Safe Loader

Roey Gross, Yair Lasri



Instructor: Arie Haenel

2024 – 2nd Sem.  
Jerusalem College Of Technology (JCT)

Computer Science

Table of content:

[Abstract 2](#_Toc172228914)

[Introduction 2](#_Toc172228915)

# Abstract

Dynamically linked libraries (DLLs) are critical components that enable executables (EXEs) to extend their functionality by loading additional code at runtime. Sometimes a high-authorization EXE can load DLLs without adequately verifying their signatures. The EXE is assumed to be trusted, but the DLLs aren’t. An attacker can exploit this weakness by replacing legitimate DLLs with malicious ones, compromising the integrity of the signed EXE.

To counter this threat, we develop a robust framework for signing DLLs and ensuring that an EXE validates these signatures at load time, before any library code is executed.

Our solution involves a signer program and a validating program. The signer program signs the DLLs. The validating program uses IAT hooking to intercept the EXE's DLLs loading proccess, verify the DLL's signature, and load only trusted DLLs.

We will demonstrate that our approach ensures that even if an attacker can add or replace DLLs, only verified DLLs will be executed, maintaining the integrity and security of the system, and completing the chain of trust.

# Introduction

Imagine a scenario where a meticulously signed executable, trusted by its users and the operating system, unwittingly becomes a gateway for malicious actors. This threat unfolds when attackers exploit a critical vulnerability: the unchecked validation of dynamically linked libraries (DLLs) loaded by the executable.

In this typical case, the attacker gains access to the directory housing the signed executable. Here, they discreetly substitute genuine DLLs with their malicious counterparts, designed to execute unauthorized commands or steal sensitive information. Despite the executable being rigorously authenticated through Windows' signing mechanisms, these rogue DLLs evade detection, leveraging the executable's implicit trust to operate within the system undetected.

To further complicate matters, attackers can also exploit the system's search paths for DLLs. By placing a rogue DLL with the same name in a directory early on the search path, they ensure that their malicious DLL is loaded instead of the legitimate one intended for use by the executable. This stealthy maneuver exploits the executable's reliance on system search paths, effectively bypassing the intended security measures based on executable signing alone.

While users diligently verify the integrity of their executables, often utilizing User Account Control (UAC) to elevate privileges only for signed and validated executables, the oversight lies in the unchecked DLLs. These secondary components, essential for extending an executable's functionality, lack the same level of scrutiny, creating a vulnerable entry point into the system.

This project intervenes precisely at this juncture of vulnerability. By developing a comprehensive framework for DLL signing and validation, we aim to fortify executables against such malicious DLL manipulations. Through innovative approaches like DLL injection and IAT hooking, our solution ensures that only authenticated DLLs accompany trusted executables, safeguarding system integrity from DLL-based exploits.