SAND2022-9736D

The Center for Cyber Defenders
Expanding computer security knowledge

Shared Object Injection for Memory Usage Tracking

David Seo, Harvard University

Project Managers: Dan Chivers, Org 5638
Rachel Glockenmeier, Org 5634

Abstract

Our project aims to create a fast, automated utility to track memory allocations in an already-running process on ESXi. We wrote code that attaches to the process via ptrace and forces the process to link to a shared object that defines hooks for malloc and free. We found that our linking method is effective for 64 bit single-threaded processes, and it seems extensible to 32 bit processes as well. PLT (Procedure Linkage Table) hooking, which we explain later, is still in development. More testing is required to see if our design is compatible with multithreaded processes.

Introduction

ESXi is a closed source hypervisor developed by VMware that is often used for cloud computing applications. We seek to examine how processes behave on ESXi, including their memory usage. To that end, we would like to build a utility that can track calls to malloc and free in real time and report information about these calls, including allocation size and address.

Although ESXi emulates Linux in terms of its general filesystem layout and organization of processes by PIDs (Process IDs), it differs from Linux in several key aspects that make our task non-trivial. For example, ESXi does not support the proc filesystem, which makes process memory introspection much more difficult. Through this project, we wish to determine whether shared objection injection, a well-known method for redefining functions in already-running processes on Unix-like systems, is extensible to ESXi. We also wish to determine whether shared objection injection can successfully enable fine-grained, real-time memory usage tracking.

Methods

Our design has two high-level parts: injecting our shared object into the process and then retrieving the information that our installed hooks report.

The first part can be further split up into two parts: using ptrace to force the process to link the shared object, then modifying the process' PLT to force calls to malloc and free to redirect to our hooks in the shared object. To go into more detail, we can use ptrace with the PTRACE_ATTACH flag to attach to the currently-running process, then use ptrace with the PTRACE_SETREGSET and PTRACE_POKEDATA flags to set up the registers and code to call the function that we desire: dlopen.

We then parse the ELF (Executable and Linkable Format) header of the process' respective executable to locate the PLT.

At a high level, the PLT allows a program to jump to functions defined in external libraries, for which a hardcoded offset into the program's own text or data segment will not suffice. The PLT stores the dynamically resolved addresses of these functions in an entry that also includes the symbol corresponding to the function. We walk the PLT until we find the entries corresponding to malloc and free, and insert the addresses of our hook functions at the respective entries in the PLT. After this patch, calls to malloc and free will automatically be redirected to our hook functions via the PLT.

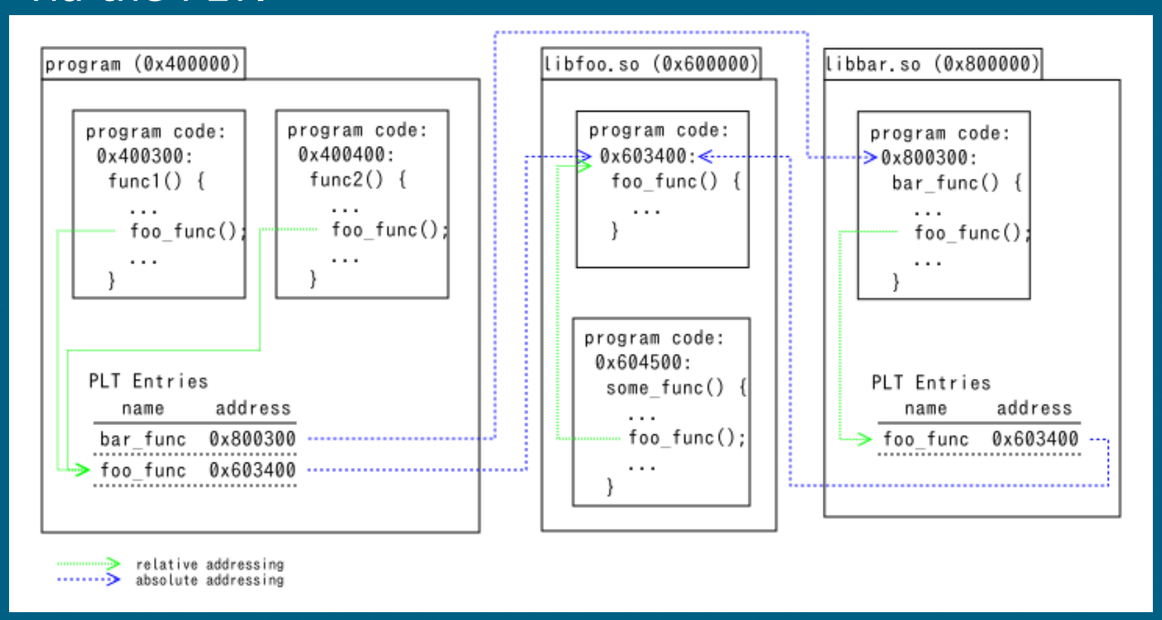

Figure: Flowchart of a function call and PLT read[1]

For now, we log information from the malloc and free hooks by printing to stderr, but in the future we hope to send UDP (User Datagram Protocol) datagrams containing the data to a local port and use a sniffer process to collect the information so that multiple processes can be tracked simultaneously.

Results

We were able to successfully attach to 64 bit processes in ESXi and inject the shellcode that links our shared object into the process. We would like to test our techniques on other kinds of processes, including multithreaded 64 bit processes as well as 32 bit processes, both of which will likely require some refactoring of our existing code.

The implementation of our PLT hooking technique is still in progress.

Discussion

From our results, our methods seem promising and applicable to ESXi. In the future we hope to successfully implement PLT hooking and test our tool on a variety of processes to identify edge cases where our methods do not work.

References

[1] https://github.com/kubo/plthook

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2017-XXXXC.

U.S. DEPARTMENT OF ENERGY
National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.