DUPEFS: Leaking Data Over the Network With Filesystem Deduplication Side Channels

Andrei Bacs and Saidgani Musaev, VUSec, Vrije Universiteit Amsterdam;

Kaveh Razavi, ETH Zurich; Cristiano Giuffrida and Herbert Bos,

VUSec, Vrije Universiteit Amsterdam

FAST 2022

Background

- **Filesystem&Dedup**
  - Inline Deduplication
  - Basic write workflow
  - Data identifiers
  - Deduplication tables
  - Deduplication granularity

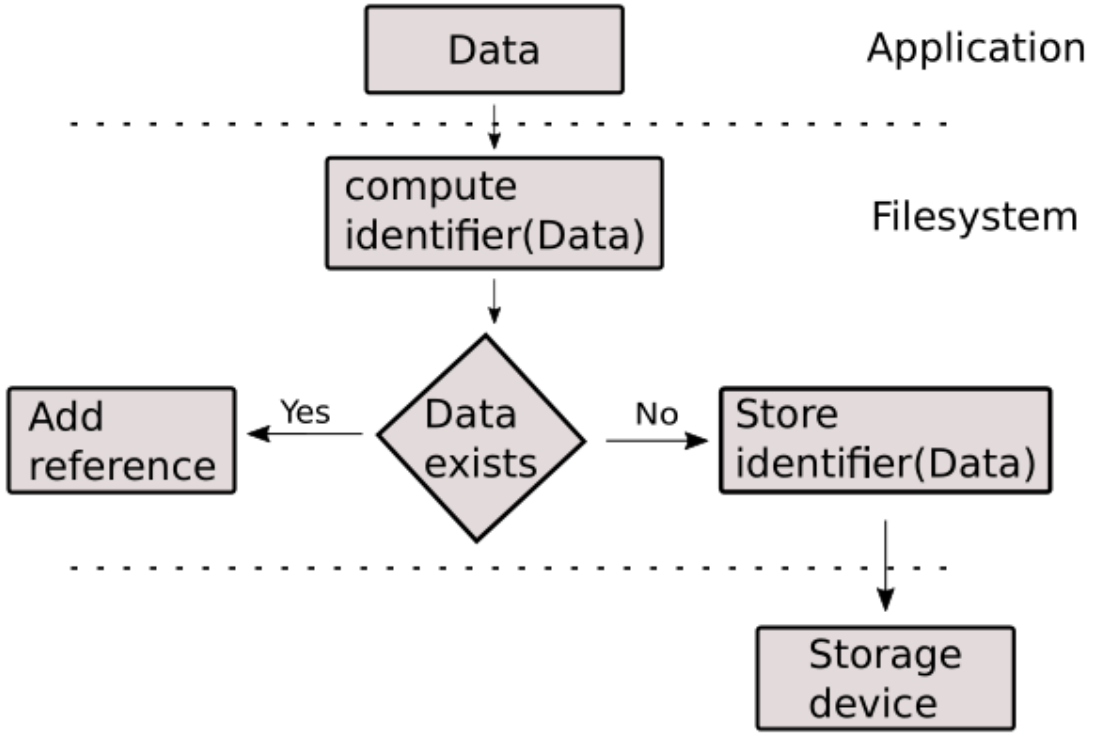
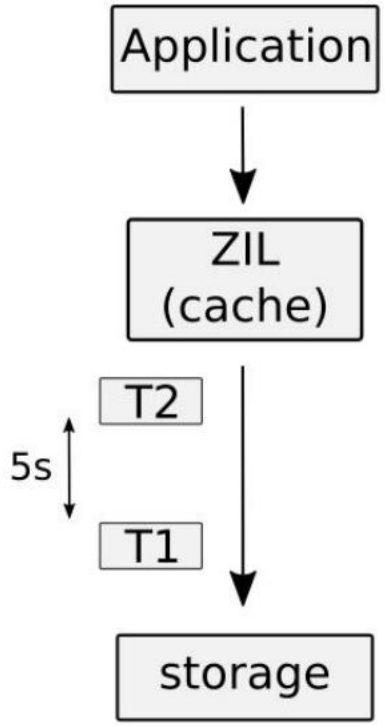
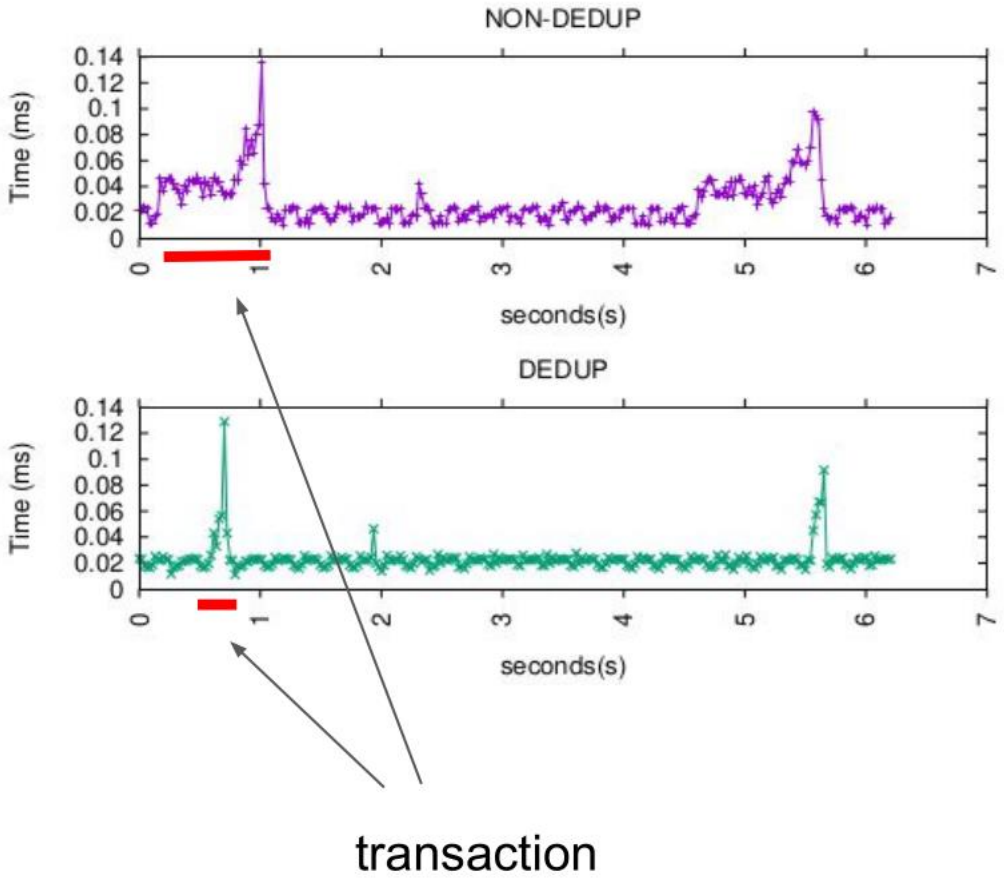


