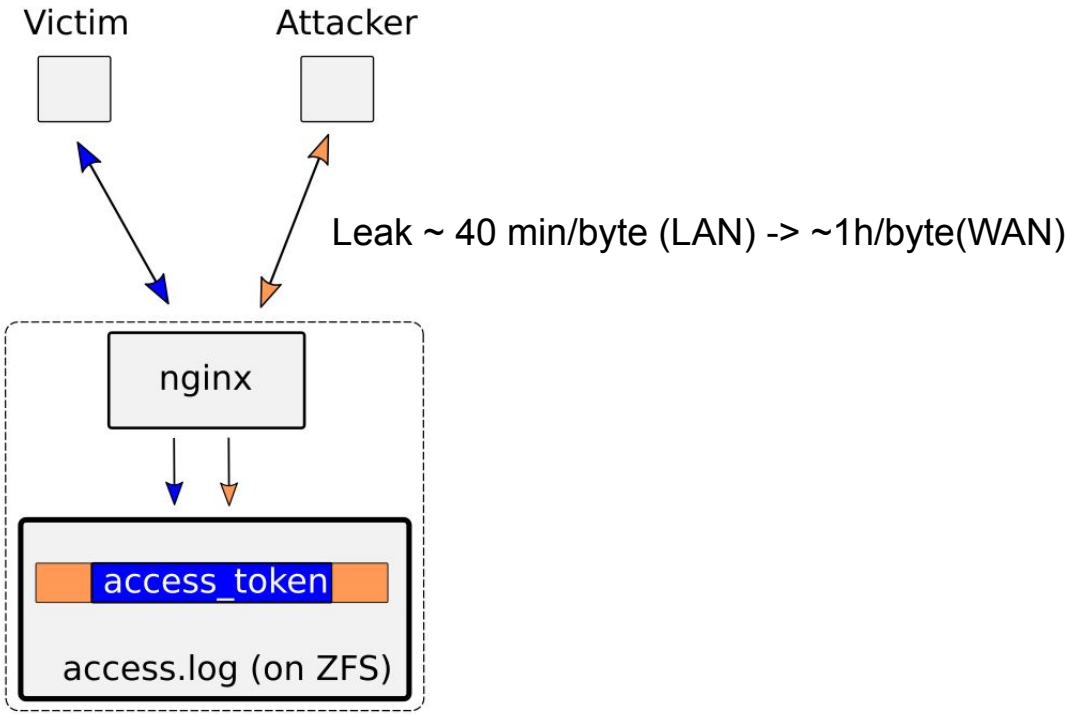
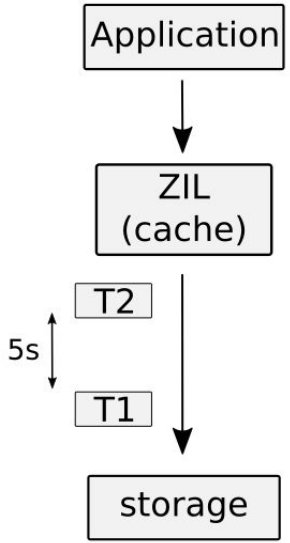
DUPEFS: Leaking Data Over the Network With Filesystem Deduplication Side Channels

Andrei Bacs, Saidgani Musaev, Kaveh Razavi, Cristiano Giuffrida and Herbert Bos

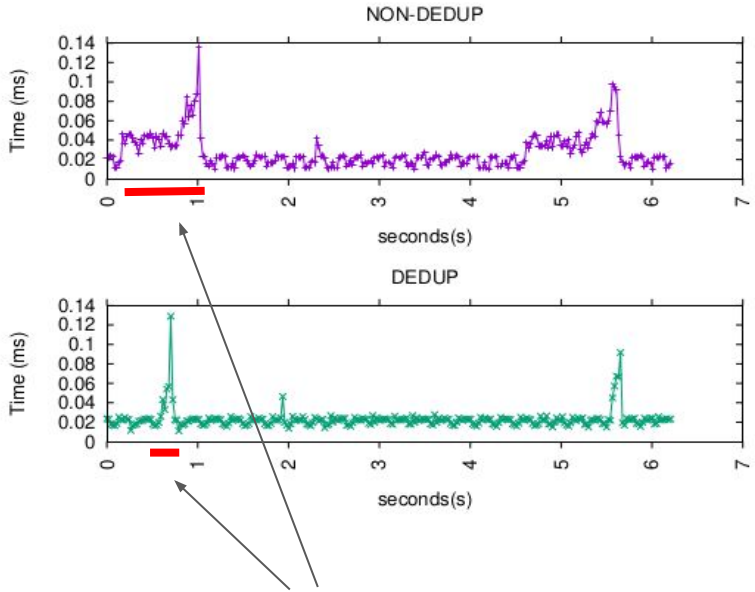
Filesystem deduplication introduces security risks



