

# Reverse Shell with Anti-Debugging and PEB Manipulation

By Reda Ouzidane

April 9, 2025

## Abstract

This document explains a C++ program that implements a reverse shell with an anti-debugging technique. The program checks for the presence of a debugger by manipulating the `BeingDebugged` flag in the Process Environment Block (PEB). If no debugger is detected, the program connects to a remote server and listens for commands to execute.

## 1 Introduction

In the context of cybersecurity, especially in malware development, it is crucial to evade detection by debuggers. Debuggers allow analysts to inspect and manipulate the program's execution, which can hinder the analysis of malicious software. One way to protect a program from being debugged is by checking the `BeingDebugged` flag in the Process Environment Block (PEB). If this flag is set, it indicates that a debugger is attached.

This document will walk you through a C++ program that checks for a debugger using the PEB and proceeds to run a reverse shell if no debugger is found.

## 2 Code Explanation

### 2.1 Anti-Debugging

The function `CustomError()` manipulates the `BeingDebugged` flag in the PEB. Here's how it works:

- `mov rax, fs:[0x60]`: Retrieves the address of the PEB.
- `movzx eax, byte ptr [rax + 2h]`: Checks the `BeingDebugged` flag.
- `jnz PATCH`: Jumps to the patch section if the flag is set (debugger detected).
- `mov byte ptr [rax + 2h], 0`: Sets the `BeingDebugged` flag to 0, which tricks the program into thinking no debugger is present.

### 2.2 Reverse Shell

The function `reverseShell()` creates a socket connection to a remote server. Once connected, it redirects the standard input and output to the socket and listens for commands. These commands are then executed using the `system()` function.

### 2.3 Main

In the main function, `CustomError()` is called first to check for a debugger. If no debugger is detected, the reverse shell is initiated with the target IP and port.

## 3 Conclusion

This program demonstrates how to create a simple reverse shell with anti-debugging capabilities. The anti-debugging technique manipulates the PEB to avoid detection by debuggers, while the reverse shell provides remote access to the compromised system.

This approach is commonly used in **malware** and **Red Team operations** to maintain persistence and evade analysis.