**CS5231 Project Support: Trace-based Exploit Detection**

# Introduction

The goal of CS5231 is to get your hands dirty and understand what is really going on under the hood. In class, you have seen how debuggers can be used to examine the state of a program during runtime. For this project, you are tasked to take things one step further and build tools to draw interesting insights based on the execution trace of a program.

This document helps you to setup or build tools to collect execution traces of a program.

# Collecting Execution Trace

There are numerous tools out there that can help you to get execution traces of a program. You can use existing instruction-tracing tools, or write your own tool.

# Method 1: Using the Peekaboo tool:

Here, we provide detailed instructions on how to obtain such execution traces using [*peekaboo*](https://github.com/melynx/peekaboo), a tool based on [DynamoRIO](https://dynamorio.org/).

First, you need to download DynamoRIO. To do so, you can go to the [DynamoRIO website](https://dynamorio.org/page_releases.html) and download the appropriate package based on your OS. We tested and verified DynamoRIO version **9.0.1**, but you are free to use other versions if you wish.

Using the **libpeekabo\_dr.so** file in the project folder, you can obtain the execution trace of a program called **my\_binary** with argument **args** using the following command:

|  |
| --- |
| **$ DynamoRIO-PATH/bin64/drrun -c libpeekaboo\_dr.so -- my\_binary args** |

The traces should be stored in a directory named **my\_binary-PID** in your current working directory. After generating the trace, you can use the **readtrace** program provided in the project folder to inspect the generated trace. You can see [*peekaboo*’s homepage](https://github.com/melynx/peekaboo) to learn how to interpret the generated trace.

|  |
| --- |
| **$ readtrace -h**  **$ readtrace -y my\_binary-PID/PID/ # print syscalls instructions** |

Note: You might need to install **libcapstone3** to use the **readtrace** program:

|  |
| --- |
| **$ sudo apt install libcapstone-dev # recommended**  **$ sudo apt install libcapstone3 # alternative 1 $ sudo apt install binutils-dev # alternative 2** |

# Method 2: Write your own tool

Alternatively, you can take things one step further and detect the vulnerabilities by analysing the runtime information as the program executes. To do so, you can develop your own client (similar to **libpeekabo\_dr.so**) for DynamoRIO. In the **samples** folder of DynamoRIO, you can find some sample clients written in C that use DynamoRIO API to achieve a specific task. After you compile them to a shared object (**.so**), you can run them alongside DynamoRIO by using the **–c** flag, similar to how you run *peekaboo*.

# Appendix A: Generating *peekaboo* binaries from source

If the provided **libpeekabo\_dr.so** and **readtrace** does not work for your OS, you can try to build the peekaboo project from its source.

|  |
| --- |
| **$ sudo apt install git cmake build-essentials libcapstone-dev**  **$ git clone** [**https://github.com/melynx/peekaboo.git**](https://github.com/melynx/peekaboo.git)  **# Generating C/C++ library**  **$ cd peekaboo/libpeekaboo**  **$ make**  **# Generating peekaboo shared object**  **$ cd peekaboo/peekaboo\_dr**  **$ mkdir build**  **$ cd build**  **$ DynamoRIO\_DIR=($DynamoRIO\_PATH) cmake ..**  **$ make**  **# Generating readtrace**  **$ cd peekaaboo/**  **$ make** |