Linux kernel vulnerabilities:
State-of-the-art defenses and open problems

Haogang Chen, Yandong Mao, Xi Wang, Dong Zhou*
Nickolai Zeldovich and M. Frans Kaashoek

MIT CSAIL
*Tsinghua University

Why Study Kernel Bugs?

- An OS kernel is the TCB for many systems
- Despite much research, still many bugs...
- Any open research problems?

Contribution

- Classification of kernel bugs
  - 141 vulnerabilities from CVE (Jan 2010 ~ Mar 2011)
- Studies of 9 existing tools
  - They only address a small subset of vulnerabilities
- Findings
  - Semantic and DoS vulnerabilities are open problems

Vulnerability, Exploit & Impact

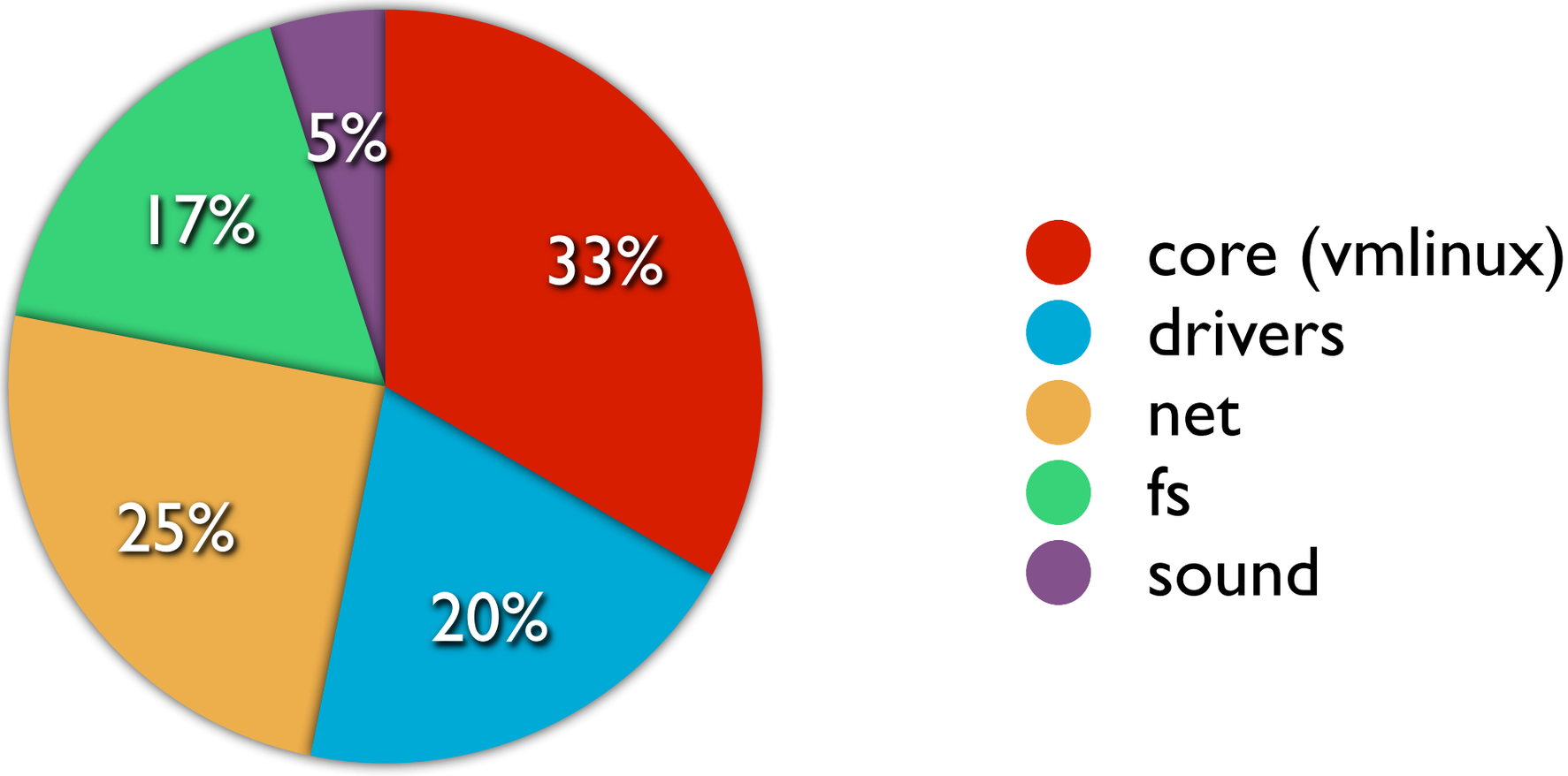
- Vulnerabilities
  - Programming mistake the developers made
- Exploits
  - An attack that takes advantage of the vulnerability
- Impacts
  - Direct consequence of an exploit

