第 6 讲：The Programming Languages of OS

第三节：A Study of Bugs on Linux

陈渝

清华大学计算机系

yuchen@tsinghua.edu.cn

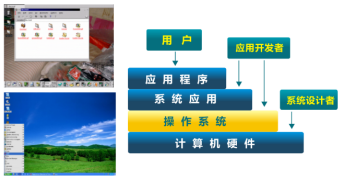
2020 年 3 月 22 日

Introduction

Why study faults/bugs/Vulnerabilities in OS code?

- Find the relations between Language and OS

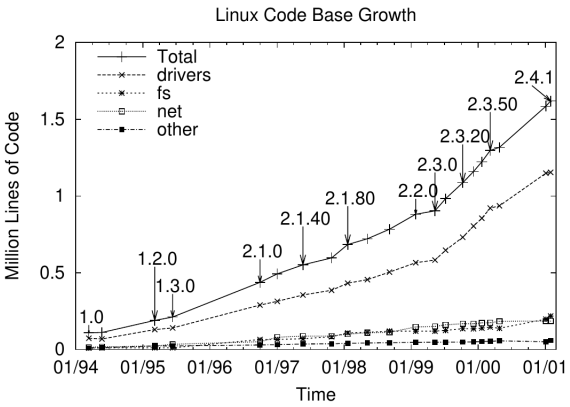
Introduction



Some questions

- Where are the errors?
- How are bugs distributed?
- How long do bugs live?

Introduction – 19 years ago



19 years ago

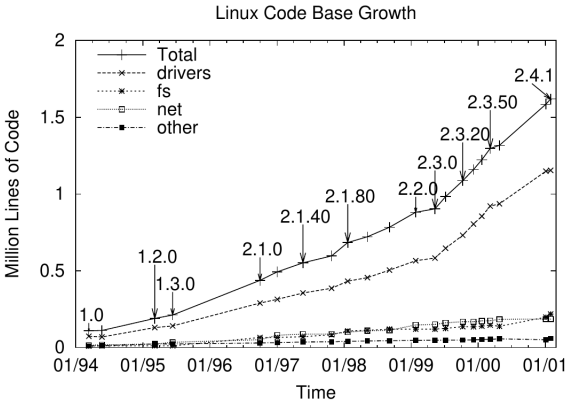
- In SOSP'01, Chou et al. studied faults (errors/bugs) in Linux code.
- Faults collected using static analysis.
- Faults collected in Linux 1.0 (1994) to 2.4.1 (2001). Primarily "development" versions. x86 code.

Introduction – 19 years ago

The twelve bug checkers

Check	Nbugs	Rule checked
`Block`	206 + 87	To avoid deadlock, do not call blocking functions with interrupts disabled or a spinlock held.
`Null`	124 + 267	Check potentially NULL pointers returned from routines.
`Var`	33 + 69	Do not allocate large stack variables ($> 1K$) on the fixed-size kernel stack.
`Inull`	69	Do not make inconsistent assumptions about whether a pointer is `NULL`.
`Range`	54	Always check bounds of array indices and loop bounds derived from user data.
`Lock`	26	Release acquired locks; do not double-acquire locks.
`Intr`	27	Restore disabled interrupts.
`Free`	17	Do not use freed memory.
`Float`	10 + 15	Do not use floating point in the kernel.
`Real`	10 + 1	Do not leak memory by updating pointers with potentially `NULL` realloc return values.
`Param`	7	Do not dereference user pointers.
`Size`	3	Allocate enough memory to hold the type for which you are allocating.

