



## FORENSIC ANALYSIS OF MOBILE APPLICATION SECURITY USING THE IDFIF V2 FRAMEWORK

<sup>1,\*</sup>Abdul Aziz Setiawan, <sup>2</sup>Imam Sutanto

<sup>1,2</sup>Department of Computer Science, Universitas Esa Unggul, Jakarta, Indonesia

<sup>1,\*</sup>abdulsetawan19@student.esaunggu.ac.id, <sup>2</sup>imamsutanto@esaunggu.ac.id

\*correspondence email

### Abstract

*Mobile application security is a critical concern amid the rising frequency of cyber attacks, particularly on the Android platform. This research investigates mobile application vulnerabilities through the lens of the Integrated Digital Forensics Investigation Framework (IDFIF) version 2, with an emphasis on the Laboratory Process stage. Using the Mobile Security Framework (MobSF) for static and dynamic analysis, supported by the Genymotion emulator, the study identifies several vulnerabilities in the tested mobile application. These include malicious permissions (e.g., READ\_EXTERNAL\_STORAGE and WRITE\_EXTERNAL\_STORAGE), the use of outdated v1 signature schemes susceptible to Janus attacks, and the ability to bypass debugging protections, allowing for potential manipulation of the application. Notably, no vulnerabilities were detected in the SSL Pinning process. The findings underscore the importance of addressing these vulnerabilities by removing malicious permissions, updating cryptographic certificate mechanisms, and encrypting sensitive data. Moreover, this research highlights the effectiveness of IDFIF v2 in systematically identifying and analyzing mobile application vulnerabilities. By linking these findings to practical security mitigation measures, the study contributes to the development of more robust mobile security protocols in the future.*

**Keywords:** IDFIFv2, Vulnerability Analysis, Mobile Application Security, Dynamic Analysis, MobSF

### INTRODUCTION

Mobile applications are now an integral part of daily life, with the number of internet users in Indonesia reaching 212.9 million by 2023[1]. While this surge in mobile app usage offers significant benefits, it also increases the risk of cyberattacks. A report by Surfshark[2] indicates that Indonesia faced over 700 million cyberattack incidents in 2022 alone, a sharp increase in cyber threats targeting mobile platforms. Mobile applications, particularly those on the Android operating system, are frequent targets due to their vast user base and relatively open security policies[3]–[5]. Studies by Kusreynada and Barkah[6] emphasize that mobile apps lacking robust security mechanisms—such as malicious permissions and insecure data transmission protocols—are highly susceptible to exploitation. Further, Anwar’s research [4] highlights specific vulnerabilities in Android-based e-commerce apps, particularly regarding data storage and authentication.

Despite the clear risks, there remains a significant gap in the forensic investigation of mobile application security. Existing research in mobile forensics has often focused on general security practices or applied frameworks without delving into a structured[7], [8], systematic approach to vulnerability detection. This paper addresses this gap by applying the Integrated Digital Forensics Investigation Framework (IDFIF) version 2[9]–[11], which provides a structured methodology encompassing preparation, evidence collection, laboratory processing, and the presentation of results. By incorporating both static and dynamic analysis methods, this study aims to offer a comprehensive analysis of mobile application vulnerabilities and their associated security risks.

