ADVERSARY AND HARMONY,
THE EVOLUTION OF
AI SECURITY

搭配模糊測試對Linux核心SMB遠端檔案系統進行漏洞挖掘

Pumpkin 🎃 (@u1f383)

About

- DEVCORE 前實習生
- Security and Systems Lab (SENSE Lab) in NYCU
- CTF team TSJ 成員

Outline

- Background
- KSMBD
- Environment
- Result
- Conclusion

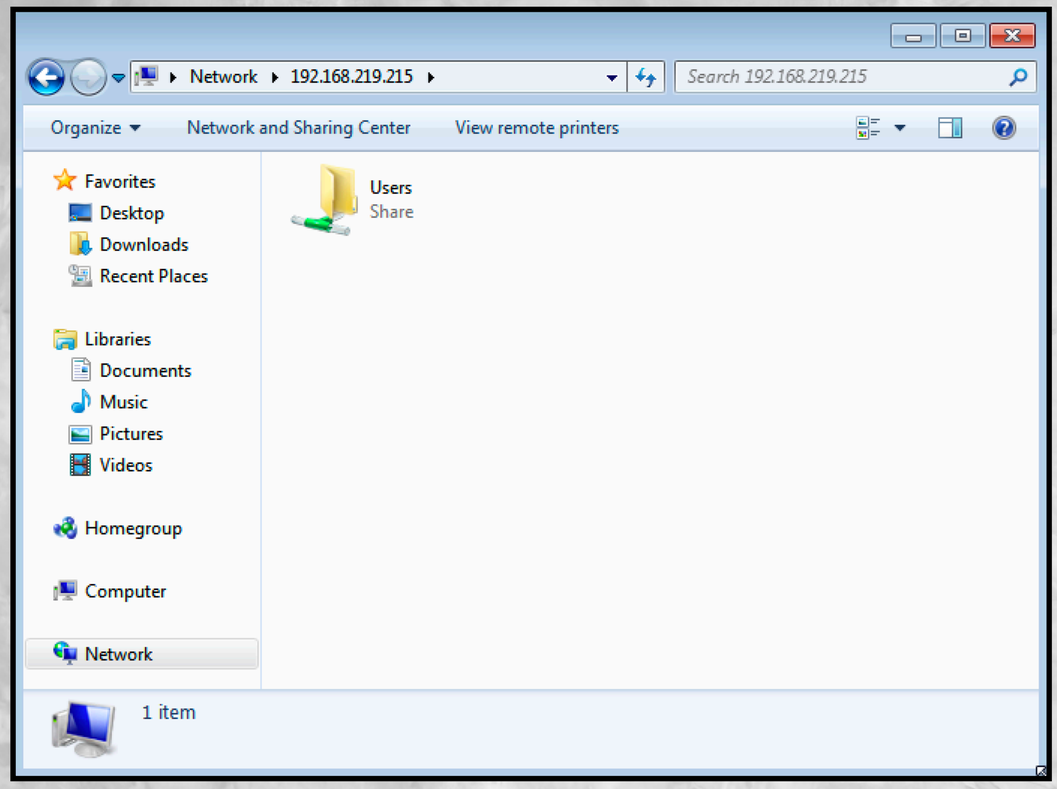
Background

Remote File System

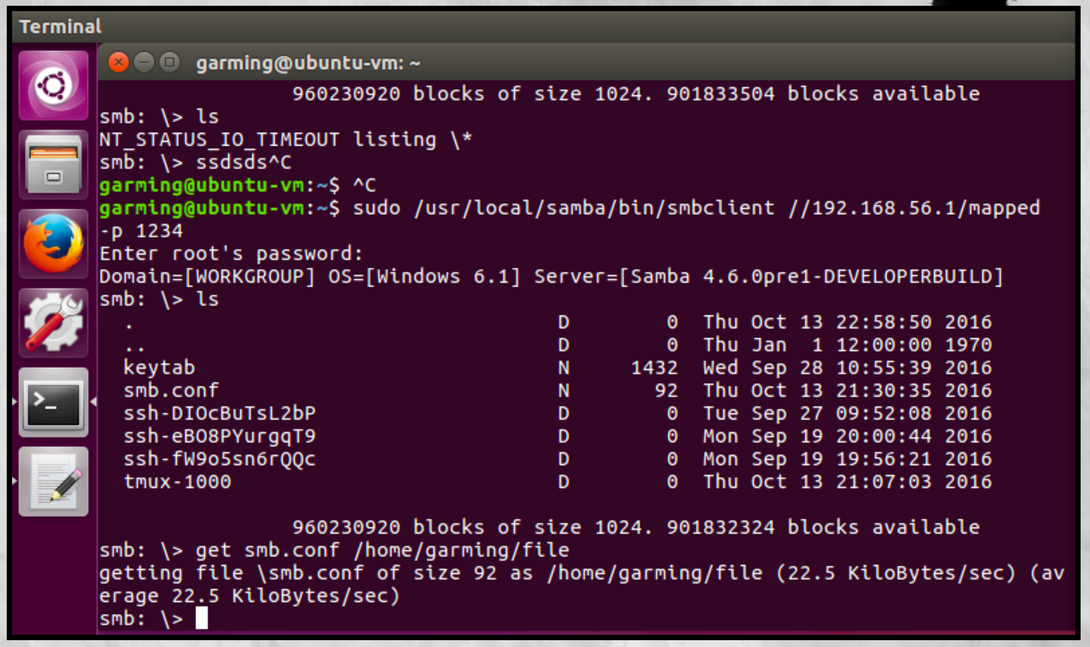
- 使用伺服器的硬碟來儲存檔案
- 透過網路傳輸來操作檔案
- 用戶端能以普通硬碟的方式來使用

- 常見應用：NAS、Windows 網路芳鄰
- 常見協議：FTP、NFS、SMB

Remote File System



Windows (圖1)



Linux (圖2)

圖1. WINDOWS SMB SHARES
圖2. samba-garming

Remote File System

- Linux kernel 實作了下列兩個協議的用戶端與服務端
  - SMB
    - Client - CIFS (Common Internet File System)
    - Server - KSMBD (Kernel SMB Daemon)
  - NFS
    - Client - NFS (Network File System)
    - Server - NFSD (NFS Daemon)

Remote File System

- Linux kernel 實作了下列兩個協議的用戶端與服務端
  - SMB
    - Client - CIFS (Common Internet File System)
    - Server - KSMBD (Kernel SMB Daemon)
  - NFS
    - Client - NFS (Network File System)
    - Server - NFSD (NFS Daemon)

Fuzzing101

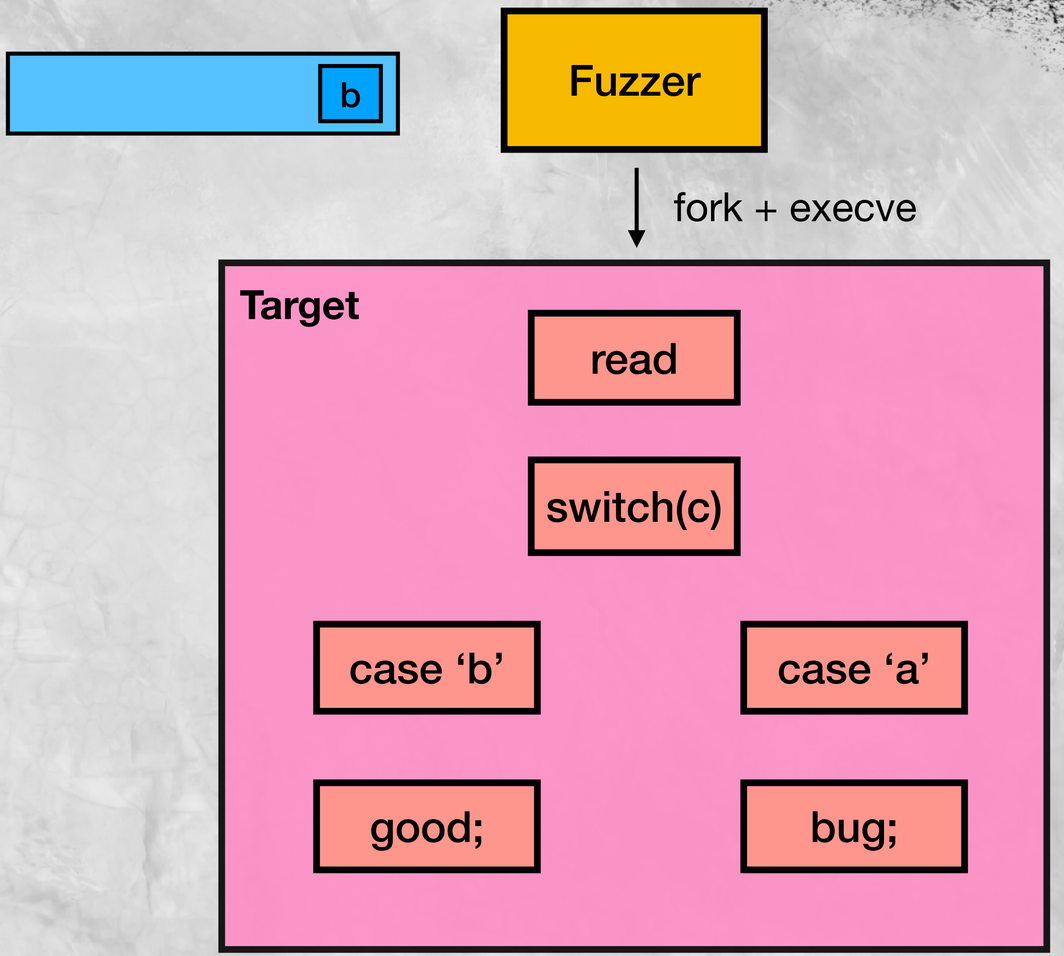
- 一種自動化找出程式漏洞的機制
- 根據對象不同，可以再細分為：
  - Userspace fuzzer，像是 AFL、AFL++
  - Kernel fuzzer，像是 syzkaller、kAFL
  - Hypervisor fuzzer，像是 hypercube、NYX