Introduction – 19 years ago



Often condensed to the most important finding: "Drivers are the one major source of bugs in operating systems" , which becomes the scientific fundament for a huge body of OS research:

- Mike Swift: Nooks, Microdrivers , Carbon
- Tanenbaum: Minix 3
- UNSW: Dingo + Termite
- Gun Sirer: Reference Validation
- Microsoft: Singularity + Signed Drivers

Introduction – 19 years ago

deadlock/data race related bug

To avoid deadlock, do not call blocking functions with interrupts disabled or a spinlock held.

```
// A) Call schedule() with interrupts disabled
asm volatile ("cli");
schedule();
asm volatile ("sti");

// B) Call blocking function with lock held
//     (BlockLock)
DEFINE_SPINLOCK(l);
unsigned long flags;
spin_lock_irqsave(&l, flags);
..
void *foo = kmalloc(some_size, GFP_KERNEL);
```

Introduction – 19 years ago

NULL/Free related bug

Check potentially NULL pointers returned from routines.

```
my_data_struct *foo =
     kmalloc(10 * sizeof(*foo), GFP_KERNEL);
foo->some_element = 23;
```

Do not use freed memory

```
free(foo);
foo->some_element = 23;
```

Introduction – 19 years ago

Var related bug

Do not allocate large stack variables (>1K) on the fixed-size kernel stack.

```
void some_function()
{
    char array[1 << 12];
    char array2[MY_MACRO(x,y)]; // not found
    ...
}
```

Introduction – 19 years ago

Range related bug

Always check bounds of array indices and loop bounds derived from user data.

```
int index = -1;
int n = copy_from_user(&index, userptr,
                       sizeof(index));
if (!n) {
  kernel_data[index] = 0x0815;
}
```

Introduction – 9 years ago

Lines of Code in Linux

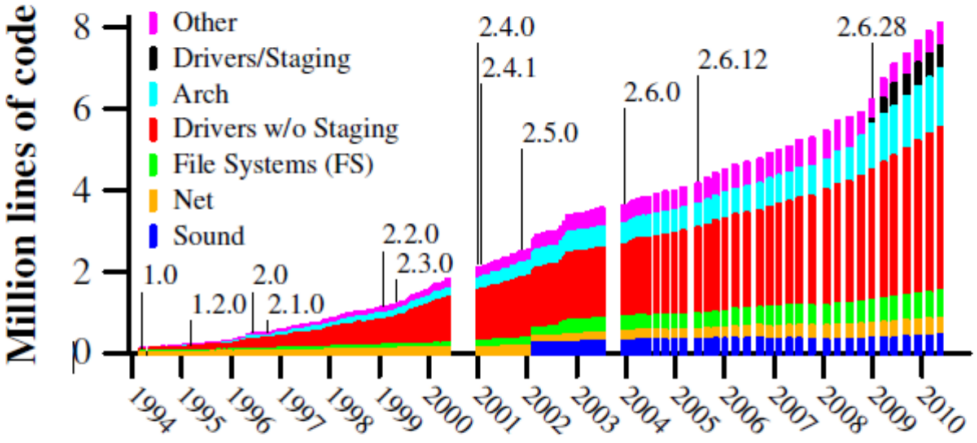
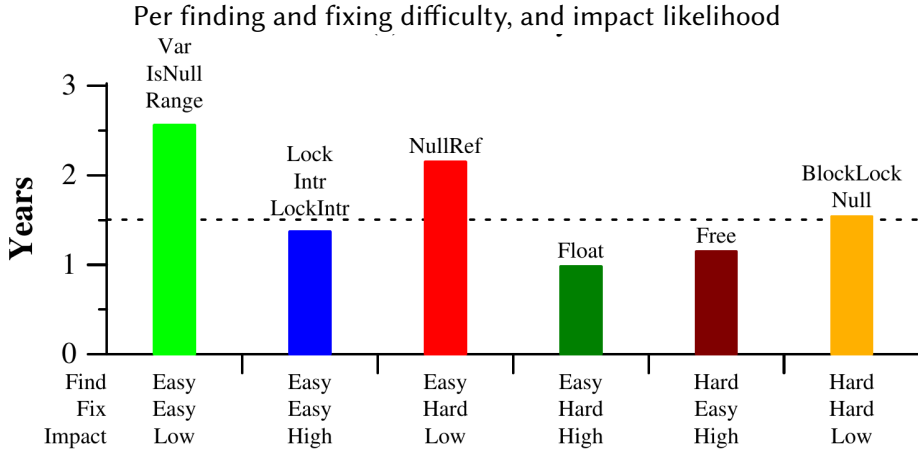


