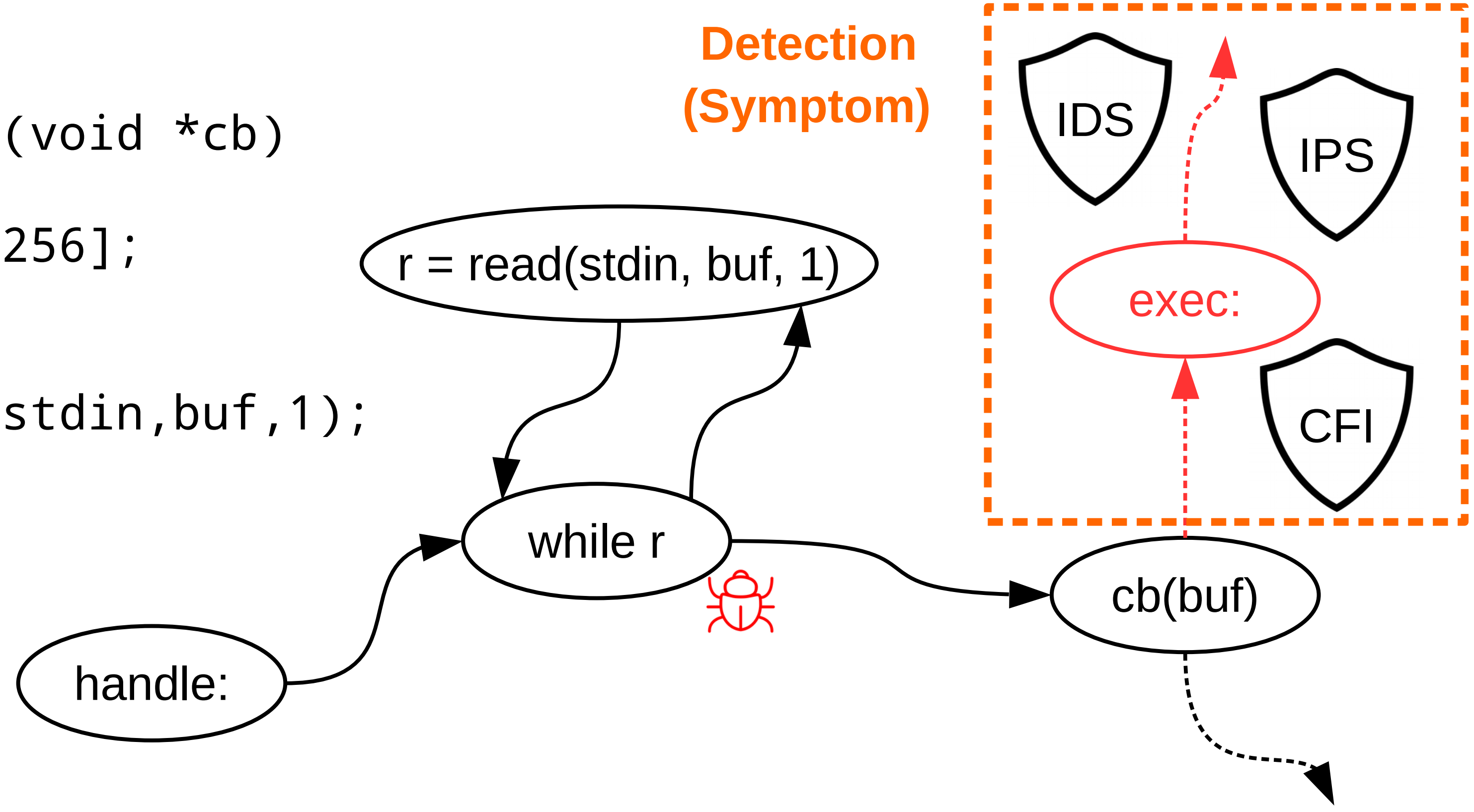
ARCUS: Symbolic Root Cause Analysis of Exploits in Production Systems