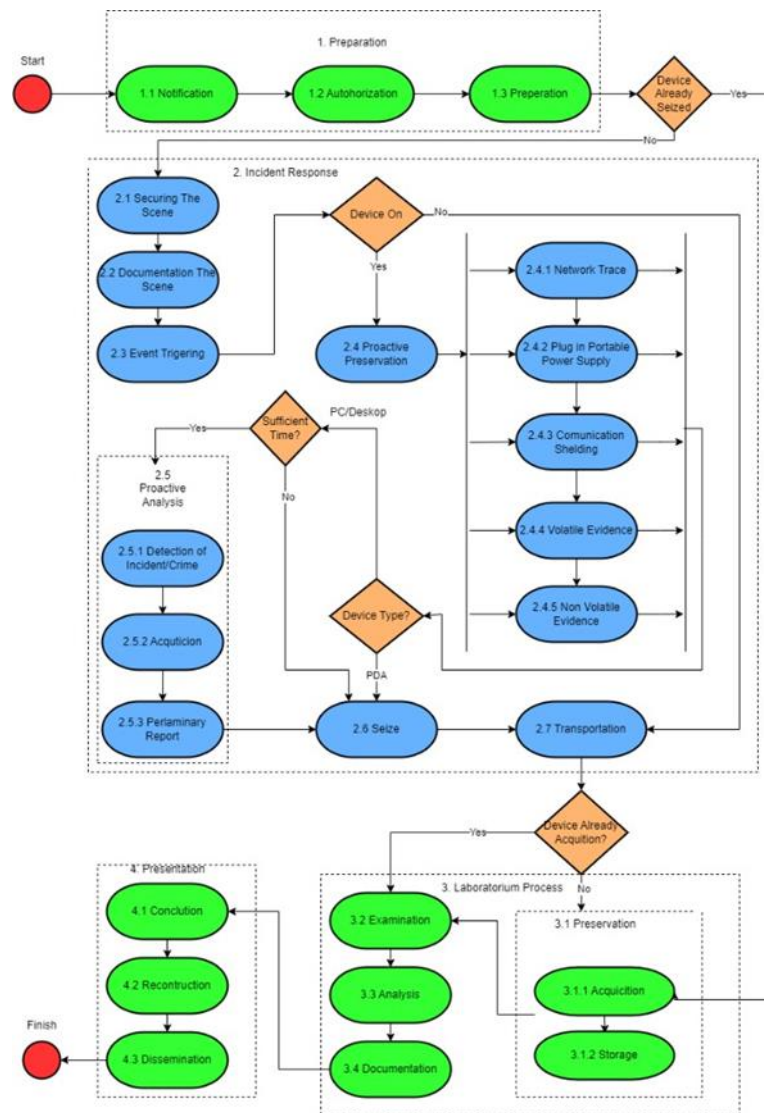
IDFIF v2 is an advanced forensic framework that enables systematic[12], thorough analysis of mobile applications. While other forensic frameworks exist, such as the Digital Forensics Research Workshop (DFRWS) and the Mobile Forensic Analysis Toolkit (MFAT)[13], IDFIF v2 distinguishes itself with its holistic approach, focusing not only on vulnerability detection but also on maintaining data integrity throughout the investigative process. This paper justifies the use of IDFIF v2 over alternative frameworks due to its structured methodology and compatibility with modern forensic tools like the Mobile Security Framework (MobSF)[14]. MobSF, an open-source tool for both static and dynamic analysis[15], [16], has been proven effective in identifying a range of security vulnerabilities, including weaknesses in SSL validation and authentication processes [5]. Recent studies have demonstrated that MobSF is an effective tool in uncovering security flaws, making it an ideal choice for this research.

The novelty of this study lies in its systematic application of IDFIF v2 to mobile app security, with a particular focus on Android-based applications in Indonesia[17], [18]. Unlike previous studies that have concentrated on general security concerns, this research aims to fill the gap in mobile forensic analysis by providing a structured approach to vulnerability detection and mitigation. The results of this study have significant implications for improving security protocols in mobile apps[19], [20], contributing to safer user experiences, particularly in the context of rapidly evolving threats. Recent cyberattacks, such as the massive data breach in Indonesia's e-commerce sector in 2023, underscore the urgency of improving mobile application security. Such incidents highlight the need for robust forensic analysis methods to identify vulnerabilities and safeguard user data.

In conclusion, this study contributes to advancing mobile forensic analysis by utilizing the IDFIF v2 framework in conjunction with MobSF, offering a new approach to identifying and mitigating vulnerabilities in mobile applications. This research is crucial for enhancing mobile security protocols, particularly in Indonesia, where mobile app usage and cyber threats continue to grow.

## METHODS

This study uses the Integrated Digital Forensics Investigation Framework (IDFIF) v2[21] for a systematic and structured approach to mobile application security forensics. The IDFIF v2 framework, which consists of sequential stages such as preparation, incident response, laboratory processing, and presentation, ensures that each phase is conducted with integrity and accuracy to provide reliable analytical results. The framework's structured nature supports reproducibility, a key aspect of any forensic investigation. The framework used in this study is illustrated in Figure 1.



*Fig 1. IDFIFv2 Framework proposed*

The IDFIF v2 framework can be outlined in several key stages. The first stage, Preparation, involves notifying relevant stakeholders about the investigation, securing any necessary permissions, and setting up the tools and environment needed for analysis. In the Incident Response phase, the focus shifts to securing the scene, preserving digital evidence, and taking measures to prevent contamination[22]. The Laboratory Process follows, where the digital evidence undergoes thorough examination and detailed analysis, forming the heart of the forensic investigation. Finally, in the Presentation stage, the results of the analysis are compiled, thoroughly reviewed, and presented in a report that adheres to legal and ethical standards, ensuring transparency and accountability throughout the process. The details of each stage will be further explained in the subsequent subsections.