Fuzzing101

- 載入目標程式
- 產生隨機輸入
- 取得執行回饋
- 調整變異策略
- 觸發程式漏洞

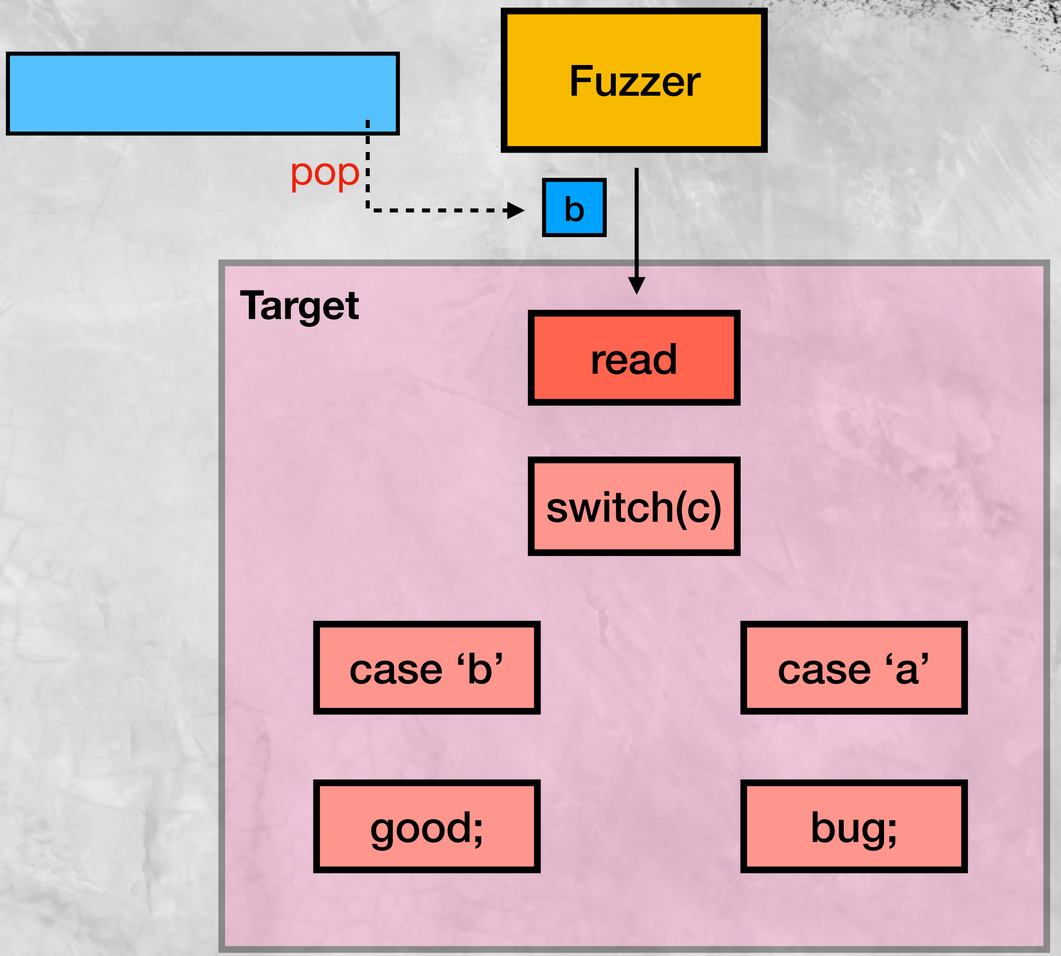
Fuzzing101

- 載入目標程式
- 產生隨機輸入
- 取得執行回饋
- 調整變異策略
- 觸發程式漏洞



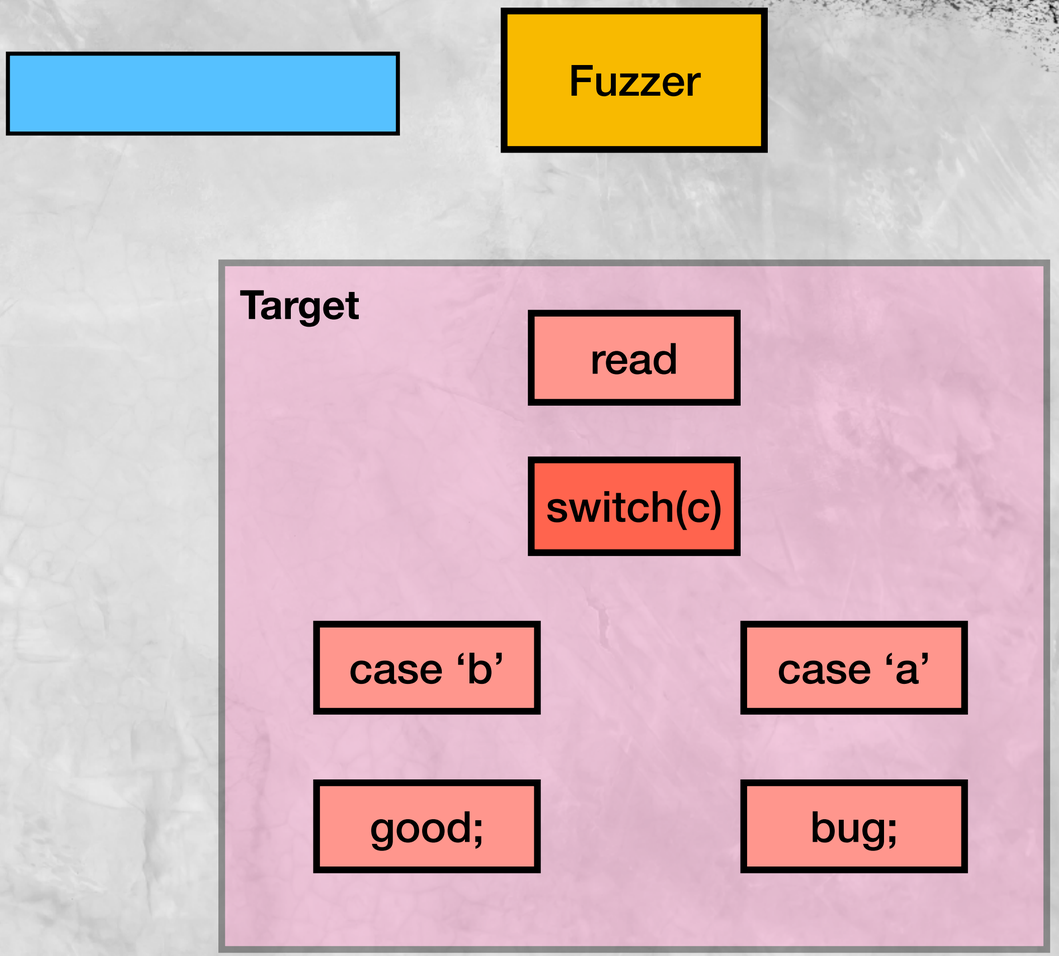
Fuzzing101

- 載入目標程式
- 產生隨機輸入
- 取得執行回饋
- 調整變異策略
- 觸發程式漏洞



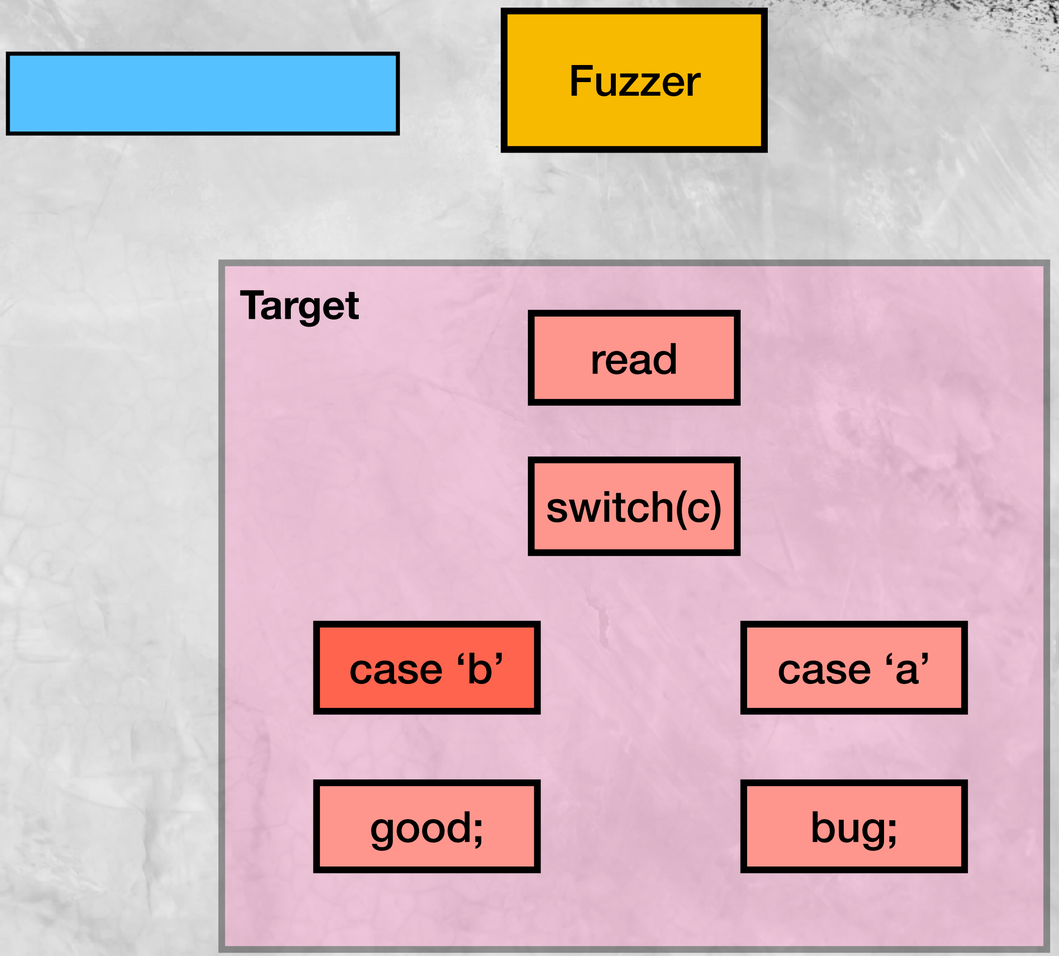
Fuzzing101

- 載入目標程式
- 產生隨機輸入
- 取得執行回饋
- 調整變異策略
- 觸發程式漏洞



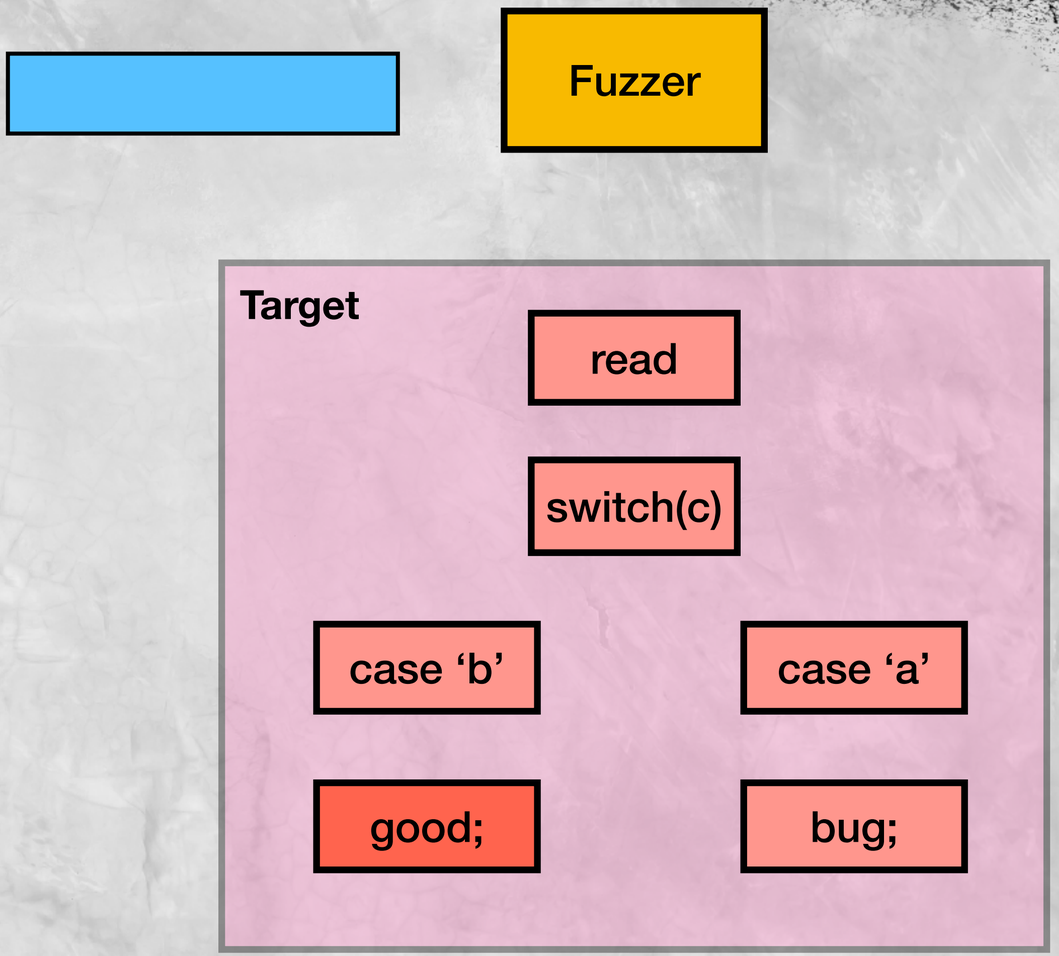
Fuzzing101

- 載入目標程式
- 產生隨機輸入
- 取得執行回饋
- 調整變異策略
- 觸發程式漏洞



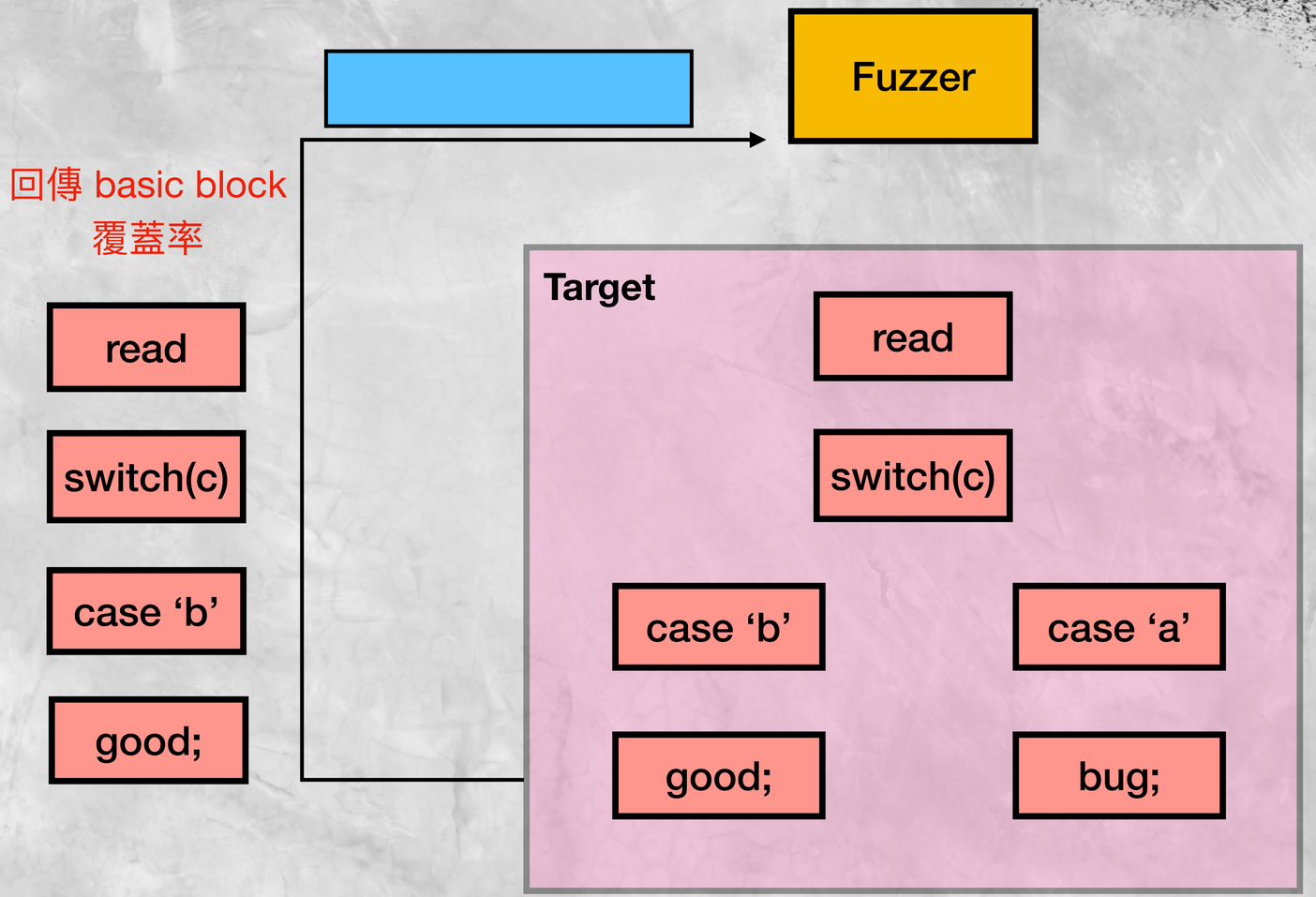
Fuzzing101

- 載入目標程式
- 產生隨機輸入
- 取得執行回饋
- 調整變異策略
- 觸發程式漏洞



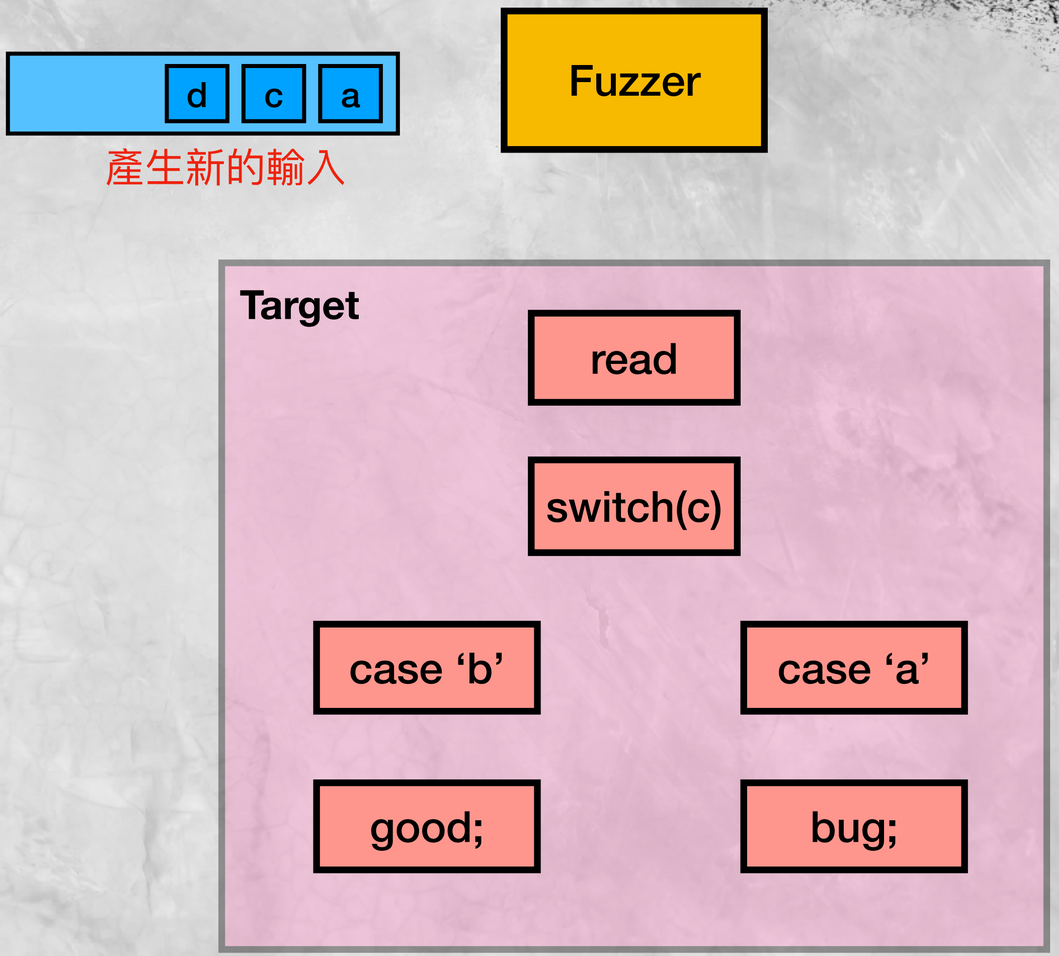
Fuzzing101

- 載入目標程式
- 產生隨機輸入
- 取得執行回饋
- 調整變異策略
- 觸發程式漏洞



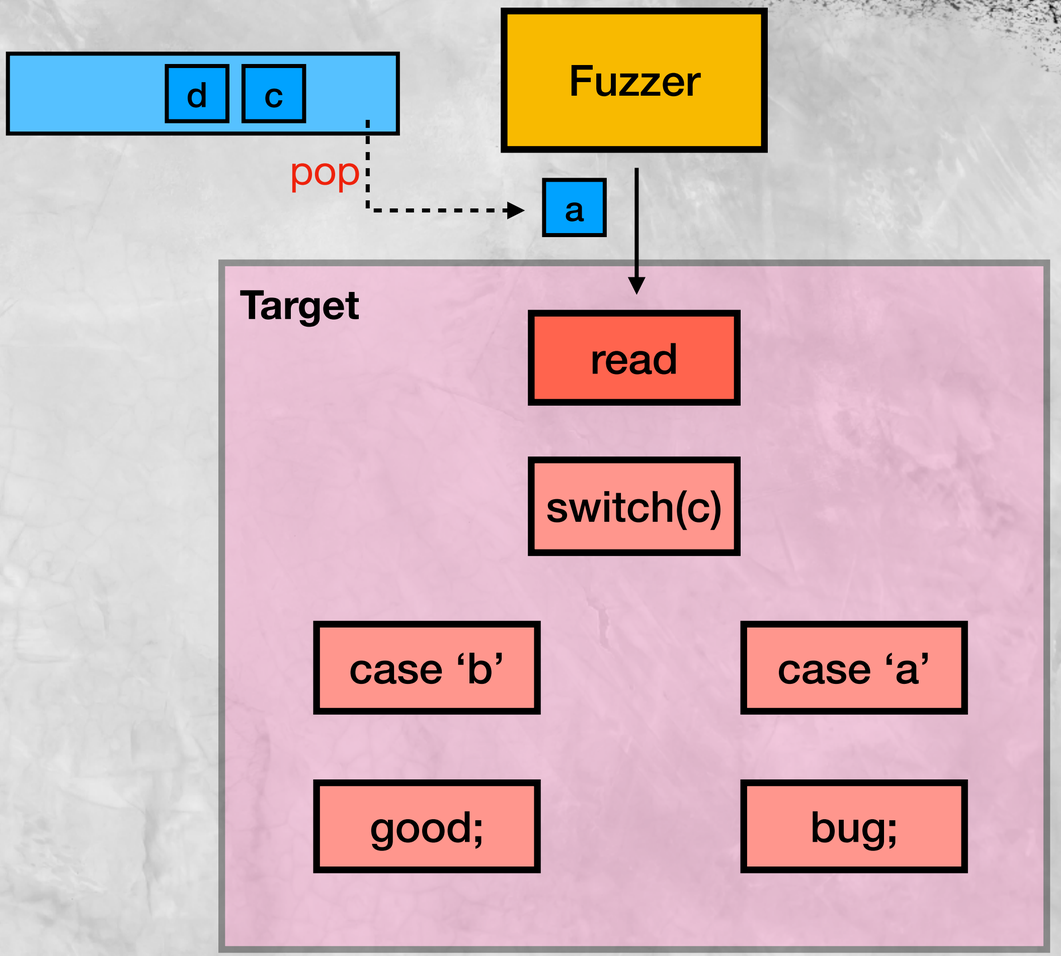
Fuzzing101

- 載入目標程式
- 產生隨機輸入
- 取得執行回饋
- 調整變異策略
- 觸發程式漏洞



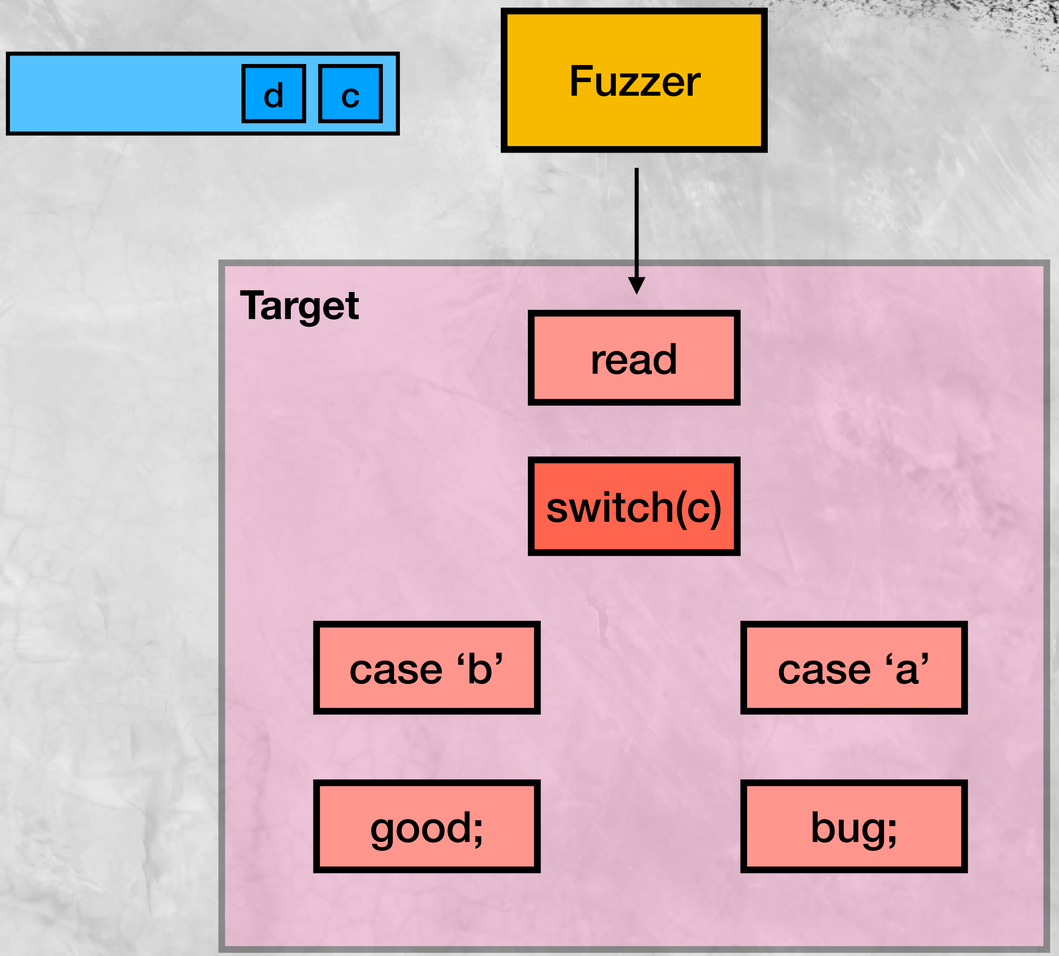
Fuzzing101

- 載入目標程式
- 產生隨機輸入
- 取得執行回饋
- 調整變異策略
- 觸發程式漏洞



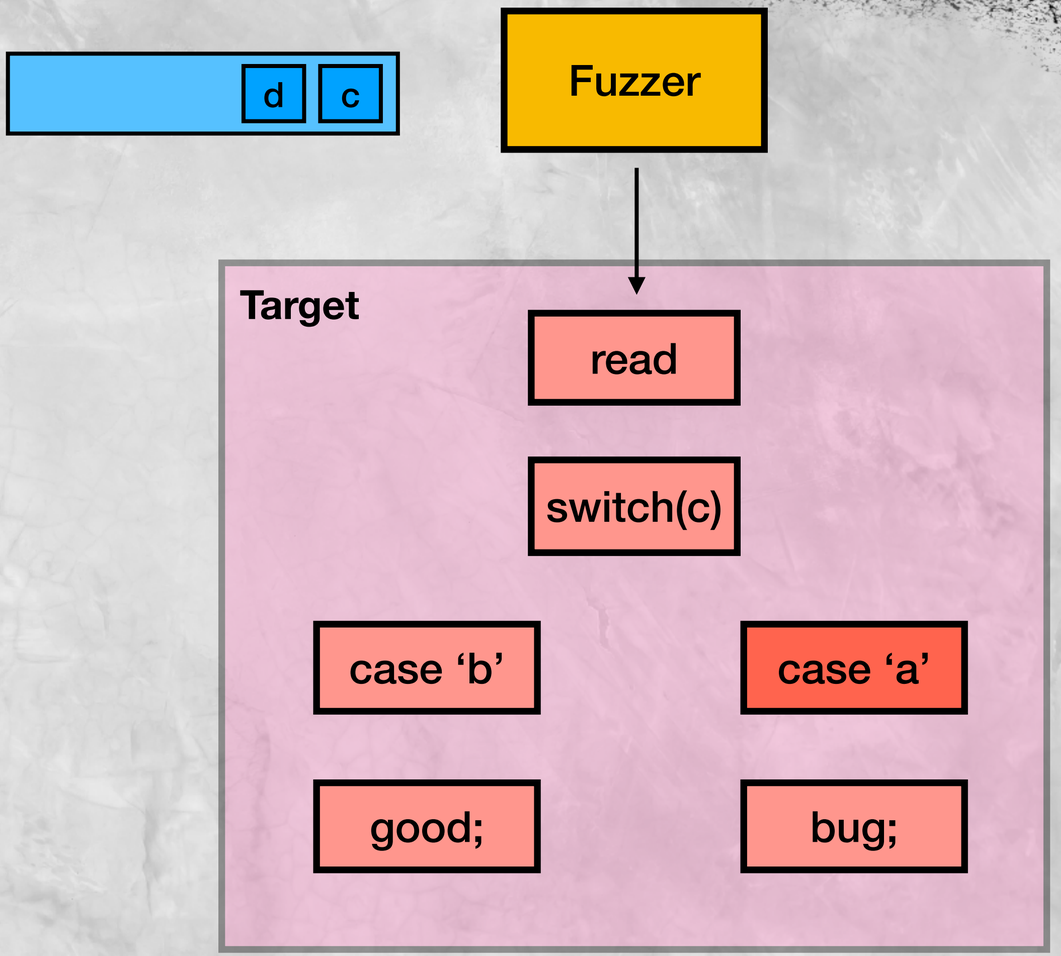
Fuzzing101

- 載入目標程式
- 產生隨機輸入
- 取得執行回饋
- 調整變異策略
- 觸發程式漏洞



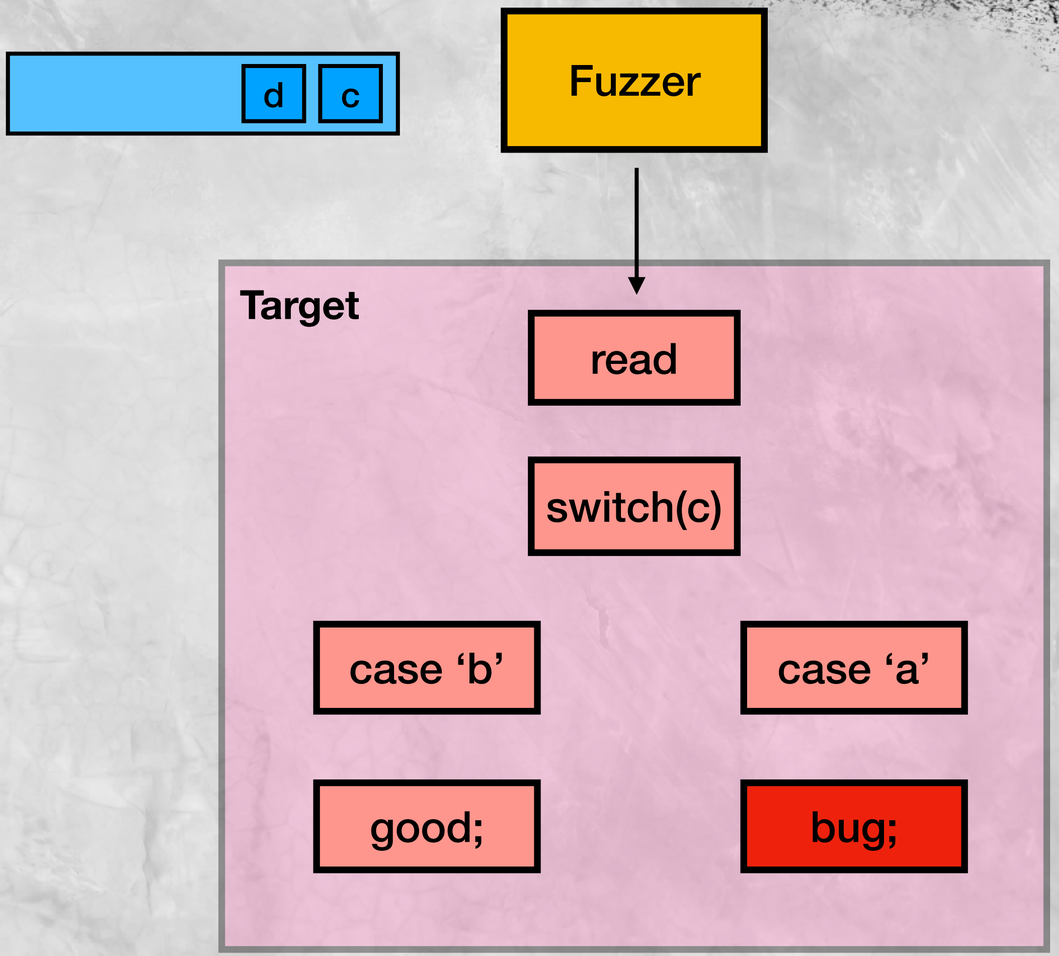
Fuzzing101

- 載入目標程式
- 產生隨機輸入
- 取得執行回饋
- 調整變異策略
- 觸發程式漏洞



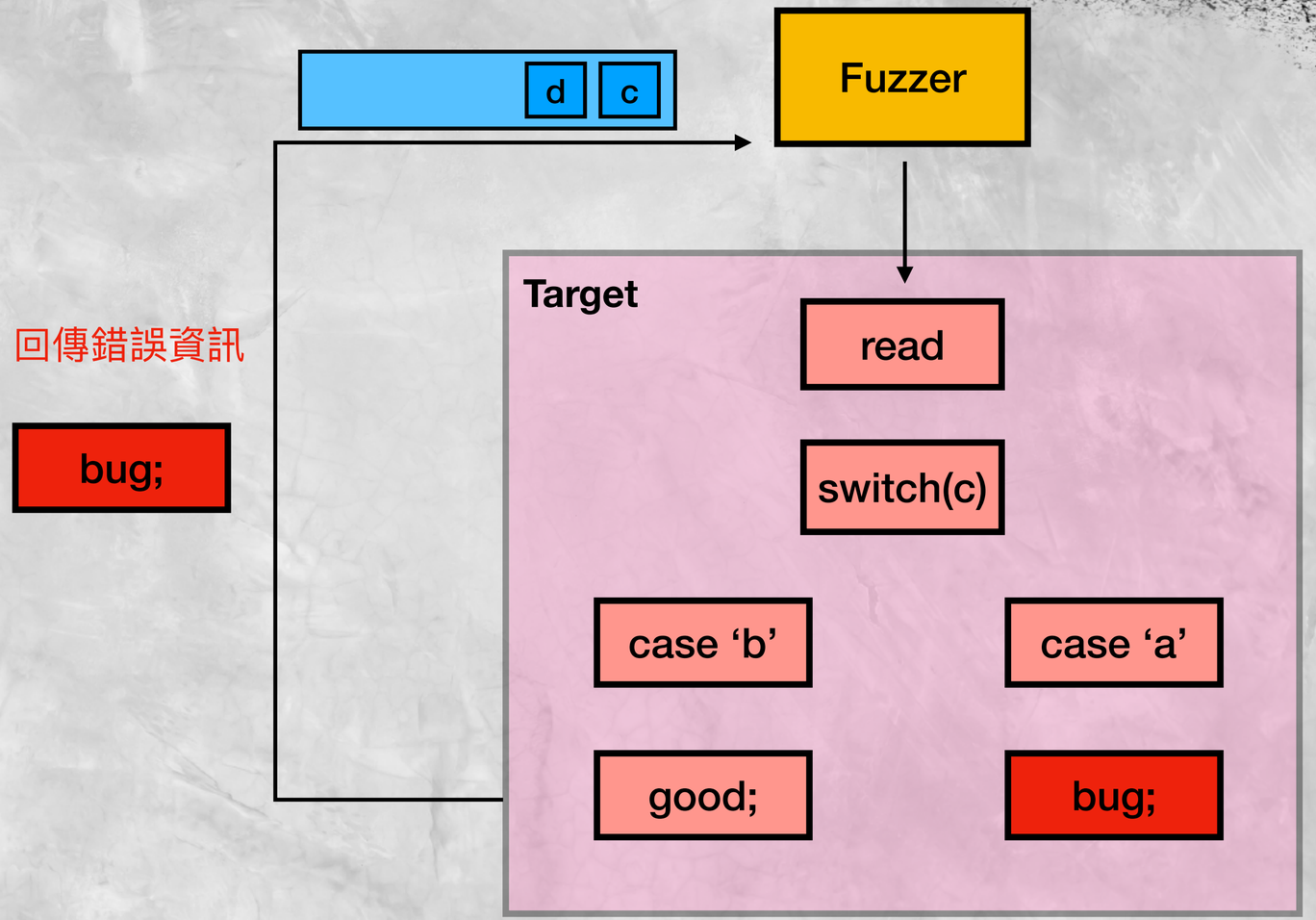
Fuzzing101

- 載入目標程式
- 產生隨機輸入
- 取得執行回饋
- 調整變異策略
- 觸發程式漏洞



Fuzzing101

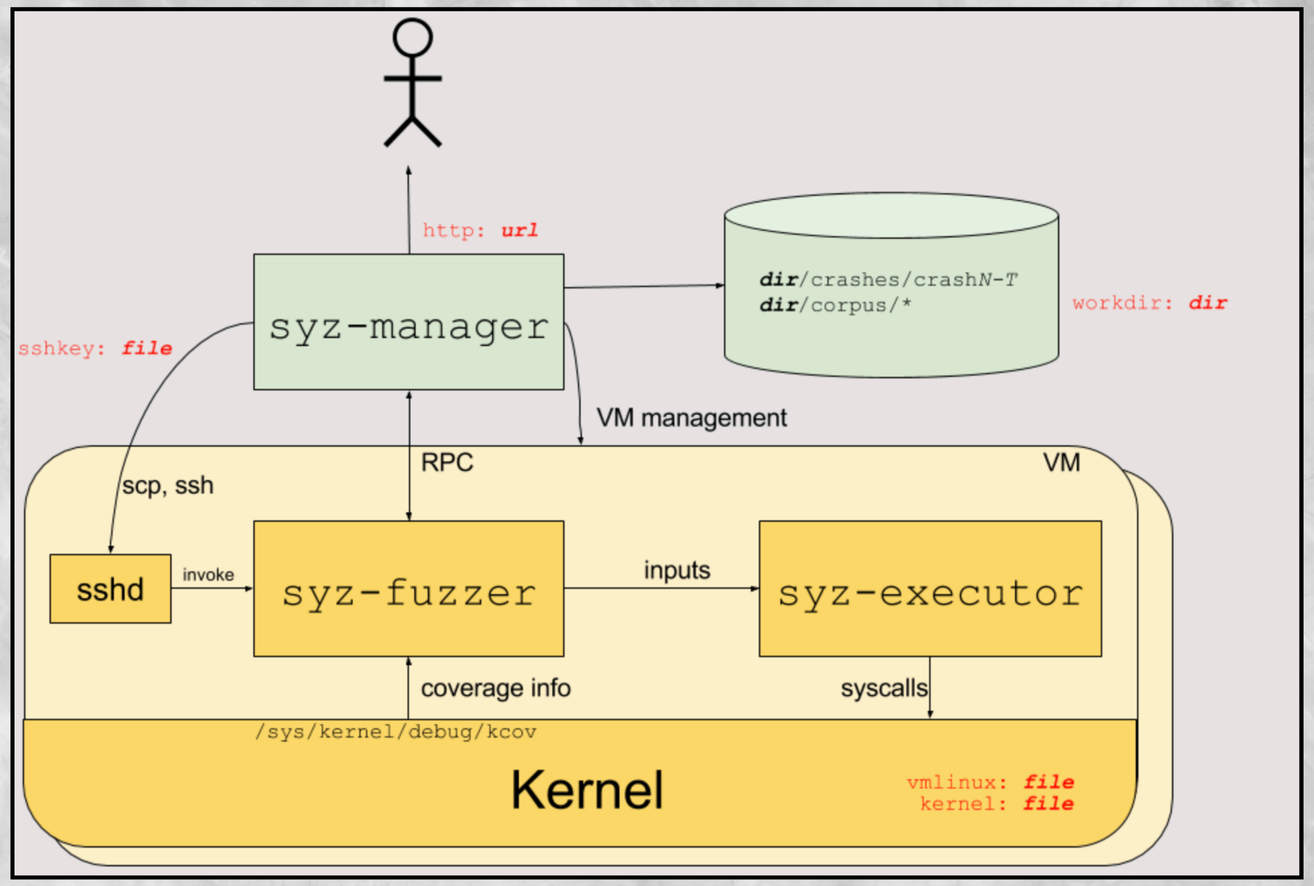
- 載入目標程式
- 產生隨機輸入
- 取得執行回饋
- 調整變異策略
- 觸發程式漏洞



Syzkaller

- 以系統呼叫參數與順序為變異對象的 kernel fuzzer，並具有以下特色：
  - 提供友善的 GUI
  - 簡單使用並且方便架設
  - 使用者能客製化部分的測試方法

Syzkaller



