ARCUS: Symbolic Root Cause Analysis of Exploits in Production Systems

Carter Yagemann, Matthew Pruett, Simon Chung, Kennon Bittick, Brendan Saltaformaggio, Wenke Lee
Detecting Exploits

```
void handle(void *cb)
{
    char buf[256];
    int r=1;
    while r
        r=read(stdin,buf,1);
    cb(buf);
}
```
Why Symptoms?

- Easiest to detect
  - Manifestation of behavior
- But how do we fix it?
  - Input filters
  - Function hardening
- Brittle, expensive
void handle(void *cb)
{
    char buf[256];
    int r=1;
    while r
        r = read(stdin, buf, 1);
    cb(buf);
}
Real-World Cases Are Harder

Average Distance: 11,722 basic blocks
Patching Postmortem is Hard

What data is there?

- Crash dumps? Corruptible
- System logs? Symptoms Only
- Concrete inputs? Privacy, Reproducibility

Developers ignore bugs they don’t understand
Our Solution: ARCUS

Analyzing Root Cause Using Symbex

Use “What-If” questions to test the impact of particular inputs on the satisfiability of security violations
ARCUS Pipeline

- Buffer Overflow
- Integer Overflow
- Use-After-Free
- Double Free
- Format String

Vulnerability-class-specific Satisfiability Tests

- Location
- Preliminary Patch

User Program

Control Flow Trace

Intermediate Symbolic State

Sat? Root Cause Report

Dev

How?
“Hardware is the New Software”

- Architecture Events
- Cache Events
- Power Events
- Control Flow Events

Tracing Facilities

Modern CPU

Complete
~7% overhead
$ ./ntpq -4 ['python -c 'print "A" * 300'`]  
Name or service not known  
*** stack smashing detected ***: <unknown> terminated

I’m going to use source code for clarity, ARCUS works directly on binaries
The State Explosion Problem

1. int openhost(const char *hname, ...) {
2.  char *cp;
3.  char name[256];
4.  
5.  cp = hname;
6.  if (*cp == '[') {
7.    cp++;
8.    for (i = 0; *cp && *cp != ']'; cp++, i++)
9.      name[i] = *cp;
10.   if (*cp == ']') {
11.      name[i] = '\0';
12.      hname = name;
13.    } else return 0;
14.   /* [... ] */
}

Symbols

hname := [s1,s2,...]
name := [ ]
cp := {}
ret_ptr := {c1}

How many times should Line 9 iterate?

11
Solution: Control Flow Trace

1. int openhost(const char *hname, ...) {
2.   char *cp;
3.   char name[256];
4.   
5.   cp = hname;
6.   if (*cp == '[') {
7.     cp++;
8.     for (i = 0; *cp && *cp != ']'; cp++, i++)
9.       name[i] = *cp;
10.    if (*cp == ']') {
11.      name[i] = '\0';
12.      hname = name;
13.    } else return 0;
14.   }/* [...] */
Localizing Root Cause

Intermediate Symbolic States

What if we didn’t corrupt the pointer?

What if we didn’t corrupt the pointer?

exit loop
What’s Different?

Preliminary Patch:

```c
for (i=0; *cp && *cp != ']' && index(']', hname) <= 257; cp++, i++)
```
Real-World Evaluation

- Tested **27** exploits for **20** real Linux programs
- **100%** detection rate
- **100%** consistency between proposed patch and official patch (where available)
- **4** new 0-days found!
  - **3** CVEs issued
  - Patched by developers using ARCUS’ reports
See Paper For:

- Additional vulnerability classes
- How ARCUS interfaces with Intel PT
- Interesting case studies
- Additional experiments
Thank You!

Carter Yagemann
yagemann@gatech.edu
carteryagemann.com

Code & Data: github.com/carter-yagemann/arcus

Co-Authors: Matthew Pruett, Simon Chung, Kennon Bittick, Brendan Saltaformaggio, Wenke Lee