Automated Reverse Engineering Project Proposal
Factors Influencing Gadget Discovery with ROPGadget
EEL4930
October 18, 2024

Shane Ferrell

Goal:

The purpose of this project is to examine the factors influencing an attacker's ability to discover return-oriented programming (henceforth referred to as ROP) gadgets in x86 executable code. This project investigates one of the main claims made in the 2007 paper by Shacham, which states that for any sufficiently large x86 binary, there will exist at least one chain of gadgets that enables arbitrary computation.

Tools:

- gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0

GCC (GNU Compiler Collection) is used to create the benchmark dataset of executable binaries in x86 architecture from various C++ programs.

- ROPGadget 7.5

ROPGadget is an open-source tool that automates the discovery of gadgets for ROP exploitation in executable binaries. ROPGadget will be used to identify gadgets in the benchmark dataset.

Dataset:

The dataset consists of 18 programs written in C++ that are categorized under 3 purposes: Arithmetic Operations, String Manipulation, and File I/O. For each purpose, the programs are categorized under 2 structures: Dense Loops and Sparse Function Calls. Each program has also been implemented with 3 code sizes: small, medium, and large. The source code for each program is in a file called main.cpp.

Each main.cpp program has been compiled with 4 levels of optimization using gcc: -O0, -O1, -O2, and -O3 and linked both statically and dynamically with a Bash script compile.sh, producing 144 executable binaries.

The 18 programs, 144 executable binaries that make up the dataset, and the script for compiling these executables can be found at the following repository: https://github.com/shaneferrellwv/gadget-discovery

Description:

This project consists of two main tasks: gadget discovery and analysis.

Gadget discovery will consist of creating a script that uses ROPGadget to discover ROP gadgets in the binary test executables. This script will output the number of gadgets discovered and the amount of time needed to complete the search for these gadgets.

Analysis will consist of performing a regression analysis for how the independent variables of:

1) program purpose (arithmetic operations, string manipulation, file I/O)
2) source code size
3) executable binary size
4) program structure (dense loops vs. sparse function calls)
5) compiler optimization level
6) linking (dynamic vs. static)

influence the dependent variables of:

1) number of gadgets discovered by ROPGadget
2) time to complete search of gadgets by ROPGadget

Challenges & Considerations:

The largest challenge identified in this project is the ability to quantify the independent variables (program purpose, program structure, compiler optimization level, and linking). Bias may exist in how the dataset was created and categorized, and I am continuing to keep this in consideration as the analysis stage begins.

Another consideration is the accuracy of ROPGadget to discover gadgets. ROPGadget is both frequently and recently updated with new features. I have no previous experience using ROPGadget for gadget discovery, and I am excited to learn more about how it works and additional features that can aid in my analysis.

References:

Shacham, H. (2007). *The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86)*. Proceedings of the 14th ACM Conference on Computer and Communications Security. https://doi.org/10.1145/1315245.1315313