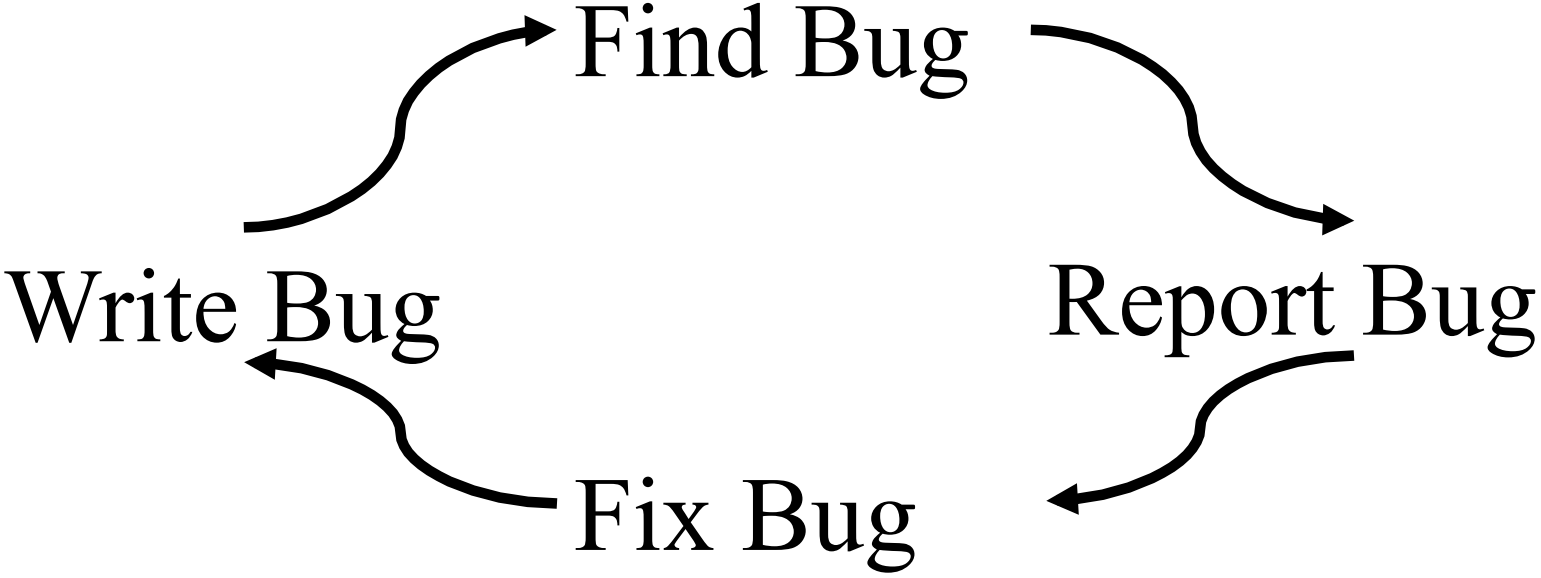
Dynamic Test Generation To Find Integer Bugs in x86 Binary Linux Programs