圖片來源：[syzkaller github repo]

Syzkaller

- sys-manager - 執行於 host，主要用來執行 VM
- sys-fuzzer - 負責產生 syscall sequence，並根據覆蓋率調整測試策略
- sys-executor - 負責執行 syscall sequence

Syzkaller

- Syzlang
  - 在 syzkaller 架構中被用來定義 system call 格式的語言
  - 可以根據不同子系統呼叫做客製化的描述

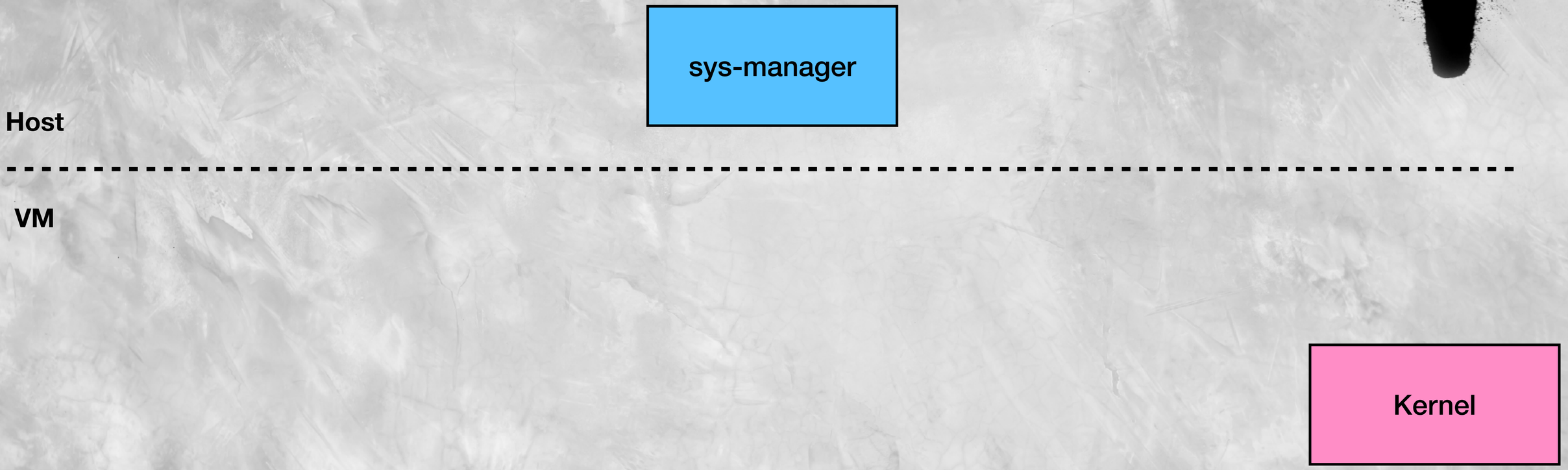
```
open(file filename, flags flags[open_flags], mode flags[open_mode]) fd
read(fd fd, buf buffer[out], count len[buf])
close(fd fd)
open_mode = S_IRUSR, S_IWUSR, S_IXUSR, S_IRGRP, S_IWGRP, S_IXGRP, S_IROTH, S_IWOTH, S_IXOTH
```

Syzkaller

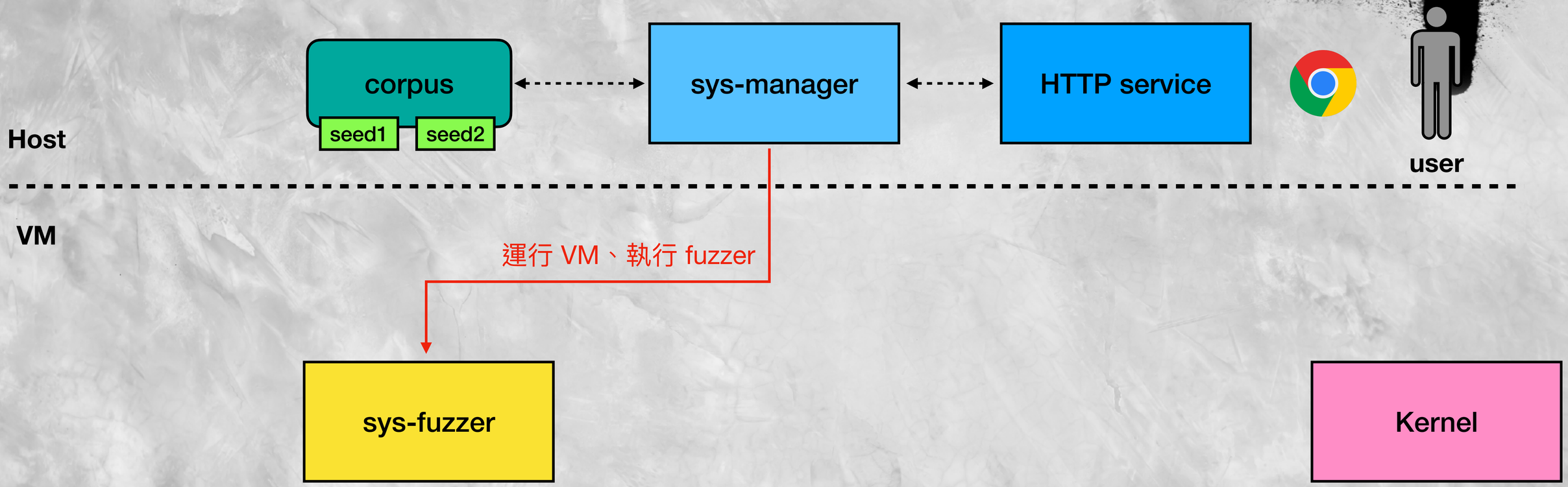
- Program
  - 描述經過動態處理後的形式，定義了 syscall sequence
  - 此種形式作為 seed 存放在 corpus
  - 每行都定義了一次 syscall 的呼叫，此外還能保存回傳值並繼續使用

```
r0 = syz_open_dev$loop(&(0x7f00000011c0), 0x0, 0x0)
r1 = openat$6lowpan_control(0xffffffffffffff9c, &(0x7f00000000c0), 0x2, 0x0)
ioctl$LOOP_SET_FD(r0, 0x4c00, r1)
```

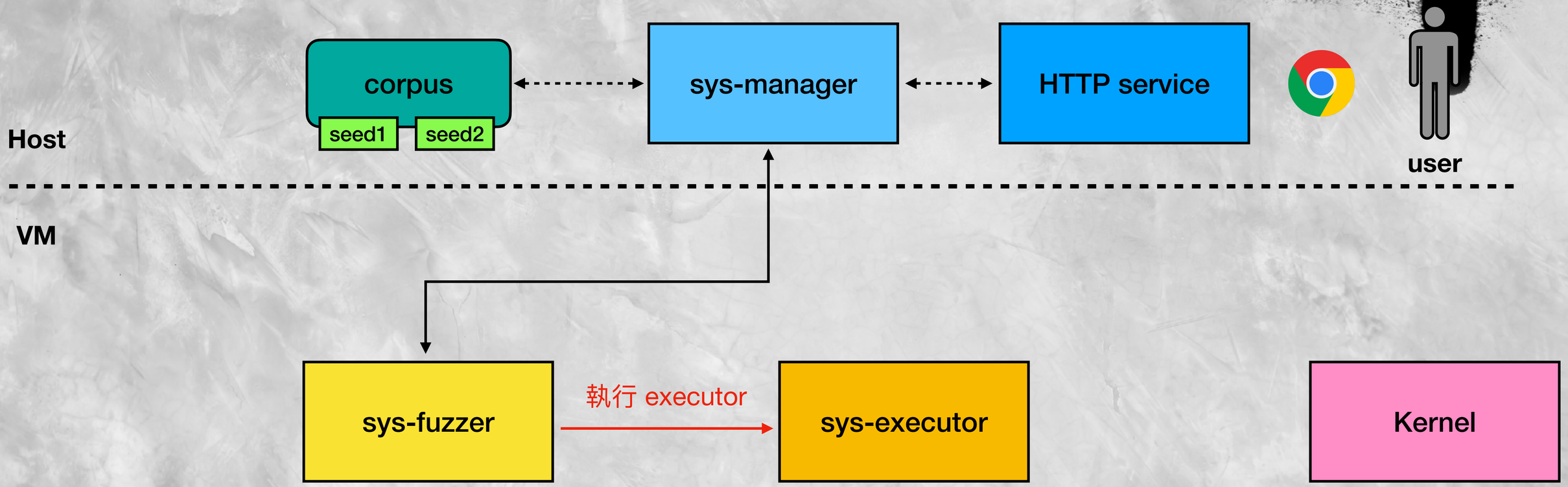
Syzkaller



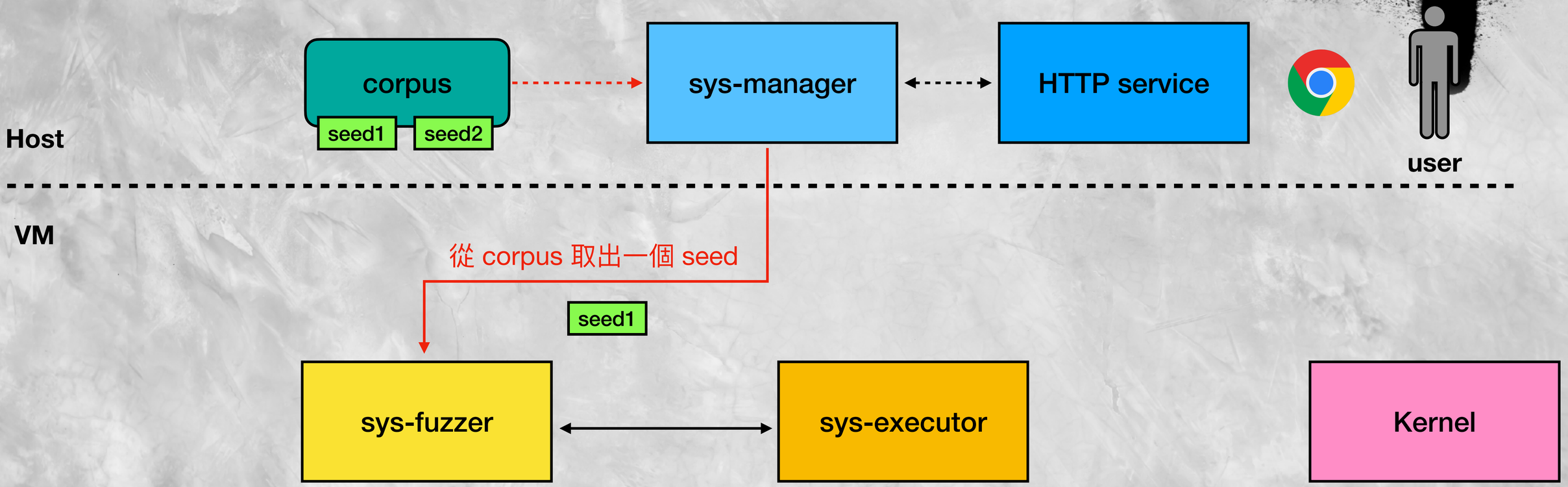
Syzkaller


corpus
seed1
seed2
sys-manager
HTTP service
user
Host
VM
運行 VM、執行 fuzzer
sys-fuzzer
Kernel

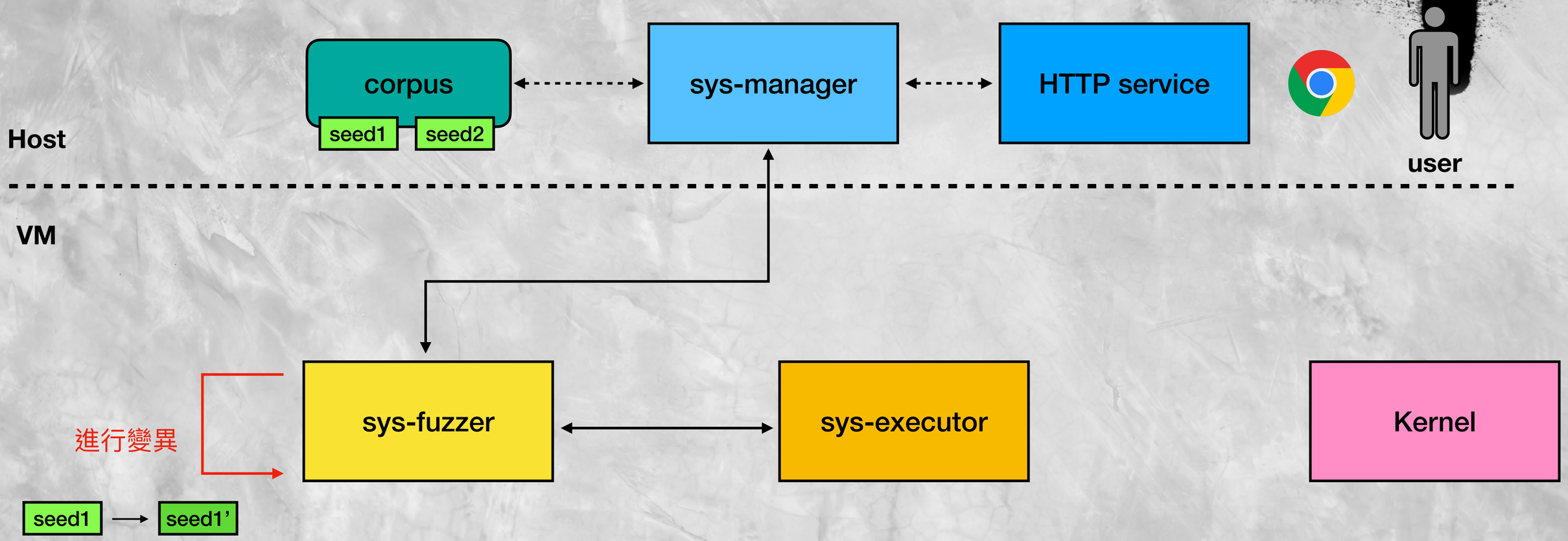
Syzkaller


corpus
seed1
seed2
sys-manager
HTTP service
user
Host
VM
sys-fuzzer
執行 executor
sys-executor
Kernel

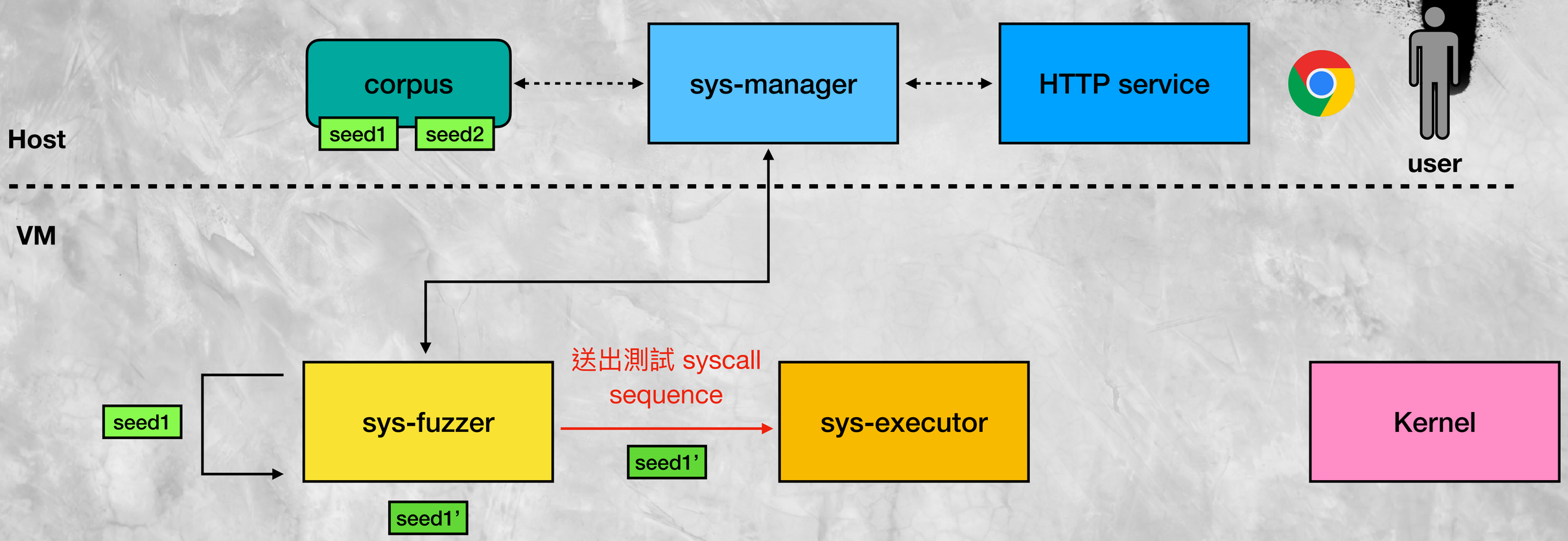
Syzkaller


corpus
seed1
seed2
sys-manager
HTTP service
user
Host
VM
從 corpus 取出一個 seed
seed1
sys-fuzzer
sys-executor
Kernel