Figure 1: Write path with deduplication.

Deduplication timing side channel



Write path with deduplication



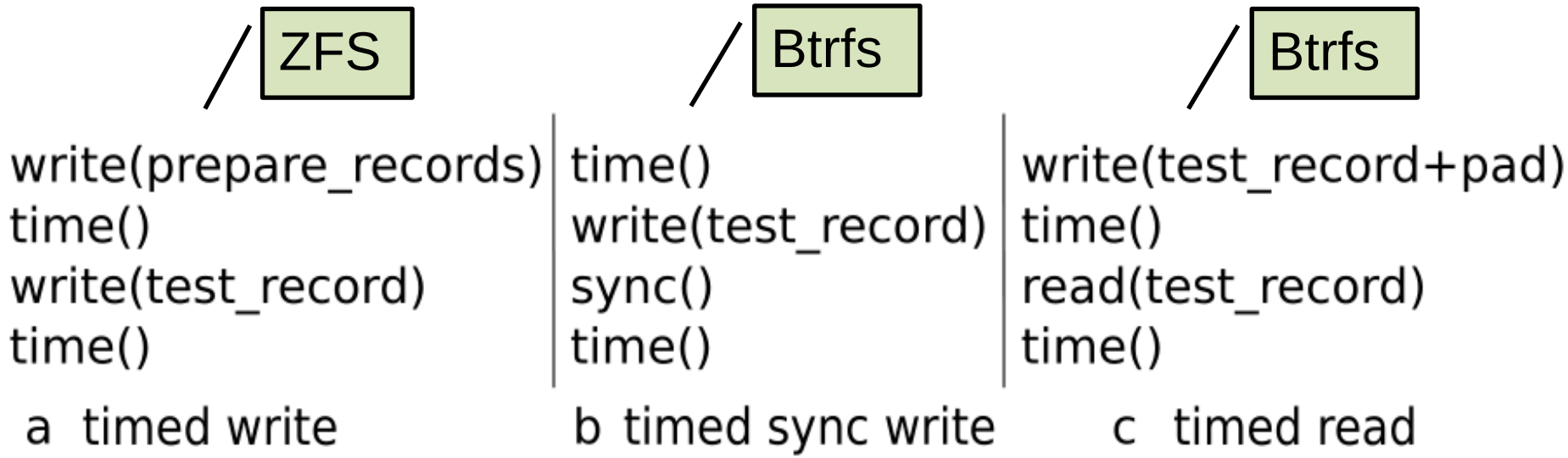
Transaction is shorter (pulse width) for duplicate data

Background

- **ZFS**
  - Transactional copy-on-write filesystem
  - Deduplication record(128KB)&deduplication table(DDT)
- **Btrfs**
  - Extent(128KB)： The data are stored in extents
  - COW filesystem
- **Primitive**
  - The timed write primitive
  - The timed read primitive

Attack Primitives

- carefully-chosen sequence of file operations



Attack Challenges

- **Filesystem asynchronous I/O operations**
  - intermediary caches
  - transactional behavior
  - ✓Exploitation technique: filesystem cache massaging
- **Deduplication granularity**
  - typical record size 128KB (ZFS and Btrfs)
  - ✓Exploitation technique: alignment probing
- **Week amplification in a remote attack**
  - ✓Exploitation technique: secret spraying

Threat Model

- Attacker and victim have access to the same filesystem
- The filesystem uses inline deduplication and default settings
- No limit on I/O operations

DUPEFS Exploitation

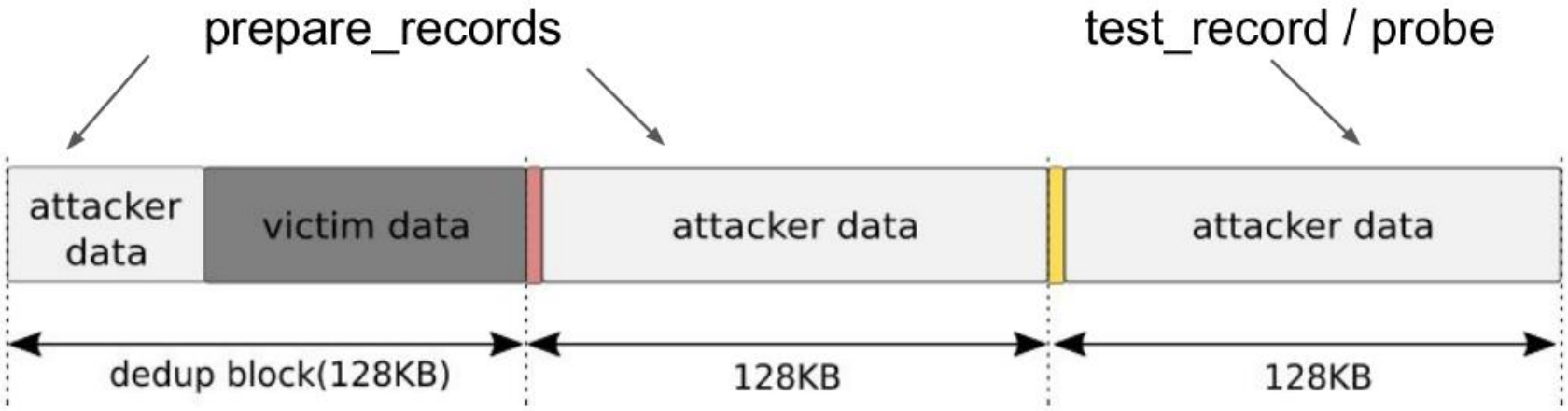
- **Data fingerprinting**
  - reveal the presence of existing known but inaccessible data
- **Data exfiltration**
  - exfiltrate secret data from a system (or sandbox)
- **Data leak**
  - leak secret data from a remote system
  - alignment probing
  - secret spraying

