Poster: De-obfuscation System for Obfuscation Techniques based on Trampoline Code

Gwangyeol Lee[*], Minho Kim, Jeong Hyun Yi, and Haehyun Cho

Soongsil University, Seoul, Korea
gwangyeal@gmail.com, mhkim37@soongsil.ac.kr, {jhyi,haehyun}@ssu.ac.kr,

Abstract

Security researchers work to analyze and counteract malware. However, attackers continuously attempt to evade these analyses, using obfuscation techniques. While previous studies proposed many methodologies, they failed to address OEP obfuscation and API obfuscation, resulting in unpacking failures. In this work, we propose an automatic de-obfuscation system against OEP obfuscation and API obfuscation. Our system analyzes the memory dump of packed programs, detects trampoline codes, and identifies obfuscated data for the program reconstruction.

Keywords: OEP obfuscation, API obfuscation, De-obfuscation, Unpacking

1 Introduction

Malware remains a major threat, causing significant damage across servers, PCs, and mobile devices [1]. While many methods exist for analyzing malware, attackers continually implement to evade these techniques [2]. They often use *packers* to apply obfuscation techniques. Modern packers protect code and data to complicate program analysis [3, 4]. They use trampoline code to conceal the OEP (Original Entry Point) and IAT (Import Address Table). In this work, we propose an effective de-obfuscation system targeting OEP obfuscation and API obfuscation techniques.

2 Obfuscation Techniques

Obfuscation techniques restore protected data to memory at runtime using trampoline codes. In this section, we introduce two main obfuscation techniques that leverage trampoline codes.

OEP obfuscation either positions a trampoline code at the OEP or modifies the runtime stack to trigger it. API obfuscation links the trampoline code's address in the IAT or modifies the API call to trampoline code [5].

3 Proposed System

In this study, we aim to create an automated de-obfuscation system that detects, executes, and analyzes trampoline codes, then patches the programs for unpacking. Specifically, we focus on de-obfuscating the techniques discussed in section 2 to unpack obfuscated programs.

[*]Corresponding author: Soongsil University, Seoul, 06978, Korea, Email: gwangyeal@gmail.com

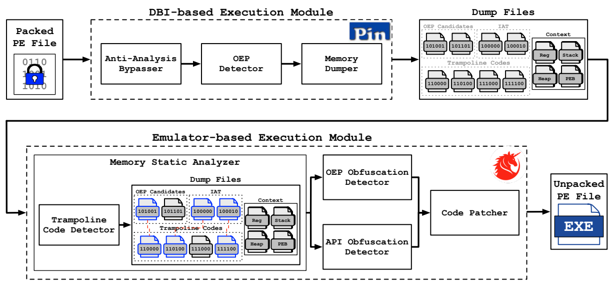

Figure 1: The overview of De-obfuscation System.

Figure 1 provides the overview of our system. It runs an obfuscated program to capture in-memory data, identifies all trampoline codes, and independently executes them to determine their obfuscation purpose. The system then de-obfuscates and reconstructs the program, ensuring a de-obfuscated OEP and Import Table.

4 Evaluation Results and Future Work

We evaluated our system using the sample program, obfuscated with Themida [3], VMProtect [4], ASProtect [6], Enigma [7], and Obsidium [8]. The results are shown in Table 1. We found that ASProtect and Obsidium don't employ trampoline codes for OEP obfuscation, while Themida does at the OEP. VMProtect and Enigma trigger the trampoline code post-OEP using stack manipulation. For API obfuscation, all packers call trampoline code either directly or indirectly. Despite ASProtect's use of an argument-sensitive trampoline code, we identified the obfuscated APIs. Our next step is to test our system against real-world obfuscated malware.

	OEP obfuscation	API obfuscation
Themida	✓	✓
VMProtect	△	✓
ASProtect	–	✓
Enigma	△	✓
Obsidium	–	✓

Table 1: Comparison of De-obfuscation Results.

Acknowledgement: This work was supported by the National Research Foundation of Korea (NRF) Grant through the Korean Government (MSIT) under Grant NRF-2021R1A4A1029650.

References

[1] AV-TEST. Statistics of malware. https://www.av-test.org/en/statistics/malware/, 2023.

[2] FireEye. M-trends 2020. https://content.fireeye.com/m-trends/rpt-m-trends-2020, 2020.

[3] Oreans Technologies. Themida. https://www.oreans.com/Themida.php, 2004–2022.

[4] VMProtect Software. VMProtect Software Protection. https://vmpsoft.com/, 2003–2022.

[5] B. Cheng, J. Ming, E. Leal, H. Zhang, J. Fu, G. Peng, and J. Marion. {Obfuscation-Resilient} executable payload extraction from packed malware. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 3451–3468, 2021.

[6] StartForge. ASProtect. http://www.aspack.com/asprotect32.html/, 2007–2022.
[7] The Enigma Protector. Enigma protector. https://enigmaprotector.com/, 2004–2022.
[8] Obsidium Software. Obsidium software protection system. https://www.obsidium.de/, 2022.