Vulnerabilities vs. Impacts

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

- Buffer and integer overflows are the top threat to the kernel's integrity
- Denial-of-service is the most common type of impacts
- Many bugs lead to the violation of high-level security policies

Distribution of Vulnerabilities



- Vulnerabilities are evenly distributed in all parts of kernel
- A technique that targets at only one of them is of limited use

Effectiveness of Existing Tools

Tools	Vulnerabilities Prevented	Limitations
BGI	15/14	• only for loadable modules • only attacks that cross the boundary
SUD Nooks	28/141	• only for device drivers
SecVisor	0	• only code injection attacks
Raksha	0	• only pointer injection attacks • require special hardware
Kmemcheck, Secure-dealloc	24/141	• only kernel heap or stack • no guarantee to detect all
Smatch, Saturn	?	• precise analysis is hard to scale

- Solutions that focus on one part of the kernel are insufficient
- Tools that focus on a certain class of exploits are of limited use

Open Research Problems

- Semantic vulnerabilities
- Denial-of-service attacks

Semantic Vulnerabilities

- Violation of high-level security policies
  - e.g. change file permissions without being the owner
  - Can easily be exploited to gain privilege

```
1  --- a/fs/btrfs/acl.c
2  +++ b/fs/btrfs/acl.c
3  @@ -160,3 +160,6 @@ static int btrfs_xattr_acl_set(...
4          int ret;
5          struct posix_acl *acl = NULL;
6
7  +       if (!is_owner_or_cap(dentry->d_inode))
8  +               return -EPERM;
9  +
```

```
10
11 --- a/fs/gfs2/file.c
12 +++ b/fs/gfs2/file.c
13 @@ -218,6 +218,11 @@ static int do_gfs2_set_flags(...
14 +       error = -EACCES;
15 +       if (!is_owner_or_cap(inode))
16 +               goto out;
17 +
18 +       error = 0;
```

Figure: Patch for CVE-2010-2071 (btrfs) and CVE-2010-1641 (gfs2)
(Setting extended attributes for arbitrary files)

Semantic Vulnerabilities

- Other semantic bugs in filesystems
  - CVE-2010-2537 (btrfs): overwrite append-only files
  - CVE-2010-1636 (btrfs): read write-only files
  - CVE-2009-4131 (ext4): overwrite arbitrary files
  - CVE-2010-2066 (ext4): overwrite append-only files
  - CVE-2010-2226 (xfs): read write-only files
  - CVE-2010-1146 (reiserfs): expose fs-private metadata
- Others exist in network protocols as well

Challenge & Possible Solutions

- Challenge
  - High-level policies are implicit (unlike memory safety)
  - Policy do not map to kernel interfaces (e.g. append)
- Possible solutions
  - Annotate the VFS layer, and check all implementations
  - Infer policies from the majority and detect bugs as deviant behaviors [Engler01]
  - Separate privilege using users or applications as principals, instead of kernel modules [Zeldovich08]

Denial-of-service Attacks

- Cause the kernel to kill a running process, or make the kernel hang

- Almost every vulnerability can lead to DoS

DoS Attacks Matter

- DoS vulnerabilities might be exploitable in different ways
  - E.g., the *econet* exploit uses a null pointer dereference vulnerability (CVE-2010-3849) to trigger another memory corruption attack
  - E.g., double-free or use-after-free bugs could be used to corrupt memory
- Any kernel weakness can turn out to be a security threat [Arnold09]

Challenges & Possible Solutions

- Challenges
  - How to recover and continue execution?
  - How to terminate the buggy component without violating kernel invariants (release locks, etc.)?
- Compile-time analysis
  - Find null-derefs [Dillig07] and infinite loops [Cook06]
- Runtime recovery
  - Shadow drivers and recovery domains [Lenharth09]
  - Are there general-purpose techniques?

Conclusion

- We have a long way to go in making existing OS kernels secure
- State-of-the-art defense techniques address only a small subset of vulnerabilities
- Semantic bugs and DoS attacks pose challenging research problems

Q & A

Linux kernel vulnerabilities:

State-of-the-art defenses and open problems

Haogang Chen, Yandong Mao, Xi Wang, Dong Zhou*
Nickolai Zeldovich and M. Frans Kaashoek

MIT CSAIL
*Tsinghua University

Distribution of Vulnerabilities

Vulnerability	Total	core	drivers	net	fs	sound
Missing pointer check	8	4	3	1	0	0
Missing permission check	17	3	1	2	11	0
Buffer overflow	15	3	1	5	4	2
Integer overflow	19	4	4	8	2	1
Uninitialized data	29	7	13	5	2	2
Null dereference	20	9	3	7	1	0
Divide by zero	4	2	0	0	1	1
Infinite loop	3	1	1	1	0	0
Data race / deadlock	8	5	1	1	1	0
Memory mismanagement	10	7	1	1	0	1
Miscellaneous	8	2	0	4	2	0
Total	141	47	28	35	24	7

Vulnerabilities vs. Tools

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