# Forensic Analysis of Gemini's Facilitation of Malware Generation and Distribution*

Gayoon Kim[1], Minjung Yoo[1], Seunghyun Park[1], and Seongmin Kim[1,*]

Sungshin Women's University, Seoul, Korea
{20240926, 20211079, 20211058, *sm.kim*}@sungshin.ac.kr

**Abstract**

Recent reports highlight the misuse of generative AI for malware development. In particular, Gemini's integration with Google services allows the immediate distribution of generated code, complicating efforts to reconstruct user activity when relying solely on chat logs. To address this challenge, this study conducts a forensic analysis of artifacts produced during code generation and sharing through the mobile Gemini application, and proposes an approach for investigating AI-assisted malicious activities.

*Keywords* – Google Gemini · Mobile Forensics · Digital Trace Analysis · Google Takeout

## 1 Introduction

Advances in generative AI have enabled the creation of complex code without requiring programming expertise. Recent studies show that Large Language Models (LLMs) can be prompted to produce malicious code [1],and that Gemini, in particular, demonstrates high capability in generating phishing and social engineering content [2]. Unlike conventional LLMs, Gemini is tightly integrated with Google services such as Gmail, Colab, and Google Drive, allowing generated code to be instantly shared or modified. This integration increases the potential for malicious exploitation, including automated malware distribution.

At the same time, such integration leaves residual forensic artifacts across multiple platforms. These artifacts may persist even after conversations are deleted, enabling the reconstruction of user activity and inference of operational intent. This study investigates the forensic implications of Gemini-based code generation in mobile environments, focusing on logs and artifacts produced through its integration with Google services.

## 2 Methodology

To investigate forensic artifacts associated with Gemini-based code generation and distribution, real-time log data and cloud-stored records were analyzed in conjunction. ADB Logcat revealed user account identifiers associated with the Gemini application, enabling the selection of corresponding target accounts for Google Takeout. Logcat entries such as `#sendRobinQuery` provided timestamps of prompt submissions, confirming user-initiated interactions, while `APSResolverComparator: AppPredictionService` logs identified frequently used applications during content sharing—supporting the prioritization of service-level artifact extraction. However, although these entries confirm that a query was issued to Gemini, both

*Proceedings of the 9th International Conference on Mobile Internet Security (MobiSec'25), Article No. P-34, December 16-18, 2025, Sapporo, Japan. 

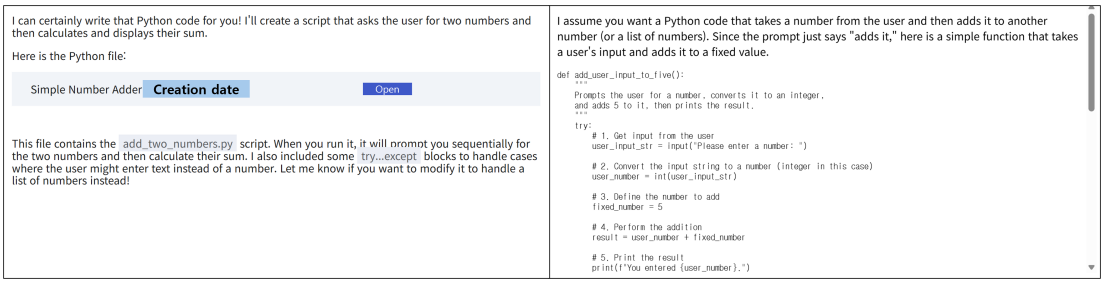


Figure 1: Difference in code visibility in Gmail export of Gemini-generated code: Canvas export (left) vs. standard prompt export (right)

user prompts and Gemini's responses appear as "REDACTED" and cannot be directly recovered from Logcat. Therefore, Takeout data is essential for retrieving preserved content and correlating it with specific user actions.

Gemini's export mechanisms disseminate generated code across multiple Google services. When exporting to Gmail, if the code is produced through standard prompt-based interactions, both the code and Gemini's explanation appear directly within the email body. In contrast, Canvas-based exports omit the code from the message, displaying only descriptive text and redirecting access to an external file, as shown in Figure 1. Exporting to Colab (available only on the Web) creates a notebook in Google Drive that retains the complete code along with its execution context. These behavioral differences determine how artifacts are preserved within Google Takeout, which serves as the primary source for cross-service forensic validation.

For Colab exports, Google Takeout provides not only the primary notebook file (e.g., Simple Number Adder), which contains the generated code and explanatory output, but also a corresponding metadata file (e.g., Simple Number Adder-Information) detailing creation history and account association. These metadata files include fields, such as `last_modified_by_me` and `last_modified_by_any_user` to identify the editor, and `permissions` to indicate sharing activity and access roles. The `execution_count` field specifies whether the code was executed, while `"metadata": {"colab": {"from_bard": true}}` confirms that the notebook originated from Gemini (formerly Bard). This provenance marker establishes a verifiable link between AI-assisted code generation and subsequent dissemination.

# 3 Conclusion

This study confirms that Gemini-based code generation and distribution activities are not confined to a single application but leave forensically significant artifacts across interconnected Google services. These artifacts enable the identification of file creation events, modification histories, and participant involvement. Collectively, they demonstrate the forensic value of such data in tracing coordinated actions related to malware development and reconstructing the operational pathways of the actors involved.

# References

[1] Evangelos Dragonas, Costas Lambrinoudakis, and Panagiotis Nakoutis. Forensic analysis of openai's chatgpt mobile application. *Forensic Science International: Digital Investigation*, 50:301801, 2024.

[2] Orçun Çetin, Baturay Birinci, Çağlar Uysal, and Budi Arief. Exploring the cybercrime potential of llms: A focus on phishing and malware generation. In *European Interdisciplinary Cybersecurity Conference*, pages 98–115. Springer, 2025.