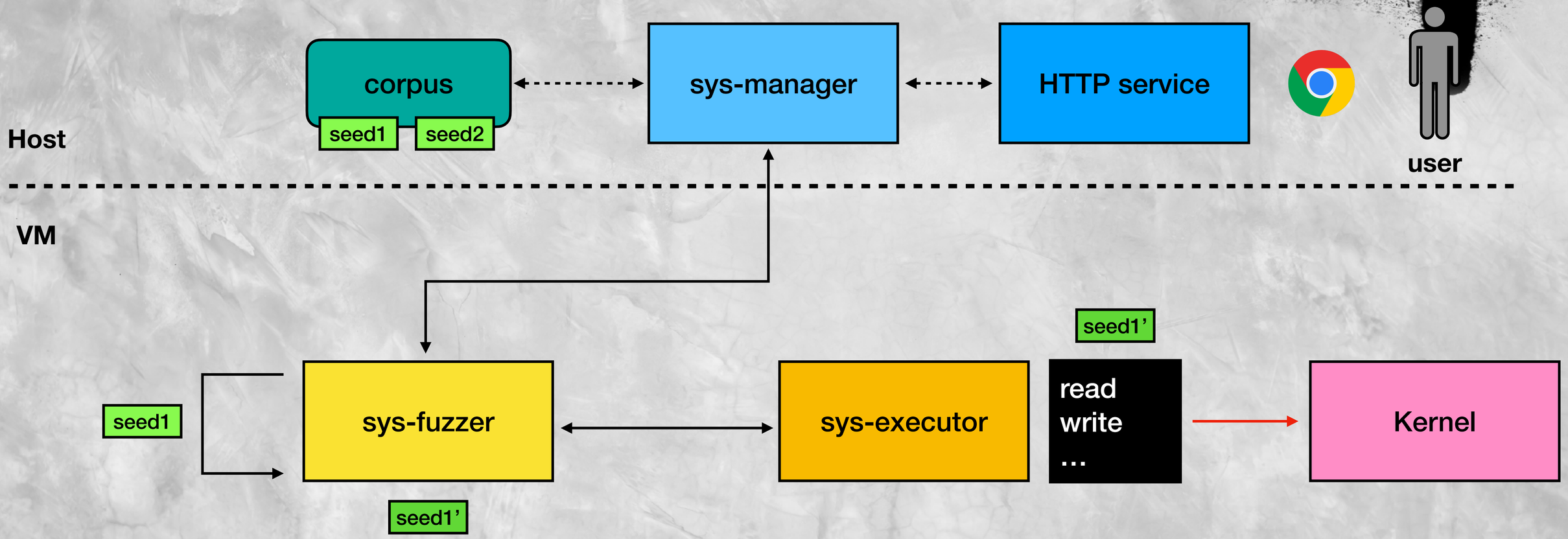
Syzkaller


corpus
seed1
seed2
sys-manager
HTTP service
user
Host
VM
sys-fuzzer
sys-executor
Kernel
進行變異
seed1
seed1'

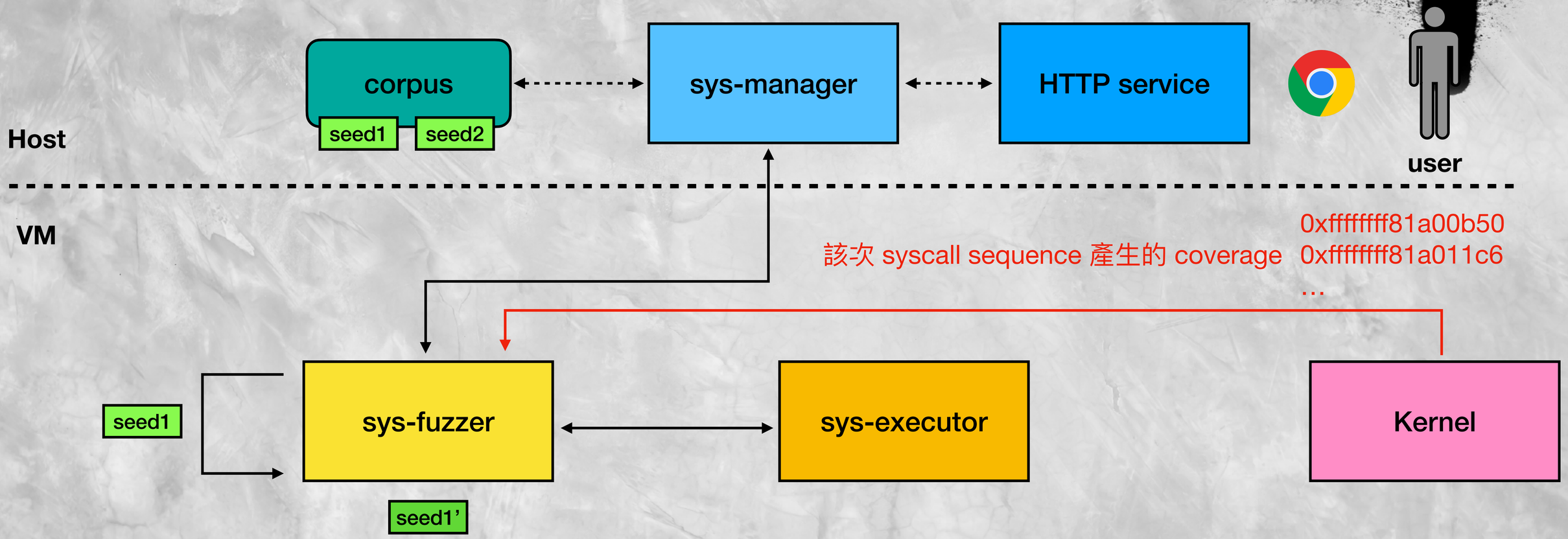
Syzkaller


corpus
seed1
seed2
sys-manager
HTTP service
user
Host
VM
seed1
sys-fuzzer
seed1'
送出測試 syscall sequence
seed1'
sys-executor
Kernel

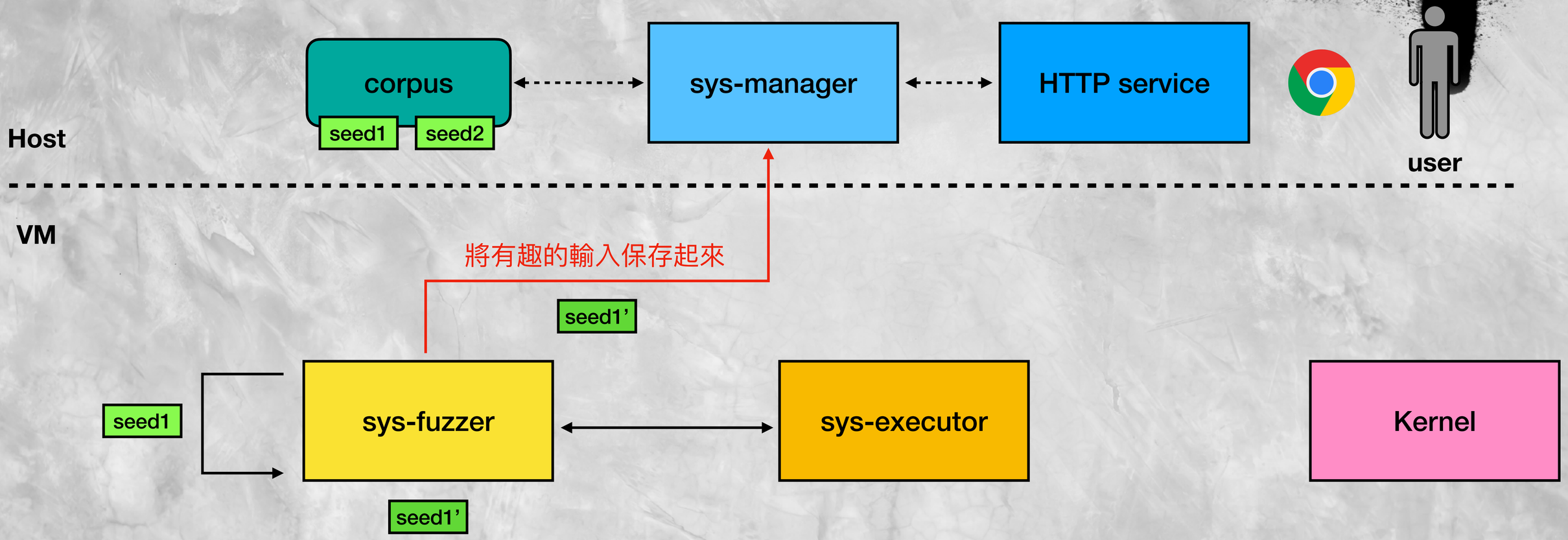
Syzkaller


corpus
seed1
seed2
sys-manager
HTTP service
user
Host
VM
seed1
sys-fuzzer
seed1'
seed1'
sys-executor
read
write
...
Kernel

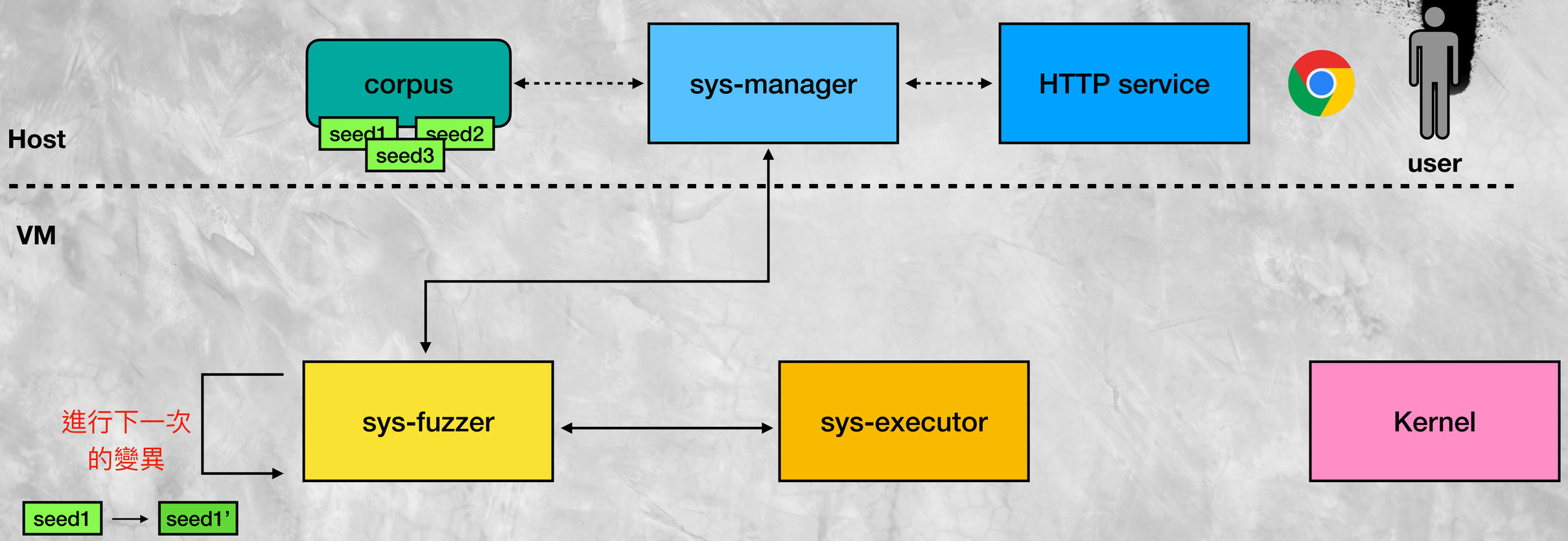
Syzkaller


corpus
seed1
seed2
sys-manager
HTTP service
user
Host
VM
該次 syscall sequence 產生的 coverage
0xffffffff81a00b50
0xffffffff81a011c6
...
seed1
sys-fuzzer
seed1'
sys-executor
Kernel

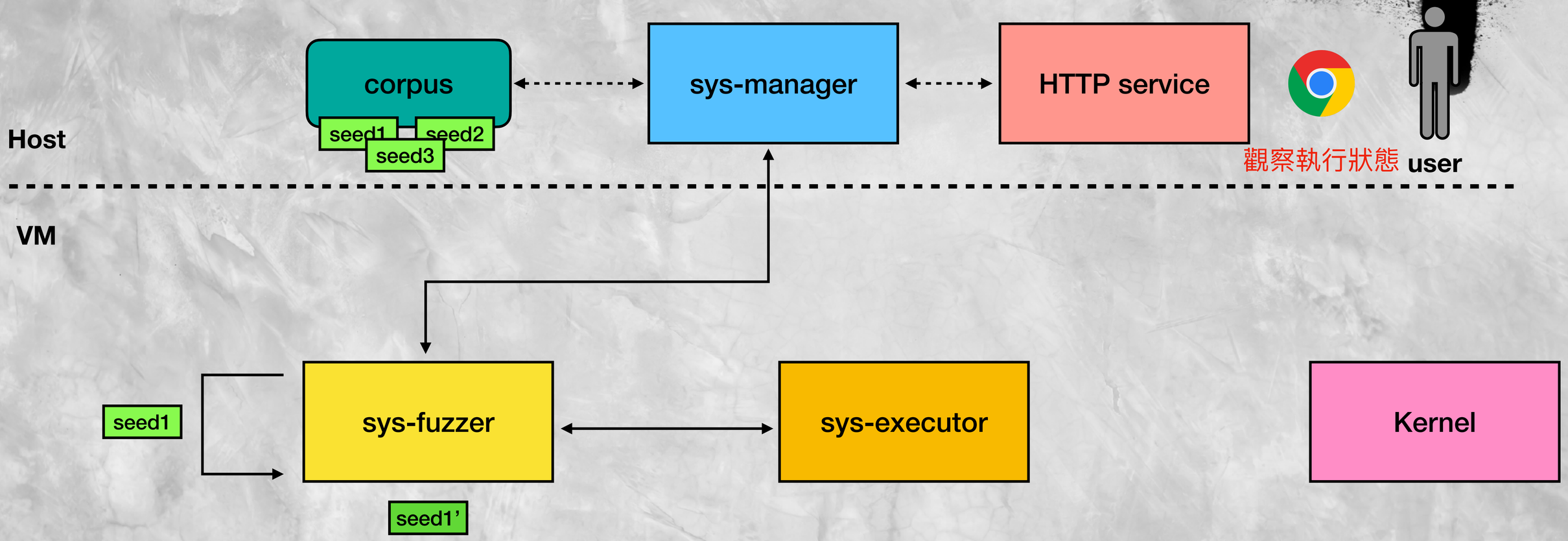
Syzkaller


corpus
seed1
seed2
sys-manager
HTTP service
user
Host
VM
將有趣的輸入保存起來
seed1'
seed1
sys-fuzzer
sys-executor
Kernel
seed1'

Syzkaller


corpus
seed1
seed2
seed3
sys-manager
HTTP service
user
Host
VM
sys-fuzzer
sys-executor
Kernel
進行下一次
的變異
seed1
seed1'

Syzkaller


corpus
seed1
seed2
seed3
sys-manager
HTTP service
Host
VM
觀察執行狀態
user
seed1
sys-fuzzer
sys-executor
Kernel
seed1'

Syzkaller

Stats:

revision	bd8b2775
config	config
uptime	61h31m45s
fuzzing	243h45m0s
corpus	19635
triage queue	0
signal	341349
coverage	130102
syscalls	2215
crash types	9 (0/hour)
crashes	19 (0/hour)
exec candidate	42062 (11/min)
exec fuzz	86044 (23/min)
exec gen	2412 (39/hour)
exec hints	176319 (47/min)
exec minimize	387629 (105/min)
exec seeds	1876 (30/hour)
exec smash	140560 (38/min)
exec total	1064521 (288/min)
exec triage	227619 (61/min)
executor restarts	3228 (52/hour)
max signal	423271 (114/min)
new inputs	40796 (11/min)
rotated inputs	4398 (71/hour)
suppressed	1 (0/hour)
vm restarts	255 (4/hour)

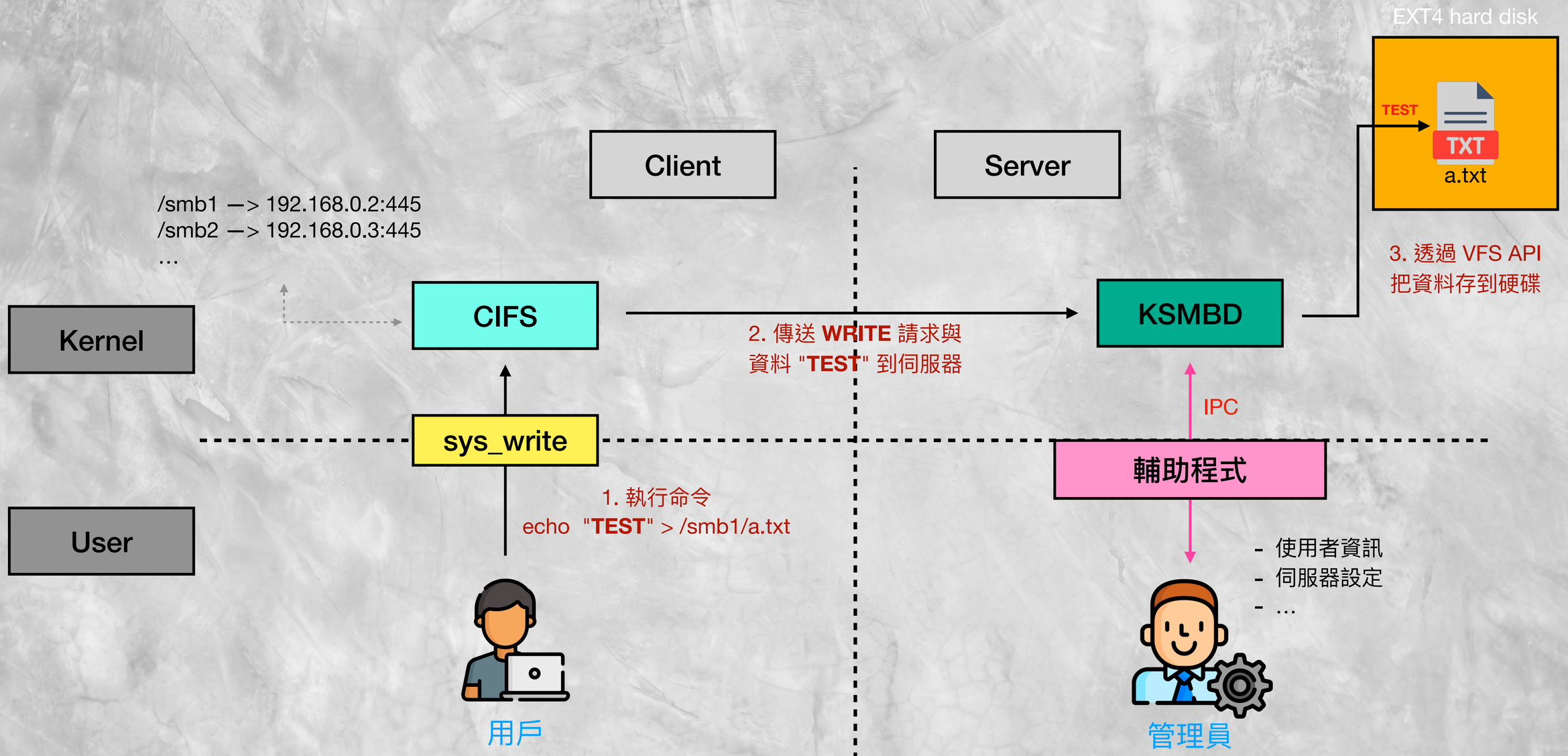
syzkaller 的執行狀態統計

Syzkaller

Description	Count	Last Time	Report
BUG: leak checking failed	309	2023/07/09 19:28	
BUG: soft lockup in __hrtimer_run_queues	1	2022/10/14 09:46	
BUG: soft lockup in corrupted	1	2023/01/12 09:04	
BUG: soft lockup in do_idle	1	2023/06/23 05:04	
BUG: soft lockup in full_proxy_read	1	2022/01/07 16:01	
BUG: soft lockup in full_proxy_write	5	2022/08/23 00:41	
BUG: soft lockup in ieee80211_tasklet_handler	5	2022/12/02 08:41	
BUG: soft lockup in inet_sendmsg	1	2021/07/22 10:01	

核心錯誤發生時的相關資訊

KSMBD


EXT4 hard disk
TEST
TXT
a.txt
Client
Server
/smb1 —> 192.168.0.2:445
/smb2 —> 192.168.0.3:445
...
Kernel
CIFS
KSMBD
3. 透過 VFS API
把資料存到硬碟
2. 傳送 WRITE 請求與
資料 "TEST" 到伺服器
IPC
sys_write
輔助程式
User
1. 執行命令
echo "TEST" > /smb1/a.txt
- 使用者資訊
- 伺服器設定
- ...
用戶
管理員

Overview

- KSMBD 是 Linux kernel SMB server 的實作
- 由於是較新的子系統，目前只支援 SMB 協議的部分功能
- 接下來會以兩個面向來介紹：
  - Connection setup (建立連線)
  - File operation (檔案操作)

Connection Setup

- SMB 是以帳號密碼做為認證方式
- 建立連線的過程中，伺服器會發送 session ID 來辨別不同的用戶端
- 連線完，伺服器將對應的 session ID 設置為已認證

```
root@syzkaller:/# cat /usr/local/etc/ksmbd/ksmbd.conf
[MyShare]
        path = /root/MyShare
        read only = no
root@syzkaller:/#
```