Figure 1. Linux directory sizes (in MLOC)

Introduction – 9 years ago – Comparative fault count

Checker	Chou *et al.* checked	Chou *et al.* unchecked	Our results
Block	206	87	N/A
BlockLock	N/A	N/A	43
Null	124	267	98
Var	33	69	13
Inull	69	0	N/A
IsNull	N/A	N/A	36
NullRef	N/A	N/A	221
Range	54	0	11
Lock	26	0	5
Intr	27	0	2
LockIntr	N/A	N/A	6
Free	17	0	21
Float	10	15	8
Size	3	0	3
Total	569	438	467

Introduction – 9 years ago



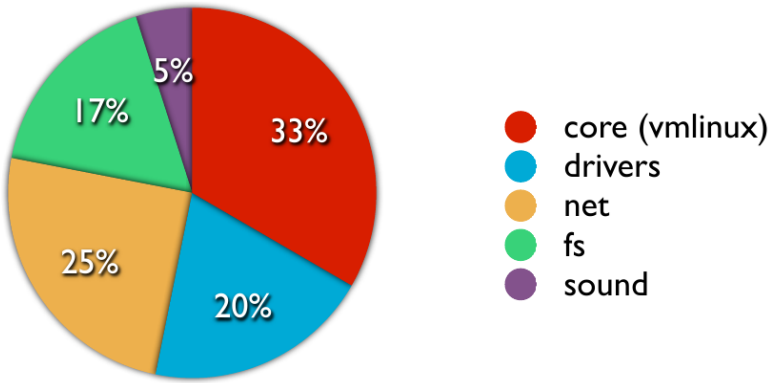
Introduction – 9 years ago – Linux kernel vulnerabilities

141 vulnerabilities from CVE (Jan 2010 Mar 2011)

Vulnerability	Mem. corruption	Policy violation	DoS	Info. disclosure	Misc.
Missing pointer check	6	0	1	2	0
Missing permission check	0	15	3	0	1
Buffer overflow	13	1	1	2	0
Integer overflow	12	0	5	3	0
Uninitialized data	0	0	1	28	0
Null dereference	0	0	20	0	0
Divide by zero	0	0	4	0	0
Infinite loop	0	0	3	0	0
Data race / deadlock	1	0	7	0	0
Memory mismanagement	0	0	10	0	0
Miscellaneous	0	0	5	2	1
Total	32	16	60	37	2

Introduction – 9 years ago – Linux kernel vulnerabilities

Distribution of 141 Vulnerabilities(Jan 2010 Mar 2011)



- Vulnerabilities are evenly distributed in all parts of kernel

Introduction – 9 years ago – Linux kernel vulnerabilities

Number of vulnerabilities that existing runtime tools can prevent

Vulnerability	BGI	SecVisor	SUD	Raksha	kmemcheck	SD
Missing pointer check	0	0	3	0	0	0
Missing permission check	0	0	1	0	0	0
Buffer overflow	1	0	1	0	0	0
Integer overflow (D)	0	0	1	0	0	0
Integer overflow (I)	0	0	1	0	0	0
Integer overflow (E)	0	0	3	0	0	0
Uninitialized data	0	0	13	0	1	23
Null dereference	11	0	3	0	0	0
Divide by zero	2	0	0	0	0	0
Infinite loop	0	0	1	0	0	0
Data race / deadlock	0	0	1	0	0	0
Memory mismanagement	1	0	1	0	0	0
Miscellaneous	0	0	0	0	0	0

- A technique that targets at only one of them is of limited use

References

- An empirical study of operating systems errors, SOSP 2003
- Linux kernel vulnerabilities: State-of-the-art defenses and open problems,APSYS 2011
- Faults in linux: ten years later,ASPLOS 2011