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

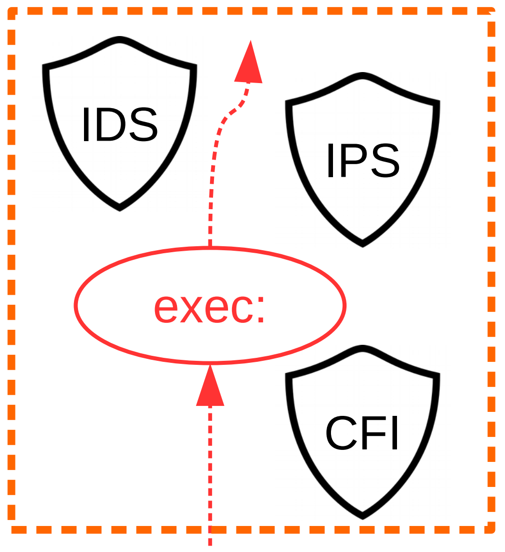
Detecting Exploits

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

User Program



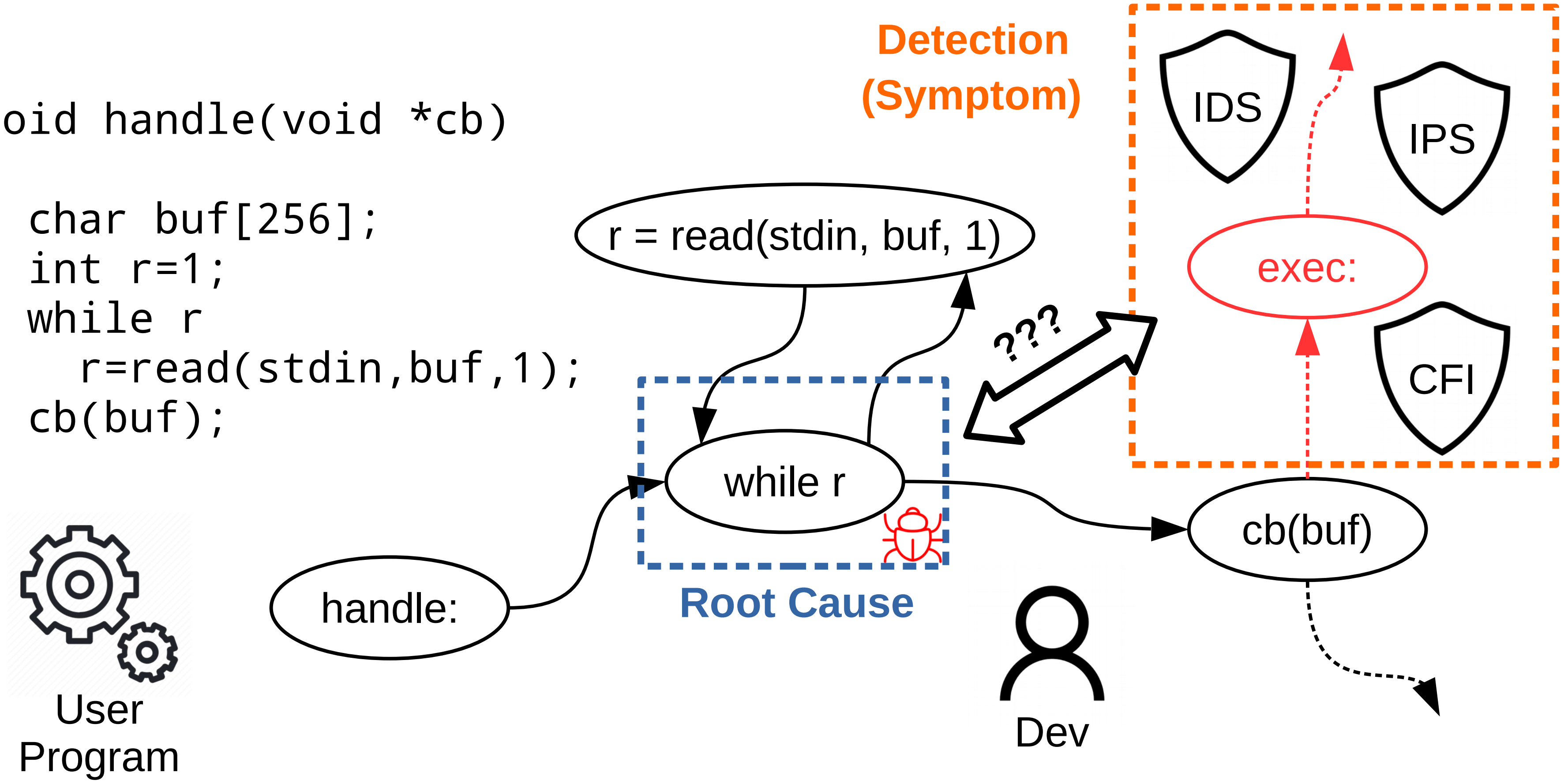
Why Symptoms?