Alignment probing

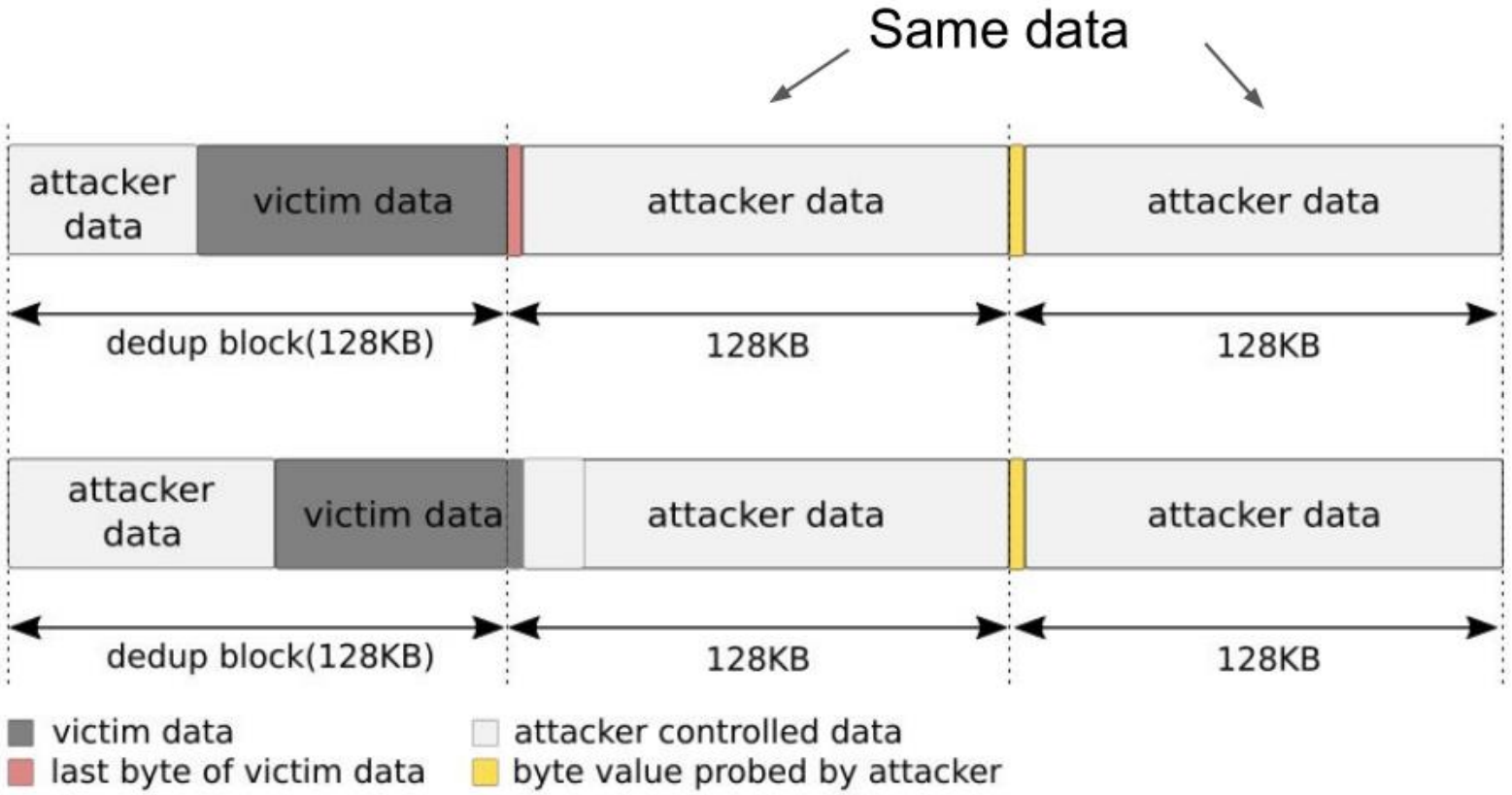
- Enables byte granularity
- Reduces entropy

Timed write primitive

```
write(prepare_records)
time( )
write(test_record)
time( )
```

