33RD USENIX SECURITY SYMPOSIUM

上海科技大学
ShanghaiTech University

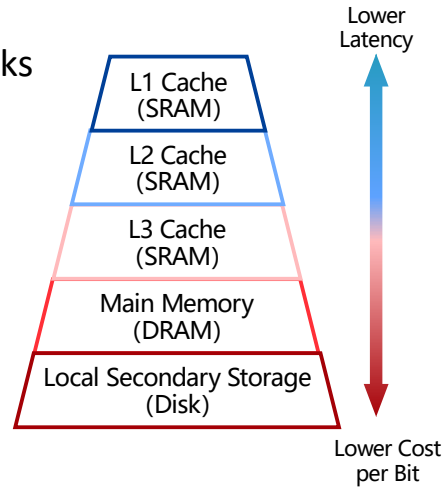
Sync+Sync: A Covert Channel Built on `fsync` with Storage

Qisheng Jiang and Chundong Wang
ShanghaiTech University

Presenter: Jian Zhang, Rutgers University

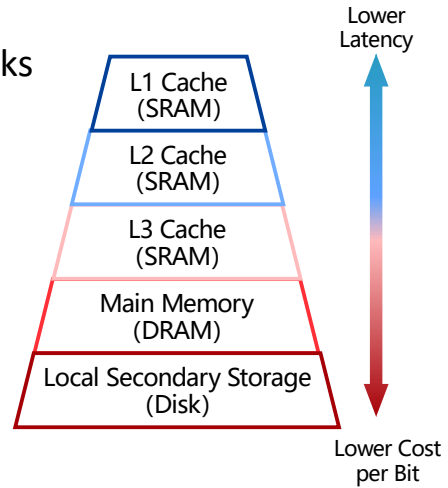
Background

- Covert Channel and Side-Channel Attacks
  - Memory Hierarchy



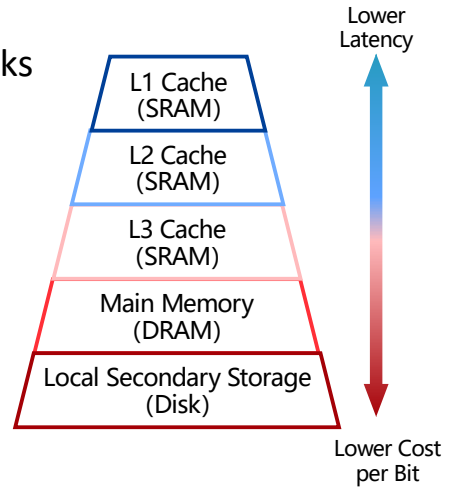
Background

- Covert Channel and Side-Channel Attacks
  - Memory Hierarchy
- **`fsync`**
  - Unprivileged system calls
  - Synchronously flush data to storage
  - Very long response latency



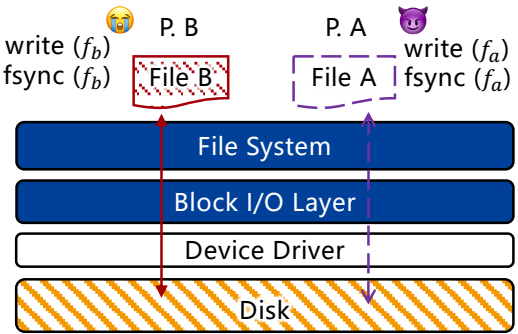
Background

- Covert Channel and Side-Channel Attacks
  - Memory Hierarchy
- **`fsync`**
  - Unprivileged system calls
  - Synchronously flush data to storage
  - Very long response latency
- **`fsync`** contention
  - Multiple files
  - Even longer response latency



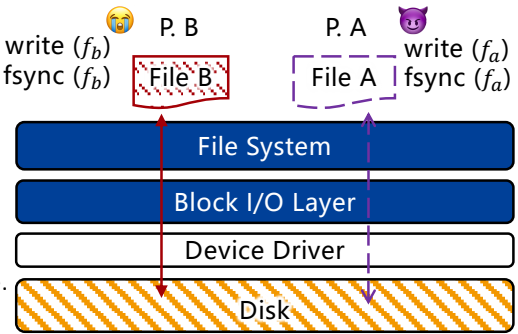
fsync Contention

- Program A and Program B
  - **`ftruncate`** / write / idle + **`fsync`**
- Only measure the **`fsync`** latency of Program A
  - A+B: about 2x A-only