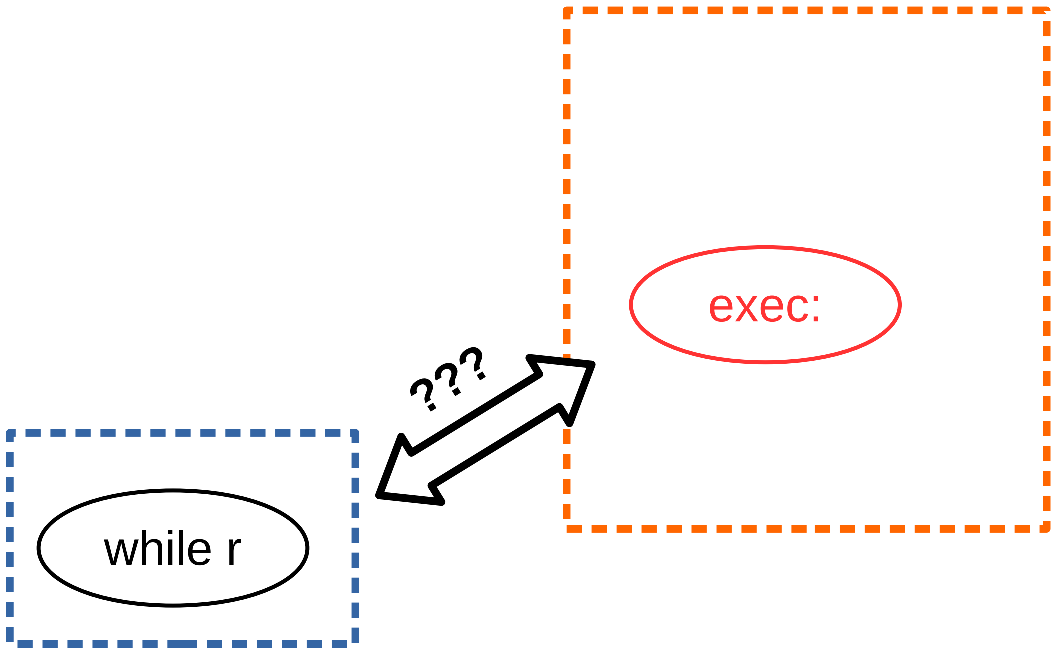
- Easiest to detect
  - Manifestation of behavior
- But how do we fix it?
  - Input filters
  - Function hardening
- Brittle, expensive