Deduplication timing side channel



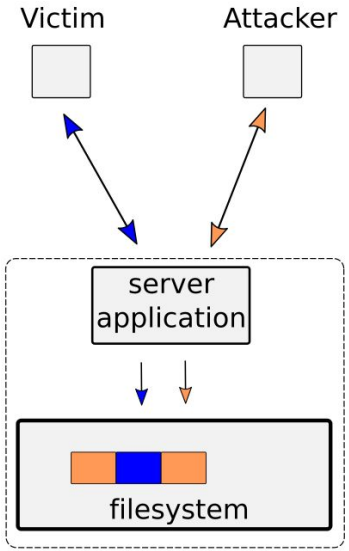
Write path with deduplication



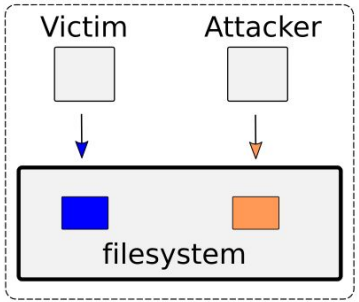
Transaction is shorter (pulse width) for duplicate data

Threat model

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



Remote: data leak



Local: data fingerprinting, exfiltration

Attack challenges

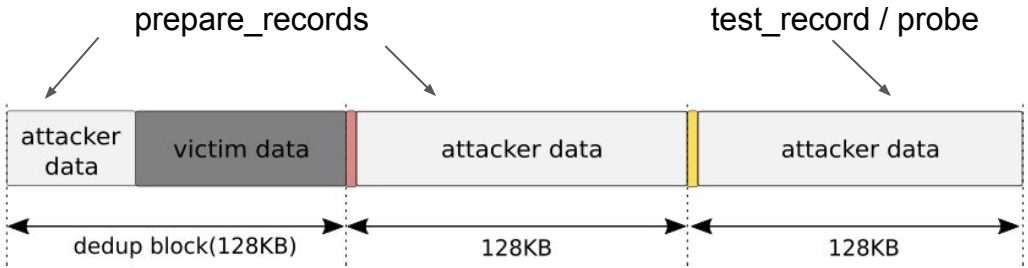
- Filesystem asynchronous I/O operations
  - intermediary caches
  - transactional behavior
  - Exploitation technique: filesystem cache massaging

- Deduplication granularity
  - typical record size 128KB (ZFS and Btrfs)
  - Exploitation technique: alignment probing