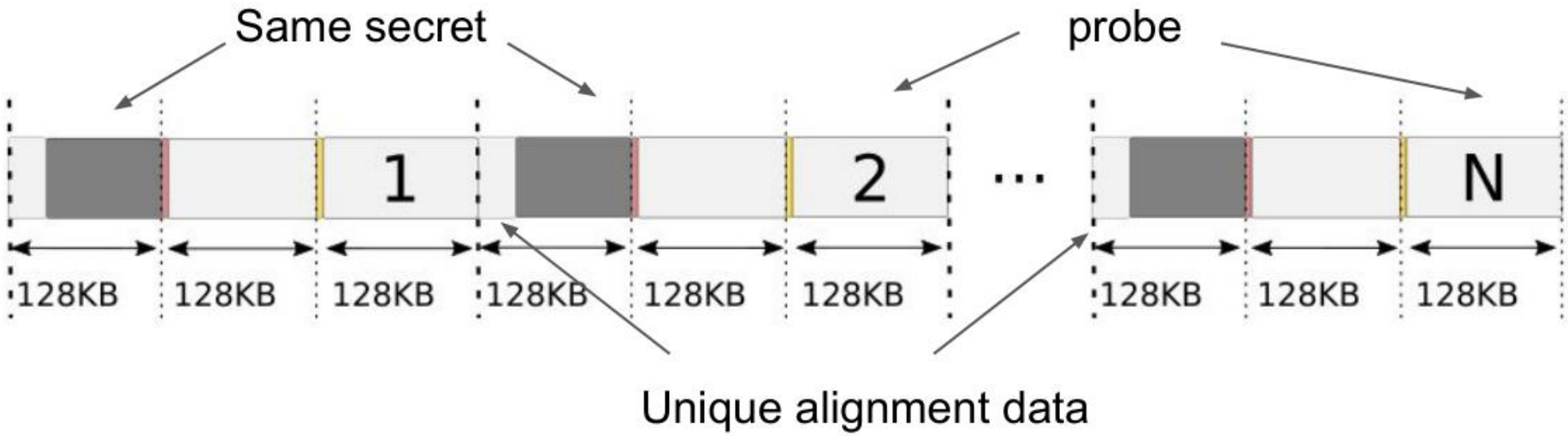


Alignment probing



Secret spraying

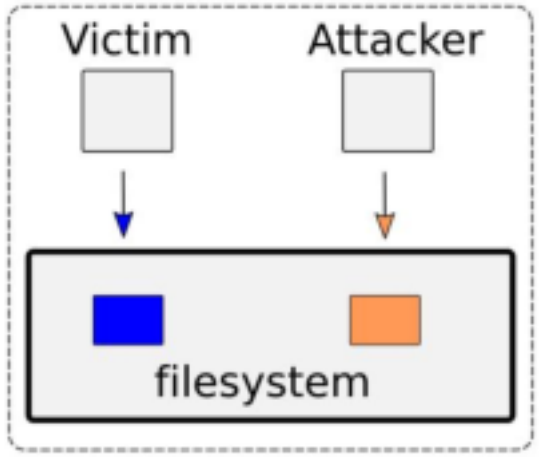
- Amplifies the timing signal
- N dedup events per correct guessed byte



Data Fingerprinting/exfiltration

➢ Data Fingerprinting

- Btrfs-based synchronous write primitive
- local exploitation scenario



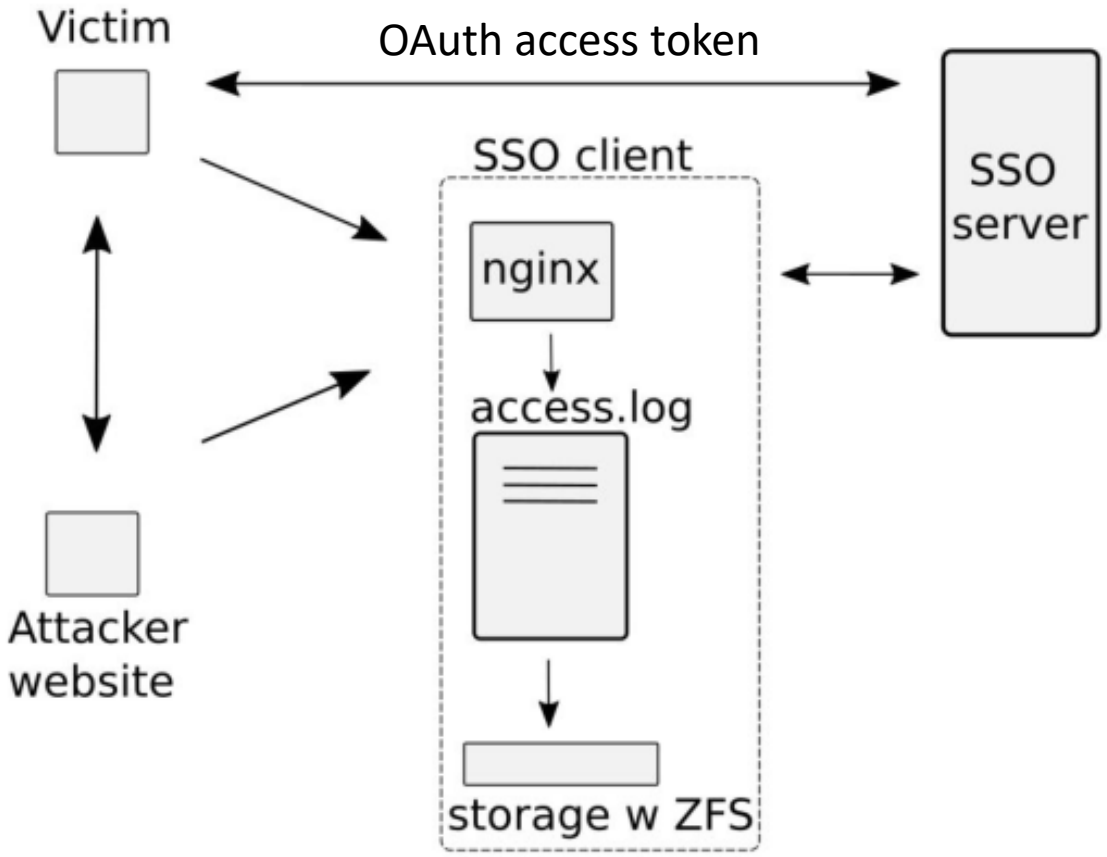
Local: data fingerprinting, exfiltration

➢ Data exfiltration

- Btrfs-based synchronous write primitive
- A local unprivileged attacker (or “sender”) seeks to exfiltrate data from a sandbox over a covert channel.