### Laboratory Process

The focus of this study is on the Laboratory Process because this stage plays a crucial role in identifying and analyzing vulnerabilities in mobile applications. The laboratory process is conducted in stages, including preservation, examination, analysis, and documentation, as illustrated in Figure 2.



*Fig 2. Laboratory Process Overview*

Figure 2 provides an overview of the Laboratory Process, outlining the key stages involved in the forensic analysis of digital evidence. The following detailed description expands on each of these stages to offer a deeper understanding of the steps taken during the laboratory process.

1. **Preservation:** The initial stage of the laboratory process is designed to maintain the integrity and validity of the digital evidence. In this case, the mobile application files collected from external storage devices (e.g., flash drives) are preserved in their original form. Evidence is copied using checksums (e.g., SHA256) to ensure data integrity, as shown in Figure 3.



*Fig 3. Evidence*

2. **Examination:** This stage involves a thorough check of the structure and content of the collected mobile application files. The examination phase is critical for understanding the application's components and identifying potential security weaknesses, such as insecure data storage or permissions that can be exploited.
3. **Analysis:** The analysis phase is where the technical assessment takes place. This study applies both static analysis and dynamic analysis techniques to identify security vulnerabilities in the mobile application. Static analysis involves inspecting the app's code and configuration files, while dynamic analysis involves observing the app's behavior during runtime on an emulator.
4. **Documentation:** After the analysis, detailed documentation is produced to record the steps taken during the forensic process, the findings, and identified vulnerabilities. The documentation is a crucial part of the forensic methodology, ensuring that the investigation is transparent and reproducible.

### Implementation

This study employs the IDFIF v2 Process method to examine vulnerabilities in mobile applications. The stages of the Laboratory Process were rigorously followed to maintain data integrity, conduct thorough analysis, and ensure the findings align with the research objectives.

### Preparation

The preparation stage involves ensuring the tools and environment are ready for the analysis. This includes preparing various tools and materials, such as laptops, flash drives, and software, to support the investigation. A detailed list of these tools and their purposes is provided in Table 1.

**Table 1.** Tools and Materials for Investigation

Tools & Materials	Description	Usability
Laptop	Hardware	Used as the main tool to run all analysis tools.

Flashdisk	Hardware	External storage for storing and transferring tested APK data.
MobSF	Software	Main tool for performing static and dynamic analysis of android application.
Ubuntu 20.04	Software	Operating system used for testing due to its stability and compatibility with forensic tools
Docker Desktop	Software	Easy setup and isolation of MobSF in a containerized environment without the need for complex manual installation.
Genymotion	Software	Android emulator used to run the application and perform dynamic analysis
Browser	Software	Used to access MobSF through the web interface.

### Notification

This step requires notifying relevant parties about the initiation of the investigation.

### Authorization

The final step includes performing a physical examination of the evidence to confirm its accessibility, as shown in Figure 3.

### Tools and Materials

Prepare the tools needed to conduct the investigation as shown in Table 1.

### Preservation

To ensure the integrity and validity of the data throughout the investigation, all collected evidence was securely stored and validated using checksums. The following steps were undertaken.

### Acquisition

The evidence, stored on a flash drive, is analyzed by duplicating its contents to preserve the integrity of the original data. Checksums are then applied to verify the data's integrity. The files collected for this purpose are shown in Figure 4, and the checksum process using SHA256 is illustrated in Figure 5.