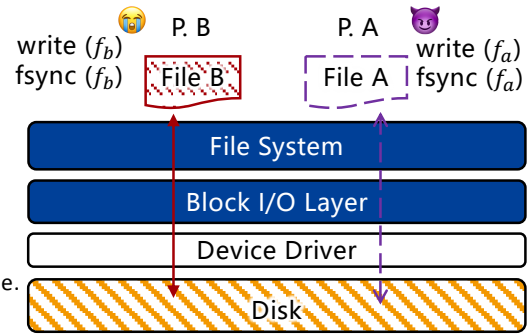
fsync Contention

- Program A and Program B
  - **`ftruncate`** / write / idle + **`fsync`**
- Only measure the **`fsync`** latency of Program A
  - A+B: about 2x A-only
- Why?
  - Contention within file system
    - Only one running transaction in Ext4 at a time.



fsync Contention

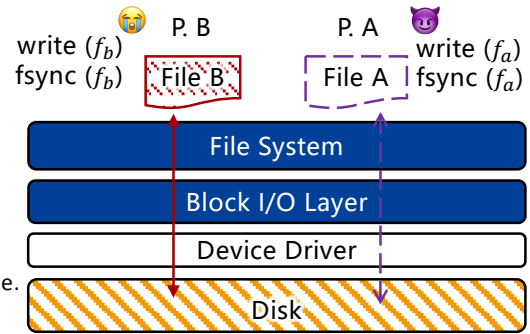
- Program A and Program B
  - **`ftruncate`** / write / idle + **`fsync`**
- Only measure the **`fsync`** latency of Program A
  - A+B: about 2x A-only
- Why?
  - Contention within file system
    - Only one running transaction in Ext4 at a time.
  - Contention within storage device
    - Software/hardware queues
    - REQ_PREFLUSH and REQ_FUA flags



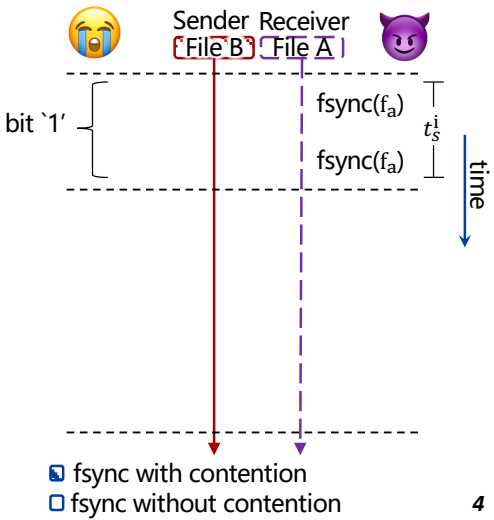
fsync Contention

- Program A and Program B
  - **`ftruncate`** / write / idle + **`fsync`**
- Only measure the **`fsync`** latency of Program A
  - A+B: about 2x A-only
- Why?
  - Contention within file system
    - Only one running transaction in Ext4 at a time.
  - Contention within storage device
    - Software/hardware queues
    - REQ_PREFLUSH and REQ_FUA flags