Data leak

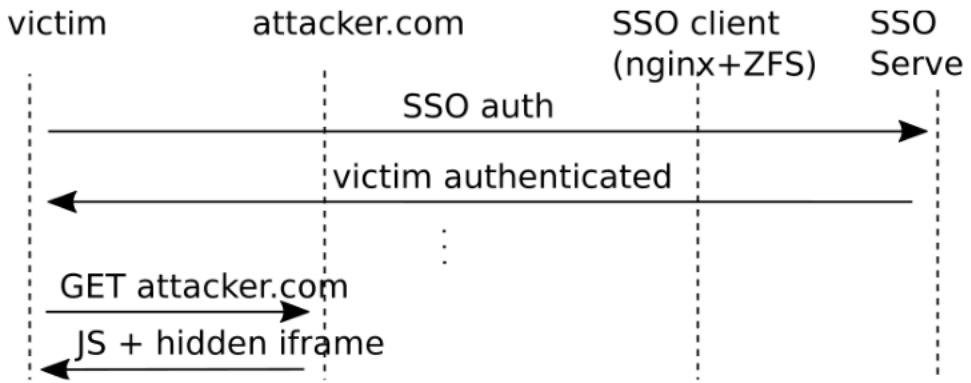
- targets access tokens of OAuth 2.0 implicit grant access scheme
- SSO client runs nginx on top of ZFS and logs requests
- access tokens are encoded in the request and do not expire during attack
- SSO client does not offer X-Frame-Options



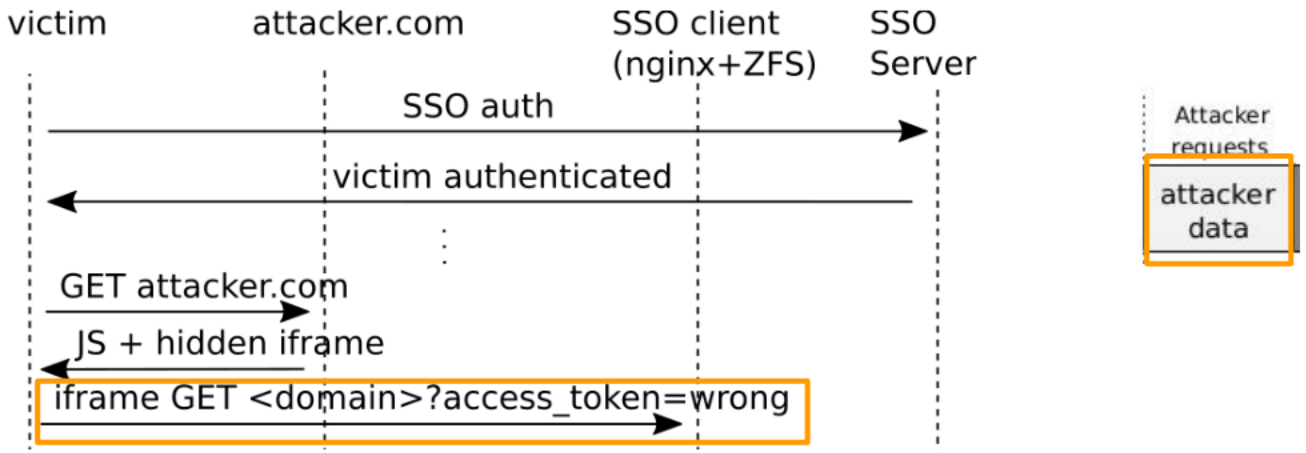
Data leak


victim
attacker.com
SSO client
(nginx+ZFS)
SSO
Server
SSO auth
victim authenticated

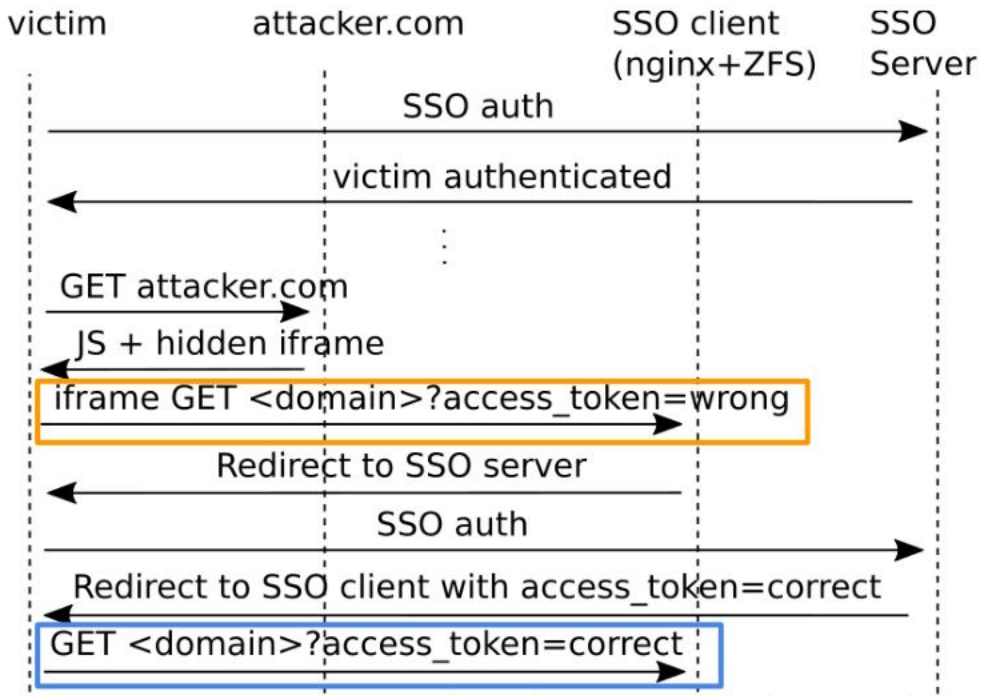
Data leak

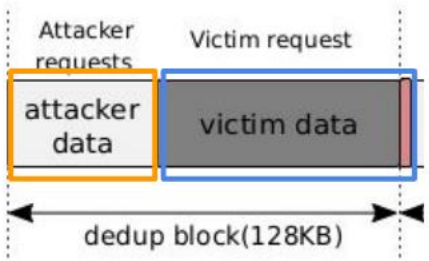

victim
attacker.com
SSO client
(nginx+ZFS)
SSO
Serve
SSO auth
victim authenticated
GET attacker.com
JS + hidden iframe