- Signal amplification
  - Exploitation technique: secret spraying

Exploitation techniques

Filesystem cache massaging
- interleave attacker and victim writes



Timed write primitive

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

Exploitation techniques

Alignment probing

- enables byte granularity

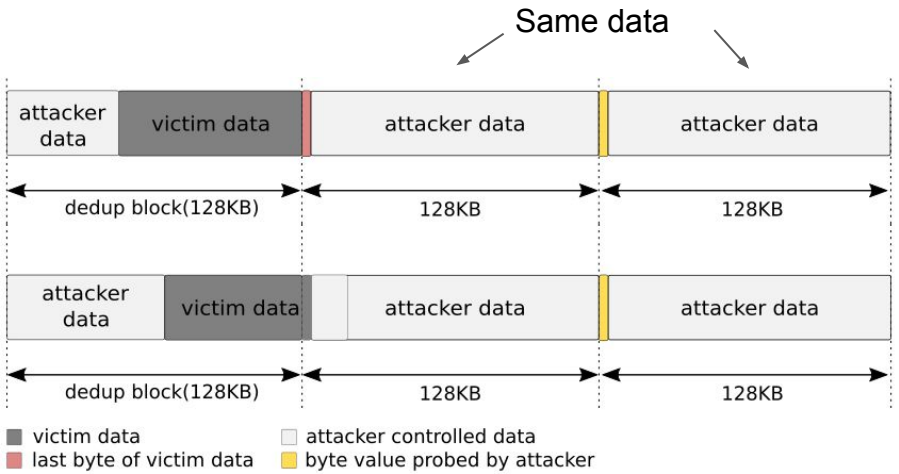
- reduces entropy

Exploitation techniques

Alignment probing

- enables byte granularity

- reduces entropy

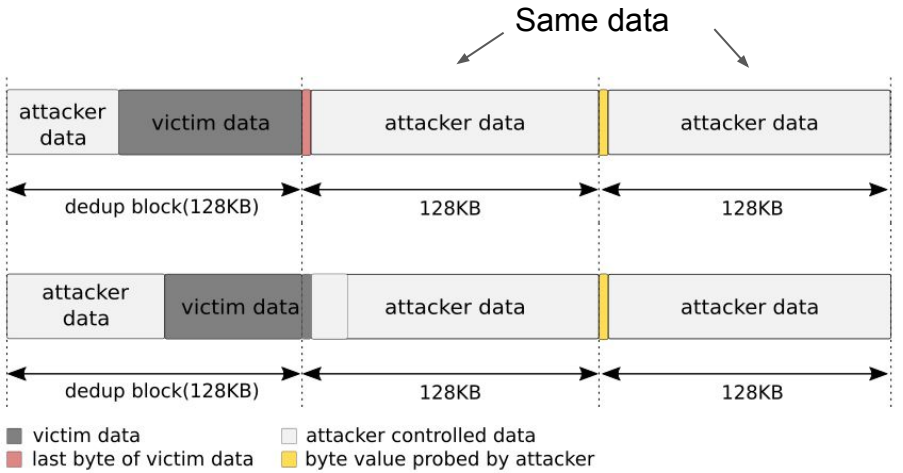


Exploitation techniques

Alignment probing

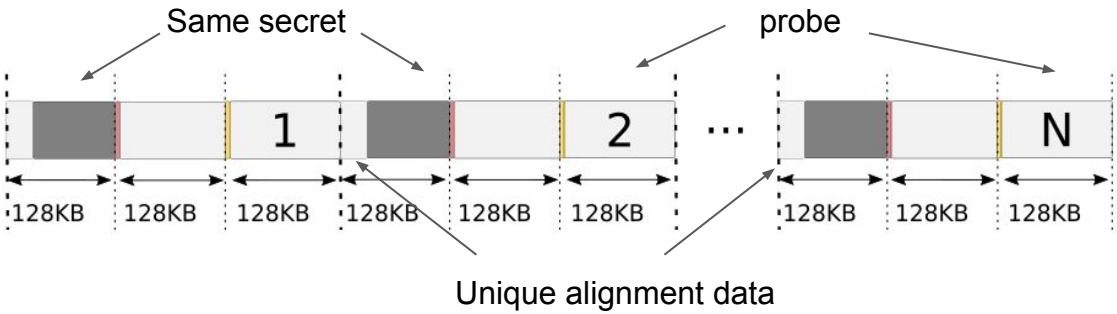
- enables byte granularity

- reduces entropy



Secret spraying

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