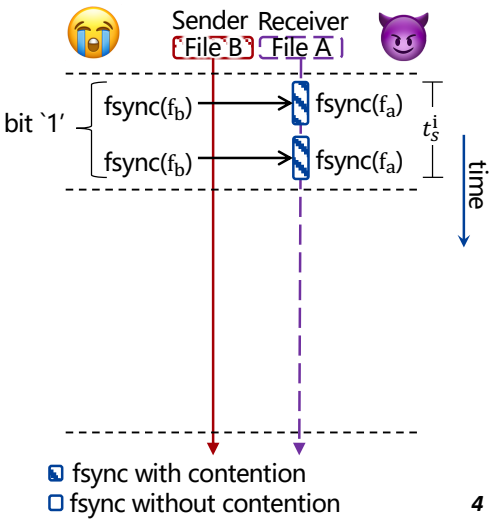
A good fit for timing-based channel



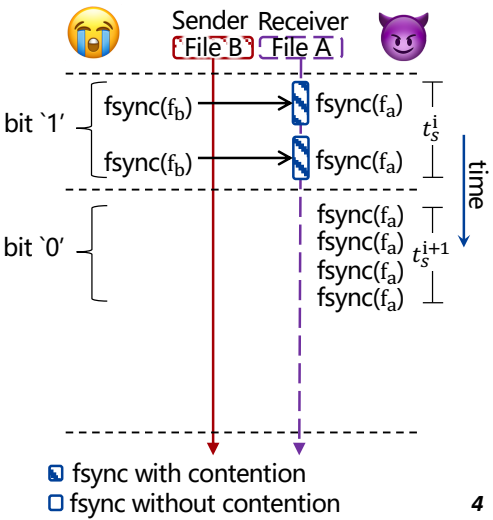
Sync+Sync - Covert Channel



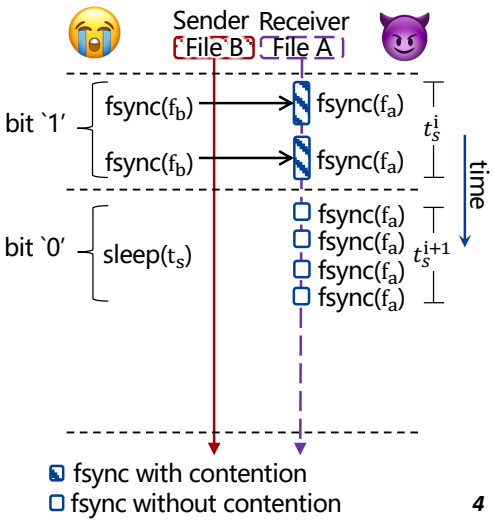
Sync+Sync - Covert Channel



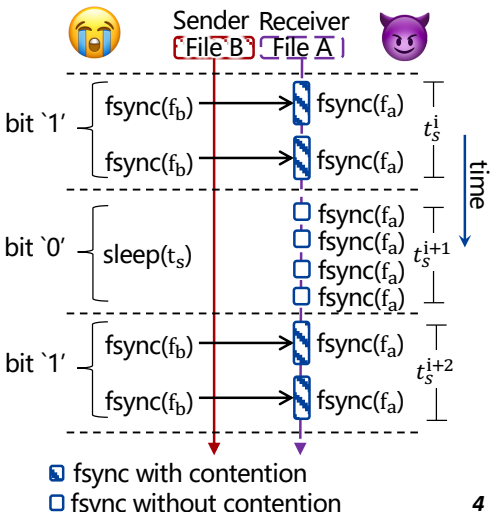
Sync+Sync - Covert Channel



Sync+Sync - Covert Channel

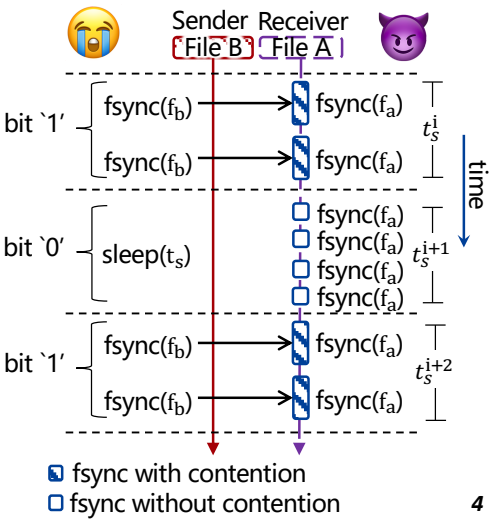


Sync+Sync - Covert Channel



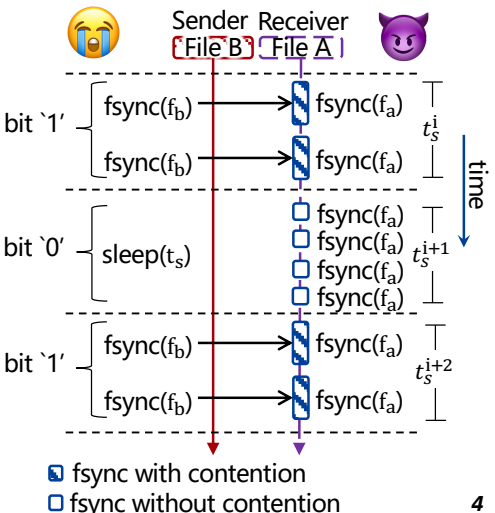
Sync+Sync - Covert Channel

- Types of Channels
  - Cross-file
    - 20,000 bps at an error rate of about 0.40%
  - Cross-container
    - Overlay file system
  - Cross-VM: disk images



Sync+Sync - Covert Channel

- Types of Channels
  - Cross-file
    - 20,000 bps at an error rate of about 0.40%
  - Cross-container
    - Overlay file system
  - Cross-VM: disk images
- Positions of two sides
  - Intra-partition
  - Inter-partition
- Various storage devices
  - SATA SSDs
  - NVMe SSDs