Data leak

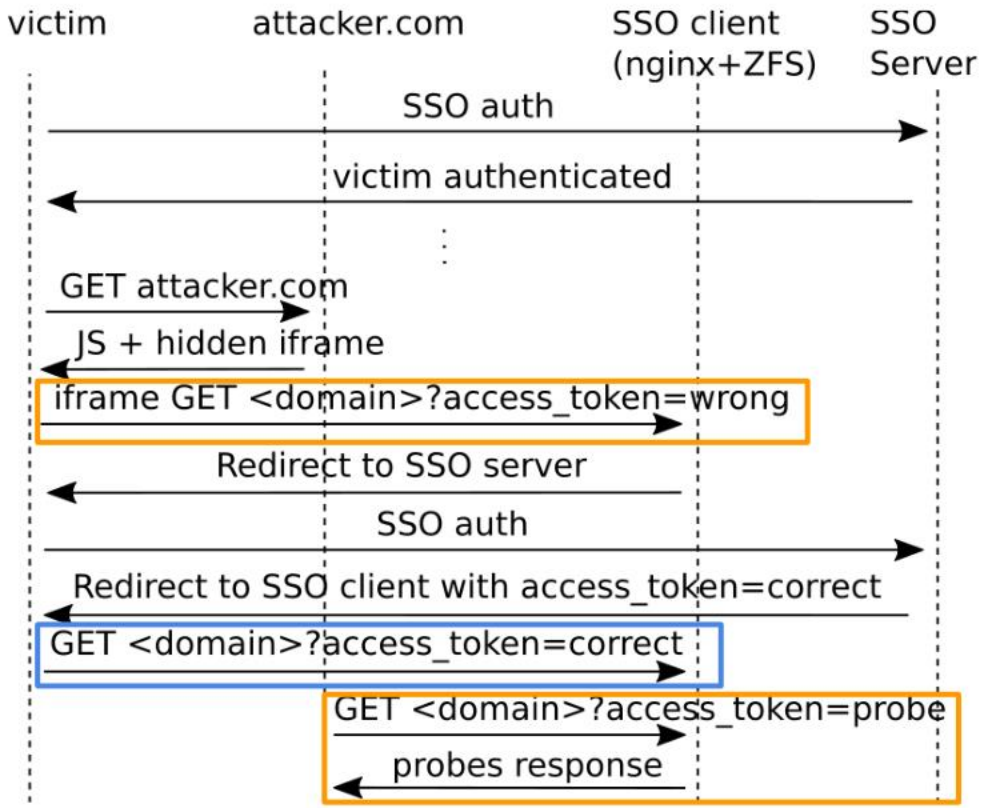

victim
attacker.com
SSO client
(nginx+ZFS)
SSO
Server
SSO auth
victim authenticated
GET attacker.com
JS + hidden iframe
iframe GET <domain>?access_token=wrong
Attacker
requests
attacker
data

Data leak


victim
attacker.com
SSO client (nginx+ZFS)
SSO Server
SSO auth
victim authenticated
GET attacker.com
JS + hidden iframe
iframe GET <domain>?access_token=wrong
Redirect to SSO server
SSO auth
Redirect to SSO client with access_token=correct
GET <domain>?access_token=correct

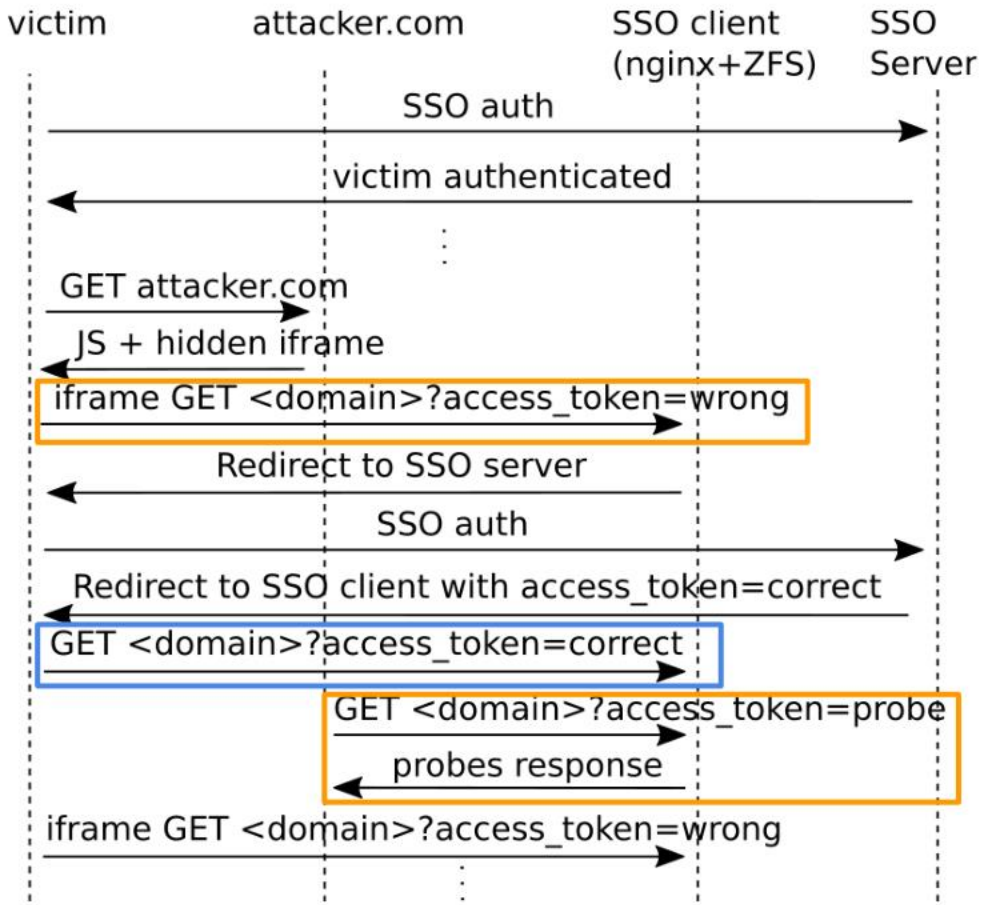

Attacker requests
Victim request
attacker data
victim data
dedup block(128KB)