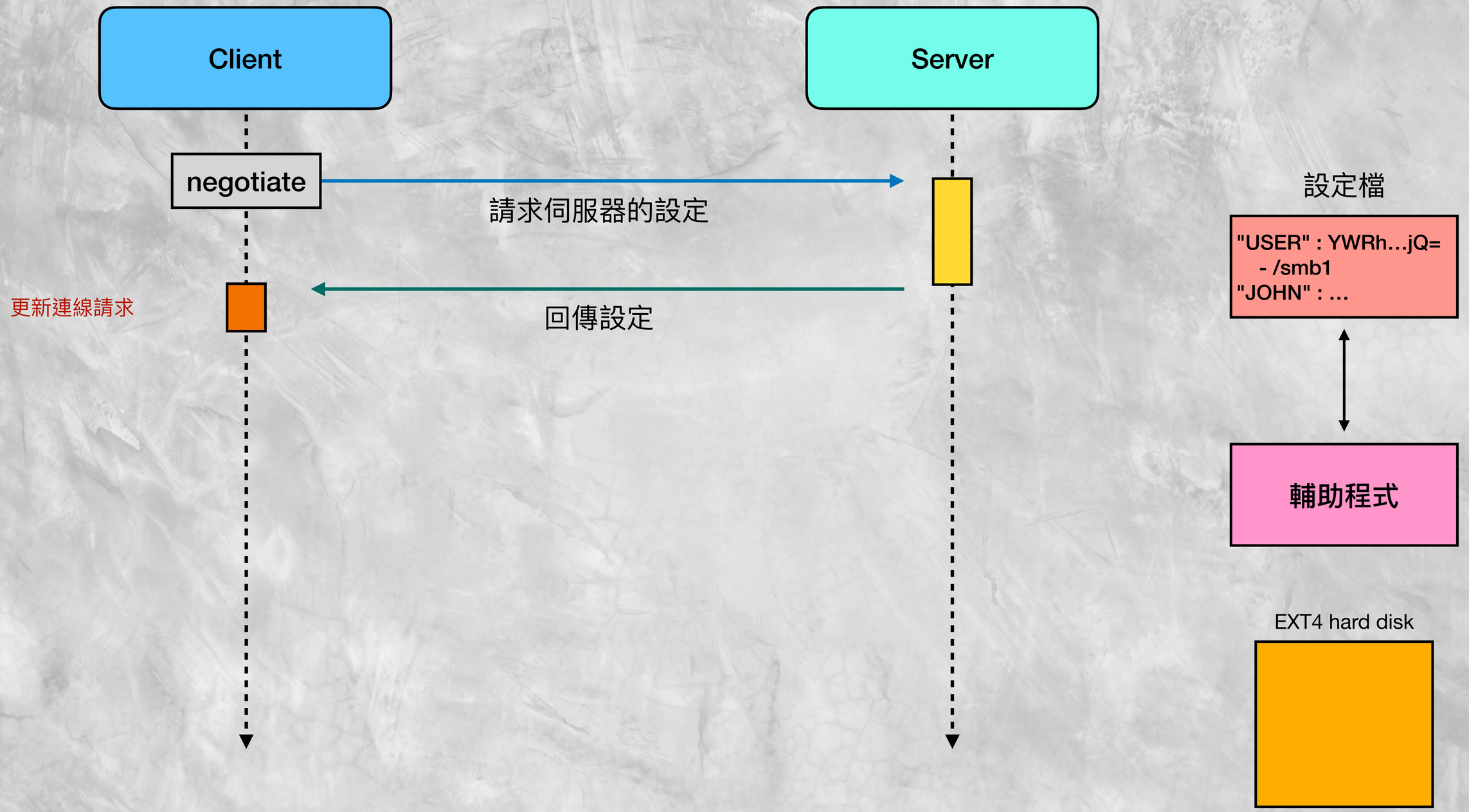
伺服器的用戶端掛載設定

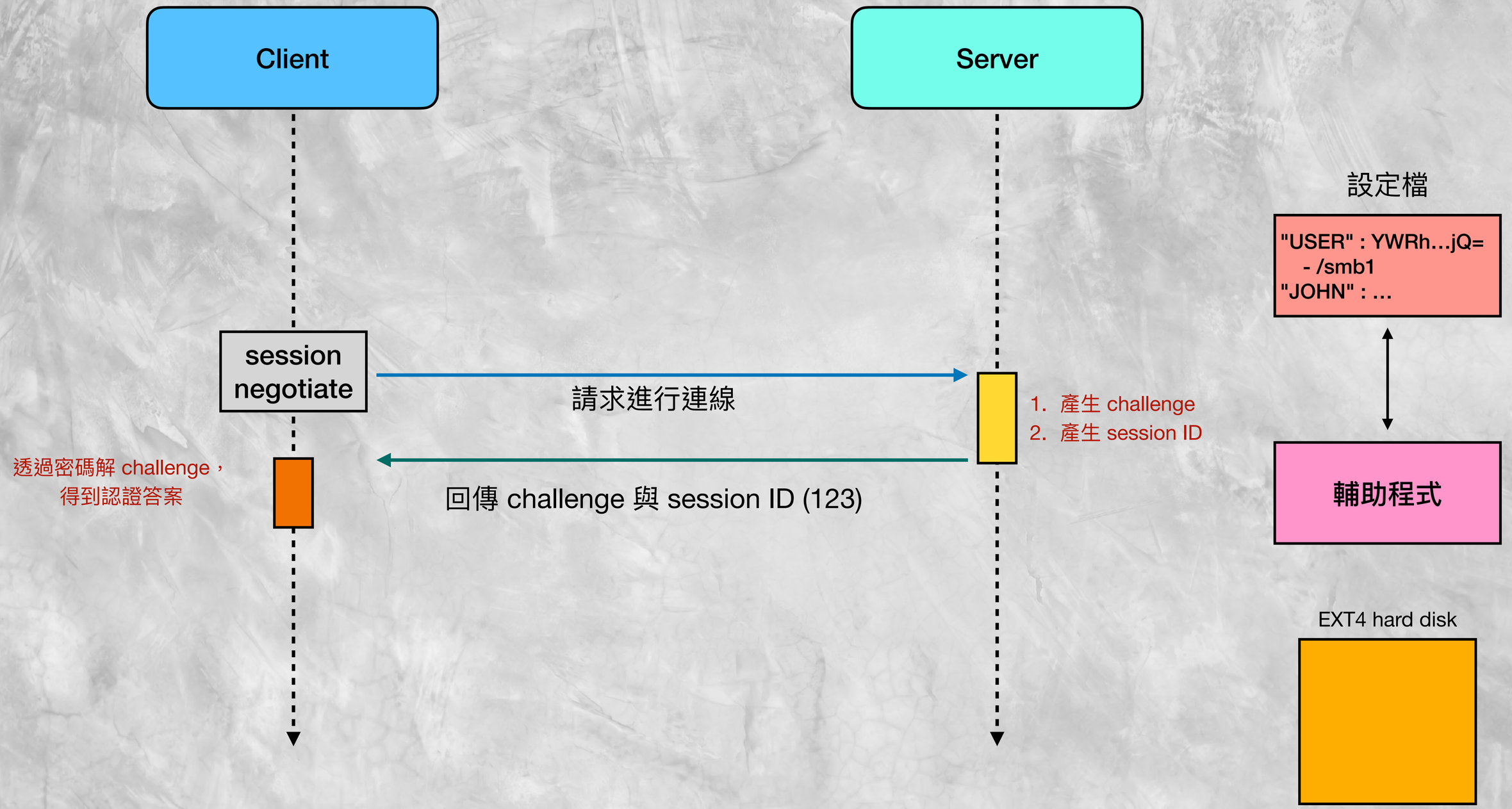
```
root@syzkaller:/# cat /usr/local/etc/ksmbd/ksmbdpwd.db
root:MpFT9WDrMpwOHe6lXoih6Q==
```

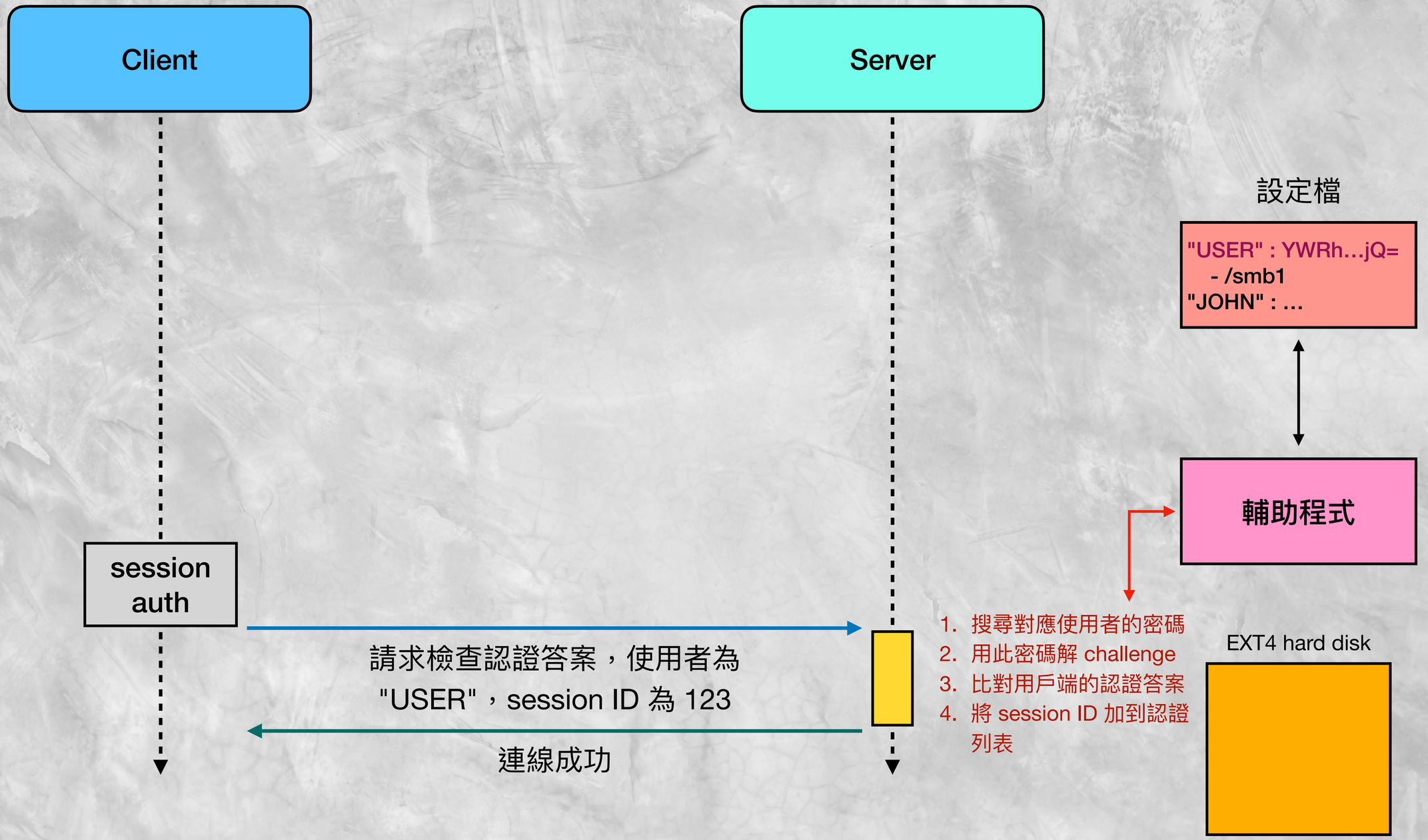
伺服器的密碼紀錄檔

```
root@syzkaller:/# ksmbd.mountd
root@syzkaller:/# mount -o user=root,password=root //127.0.0.1/MyShare /mnt
root@syzkaller:/# ls -al /mnt
total 4
drwxr-xr-x.  2 root root    0 Nov  3 09:50 .
drwxr-xr-x. 22 root root 4096 Oct 30 06:14 ..
root@syzkaller:/#
```

用戶端掛載

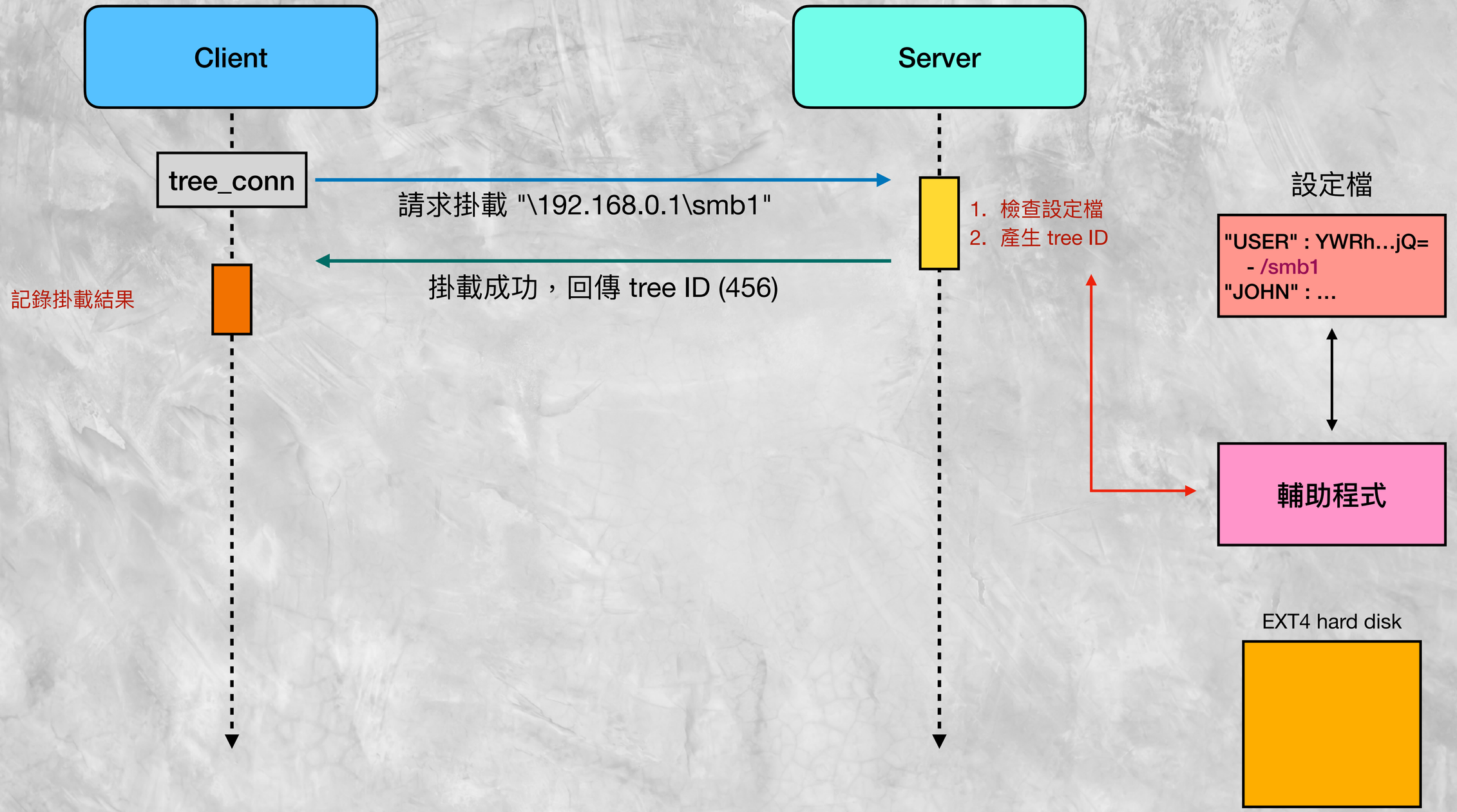

Client
Server
negotiate
請求伺服器的設定
更新連線請求
回傳設定
設定檔
"USER" : YWRh…jQ=
- /smb1
"JOHN" : …
輔助程式
EXT4 hard disk

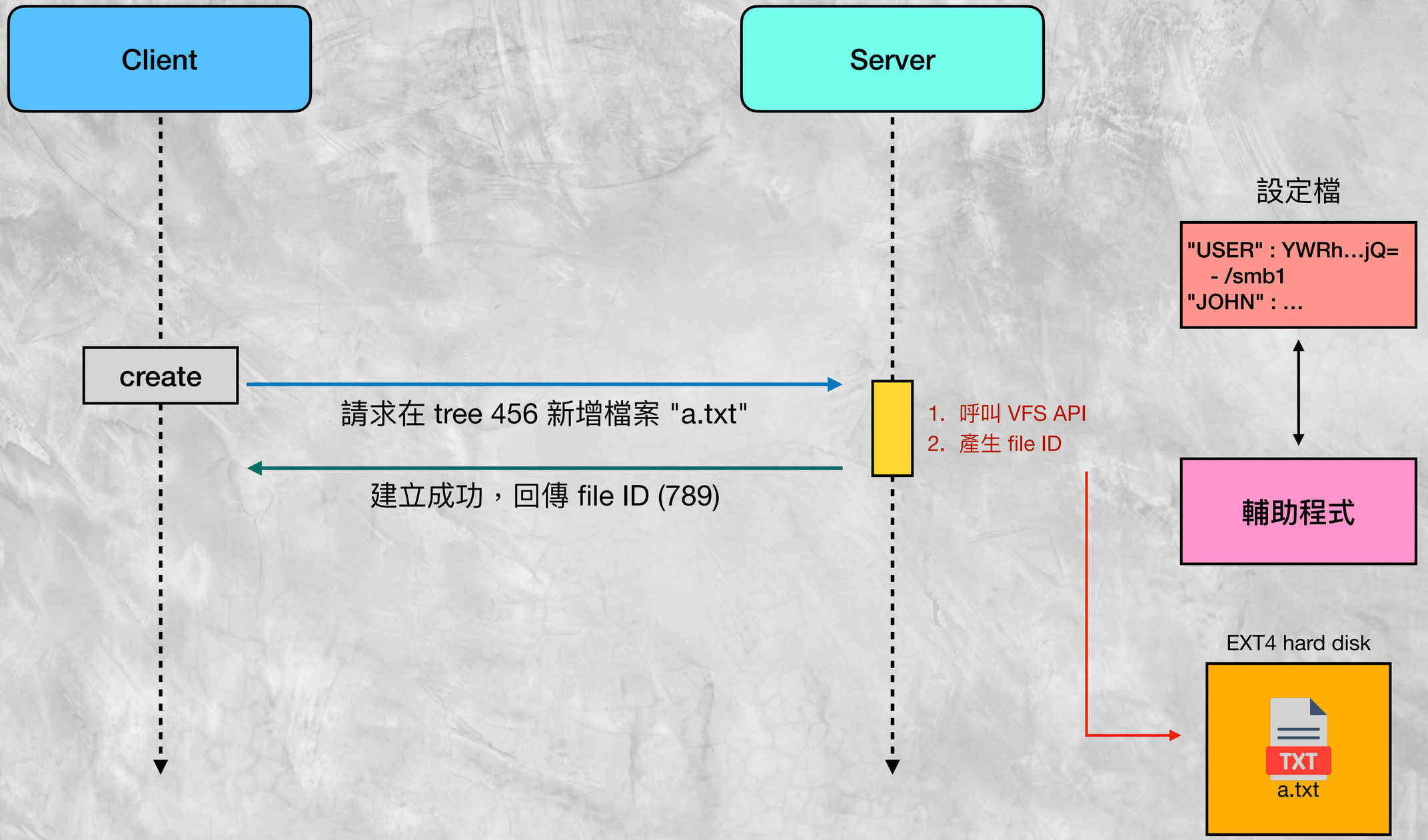

Client
Server
session negotiate
請求進行連線
1. 產生 challenge
2. 產生 session ID
透過密碼解 challenge，
得到認證答案
回傳 challenge 與 session ID (123)
設定檔
"USER" : YWRh…jQ=
- /smb1
"JOHN" : …
輔助程式
EXT4 hard disk

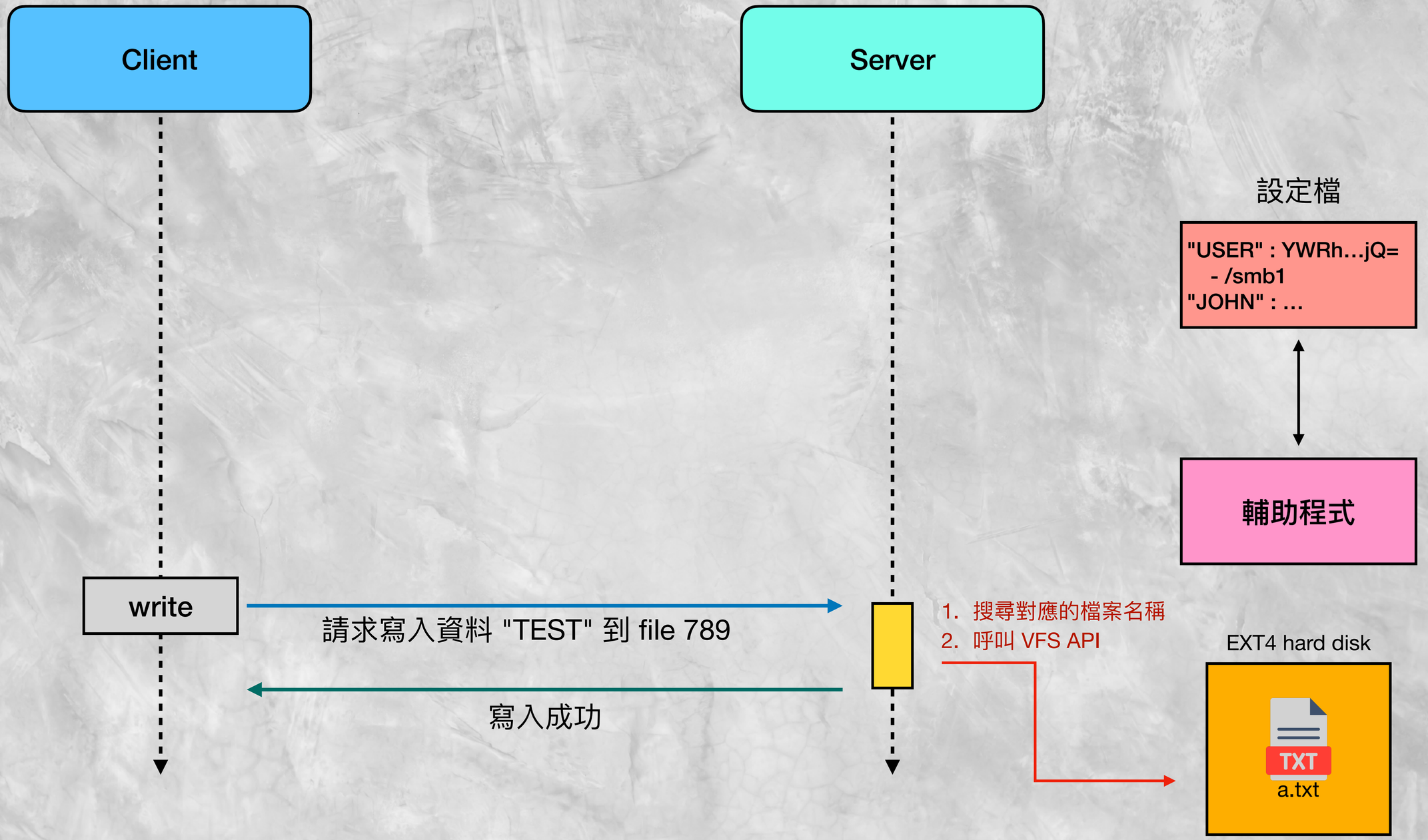

Client
Server
設定檔
"USER" : YWRh…jQ=
- /smb1
"JOHN" : …
輔助程式
session
auth
請求檢查認證答案，使用者為
"USER"，session ID 為 123
連線成功
1. 搜尋對應使用者的密碼
2. 用此密碼解 challenge
3. 比對用戶端的認證答案
4. 將 session ID 加到認證
列表
EXT4 hard disk