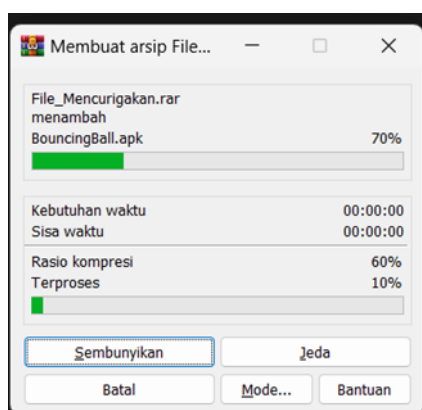


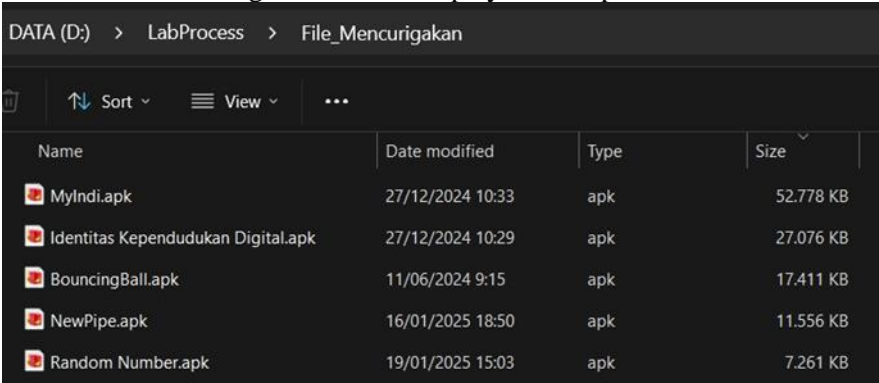
Fig 4. Files collected for checksum

e31c319cfd6459da3ea394f60391a322aeccbab0b1811c57e02cd5f32f31591e File\_Mencurigakan.rar

Fig 5. CheckSum with SHA256

Storage

The acquired data is organized into a secure directory with a clear structure to streamline the analysis process, as shown in Figure 6, which displays the suspicious APK files.



The screenshot shows a Windows File Explorer window with the address bar set to 'DATA (D:) > LabProcess > File\_Mencurigakan'. The window displays a list of five APK files with columns for Name, Date modified, Type, and Size.

Name	Date modified	Type	Size
MyIndi.apk	27/12/2024 10:33	apk	52.778 KB
Identitas Kependudukan Digital.apk	27/12/2024 10:29	apk	27.076 KB
BouncingBall.apk	11/06/2024 9:15	apk	17.411 KB
NewPipe.apk	16/01/2025 18:50	apk	11.556 KB
Random Number.apk	19/01/2025 15:03	apk	7.261 KB

Fig 6. Suspicious APK files

Examination

The initial examination of the mobile application involves checking key file attributes, such as the application name, size, and version. This step sets the foundation for further analysis.

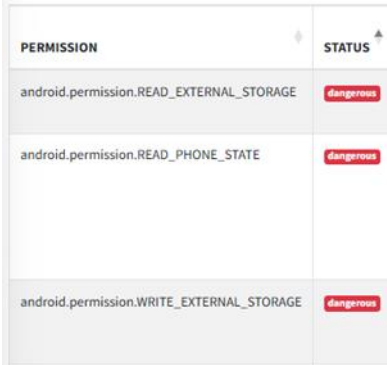
Analysis

The analysis phase involves two primary approaches: static and dynamic analysis. Both methods are applied sequentially, with the data from one APK file used as an example in the following sections.

Static Analysis

Static analysis focuses on examining the application's file structure to detect potential vulnerabilities, such as insecure configurations or suspicious permissions. For example, Figure 7 displays the application's permissions, where malicious permissions related to READ\_EXTERNAL\_STORAGE, READ\_PHONE\_STATE, and WRITE\_EXTERNAL\_STORAGE are identified. These permissions can allow the app to access external storage, retrieve phone information (e.g., phone number, call status), and write to external storage, which are potential security risks.

Figure 8 and Figure 9 highlight the certificate details and analysis. In Figure 8, the signer's certificate confirms that the application is from a legitimate developer. However, Figure 9 indicates that the application uses the v1 signature scheme, which is vulnerable to the Janus vulnerability on Android 5.0-8.0 devices when signed with only the v1 scheme. Applications on Android versions 5.0-7.0 signed with both v1 and v2/v3 schemes are also at risk.



The screenshot shows a table of permissions with two columns: PERMISSION and STATUS. Three permissions are listed, all marked as 'dangerous'.

PERMISSION	STATUS
android.permission.READ_EXTERNAL_STORAGE	dangerous
android.permission.READ_PHONE_STATE	dangerous
android.permission.WRITE_EXTERNAL_STORAGE	dangerous

Fig 7. Permission





Fig 8. Certificate

Figure 10 shows an issue with the Android manifest file, where the launch mode of activities should not be set to "single task." This configuration can result in Task Hijacking, where malicious activities from other apps can be placed on top of the application's stack. In Figure 11 and Figure 12, the application is shown to be in debug mode, allowing for logging and debugging during runtime. Debugging can be exploited by attackers using ADB (Android Debug Bridge) to modify runtime variables or bypass security mechanisms. This is evidenced in Figure 12, where the debug mode is clearly visible (debug=true).