Sync+Sync - Side-Channel Attacks

- Principle
  - Victim application: accesses files and may invoke **`fsync`**
    - E.g. Databases, Linux/Android applications, and web browsers

Sync+Sync - Side-Channel Attacks

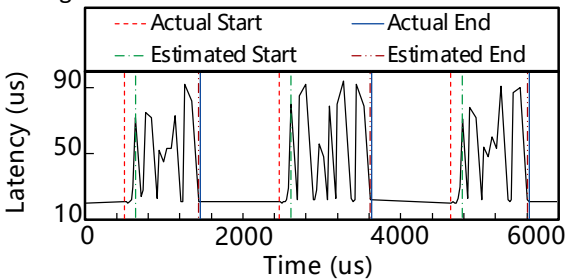
- Principle
  - Victim application: accesses files and may invoke **`fsync`**
    - E.g. Databases, Linux/Android applications, and web browsers
  - Attacker
    - On the same disk but invokes **`fsync`** on an irrelevant file periodically
    - Can observe the application's **`fsync`** behaviors

Sync+Sync - Side-Channel Attacks

- Principle
  - Victim application: accesses files and may invoke **`fsync`**
    - E.g. Databases, Linux/Android applications, and web browsers
  - Attacker
    - On the same disk but invokes **`fsync`** on an irrelevant file periodically
    - Can observe the application's **`fsync`** behaviors
- Observation
  - Different applications exhibit varying patterns of **`fsync`** calls
    - Frequency
    - Data volume to be flushed
  - The attacker can recognize different patterns of **`fsync`** calls

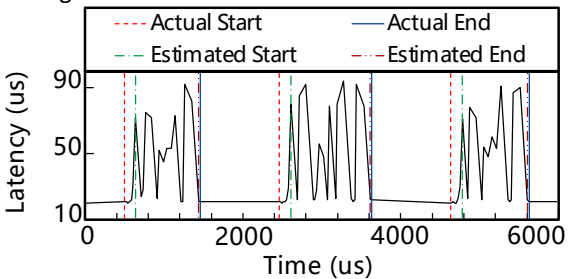
Database Operations Speculation

- Attack Design
  - Victim database
    - SQLite, Invoking **`fsync`** when committing



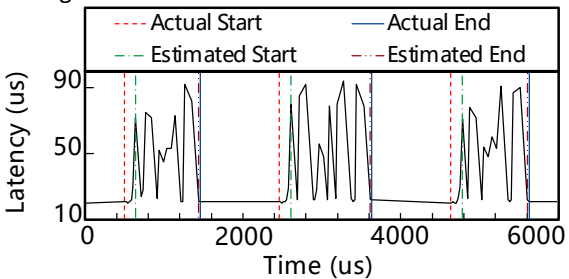
Database Operations Speculation

- Attack Design
  - Victim database
    - SQLite, Invoking **`fsync`** when committing
  - Observations
    - Attacker: Higher **`fsync`** latency
    - SQLite: **`fsync`** operations (committing a transaction)



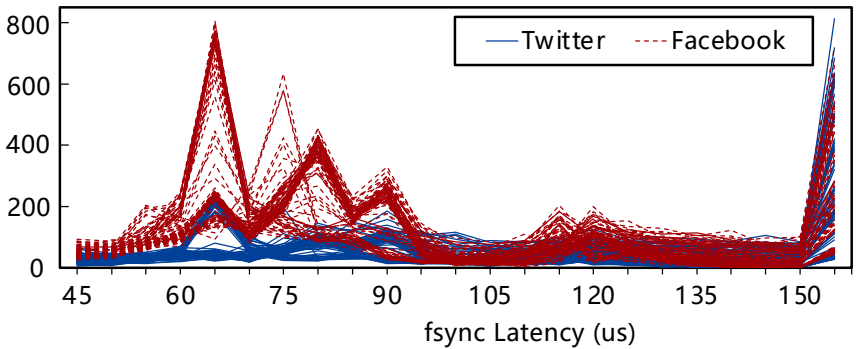
Database Operations Speculation

- Attack Design
  - Victim database
    - SQLite, Invoking **`fsync`** when committing
  - Observations
    - Attacker: Higher **`fsync`** latency
    - SQLite: **`fsync`** operations (committing a transaction)
- Information Leakage
  - Insert/Update Ratio over Time
  - B-Tree Split Detection
  - Database Operation Leakage