File Operation

- SMB 稱被掛載的目錄為 tree，而每個 tree 都有一個 tree ID
- 新增檔案時需要提供 tree ID 與檔案名稱，伺服器會產生一個 file ID
- 操作檔案時提供 tree ID 與檔案的 file ID 即可，不需要檔案名稱


Client
Server
tree_conn
請求掛載 "\192.168.0.1\smb1"
1. 檢查設定檔
2. 產生 tree ID
掛載成功，回傳 tree ID (456)
記錄掛載結果
設定檔
"USER" : YWRh...jQ=
- /smb1
"JOHN" : ...
輔助程式
EXT4 hard disk


Client
Server
create
請求在 tree 456 新增檔案 "a.txt"
建立成功，回傳 file ID (789)
1. 呼叫 VFS API
2. 產生 file ID
設定檔
"USER" : YWRh…jQ=
- /smb1
"JOHN" : …
輔助程式
EXT4 hard disk
TXT
a.txt


Client
Server
設定檔
"USER" : YWRh…jQ=
- /smb1
"JOHN" : …
輔助程式
write
請求寫入資料 "TEST" 到 file 789
寫入成功
1. 搜尋對應的檔案名稱
2. 呼叫 VFS API
EXT4 hard disk
TXT
a.txt

Attack Surface

- 檢查認證答案 (solved challenge) 的流程有問題
- 解析請求封包時，沒有檢查作為偏移或是長度使用的欄位是否合法
- 平行處理請求所造成的資料不同步
- …

Environment

KSMBD Setup

- 參考 ksmbd-tools 設定伺服器
- crontab 腳本
  - 自動啟動 ksmbd，並且每 30 秒重啟一次
  - 掛載遠端檔案系統至目錄 /mnt

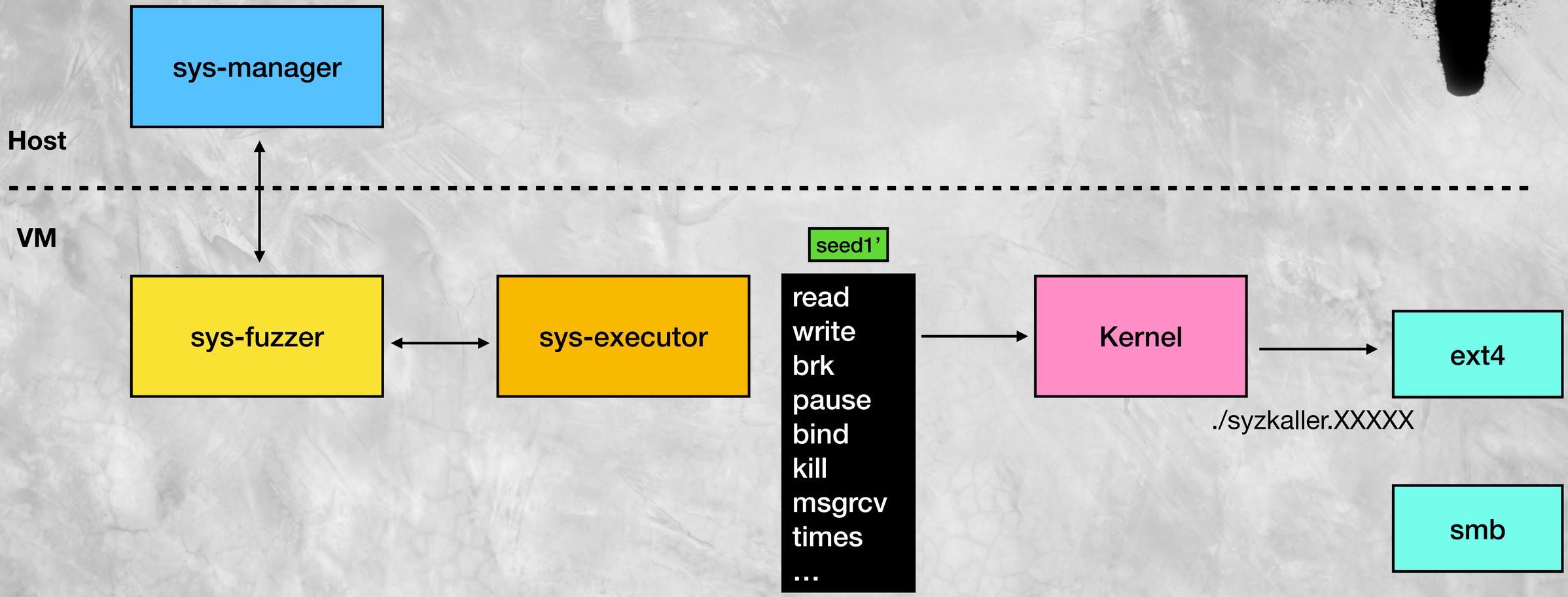
```
while true
do
        ksmbd.mountd
        mount -o user=root,password=root //127.0.0.1/MyShare /mnt
        sleep 30
        umount /mnt
        ksmbd.control --shutdown
done
```

Syzkaller Setup

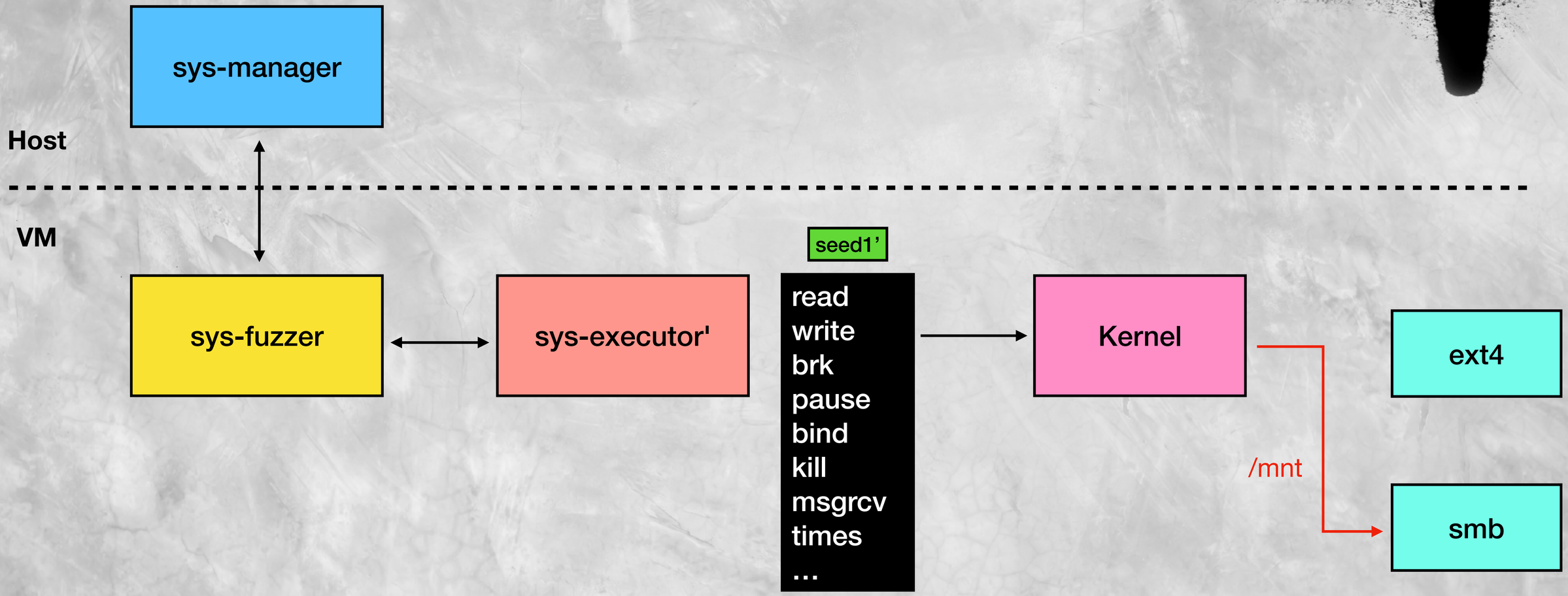
- 將 syzkaller 執行系統呼叫時的目錄設為遠端檔案系統掛載目錄

```
1  static void use_temporary_dir(void)
2  {
3      //char tmpdir_template[] = "./syzkaller.XXXXXX";
4      //char* tmpdir = mkdtemp(tmpdir_template);
5      char tmpdir_template[] = "/mnt";
6      char *tmpdir = tmpdir_template;
7      if (!tmpdir)
8          fail("failed to mkdtemp");
9      if (chmod(tmpdir, 0777))
10         fail("failed to chmod");
11     if (chdir(tmpdir))
12         fail("failed to chdir");
13 }
```

Syzkaller Setup



Syzkaller Setup

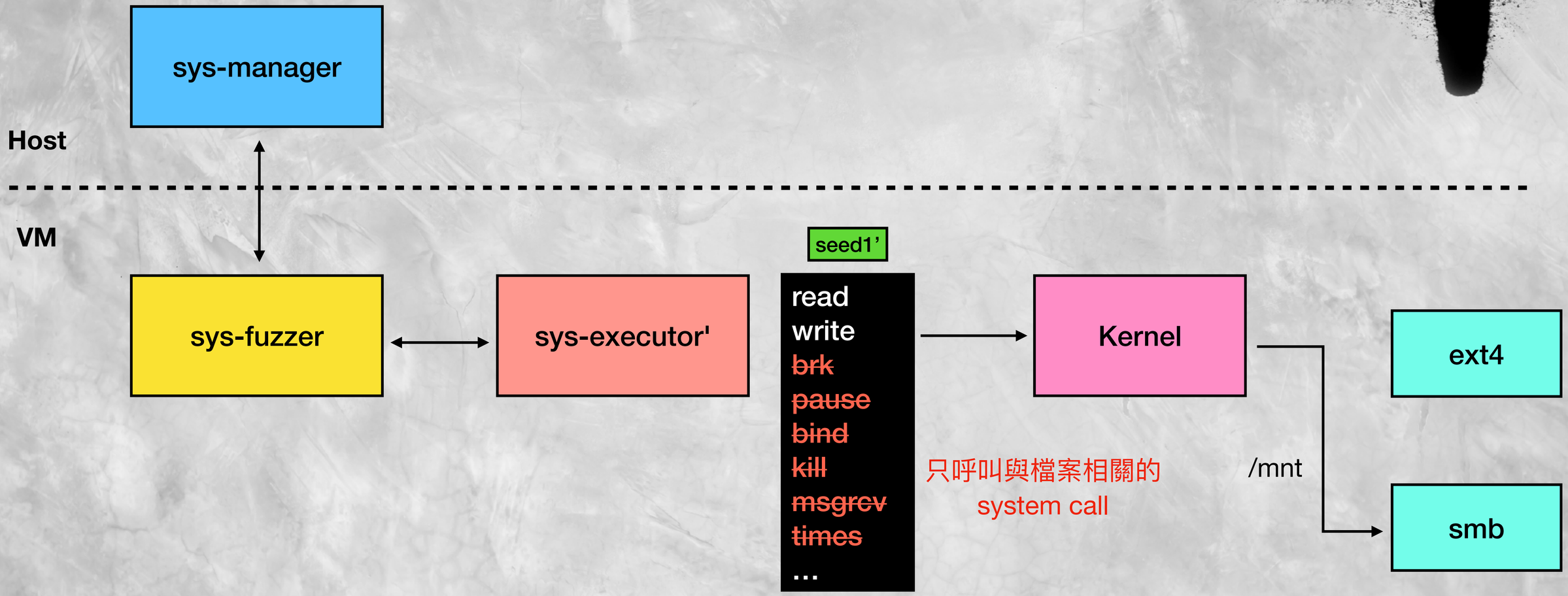


Syzkaller Setup

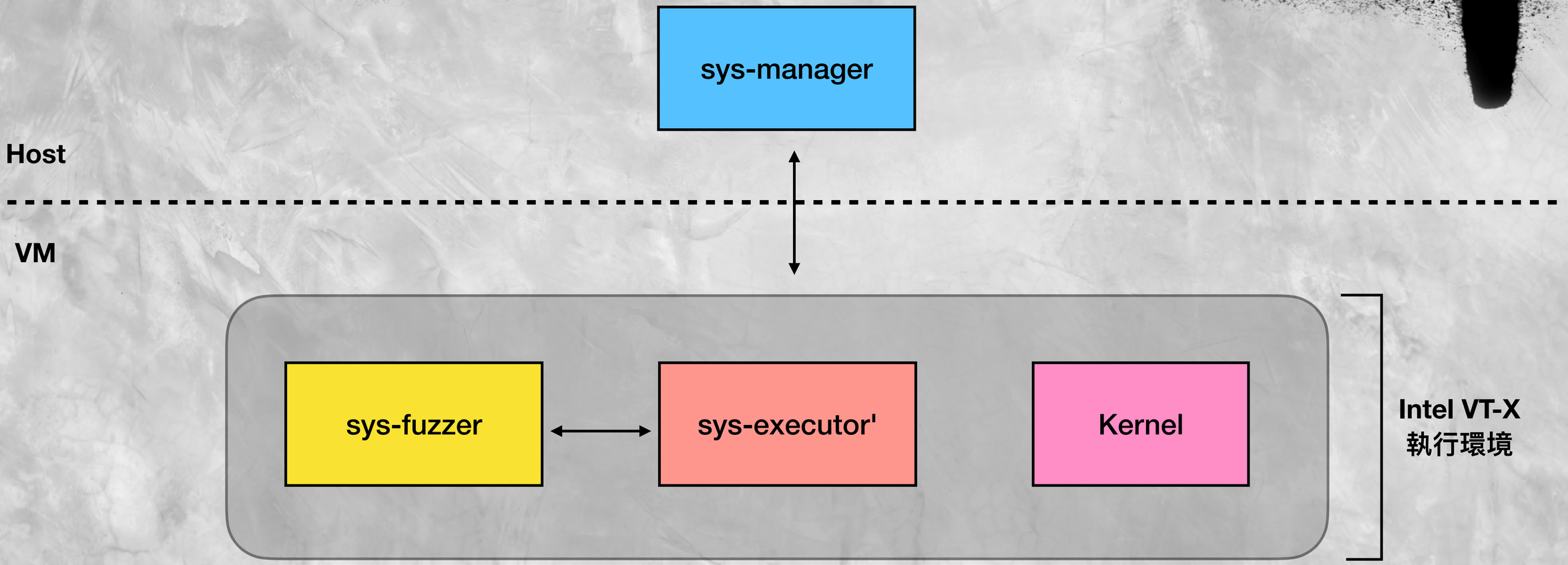
- 調整設定檔讓 syzkaller 只執行與檔案相關的系統呼叫
- 啟用 KVM 硬體虛擬化技術來加快執行速度

```
{
    "target": "linux/amd64",
    "http": "127.0.0.1:56741",
    "workdir": "/home/u1f383/syzkaller/workdir",
    "kernel_obj": "/home/u1f383/fuzz_test/linux",
    "image": "/home/u1f383/fuzz_test/linux/image/stretch.img",
    "sshkey": "/home/u1f383/fuzz_test/linux/image/stretch.id_rsa",
    "syzkaller": "/home/u1f383/syzkaller",
    "enable_syscalls": ["read","write","open",...,"fanotify_mark","statx"],
    "procs": 1,
    "type": "qemu",
    "reproduce": false,
    "vm": {
        "qemu_args": "-enable-kvm",
        "count": 1,
        "kernel": "/home/u1f383/fuzz_test/linux/arch/x86/boot/bzImage",
        "cpu": 8,
        "mem": 2048
    }
}
```

Syzkaller Setup



Syzkaller Setup



Result

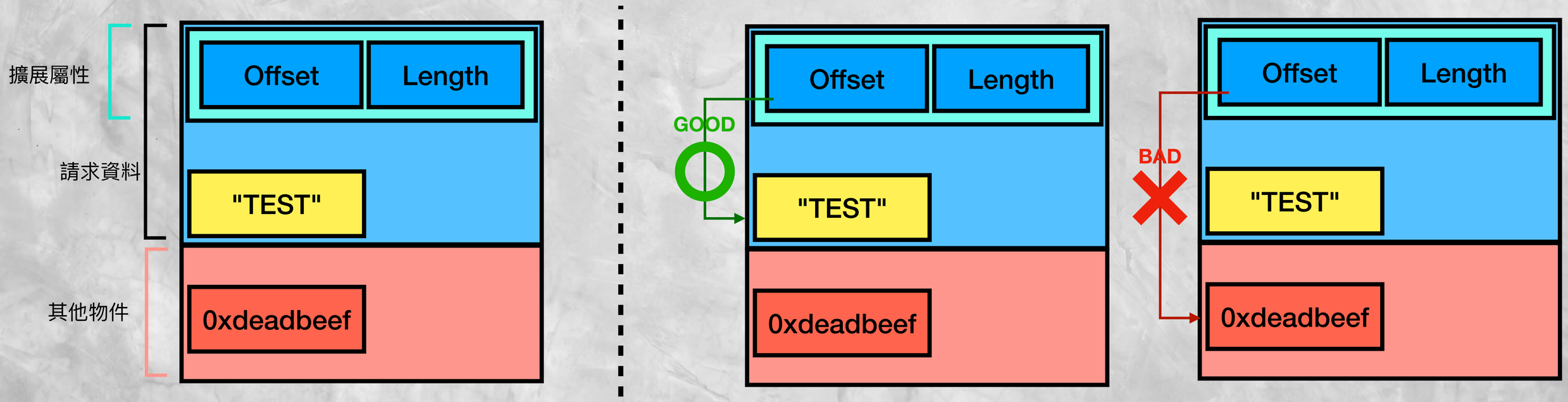
Overview

- 從環境建置到回報漏洞約一個月，一共找到四個漏洞，其中一個重複回報
  - CVE-2023-1194 - use-after-free in parse_lease_state
  - CVE-2023-1193 - use-after-free in setup_async_work (by syzkaller)
  - CVE-2023-1192 - use-after-free in smb2_is_status_io_timeout (by syzkaller)

CVE-2023-1194

- 嚴格來說應該是 out-of-bound access
- OPEN 請求的欄位包含擴展屬性，提供開發人員一個介面能客製化檔案屬性
- SMB 使用偏移 (Offset) 與長度 (Length) 紀錄擴展屬性名稱
- 然而 KSMBD 在解析時，沒有檢查名稱的偏移是否合法，造成越界存取

CVE-2023-1194



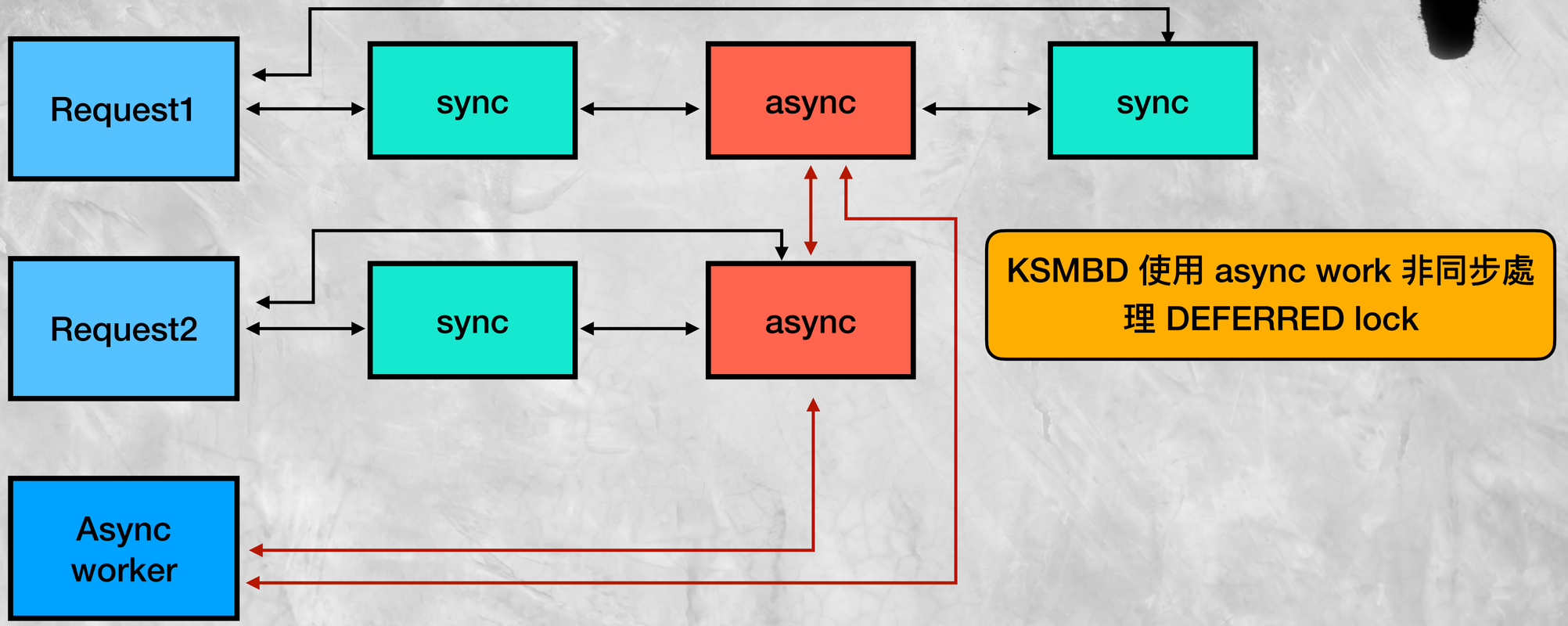
CVE-2023-1194

- 越界存取位址的字串不會被修改 - 沒有任意讀寫
- 資料在後續也不會回傳給使用者 - 沒有 info leak
- 只能在傳送很大的偏移時，讓 KSMBD 存取到非法記憶體位址而 panic，造成 DoS

CVE-2023-1193

- KSMBD 會把每個請求包裝成不同的 sync work，處理不同的操作
- 接收到 LOCK 請求時，如果 lock type 是 DEFERRED，則會新增 async work，代表要非同步的執行
- 這些 async work 會加進 async work list 中，並由 async worker 負責執行