Ideal Fix: Developer Patch!

```
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: <u>**11,722**</u> basic blocks

Patching Postmortem is Hard

What data is there?

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

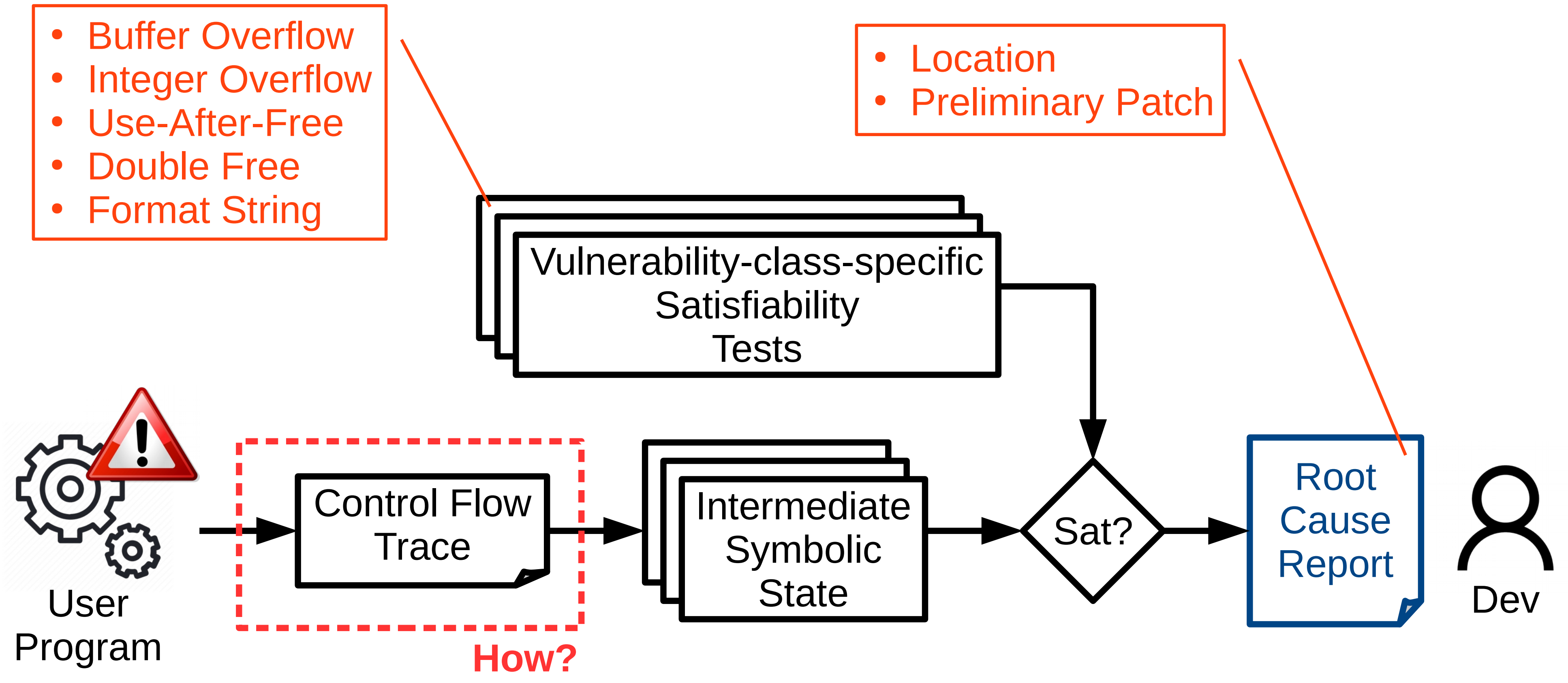
$$\boxed{\text{Developers ignore bugs they don't understand}}$$

