Bachelor's Thesis

(Course Code: XB_40001)

Mitigations of hypervisor detection techniques in Windows using HyperDbg

by

Mārcis Zariņš

(Student Number: 2781630)

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Science
in
Computer Science
at the
Vrije Universiteit Amsterdam

May 23, 2025

Certified by ....

Erik van der Kouwe
First supervisor's title
First Supervisor

Certified by ....

Mohammad Sina Karvandi
Daily supervisor's title
Daily Supervisor

Certified by ....

Second Reader

Mitigations of hypervisor detection techniques in Windows using HyperDbg

Mārcis Zariņš
Vrije Universiteit Amsterdam
Amsterdam, NL
m.zarins@student.vu.nl

ABSTRACT

The abstract.

1 INTRODUCTION

⋆ Motivation behind the work. ⋆ Some very light background information about hypervisors and their detection. ⋆ The choice of HyperDbg as the platform on which the implementation was carried out. ⋆ Overview of what this paper is about, and the goals set/achieved for the project

2 BACKGROUND

⋆ Theory behind virtualization and how hypervisors work.

⋆ Use of hypervisors in malware analysis.

⋆ Processes/malware can detect virtualizeation and change their behaviour

3 OVERVIEW

⋆ Describe the many ways how a hypervisor could be detected and how the hypervisor can(in theory) mitigate this.

⋆ Talk about which detection methods were chosen to be mitigated during this project and why. This might be more fitting in the design setion?

4 DESIGN AND IMPLEMENTATION

4.1 General Overview

General overview of the development/testing environment, how the design of the implementation was decided during the project

4.2 Implementation & Design

Talk about the chosen methods that were implemented

4.2.1 CPUID. CPUID mitigation design

4.2.2 Model Specific Registers. MSR access mitigation design

4.2.3 Windows system call interception. Implementation and the design of the approach chosen to intecept system calls from the kernel-mode

4.2.4 Modification of system call behaviour. The approach taken for modifying Windows system calls with user-mode callers, to hide hypervisor presence

4.3 Final design

Overview of the achieved implementation, talk about the tradeoffs of the design/advantages.

5 EVALUATION

How the implemented features of HyperDbg were tested. What tests were performed to measure the effectiveness of the hypervisor transparency Mention the testing environment.

5.1 Overhead Analysis

Analyze the overhead the transparency mode introduces,

5.2 Hypervisor Detection

Deploy the transparency mode against publicly mainained tools that contain common hypervisor and debugging detection methods. Compare to bare metal, VMWare and VMWare + HyperDbg The tools are VMAware, Al-Khaser, Pafish(Maybe more but only if there is a need) For specific implemented features, can deploy against self made POC that execute the mitigated detection methods.

5.3 Comparison with ScyllaHide

Compare the implemented kernel-level system call hooking approach to ScyllaHide's usermode level interception. Prove that this chosen approach is better since processes executing the system calls directly can bypass ScyllaHide but not the projects approach

5.4 Results

General compilation of the results/achieved goals from the research proposal

6 DISCUSSION

6.1 Design Limitations

Talk about the limitations of the tranparency implementation. How it could break genuine processes that expect certain data to be there or actions to execute, which the transparency features block.

6.2 Not Full Transparency

Discuss the fact that this paper only touches a few of the numerous hypervisor detection techniques and there are still many ways any process could easily detect it. Mention that some of the implemented features do still leave some footprints which could also be found.

7 RELATED WORK

Talk about the related work and papers to this project. Primarily other hypervisor detection evasion works and papers on malware detection/analysis using hypervisors

8 CONCLUSION

⋆ Talk about the achieved level of hypervisor transparency.

⋆ How well the goals of the research proposal were achieved.

⋆ I excluded the appendinx from this overview as it added not useful data, but it will be featured in the full paper