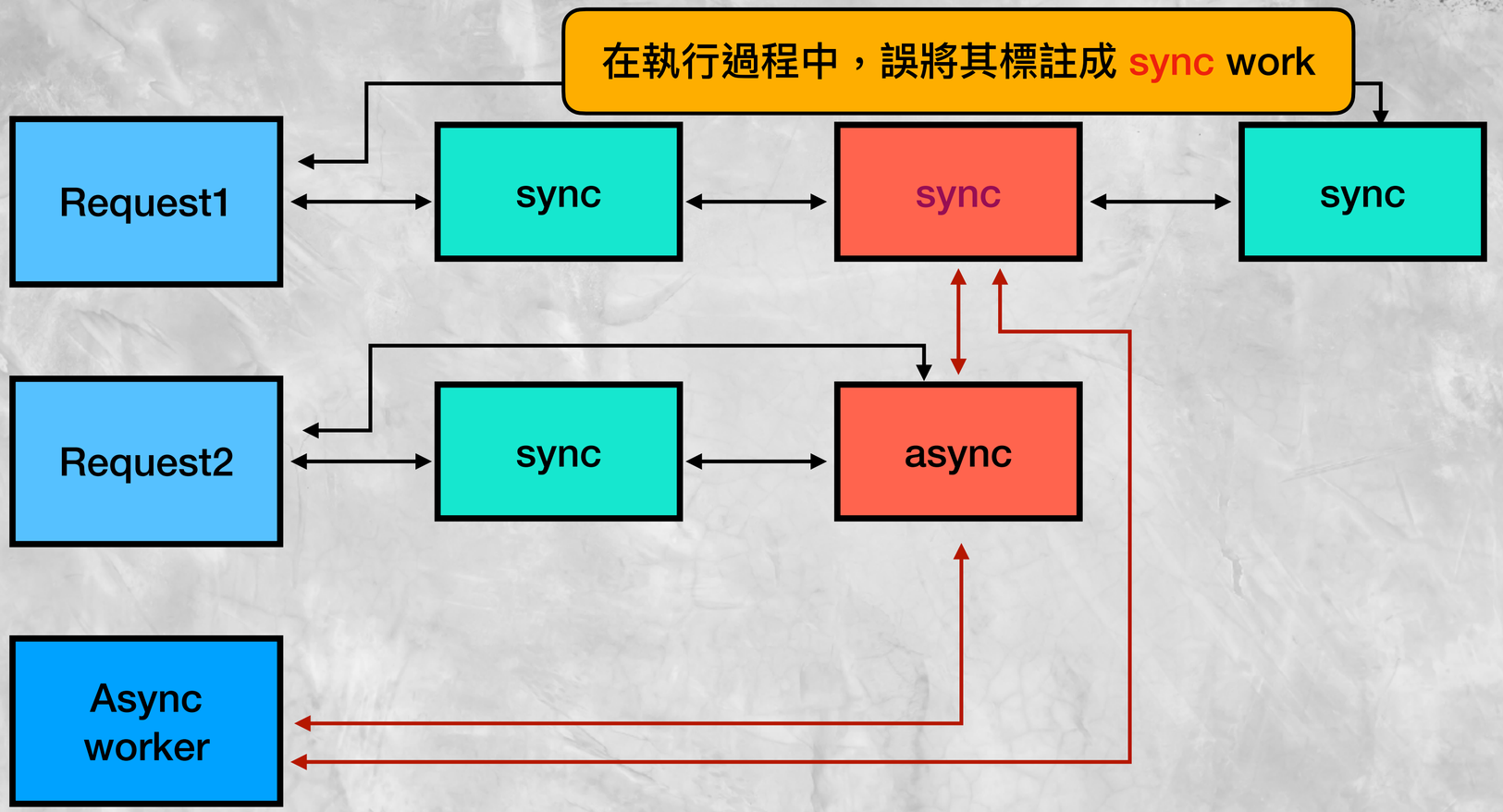
CVE-2023-1193



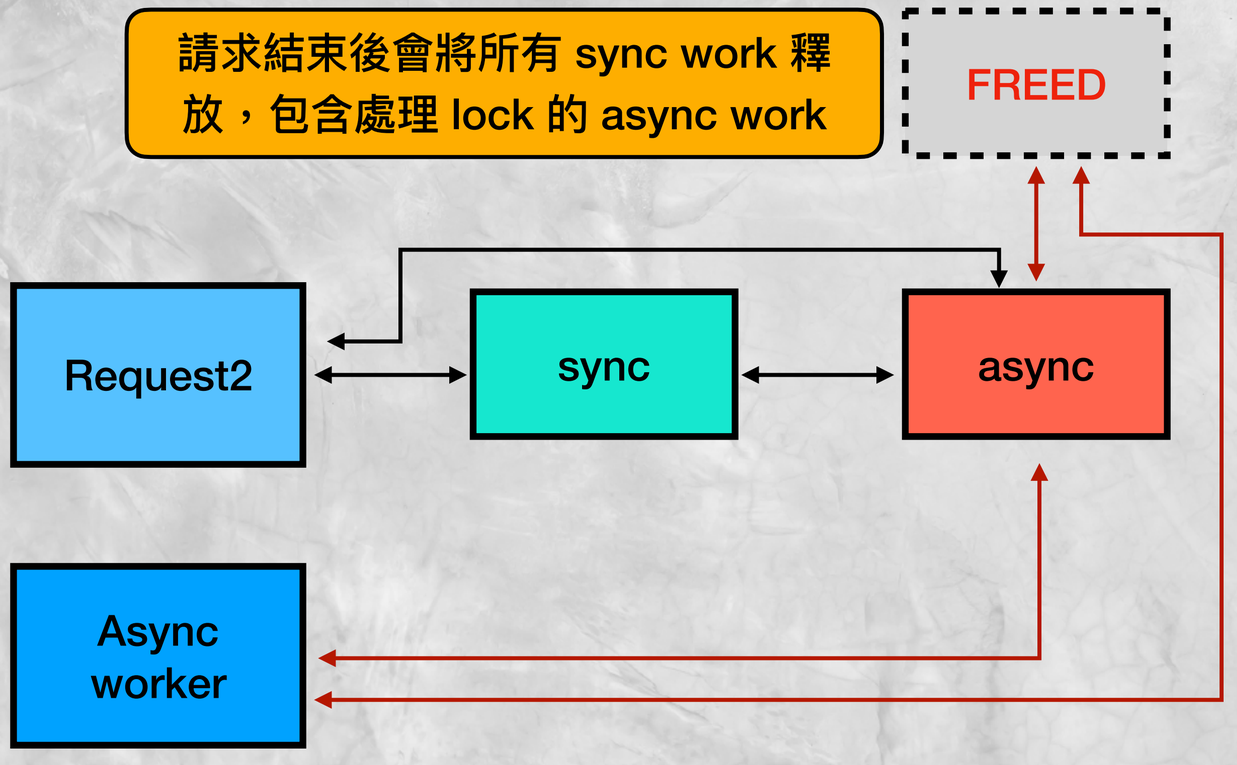
CVE-2023-1193

- KSMBD 在處理時誤將 async work 標記成 sync
- 完成請求後，KSMBD 會釋放 sync work，包含被誤標記成 sync 的 async work
- 被誤標記的 async work 沒有從 async work list 中移除
- 下次處理 DEFERRED LOCK 請求時，就會在新增 async work 時，嘗試更新已經被釋放的 async work 的 linked list 欄位，進而觸發 UAF

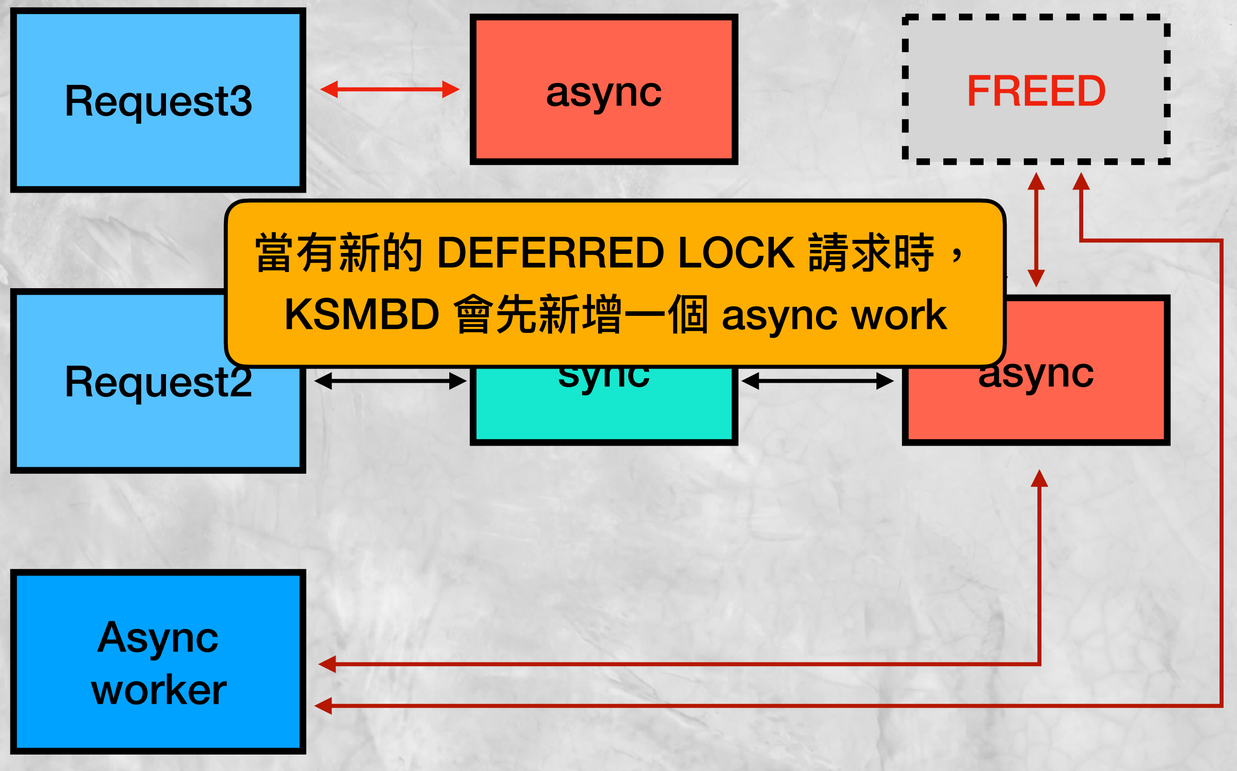
CVE-2023-1193



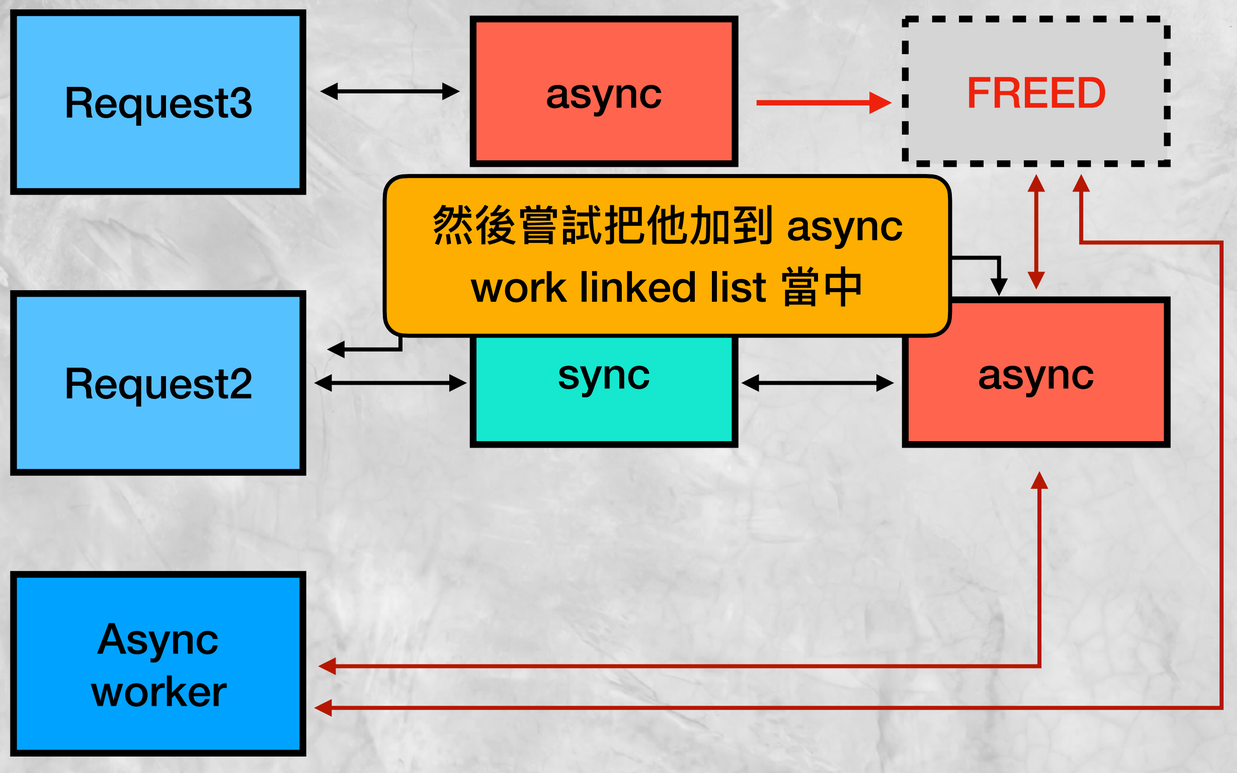
CVE-2023-1193



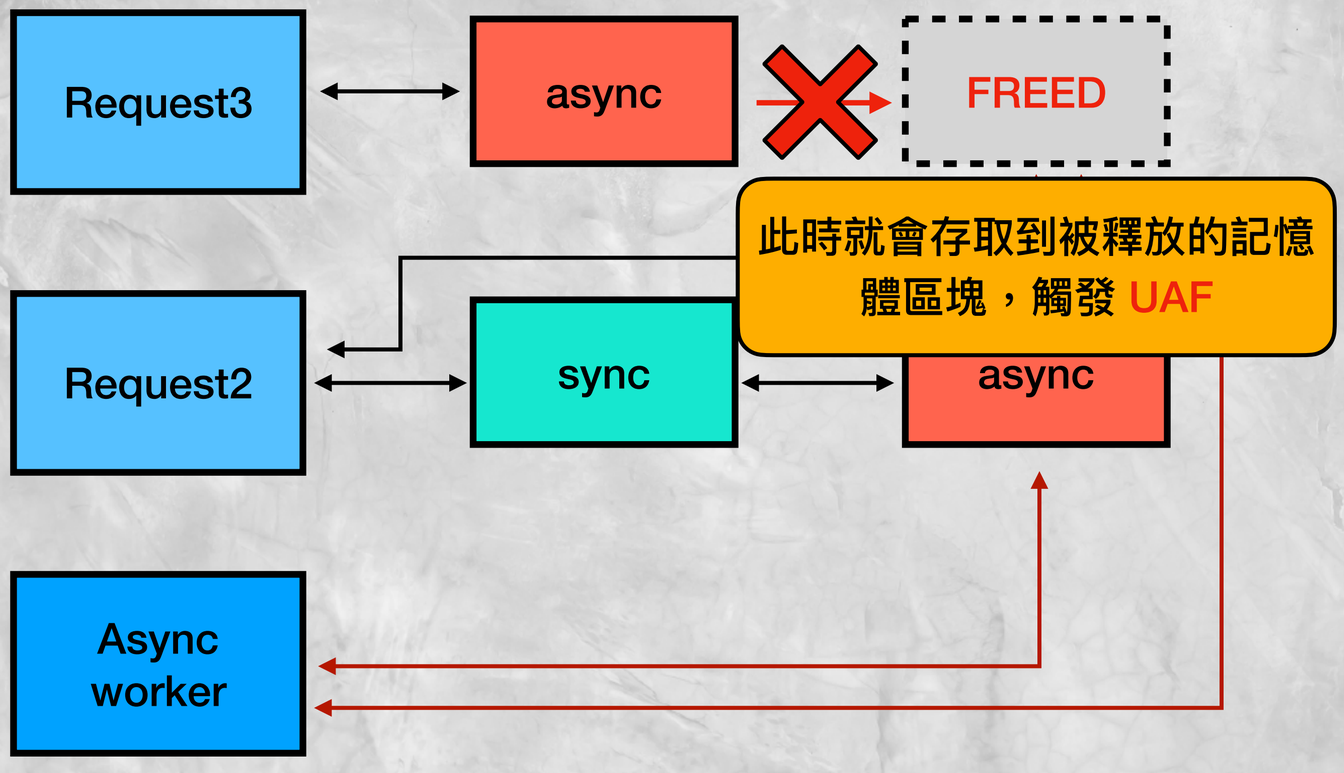
CVE-2023-1193



CVE-2023-1193



CVE-2023-1193



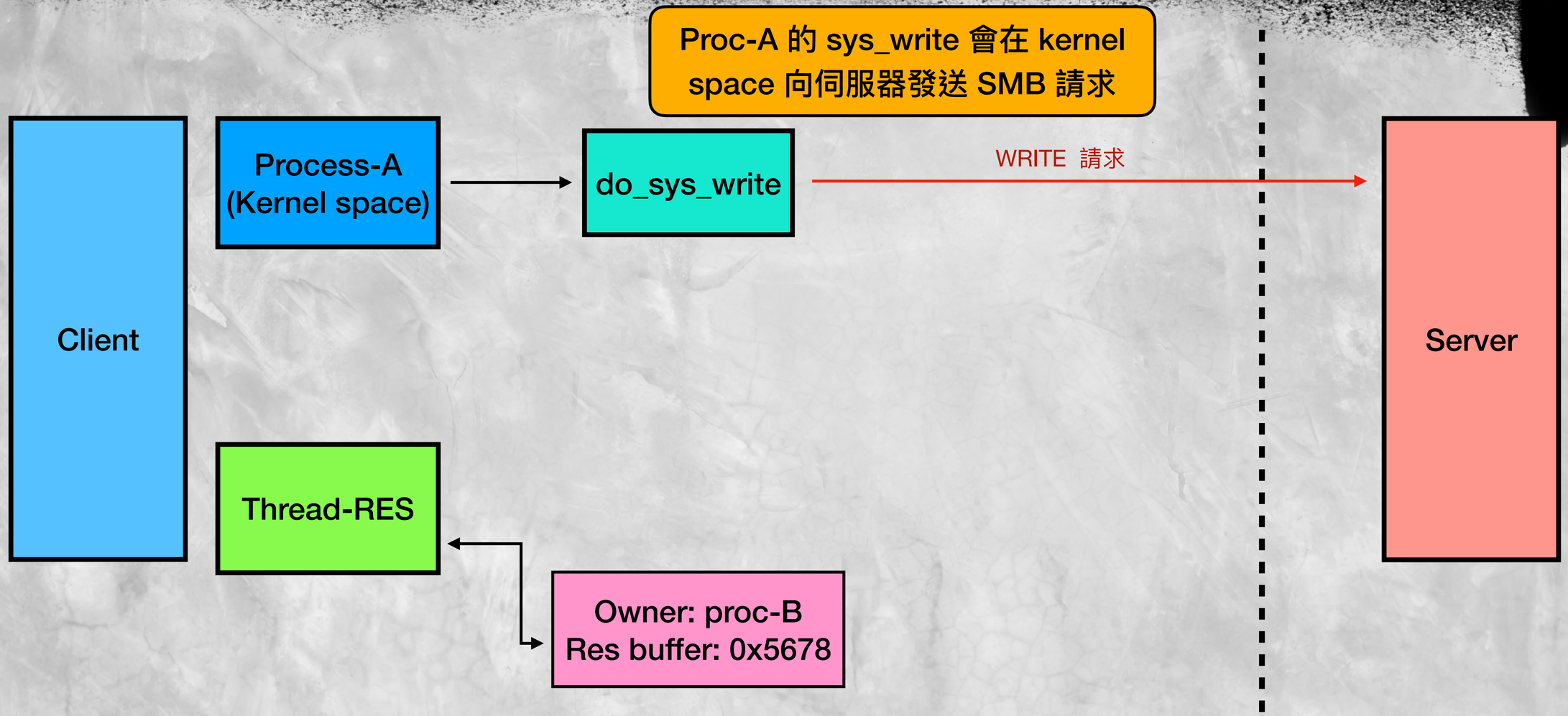
CVE-2023-1193

- 用可控物件替換掉被釋放的 async work
- Linked list insert 會寫入記憶體位址 - info leak
- Async worker 釋放 async work - 任意物件釋放
- 配合現有的攻擊手法，理想情況下可以做到提權

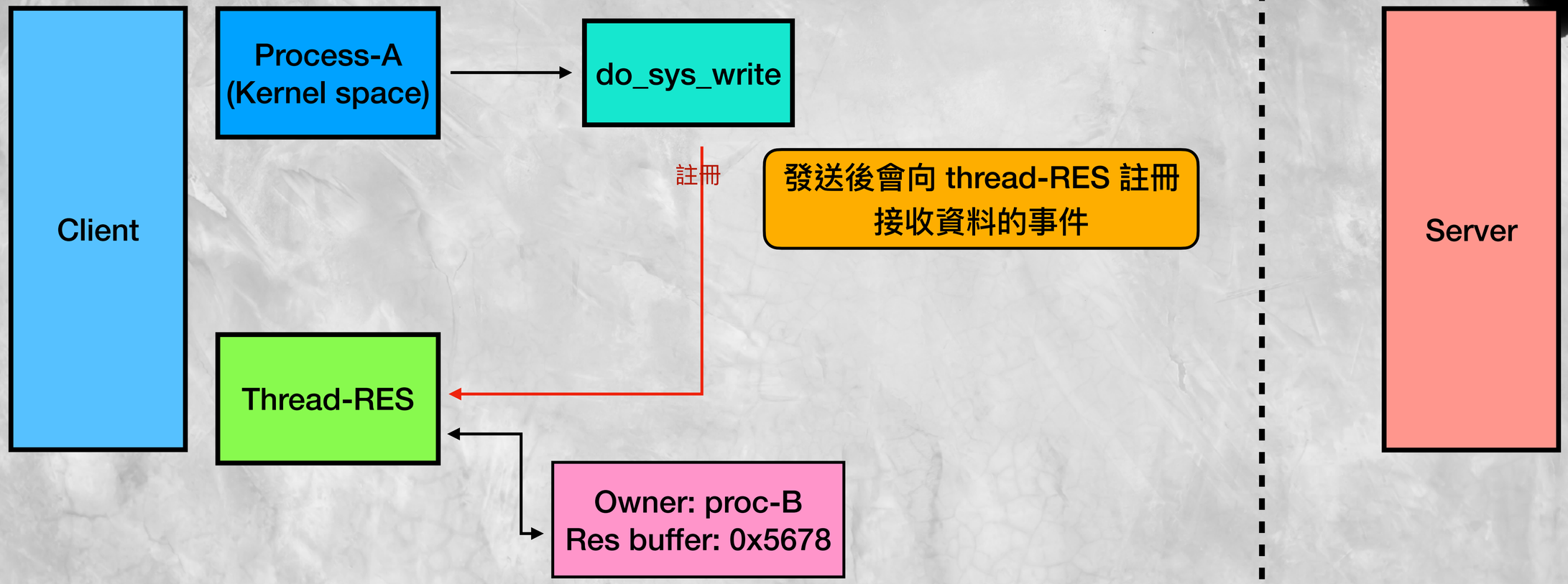
CVE-2023-1192

- CIFS 會在 process 呼叫 syscall 時，向伺服器發送 SMB 請求
- 發送請求後，process 會向 thread-RES 註冊接收事件
- Thread-RES 接收伺服器的回覆後，喚醒原 process 並轉交資料

