Exploring Multiple Execution Paths for Malware Analysis

Andreas Moser, Christopher Kruegel, and Engin Kirda
Secure Systems Lab
Technical University Vienna

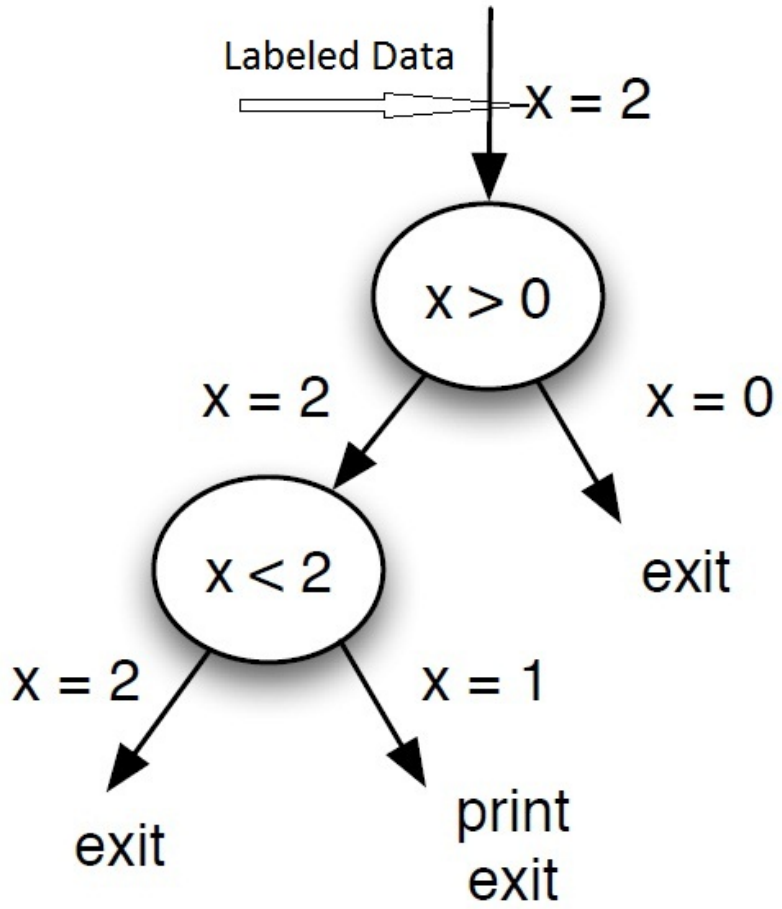
Introduction

- Malware Analysis
  - Execute unknown sample of malware in a restricted environment and observe it actions.
- Manual analysis is pain and the reason is obvious.
- Automating analysis
  - CWSandbox , Norman SandBox , TTAnalyze, Cobra etc
  - What does all these systems do in general?
    - record interaction with OS like recording system calls along with params used.

Problem

- Analysis is based on a single execution trace only
- Potential to miss a significant fraction of the behavior that a program might exhibit under varying circumstances.
  - Michelangelo virus, which remains dormant most of the time, delivering its payload only on March 6 (which is Michelangelo's birthday)
- Basic idea is to explore multiple execution paths of a program under test.
  - Depends on how the code uses inputs like current time from OS, content of a file etc

```
0:  int x;
1:  x = read_input();
2:  if (x > 0)
3:      if (x < 2)
4:          printf("ok");
5:  exit(0);
```



Exploration of multiple execution paths

Tracking Input

- Taint sources - used to assign labels to certain memory locations of interest
  - Vigilante, a taint-based system that can detect computer worms that propagate over the network. In this system, the network is considered a taint source. Each byte read from this n/w card gets new label
- Shadow Memory - Map memory to labels
- Inverse mapping - Map labels to memory

```
0:   ...
1:   x = read_input();
2:   check(x);
3:   printf("%d", x);
4:   ....
5:
6:   void check(int magic) {
7:       if (magic != 0x1508)
8:           exit(1);
9:   }
```

consistent memory update

What if the labeled data is an operand?

```
0:  char str[], *p;
1:  int sum;
2:
3:  p = str;
4:  sum = 0;
5:  while (*p >= '0' && *p <= '9') {
6:      sum = sum * 10;
7:      int c = *p - '0';
8:      sum = sum + c;
9:      p++;
10:  }
11:
12: if (sum == 82)
13:     printf("ok");
```

Code fragment

Mapping:	Constraints:
str[0] <--> l_0	
str[1] <--> l_1	

Initial state

Mapping:	Constraints:
str[0] <--> l_0	$l_2 = l_0 - 30$
str[1] <--> l_1	$l_3 = l_2$
c <--> l_2	
sum <--> l_3	

State after first loop iteration

Mapping:	Constraints:
str[0] <--> l_0	$l_2 = l_0 - 30$
str[1] <--> l_1	$l_3 = l_2$
c <--> l_5	$l_4 = 10 * l_3$
sum <--> l_6	$l_5 = l_1 - 30$
	$l_6 = l_4 + l_5$

State after second loop iteration

Saving and Restoring state

- Complete virtual address space
- Mappings
- Constraint system
- *Path constraint - conditional operation enforces a constraint on possible range of labeled arguement*

Evaluation

308 ----------> 229 ----------> 172

- 308 malware samples
- 229 samples use at least one tainted source
- 172 malware samples use these tainted bytes for control flow decisions

Tainted input sources

Interesting input sources	
Check for Internet connectivity	20
Check for mutex object	116
Check for existence of files	79
Check for existence of registry entry	74
Read current time	134
Read from file	106
Read from network	134

Relative increase in code coverage

Relative increase	Number of samples
0 % - 10 %	21
10 % - 50 %	71
50 % - 200 %	37
> 200 %	43

Lessons learnt ...

- Blaster launches DoS attack after Aug 15th
- Kriz virus first checks for a file KRIZED.TT6 in the system folder.
- rxBot - IRC based bot

Questions?