Application Information Leakage

- Application Fingerprinting
  - e.g., Twitter and Facebook



Application Information Leakage

- Website Fingerprinting
  - Most websites do not commonly use **`fsync`**.
  - Some websites invoke **`fsync`** more frequently and exhibit different I/O behaviors.
    - Due to the use of Indexed Database

Website	Average # fsync	Accuracy	F1-score
360.cn	10.6	3.3%	0.04
adobe.com	11.5	0.0%	0.00
amazon.com	16.2	13.3%	0.08
apple.com	11.5	16.7%	0.17
baidu.com	14.4	6.7%	0.03
bing.com	15.0	6.7%	0.08
booking.com	15.9	0.0%	0.00
cnn.com	15.2	6.7%	0.10
detik.com	10.1	10.0%	0.13
github.com	12.1	13.3%	0.18

Website	Average # fsync	Accuracy	F1-score
imdb.com	16.1	13.3%	0.19
jd.com	14.9	56.7%	0.65
live.com	10.6	13.3%	0.14
microsoft.com	12.1	3.3%	0.04
qq.com	**264.6**	**100.0%**	**1.00**
sina.com.cn	**40.8**	**96.7%**	**0.98**
sohu.com	14.4	46.7%	0.44
taobao.com	10.6	23.3%	0.16
tmall.com	11.4	3.3%	0.03
yahoo.co.jp	11.4	6.7%	0.06

Keystroke Attack

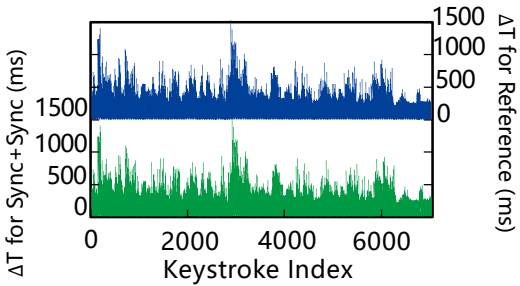
- Attack Design
  - Victim
    - For every keystroke typed by the victim, the service program auto-commits the user input by storing it in a file with an **`fsync`**
    - A keystroke -> an **`fsync`**

Keystroke Attack

- Attack Design
  - Victim
    - For every keystroke typed by the victim, the service program auto-commits the user input by storing it in a file with an **`fsync`**
    - A keystroke -> an **`fsync`**
  - Attacker
    - Increased **`fsync`** latency -> a keystroke

Keystroke Attack

- Attack Design
  - Victim
    - For every keystroke typed by the victim, the service program auto-commits the user input by storing it in a file with an **`fsync`**
    - A keystroke -> an **`fsync`**
  - Attacker
    - Increased **`fsync`** latency -> a keystroke
- Inter-keystroke Timings
  - Inter-keystroke latency (> 100ms)
  - Attacker's fsync latency (< 100us)
  - An accuracy of 99.2%



Conclusion

- Sync+Sync is the first covert channel that makes use of **`fsync`** at the persistent storage level.

Conclusion

- Sync+Sync is the first covert channel that makes use of **`fsync`** at the persistent storage level.
- Sync+Sync covert channel effectively works in various scenarios.

Conclusion

- Sync+Sync is the first covert channel that makes use of **`fsync`** at the persistent storage level.
- Sync+Sync covert channel effectively works in various scenarios.
  - Cross-partition, cross-file system, cross-container, cross-VM, and even cross-disk

Conclusion

- Sync+Sync is the first covert channel that makes use of **`fsync`** at the persistent storage level.
- Sync+Sync covert channel effectively works in various scenarios.
  - Cross-partition, cross-file system, cross-container, cross-VM, and even cross-disk
- Sync+Sync attacks affects real-world applications that utilize **`fsync`**s in their implementations.

Conclusion

- Sync+Sync is the first covert channel that makes use of **`fsync`** at the persistent storage level.
- Sync+Sync covert channel effectively works in various scenarios.
  - Cross-partition, cross-file system, cross-container, cross-VM, and even cross-disk
- Sync+Sync attacks affects real-world applications that utilize **`fsync`**s in their implementations.

A full version of the paper is available at https://arxiv.org/abs/2309.07657
Source code of Sync+Sync covert channel is available at https://github.com/toast-lab/sync-sync
Should you have any question, drop an email to toast-lab@outlook.com

上海科技大学
ShanghaiTech University

Thanks :-)

Toast Lab in ShanghaiTech (June 1st, 2024)

Jian Zhang is on the job market

https://toast-lab.tech
toast-lab@outlook.com

TOAST LAB
吐 司 实 验 室

立志成才 报国裕民