] Zicheng Zhang. 2021. On the usability (in) security of in-app browsing interfaces in mobile apps. In *Proceedings of the 24th International Symposium on Research in Attacks, Intrusions and Defenses*. 386–398.
[45] Zhongwen Zhang, Peng Liu, Ji Xiang, Jiwu Jing, and Lingguang Lei. 2015. How your phone camera can be used to stealthily spy on you: Transplantation attacks against android camera service. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*. 99–110.
[46] Zimperium. 2021. *PhoneSpy: The App-based Cyberattack Snooping South Korean Citizens.* Zimperium. https://www.zimperium.com/blog/phonespy-the-app-based-cyberattack-snooping-south-korean-citizens/
[47] Zeljka Zorz. 2022. *Retina-x admits they have suffered a data breach - help net security.* https://www.helpnetsecurity.com/2017/05/02/retina-x-data-breach/

A USER STUDY QUESTIONNAIRE

Q1: What is your age in years? [single choice]

- 0-19
- 20-29
- 30-39
- 40-49
- 50-59
- 60 and more
- Prefer not to answer

Q2: What is your gender? [single choice]

- Woman
- Man
- Prefer not to answer

Q3: Which of the following best describes the highest level of formal education that you have completed? [single choice]

- Have not completed high school
- High school or equivalent
- Bachelor or associate degree
- Graduate degree
- Other

Q4: What is your profession? [free text]
Q5: During the use of the camera, did you notice any special icon appearing in the top right corner? [free text]
Q6: If you noticed, can you describe what you recognized? Why do you think you were able to recognize it? [free text]

B USER STUDY DEMOGRAPHICS AND RESULTS

Table 3: Impact of DOA on privacy indicator recognition.

Gender		
Female	9	(20.5%)
Male	35	(79.5%)
Age		
20-29	23	(52.3%)
30-39	10	(22.7%)
40-49	11	(25.0%)
Group A (Default)		
†Yes	14	(63.60%)
‡No	8	(36.40%)
Group B (Under DOA)		
Yes	3	(13.60%)
No	19	(86.40%)

†'Yes' means the user recognized the privacy indicator.
‡'No' means the user did not recognize the privacy indicator.