TITLE	SEVERITY
Application vulnerable to Janus Vulnerability	high
Signed Application	Info

Fig 9. Certificate Analysis

Activity (com.prime31.UnityPlayerNativeActivity) is vulnerable to Android Task Hijacking/StrandHogg.	high
--	------

Fig 10. Manifest Analysis

Debug configuration enabled. Production builds must not be debuggable.	high
--	------

Fig 11. Code Analysis

```
1. package com.prime31.InAppBilling;
2.
3. /* loaded from: classes.dex */
4. public final class BuildConfig {
5.     public static final boolean DEBUG = true;
6. }
```

Fig 12. File Code

Dynamic Analysis

Dynamic involves observing the behavior of the application as it runs on the emulator to detect suspicious activity.

API monitoring in performing dynamic analysis refers to the collection and examination of data related to the functionality of the API, the monitoring process in several activities in the application being tested. Therefore, the API monitoring process is very important to ensure the smoothness and functionality of the application.

Figure 13 illustrates the SSL Pinning Bypass, which is used to validate the SSL certificate of the target server. The test reveals no security vulnerabilities in the SSL Pinning Bypass process that could compromise the application. This suggests that only a limited number of SSL certificates are accepted to establish a connection with the server, effectively preventing eavesdropping on client-server traffic through the use of counterfeit certificates.

Root detection bypass is a method used to evade the detection of root access or superuser permissions on Android devices, which allows full access to and modification of the system. Figure 14 shows that the superuser application fails to detect superuser permissions being used by other applications. Figure 15 shows Frida successfully hooking several child processes of the target application. These processes may be related to the application's anti-debugging mechanism. By hooking, Frida can modify the behavior of these child processes so that the application can no longer detect the presence of the debugger.

### Documentation

After completing both static and dynamic analyses of the mobile application stored on the flash drive, the next step is to document the results, outlining all findings and vulnerabilities identified during the examination.

### Presentation

This stage involves compiling and presenting the results of the analysis.

Based on the results of static and dynamic application tests conducted through the Laboratory Process using MobSF tools, the results show that vulnerabilities in the application are found where the static analysis in Figure 11 and 12 shows that the debugging can be manipulated which is found in the dynamic analysis in Figure 14.

### Reconstruction

#### Output

This vulnerability could allow an attacker to leverage tools such as ADB to access sensitive data, read application logs, perform code injection, or modify application behavior during runtime. If left unaddressed, an attacker could escalate access privileges, read or write application data, and expose user information. These exploits can be prevented by disabling debugging on production applications, encrypting sensitive data, and implementing runtime manipulation detection.

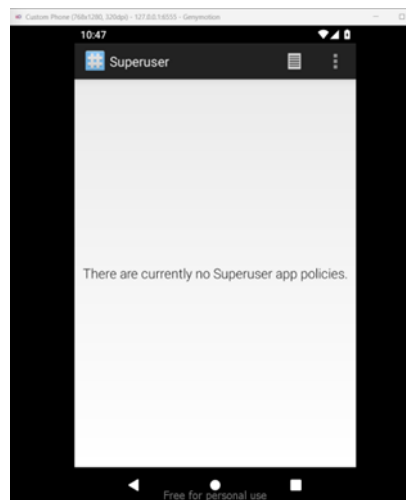
#### Frida Logs

```
Data refreshed in every 3 seconds.

Loaded Frida Script - ssl_pinning_bypa
Loaded Frida Script - dump_clipboard
Loaded Frida Script - debugger_check_bj
Loaded Frida Script - api_monitor
Loaded Frida Script - root_bypass
[SSL Pinning Bypass] okhttp Certificate
[SSL Pinning Bypass] okhttp3 Certificat
[SSL Pinning Bypass] DataTheorem trust
[SSL Pinning Bypass] Appcelerator Pinni
[SSL Pinning Bypass] Apache Cordova SSI
[SSL Pinning Bypass] Wultra CertStore.v
[SSL Pinning Bypass] Xutils not found
[SSL Pinning Bypass] httpclientandroid
[SSL Pinning Bypass] Cronet not found
[SSL Pinning Bypass] Boye AbstractVerifi
[SSL Pinning Bypass] babylon certificat
[SSL Pinning Bypass] Appmattus Certific
[SSL Pinning Bypass] certificatetransp
[API Monitor] Cannot find com.android.c
```