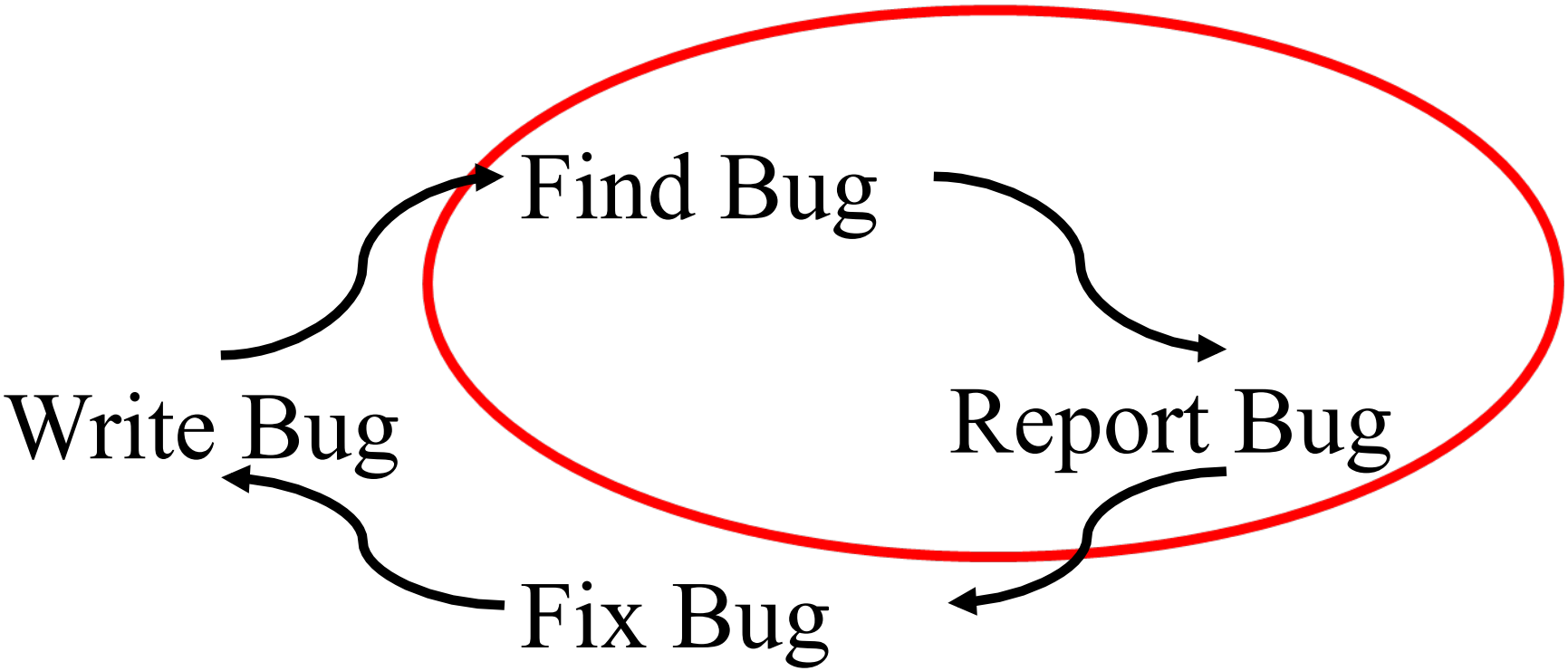
David Molnar
Xue Cong Li
David Wagner

Security Bugs Common

- 6,515 vulnerabilities reported in 2007
  - Major vendors : Adobe, Apple, Microsoft …
  - Plus many more
  - web.nvd.nist.gov/view/vuln/statistics
- Each one means patch, QA'ing, releasing

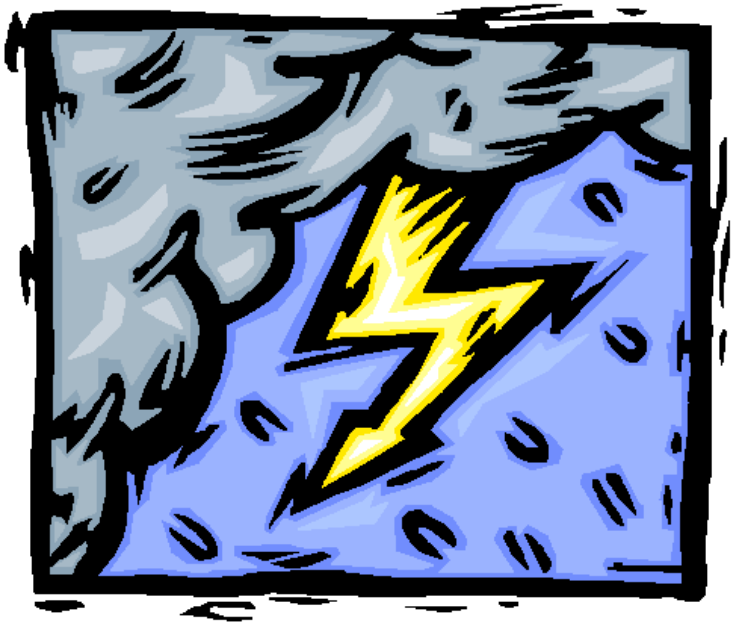


The “Bug Cycle”



The “Bug Cycle”

Technique : Fuzz Testing



Miller, Fredriksen, and So,
"An Empirical Study of the Reliability of UNIX Utilities"
http://pages.cs.wisc.edu/~bart/fuzz/fuzz.html