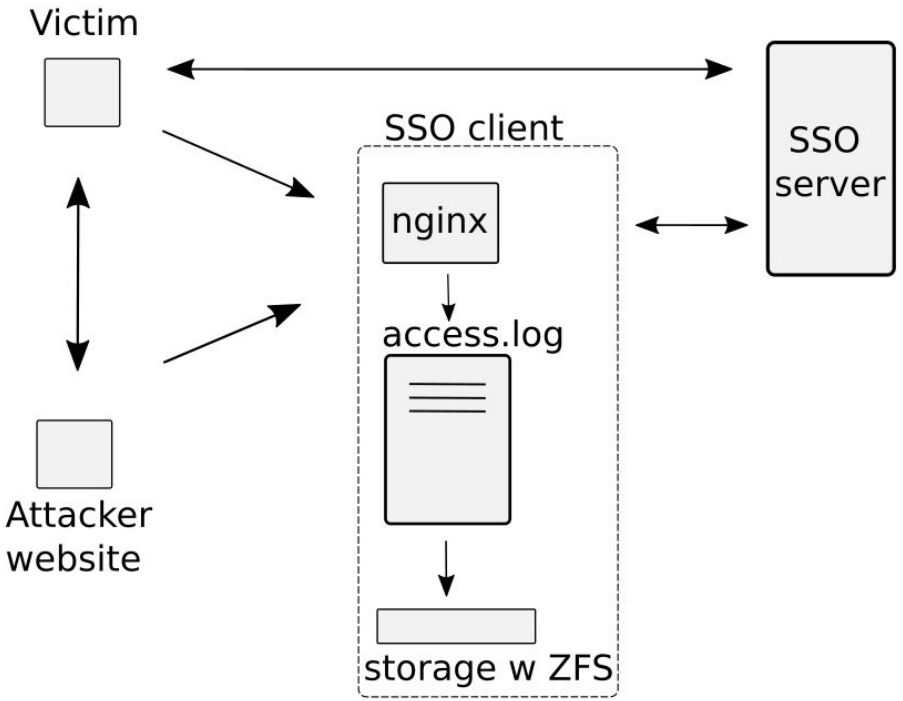
Data leak

- targets access tokens of OAuth 2.0 implicit grant access scheme

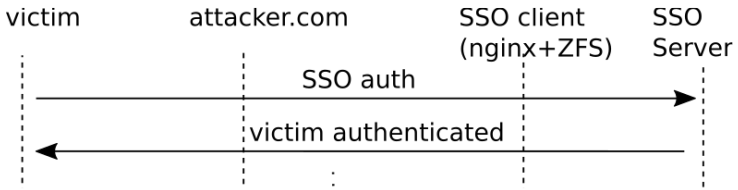
- SSO client runs nginx on top of ZFS and logs requests

- access tokens are encoded in the request and do not expire during attack

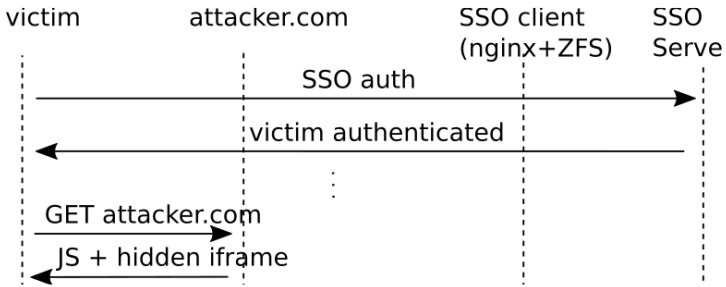
- SSO client does not offer X-Frame-Options



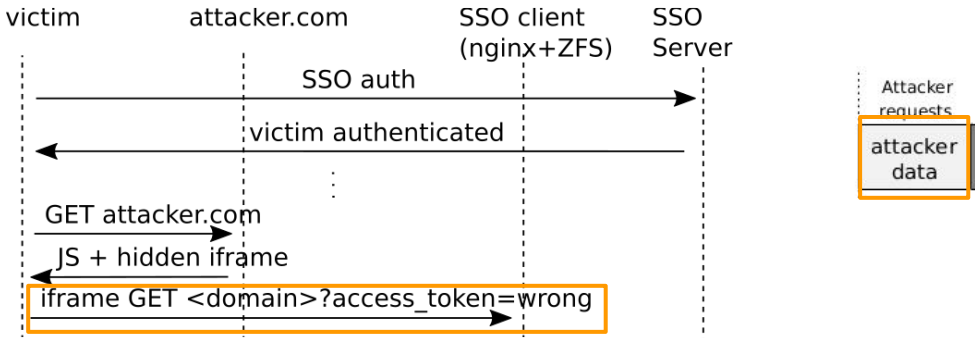
Data leak


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

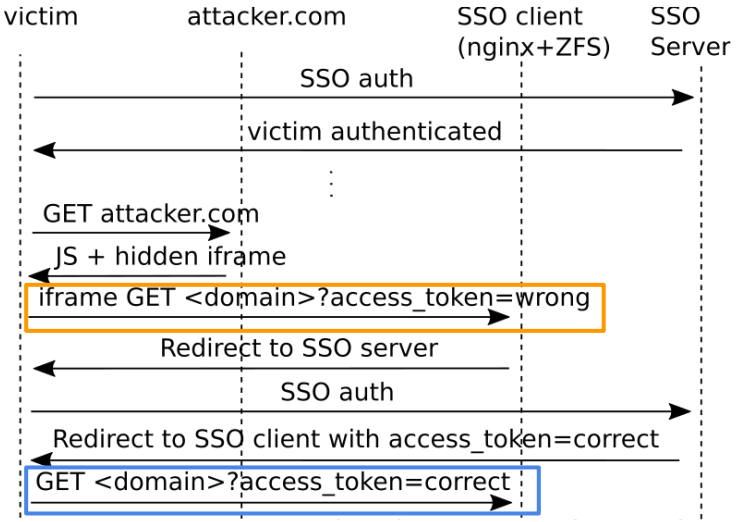
Data leak

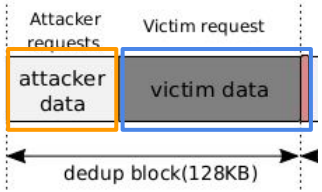

victim
attacker.com
SSO client
(nginx+ZFS)
SSO
Serve
SSO auth
victim authenticated
GET attacker.com
JS + hidden iframe

