Lab 3: CFI and Shadow Stack

Objective

The objective of this lab is to implement simple CFI and shadow stack to directly harden binary code.

Challenge Programs

DARPA CGC challenge programs: https://github.com/hengyin/cb-multios/tree/master/challenges.

Binary Rewriting

We will use datalog disassembly: https://github.com/GrammaTech/ddisasm

You can directly use its docker image:

```
docker pull grammatech/ddisasm:latest
docker run -v "`pwd`":/shared -it grammatech/ddisasm bash
cd /shared ddisasm CADET_00001 --asm CADET.s
as CADET.s -o CADET.out
ld CADET.out -e _start -o CADET_00001_rewritten
```

Task 1: CFI (60%)

Implement a simple CFI policy to protect indirect function calls (e.g., call *RAX). A simple policy can be: an indirect call can jump to any function entry.

Task 2: Shadow Stack (40%)

Implement a simple shadow stack to protect return instructions. You can allocate a large buffer to store the shadow stack in the data section.

You need to include the following in your report:

1. Important code snippets and explanations of how you implement these two protections.
2. Pick at least two programs to show that a) the protected binary can process normal inputs correctly without crashing; and b) when you provide a malicious input that hijacks the control flow, the protected binary can prevent it.

Rubrics:

For each task: Functionality (40%), Explanation (40%), Evaluation (20%)