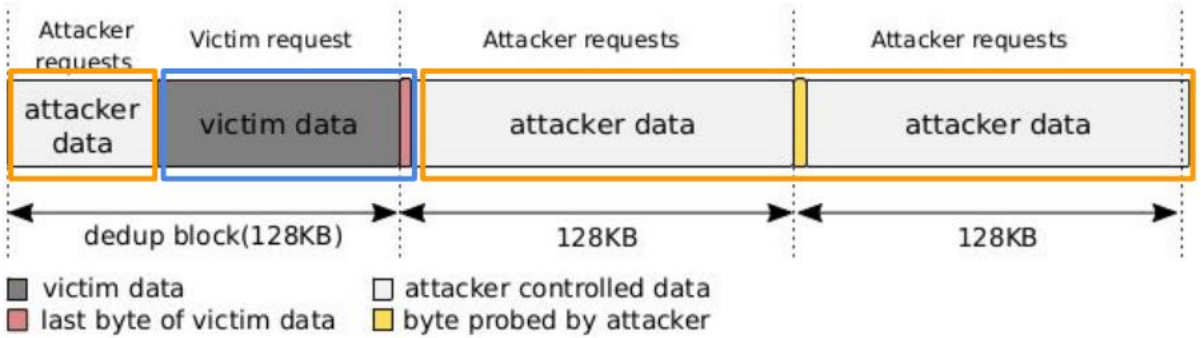
Data leak


victim
attacker.com
SSO client (nginx+ZFS)
SSO Server
SSO auth
victim authenticated
GET attacker.com
JS + hidden iframe
iframe GET <domain>?access_token=wrong
Redirect to SSO server
SSO auth
Redirect to SSO client with access_token=correct
GET <domain>?access_token=correct
GET <domain>?access_token=probe
probes response

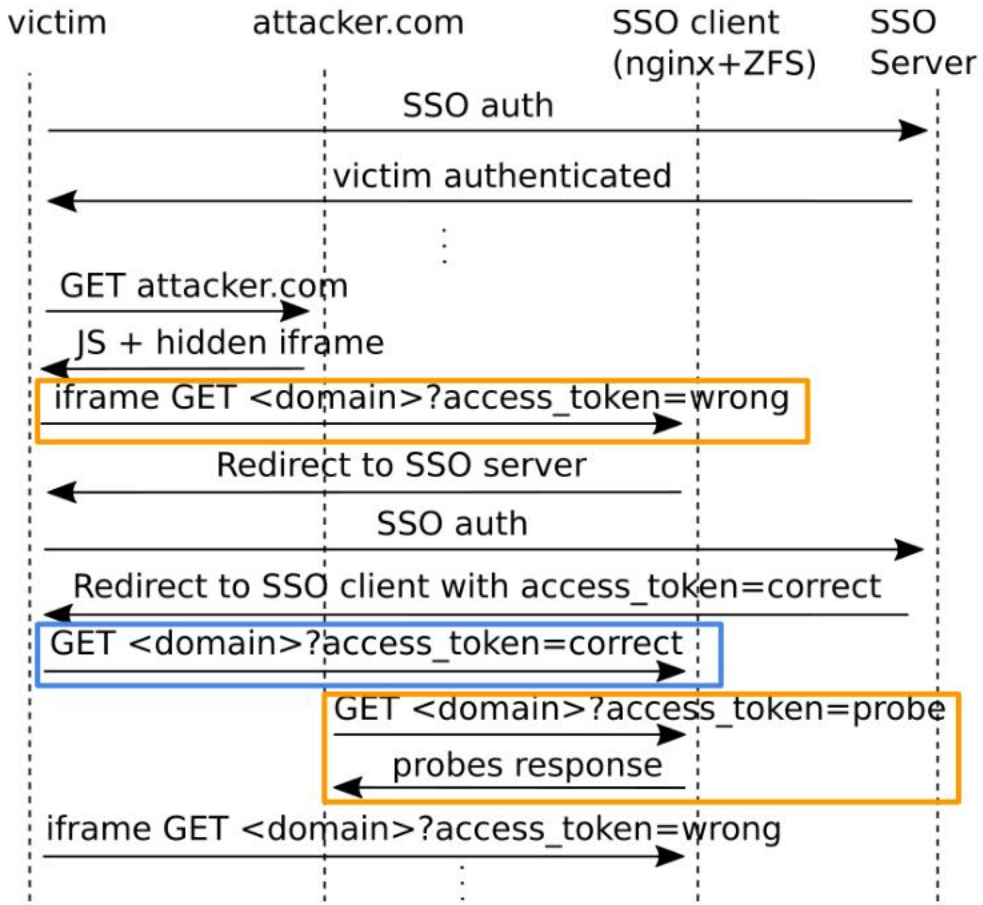

Attacker requests
Victim request
Attacker requests
Attacker requests
attacker data
victim data
attacker data
attacker data
dedup block(128KB)
128KB
128KB
victim data
last byte of victim data
attacker controlled data
byte probed by attacker

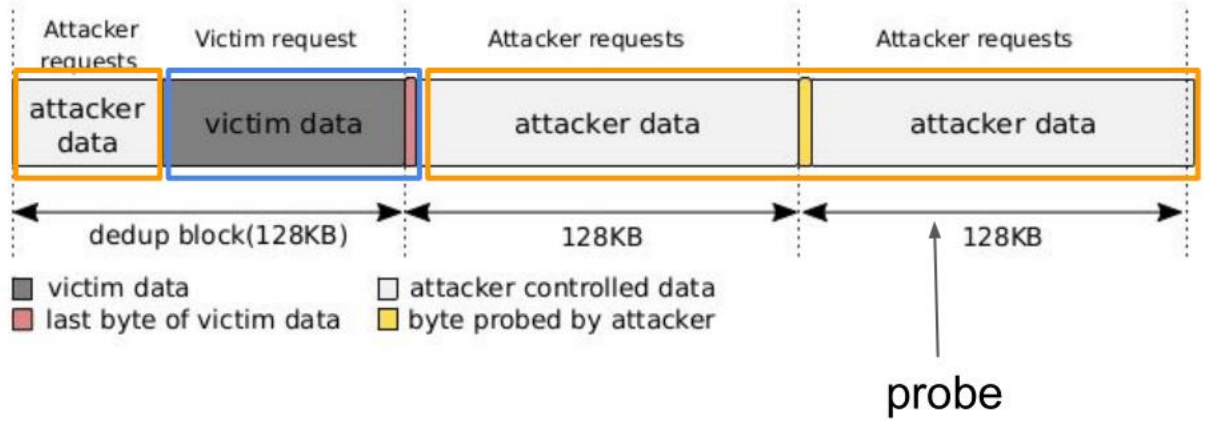
Data leak


victim
attacker.com
SSO client (nginx+ZFS)
SSO Server
SSO auth
victim authenticated
GET attacker.com
JS + hidden iframe
iframe GET <domain>?access_token=wrong
Redirect to SSO server
SSO auth
Redirect to SSO client with access_token=correct
GET <domain>?access_token=correct
GET <domain>?access_token=probe
probes response
iframe GET <domain>?access_token=wrong