Our Solution: ARCUS

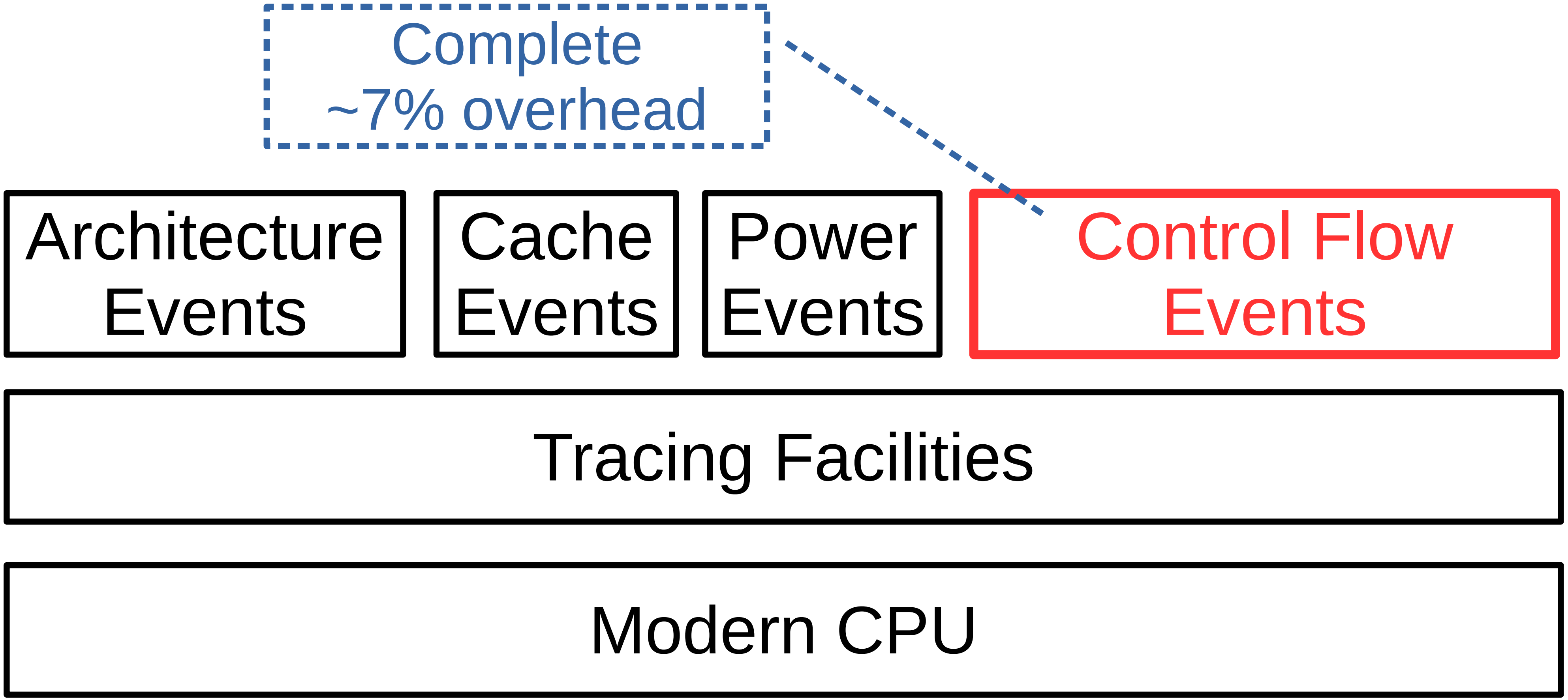
Analyzing **R**oot **C**ause **U**sing **S**ymbex

Use "What-If" questions to test the impact of particular inputs on the satisfiability of security violations

ARCUS Pipeline



“Hardware is the New Software”



Example: CVE-2018-12327

