Parallel Algorithms Project Proposal

Blake De Garza
UTeid: bd6225
UT Austin

Dustin Kattner
UTeid: dtk427
UT Austin

September 15 2023

1 Abstract

working off the work done by Brumley and et.al. from their paper in *"AEG: Automated Exploit Generation"*[1]we will set out to prove an error (exploit) state predicate is First Lattice Linear, and if so then we can apply a Linear Lattice Predicate (LLP) algorithm in order to detect the exploit predicate of a buffer overflow as defined in *Algorithm 1: Return-to-Stack-Exploit Predicate Generation Algorithm* [1, pg. 11] .

2 Background

Using QEMU to emulate a programs execution, adding in introspection to the programs state through open source tools such as GNU Debugger(GDB). We will aim to identify in a multiprocesses program, at least two or more processes, in an open source program. We intend to inject a predict such as a simple Return-To-Stack Buffer Overflow in 3 varying levels of global states (entry, middle, and end of process execution) that will allow us to search for the error (exploit) state of a Program Under Test (PUT).

Motivations for this project are to develop a systematic tool that can aid researchers to verify that 1) where a potential error (exploit) state occurs and 2) the code segment relative to source that must be vetted by a developer to fix that specific error state. Furthermore, formally we will set out to prove that a predicate is lattice linear. Once proven it's lattice linear we will move forward

Algorithm 1 Return-To-Stack-Exploit Predicate Generation excerpt from Brumley et.al.

Input$(bufaddr, \&retaddr, \mu) = R$

Output$\prod_{exploit}$

for $i = 1, len(\mu)$ **do**
  $exp + str[i] = \mu[i]$
end for
$offset = \&retaddr - bufaddr;$
$jmpTarget = offset + 8$
$expStr[offset] = jmpTarget$ ▷ $//EIPHijack$
for $i = 1, tolen(shellcode)$ **do** $expStr[offset + i] = shellcode[i];$
  return$Mem[bufaddr] == expStr[1] \wedge ... \wedge$
end for
$Mem[buffaddr + len(\mu) - 1] == expStr[len(\mu)]);$ ▷ $// \prod_{exploit}$

to a predicate search algorithm and implement a Linear Lattice Predicate (LLP) search algorithm.

References

[1] Brumley and et.al. (2010) *AEG: Automated Exploit Generation*, Proceedings of the Network and Distributed Security Symposium.