Integer Bugs

- #2 cause of vendor advisories in 2006
- Underflow/Overflow
- Value conversions
- Signed/Unsigned conversion bugs
- Poor fit with traditional runtime, static analysis
  - Static analysis: false positives
  - Runtime analysis: “benign” overflow problem

Signed/Unsigned Conversion

```
void bad(int x,char * src,char * dst)¶
{
     if (x > 800)¶
     {
          return;
     }
     else
     {
          copy_bytes(x,src, dst);
     }
}
```

Signed/Unsigned Conversion

```
void bad(int x,char * src,char * dst)¶
{
     if (x > 800)¶
     {
          return;
     }
     else
     {
          copy_bytes(x,src, dst);
     }
}
```

What if x == -1 ?

Signed/Unsigned Conversion

```
void bad(int x,char * src,char * dst)
{
     if (x > 800)
     {
          return;
     }
     else
     {
          copy_bytes(x,src, dst);
     }
}
```

-1 > 800? **No!**

Signed/Unsigned Conversion

```
void bad(int x,char * src,char * dst)¶
{
     if (x > 800)¶
     {
          return;
     }
     else
     {
          copy_bytes(x,src, dst);
     }
}
```

copy_bytes(**unsigned int** x,...

Signed/Unsigned Conversion

```
void bad(int x,char * src,char * dst)¶
{
     if (x > 800)¶
     {
          return;
     }
     else
     {
          copy_bytes(x,src, dst);
     }
}
```

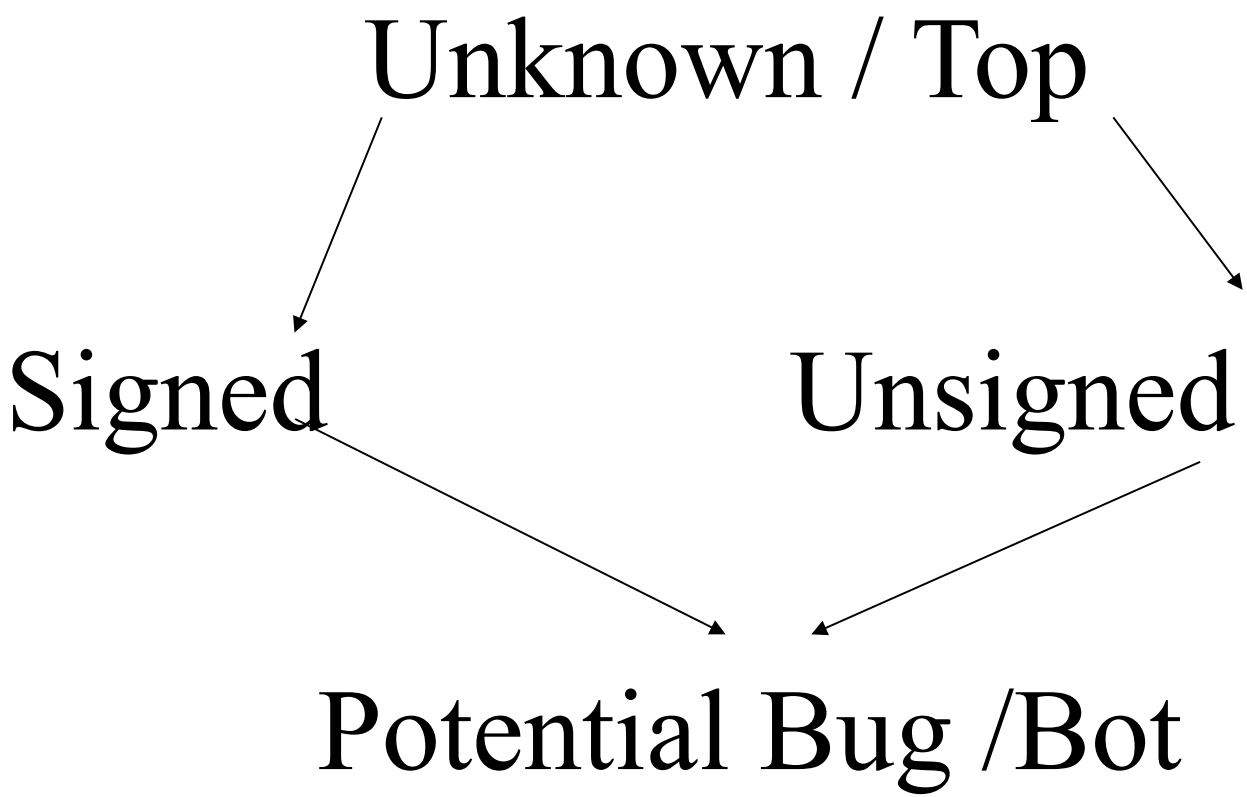
copy_bytes(**unsigned int** x,...