```
$ ./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?

Solution: Control Flow Trace

```
int openhost(const char *hname, ...) {
 char *cp;
 char name[256];

 cp = hname;
 if (*cp == '[') {
  cp++;
  for (i = 0; *cp && *cp != ']'; cp++, i++)
   name[i] = *cp;
  if (*cp == ']') {
   name[i] = '\0';
   hname = name;
  } else return 0;
  /* [...] */
```

Trace

Taken

Taken x312

Symbols

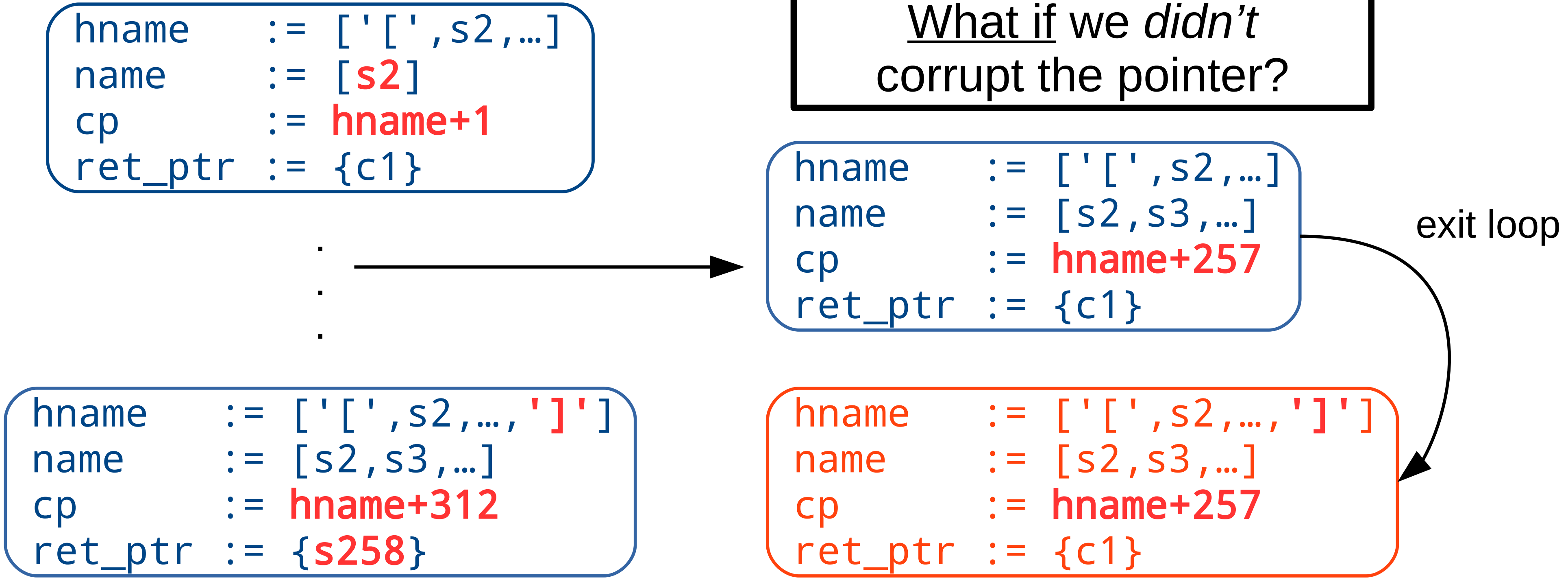
```
hname    := ['[',s2,…,']']
name     := [s2,s3,…]
cp       := hname+312
ret_ptr  := {s258}
```

Corrupted return pointer!

Localizing Root Cause

Intermediate Symbolic States

What if we *didn't* corrupt the pointer?



```
hname   := ['[',s2,…]
name    := [s2]
cp      := hname+1
ret_ptr := {c1}
```

```
hname   := ['[',s2,…,']']
name    := [s2,s3,…]
cp      := hname+312
ret_ptr := {s258}
```

```
hname   := ['[',s2,…]
name    := [s2,s3,…]
cp      := hname+257
ret_ptr := {c1}
```

```
hname   := ['[',s2,…,']']
name    := [s2,s3,…]
cp      := hname+257
ret_ptr := {c1}
```

What’s Different?

```
hname    := ['[',s2,…,']']
name     := [s2,s3,…]
cp       := hname+312
ret_ptr  := {s258}
```

Δ

```
hname    := ['[',s2,…,']']
name     := [s2,s3,…]
cp       := hname+257
ret_ptr  := {c1}
```

$$\texttt{hname[257] != ‘]’} \xleftrightarrow{\text{contradiction}} \texttt{hname[257] == ‘]’}$$

Preliminary Patch:

```
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