Data leak

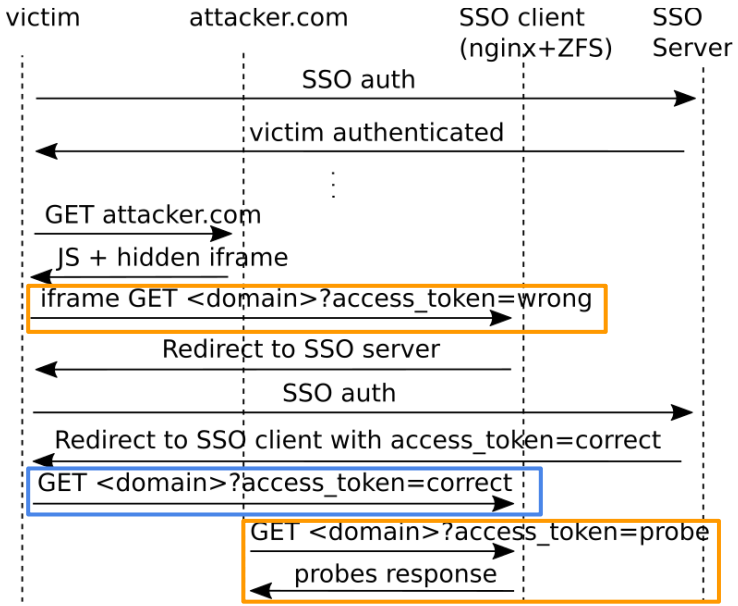

victim
attacker.com
SSO client
(nginx+ZFS)
SSO
Server
SSO auth
victim authenticated
GET attacker.com
JS + hidden iframe
iframe GET <domain>?access_token=wrong
Attacker
requests
attacker
data

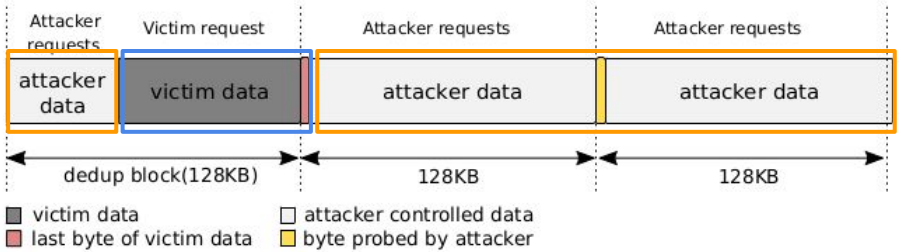
Data leak


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

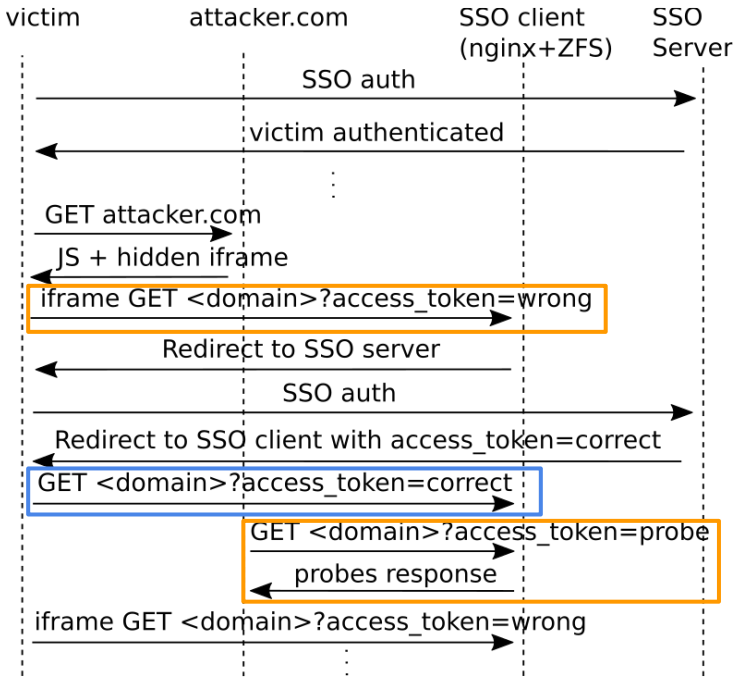

Attacker requests
Victim request
attacker data
victim data
dedup block(128KB)