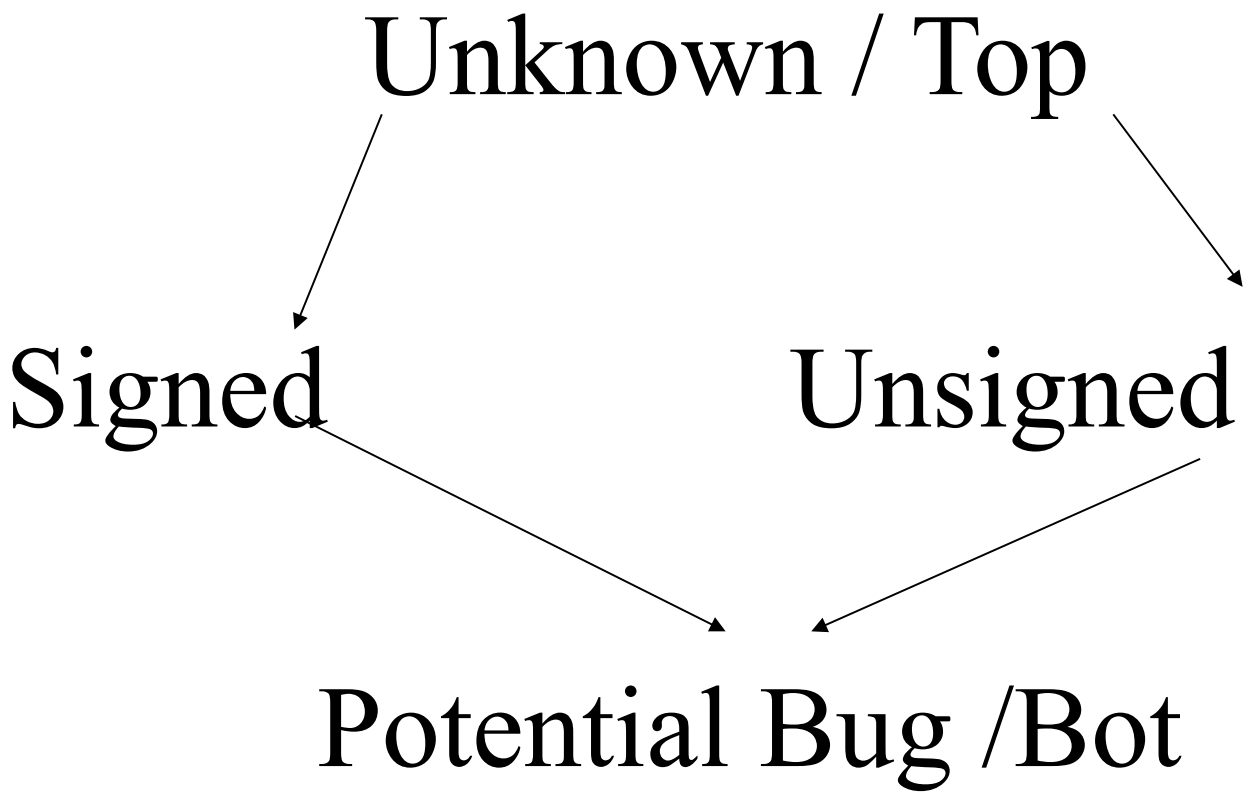
Copy a few more than 800 bytes..!

Signed/Unsigned Conversion

```
void bad(int x,char * src,char * dst)¶
{
    if (x > 800)¶
    {
        return;
    }
    else
    {
        copy_bytes(x,src, dst);
    }
}
```

Bug pattern: treat value x as signed, then as unsigned or vice versa


Unknown / Top
Signed
Unsigned
Potential Bug /Bot



Idea:

1.Keep track of type for every tainted program value

2.Use solver to **force values with type "Bot" to equal -1**

New algorithm: **infer types over long binary traces.**