*Fig 13. SSL Pinning bypass*





*Fig 14. Superuser apps*

```

- [Frida] Loaded Frida Script - debugger_check_bypass
- Starting Instrumentation
- Frida Server is already running
- Spawning com.ketchapp.bouncingball
- [Frida] Loaded Frida Script - debugger_check_bypass
- [Frida] [Debugger Check] Hook fork child process PID: 9710
- [Frida] [Debugger Check] Hook fork child process PID: 9730

```

*Fig 15. Debbing*

## RESULT AND DISCUSSIONS

This section presents the findings from the static and dynamic analysis of six Android applications using the IDFIF v2 framework, primarily focusing on identifying security vulnerabilities. The vulnerabilities discovered are categorized based on their severity and potential impact. The following sections discuss the findings in detail, providing context for each vulnerability and its practical implications.

### Static Analysis Findings

Table 2 summarizes the key vulnerabilities found in the static analysis of the applications.

**Table 2.** Static Analysis Vulnerabilities

Vulnerability categories	My IndiHome	IKD	NewPipe	Bouncing Ball	Random Number	SatuSehat
<i>Dangerous Permission</i>	✓	✓	✓	✓	✓	✓
<i>Certificate Analysis</i>				✓		
<i>Manifest Analysis</i>		✓			✓	✓
<i>Code Analysis</i>		✓	✓	✓	✓	✓
<i>Domain Malware Check</i>	✓	✓	✓	✓	✓	✓

The static analysis revealed several key vulnerabilities across the six applications. All the applications were found to request at least one dangerous permission, such as `READ_EXTERNAL_STORAGE`, `WRITE_EXTERNAL_STORAGE`, `READ_PHONE_STATE`, which attackers commonly exploit to gain unauthorized access to sensitive data on the device. These permissions pose significant risks to user privacy and data security. Additionally, NewPipe was identified with a certificate vulnerability, indicating issues with improper certificate validation or insecure communication protocols, making the app susceptible to man-in-the-middle (MITM) attacks. In terms of manifest and code analysis, several applications, including IKD, NewPipe, and Bouncing Ball, were found to have insecure configurations in their manifest files or code, leading to vulnerabilities. These included unprotected exported activities and the use of weak random number generation algorithms, which represent critical security risks.

### Dynamic Analysis Findings

The dynamic analysis, as presented in Table 3, explores vulnerabilities that can be exploited during runtime when the application is executing. It highlights behaviors such as API monitoring, SSL Pinning Bypass, and root detection bypass.

**Table 3.** Dynamic Analysis Vulnerabilities

Vulnerability categories	My IndiHome	IKD	NewPipe	Bouncing Ball	Random Number	SatuSehat
API Monitoring	✓	✓	✓	✓	✓	✓
SSL Pinning Bypass	✓				✓	
Root Detection Bypass						
Debugger Checker				✓		

The dynamic analysis revealed several key findings. API Monitoring showed that all applications exposed API endpoints, which could be monitored by attackers to gain insights into the app's internal operations, potentially enabling data manipulation or theft. SSL Pinning Bypass was found in My IndiHome and Random Number, leaving them vulnerable to Man-in-the-Middle (MITM) attacks, where attackers could intercept encrypted traffic between the client and server. This poses a serious risk for apps handling sensitive data, as attackers could steal login credentials or alter data. Root Detection Bypass was observed in IKD, though it remained unclear whether this vulnerability would allow attackers to fully access the app's features. Root access on Android devices can enable attackers to modify system files or bypass security measures. Lastly, Debugger Checker vulnerabilities were identified in IKD, allowing attackers to bypass debugger detection, potentially enabling them to inject malicious code or analyze the app's execution to extract sensitive data or exploit flaws.

### Discussion of Findings

Based on the results of both static and dynamic analysis, the identified vulnerabilities were classified according to their severity, as shown in Table 4.

**Table 4.** Vulnerability Levels and Descriptions of Analyzed Applications

Application Name	Vulnerability Level	Description
My IndiHome	Moderate Risk	It has some dangerous permissions such as SQL Injection, but there is SQLChiper to mitigate the risk of the database.