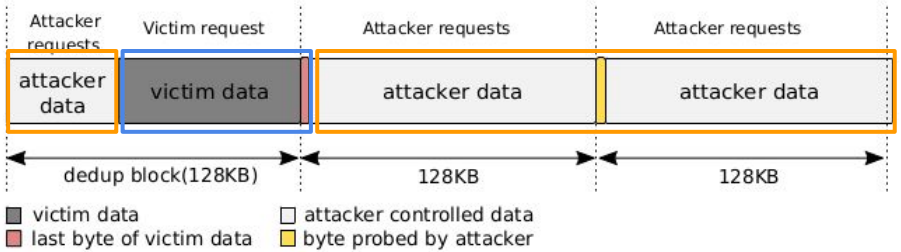
Data leak


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

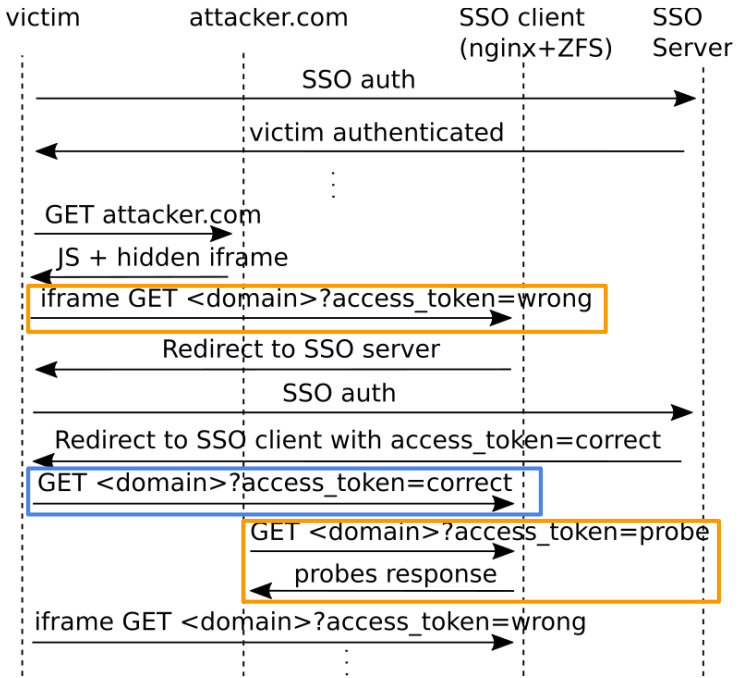

Attacker requests
Victim request
Attacker requests
Attacker requests
attacker data
victim data
attacker data
attacker data
dedup block(128KB)
128KB
128KB
victim data
last byte of victim data
attacker controlled data
byte probed by attacker

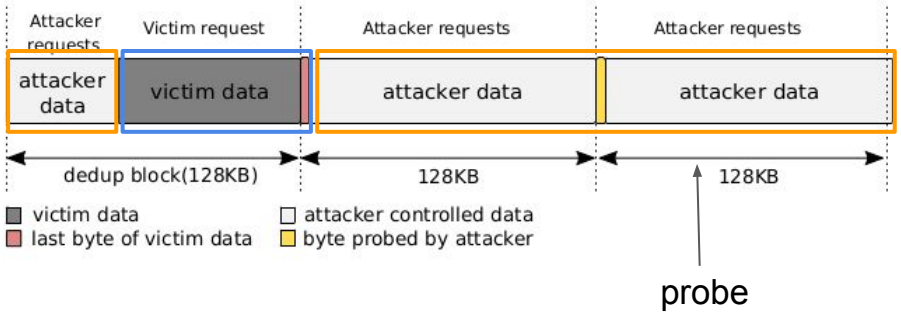
Data leak


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

Data leak

- SSO client: 4CPU, 16GB RAM, FreeBSD 10.4, ZFS and Nginx1.14

- ZIL: 10% RAM

- OAuth token 22 bytes (base64)

- LAN: 1 hop (RTT 0.1ms)

- WAN: 12 hops (RTT 2ms)

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

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
  - Renders remote attacks impractical

Conclusions

- Filesystem deduplication implementations introduce timing side channels that can be abused to leak, fingerprint or exfiltrate data

- Remote attacks leak data at byte granularity across the network

- Mitigation using a pseudo-same behavior policy without filesystem redesign

https://www.vusec.net

https://comsec.ethz.ch