Attacker requests
Victim request
Attacker requests
Attacker requests
attacker data
victim data
attacker data
attacker data
dedup block(128KB)
128KB
128KB
victim data
last byte of victim data
attacker controlled data
byte probed by attacker

Data leak





- correct byte value probe produces transaction durations < threshold

Evaluation

- **Data fingerprinting**
  - DUPEFS can reliably fingerprint thetarget data except the last sub-128 KB chunk of a file. Thus the small (181 KB and 223 KB) files have lower success rates.

Table 1: File fingerprinting

File	Type	Size	Success
config-4.11.3-200.fc25.x86_64	text	181 KB	70%
lena_color.gif	binary	223 KB	55%
libz3.so	binary	22 MB	99%
x86_64-redhat-linux-c++	binary	1 MB	99%

Evaluation

- **Data Exfiltration**

Table 2: Covert channel

N	Bit errors	Time	BR	BER	I/O
20	13	375s	0.320 bps	10.83%	76.8MB
40	14	746s	0.160 bps	11.66%	153.6MB
60	12	1591s	0.075 bps	10.00%	230.4MB
100	6	1873s	0.064 bps	5.00%	384.0MB
120	3	2387s	0.050 bps	2.50%	460.8MB

Evaluation

➢ Data Leak

- OAuth token 22 bytes (base64)
- LAN: 1 hop (RTT 0.1ms)
- WAN: 12 hops (RTT 2ms)

Table 3: LAN 1 byte data leak

Success	Attack time/byte	Probes/byte val	I/O
50%	19.2 min	200	4.9 GB
80%	25.6 min	300	7.3 GB
92%	42.6 min	400	9.8 GB
96%	78.9 min	800	19.6 GB

Table 4: WAN 1 byte data leak

Success	Attack time/byte	Probes/byte val	I/O
64%	24.5 min	200	4.9 GB
87%	38.4 min	300	7.3 GB
94%	59.7 min	400	9.8 GB
94%	110.9 min	800	19.6 GB

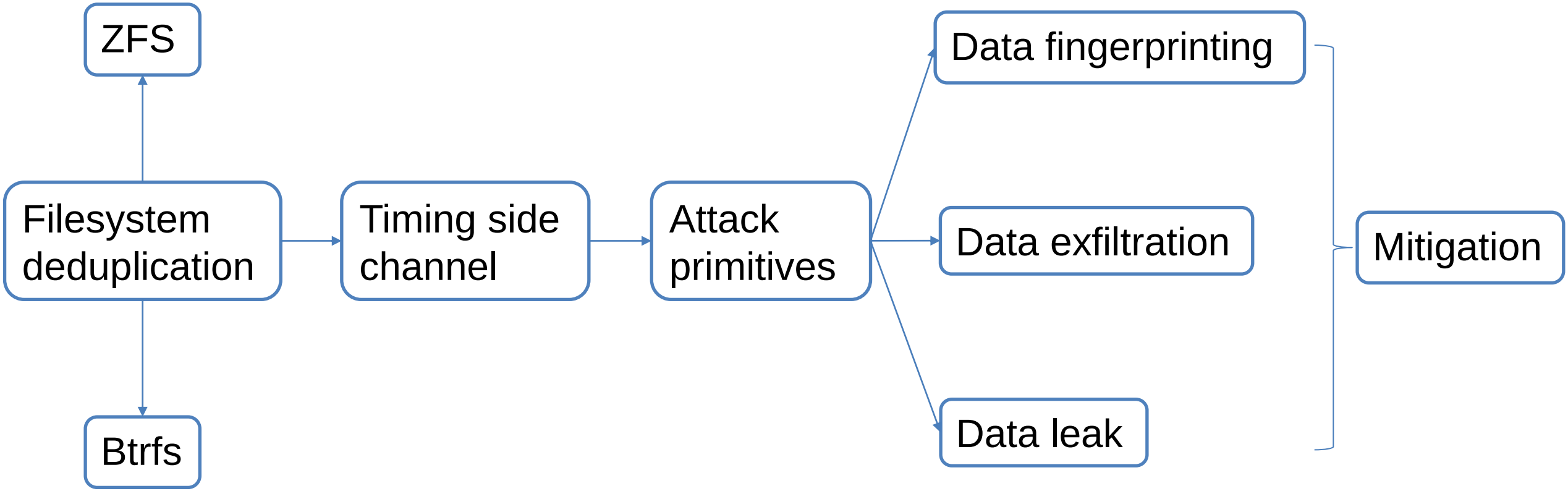
Mitigation

- **Deduplication ideal implementation**
  - Save space
  - Constant time behavior
- **Pseudo-same behavior policy**
  - Perform data overwrite for duplicate data
  - Read path: time jitter
  - Renders remote attacks impractical

Contributions

- Analyze filesystem deduplication side channels, show that despite the challenges, attackers can mount byte-level data leak attacks across the network.
- Introduce DUPEFS's novel attack primitives and demonstrate their feasibility in end-to-end attacks to leak data even across the Internet.
- Describe and analyze mitigations for such attacks.

Conclusion


ZFS
Filesystem deduplication
Btrfs
Timing side channel
Attack primitives
Data fingerprinting
Data exfiltration
Data leak
Mitigation