<i>IKD</i>	Moderate Risk	SQL Injection, Root Detection Bypass, malicious permission, and unencrypted activity, but in the dynamic test no visible hazard were found.
<i>NewPipe</i>	Extrime Risk	Vulnerable to SQL injection, has unprotected exported activities, and can bypass SSL Pinning Bypass.
<i>Bouncing Ball</i>	Ekstirme Risk	Vulnerable to SQL injection, Task Hijacking, and Janus Vulnerability.
<i>Random Number</i>	High Risk	Vulnerable to SQL injection, uses weak random algorithm, and has unprotected exported activity.
<i>SatuSehat</i>	Moderate Risk	Vulnerable to malicious permission, has exported activities, and supports cleartext communication.

A security analysis of six Android apps found varying levels of risk that could threaten data security and user privacy. Apps such as Bouncing Ball, NewPipe, and Random Number were categorized as high-risk due to vulnerabilities to SQL Injection, unprotected exported activities, and weak security implementations, including SSL Pinning bypass and the use of weak random algorithms. These vulnerabilities present significant opportunities for attackers to exploit data or manipulate application functions.

Alternatively, applications like IKD (Identitas Digital), My IndiHome, and SatuSehat fall into the medium risk category. Although IKD has vulnerabilities such as SQL Injection, malicious permissions, and unencrypted activities, the overall risk level can be mitigated with appropriate measures. My IndiHome utilizes SQLCipher to secure the database, despite having malicious permissions, while SatuSehat shows vulnerabilities in exported activities and cleartext communication. These findings emphasize the importance of app security practices, including strong data encryption, strict input validation, as well as managing app permissions to reduce the potential for exploitation.

## CONCLUSIONS

This research successfully applies the Laboratory Process in the IDFIF v2 method to analyze the vulnerability of mobile applications. Results show that the tested applications have significant vulnerabilities, including malicious permissions and weaknesses in runtime detection mechanisms. The analysis process using MobSF provides important insights for improving application security, especially in terms of data encryption, runtime detection, and debugging mode. This research confirms the importance of IDFIF v2 as an effective framework in mobile application security investigations.

For future research, combining MobSF with other tools, such as Burp Suite or OWASP ZAP, could enhance vulnerability detection, especially for network communications and APIs. Collaborating with app developers or technology companies to test the results on their applications and obtain direct feedback could further validate the effectiveness of the proposed mitigation measures. Additionally, conducting research using physical devices instead of emulators could provide a more accurate understanding of how applications interact with real hardware and network environments, closely simulating actual user conditions.

## REFERENCES

- [1] A. Ma'Arif, A. I. Cahyadi, S. Herdjunto, and O. Wahyunggoro, 'Tracking Control of High Order Input Reference Using Integrals State Feedback and Coefficient Diagram Method Tuning', *IEEE Access*, vol. 8, pp. 182731–182741, 2020, doi: 10.1109/ACCESS.2020.3029115.
- [2] J. C. Maxwell, *A Treatise on Electricity and Magnetism*, 3rd ed. Oxford: Clarendon, 1892.
- [3] N. Anwar, S. A. Akbar, A. Azhari, and I. Suryanto, 'Ekstraksi Logis Forensik Mobile pada Aplikasi E-Commerce Android', *Mob. Forensics*, vol. 2, no. 1, pp. 1–10, Mar. 2020, doi: 10.12928/mf.v2i1.1791.
- [4] N. N. Abbas, A. A. Zeerak, M. A. Javaid, and M. Hussain, 'Comparative Forensic Analysis of Android based Social Media Applications', *Mob. Forensics*, vol. 4, no. 2, pp. 102–114, Feb. 2023, doi: 10.12928/mf.v4i2.6270.
- [5] P. B. Pangestu and M. Koprari, 'Comparison of Forensic Tool Results on Android Smartphone Backup Files Using NIST Method', *Mob. Forensics*, vol. 4, no. 2, pp. 115–126, Feb. 2023, doi: 10.12928/mf.v4i2.6496.
- [6] A. Ma'arif, A. imam Cahyadi, and O. Wahyunggoro, 'CDM Based Servo State Feedback Controller with Feedback Linearization for Magnetic Levitation Ball System', *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 8, no. 3, p. 930, Jun. 2018, doi: 10.18517/ijaseit.8.3.1218.
- [7] N. Huda, L. Aulia, and M. C. Pandini, 'Identification of Plasmodium Vivax in Blood Smear Images Using Otsu Thresholding Algorithm', *Mob. Forensics*, vol. 6, no. 2, pp. 61–73, Sep. 2024, doi: 10.12928/mf.v6i1.11261.
- [8] I. R. Tuharea, A. Luthfi, and E. Ramadani, 'Social Media Metadata Forensic Ontology Model', *Mob. Forensics*, vol. 5, no. 2, pp. 1–14, Sep. 2023, doi: 10.12928/mf.v5i2.8937.
- [9] I. Riadi, A. Yudhana, and W. Y. Sulisty, 'Analisis Image Forensics Untuk Mendeteksi Pemalsuan Foto Digital', *Mob. Forensics*, vol. 1, no. 1, p. 13, Sep. 2019, doi: 10.12928/mf.v1i1.703.
- [10] A. S. M. M. Rahaman, S. Marzia, T. H. Arnob, M. Z. Rahman, and J. Akhter, 'Forensic Artifact Discovery and Suspect Profiling through Google Assistant', *Mob. Forensics*, vol. 5, no. 1, pp. 1–11, Sep. 2023, doi: 10.12928/mf.v5i1.8046.
- [11] M. E. Apriyani, R. A. Maskuri, M. H. Ratsanjani, A. Pramudhita, and R. Rawansyah, 'Forensic Digital Analysis of Telegram Applications Using the National Institute Of Justice and Naïve Bayes Methods', *Mob. Forensics*, vol. 5, no. 2, pp. 21–30, Sep. 2023, doi: 10.12928/mf.v5i2.7893.
- [12] A. H. Muhammad and G. Mandar, 'National Institute of Standard Technology Approach for Steganography Detection on WhatsApp Audio Files', *Mob. Forensics*, vol. 6, no. 2, pp. 74–82, Sep. 2024, doi: 10.12928/mf.v6i2.11287.
- [13] S. Kartoirono, I. Riadi, F. Furizal, and A. Azhari, 'Improved Breadth First Search For Public Transit Line Search Optimization', *Mob. Forensics*, vol. 5, no. 1, pp. 12–22, Mar. 2022, doi: 10.12928/mf.v5i1.7906.
- [14] G. B. Akintola, 'Evaluating the Security Vulnerabilities of the Selected Mobile Forensic Applications', *Int. J. Sci. Res. Multidiscip. Stud.*, vol. 11, no. 2, pp. 16–35, 2025.
- [15] A. S. B. Kusreynada, Sabrina Uhti, 'Android Apps Vulnerability Detection with Static and Dynamic Analysis Approach using MOBSF', *J. Comput. Sci. Eng.*, vol. 5, no. 1, pp. 46–63, 2024.
- [16] H. Shahriar, C. Zhang, M. A. Talukder, and S. Islam, 'Mobile Application Security Using

- Static and Dynamic Analysis', 2021, pp. 443–459. doi: 10.1007/978-3-030-57024-8\_20.
- [17] T. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, 'Electron Spectroscopy Studies on Magneto-Optical Media and Plastic Substrate Interface', *IEEE Transl. J. Magn. Japan*, vol. 2, no. 8, pp. 740–741, 1987, doi: 10.1109/TJMJ.1987.4549593.
- [18] H. Herpindo, R. Ristiyani, M. Rizqin Nikmatullah, and R. Ngestrini, 'Developing an Android Application for Analyzing Indonesian Syntax: A Rule and Probability-Based POS Tagging Approach', *REiLA J. Res. Innov. Lang.*, vol. 6, no. 2 SE-Articles, pp. 125–142, Jul. 2024, doi: 10.31849/reila.v6i2.14975.
- [19] M. Liyanage *et al.*, 'Enhancing Security of Software Defined Mobile Networks', *IEEE Access*, vol. 5, pp. 9422–9438, 2017, doi: 10.1109/ACCESS.2017.2701416.
- [20] Z. Trabelsi, M. Al Matrooshi, S. Al Baira, W. Ibrahim, and M. M. Masud, 'Android based mobile apps for information security hands-on education', *Educ. Inf. Technol.*, vol. 22, no. 1, pp. 125–144, Jan. 2017, doi: 10.1007/s10639-015-9439-8.
- [21] K. Ogata, *Modern control engineering*, 5th ed. New York: Pearson Education.
- [22] A. Ma'arif, A. I. Cahyadi, O. Wahyunggoro, and Herianto, 'Servo state feedback based on Coefficient Diagram Method in magnetic levitation system with feedback linearization', in *2017 3rd International Conference on Science and Technology - Computer (ICST)*, Jul. 2017, pp. 22–27. doi: 10.1109/ICSTC.2017.8011846.