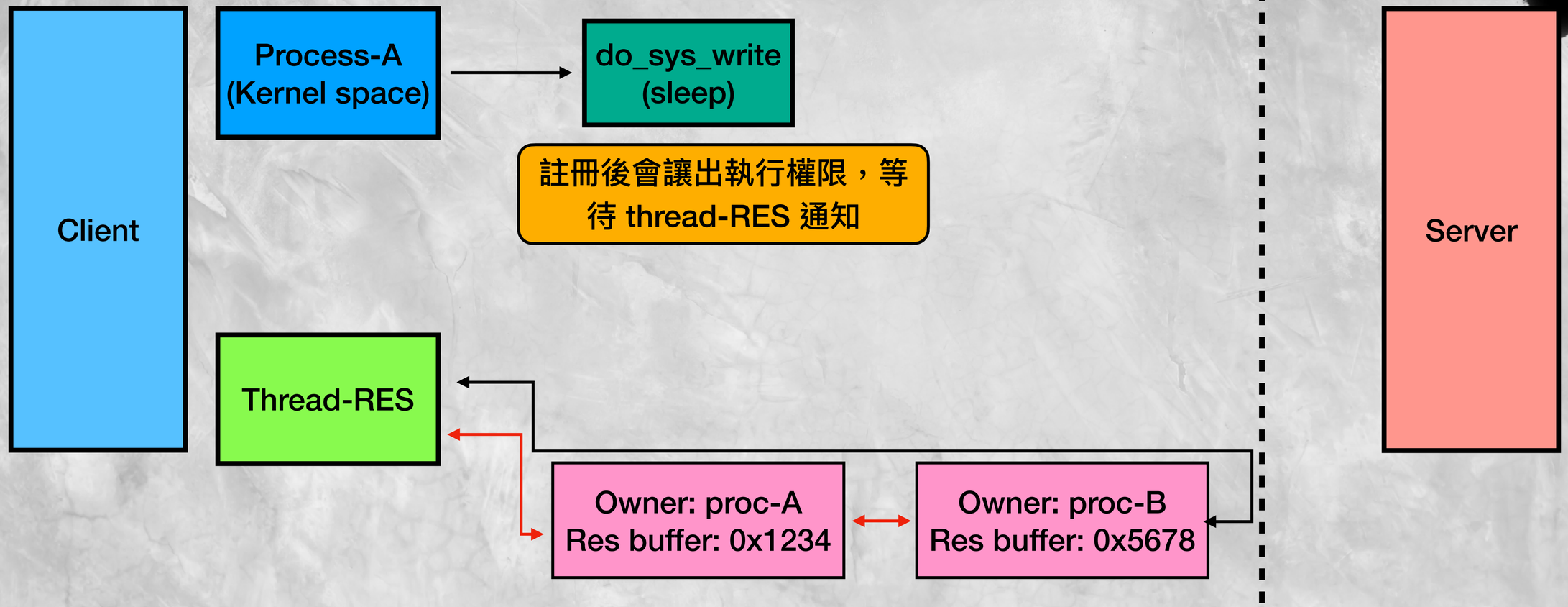
CVE-2023-1192



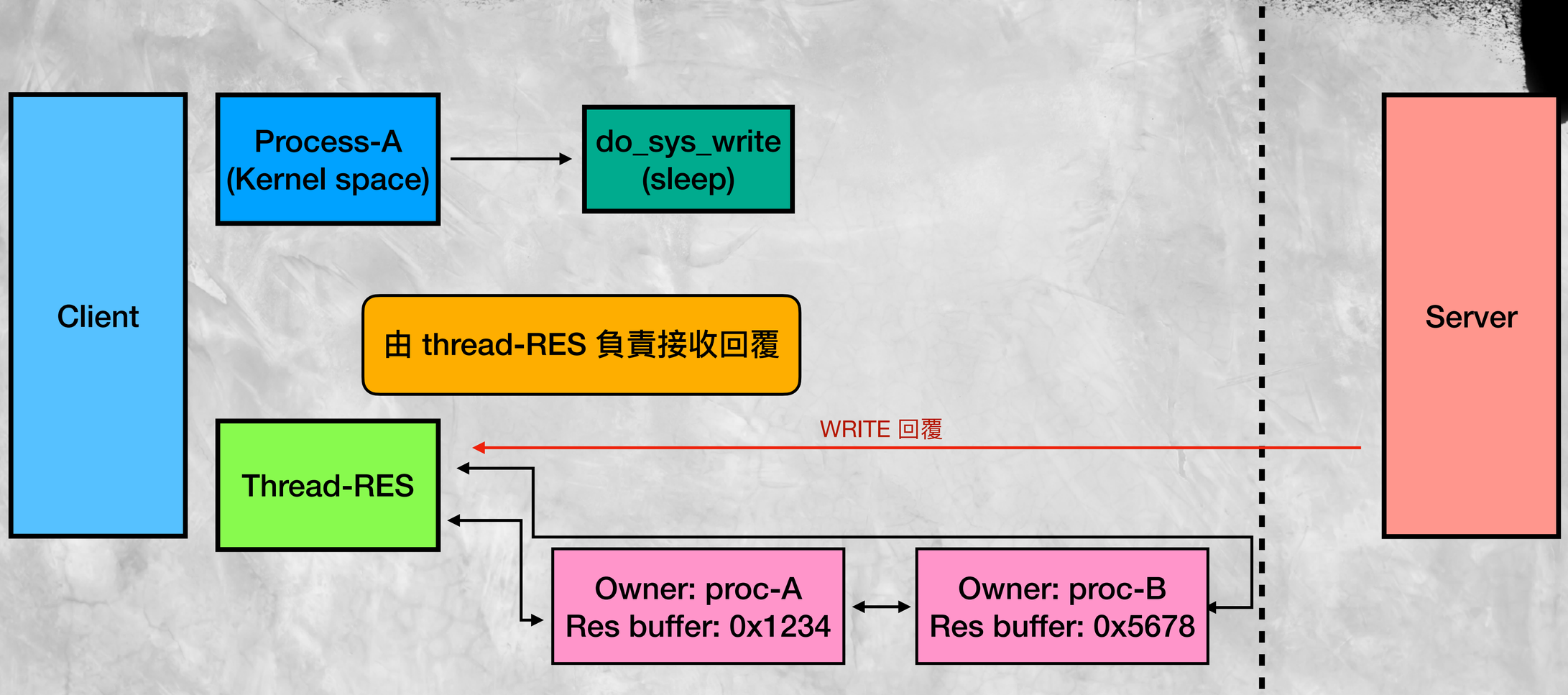
CVE-2023-1192



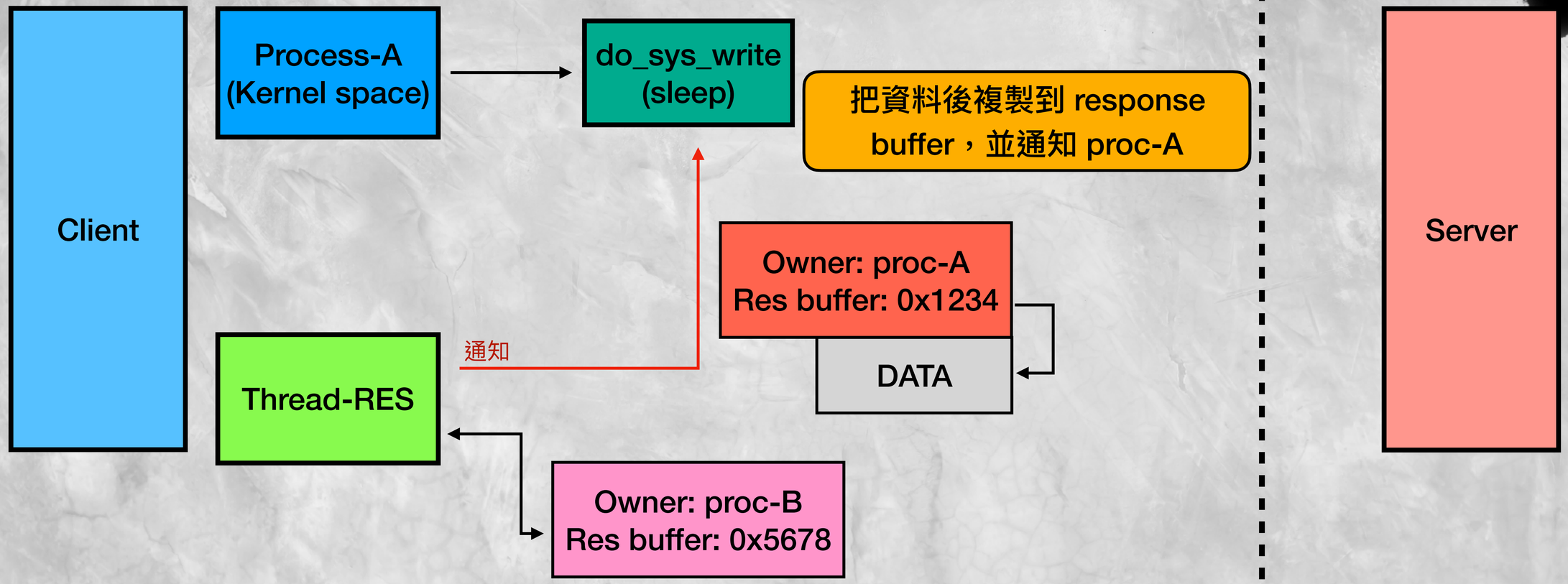
CVE-2023-1192



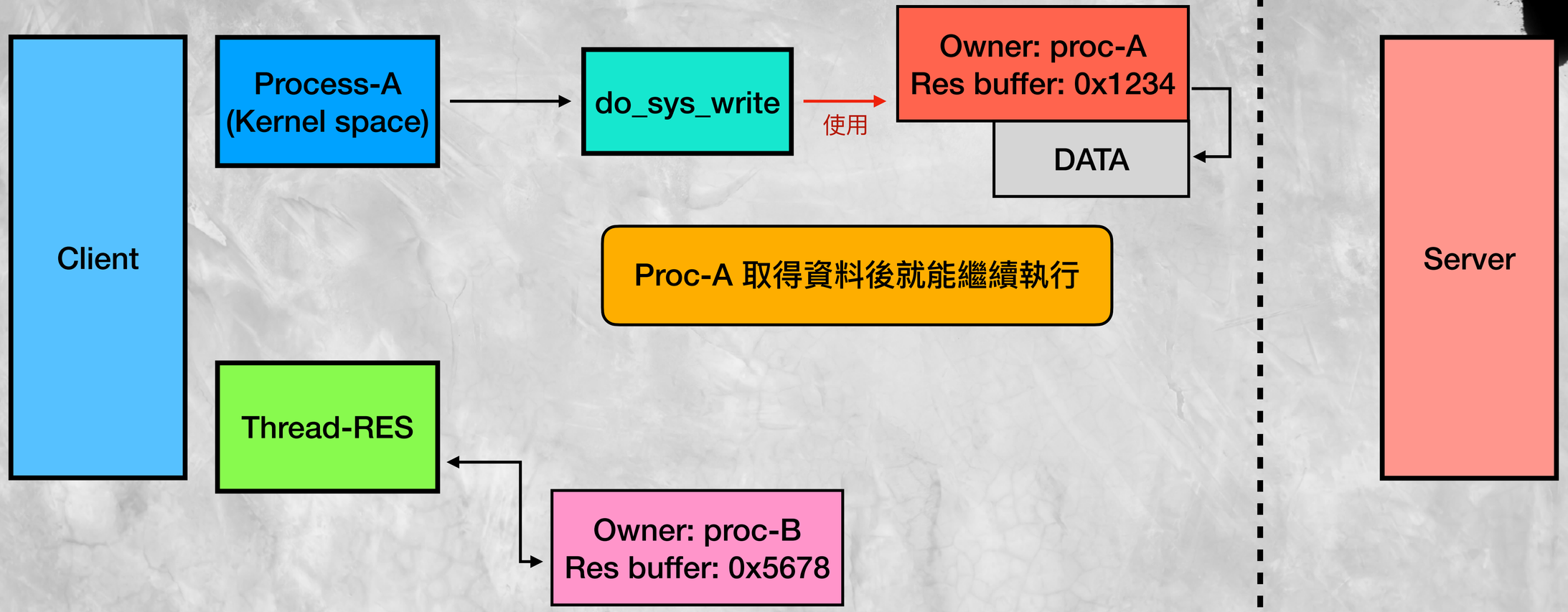
CVE-2023-1192



CVE-2023-1192



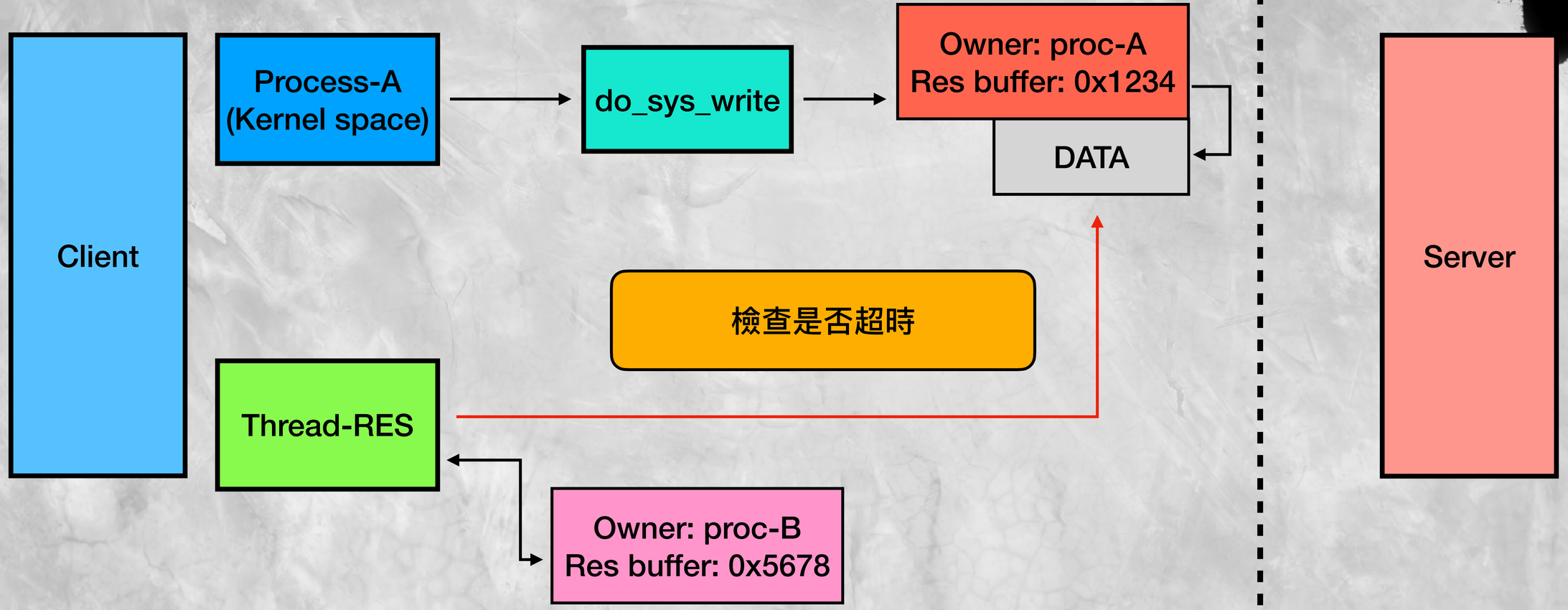
CVE-2023-1192



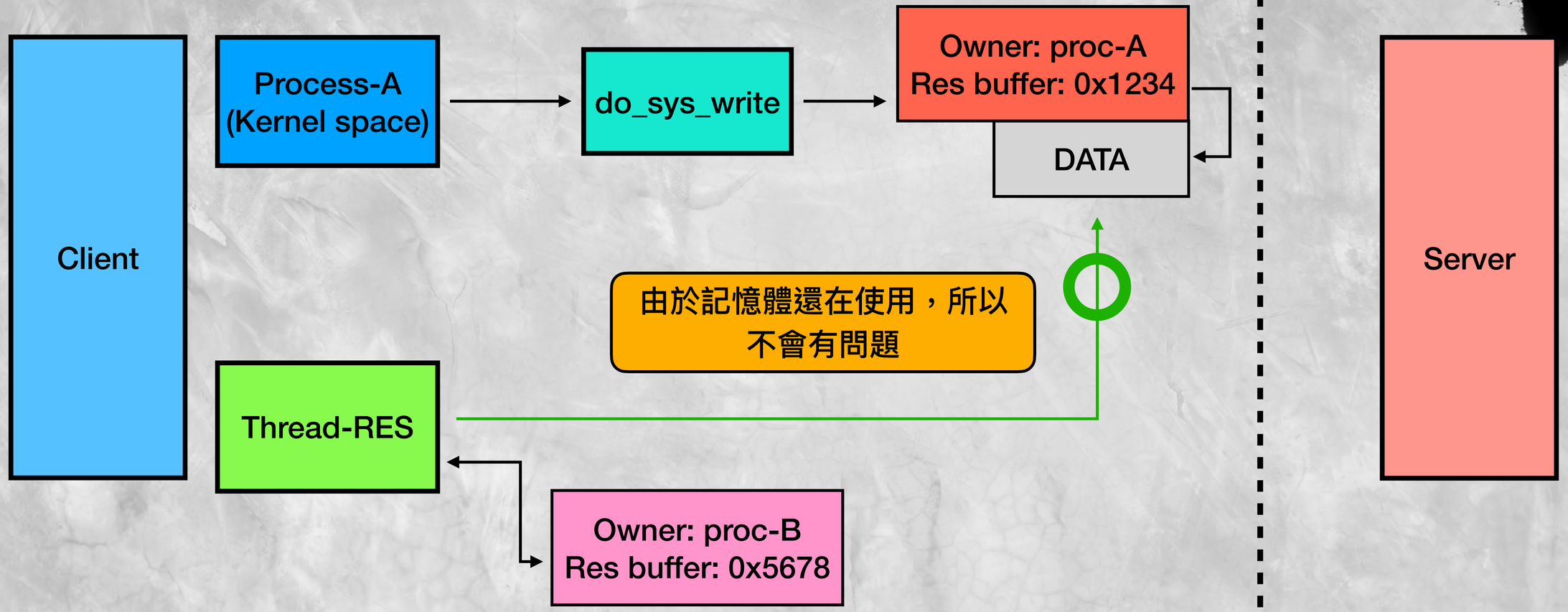
CVE-2023-1192

- Thread-RES 會在轉交後，再次存取回覆資料的記憶體，檢查是否超時
- 如果 process 還在使用回覆資料的話，不會出現問題
- 如果 process 已經使用完並釋放回覆資料，thread-RES 存取到被釋放的記憶體空間，觸發 UAF

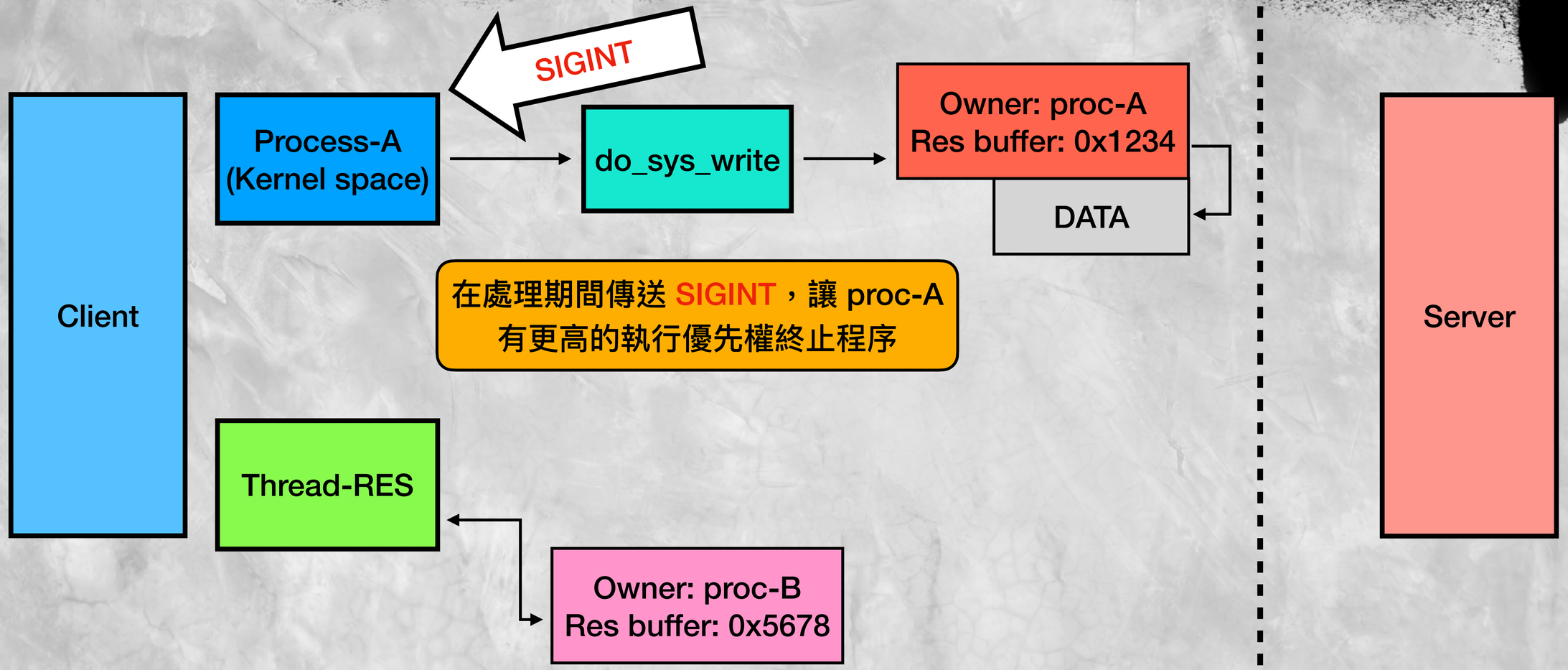
CVE-2023-1192



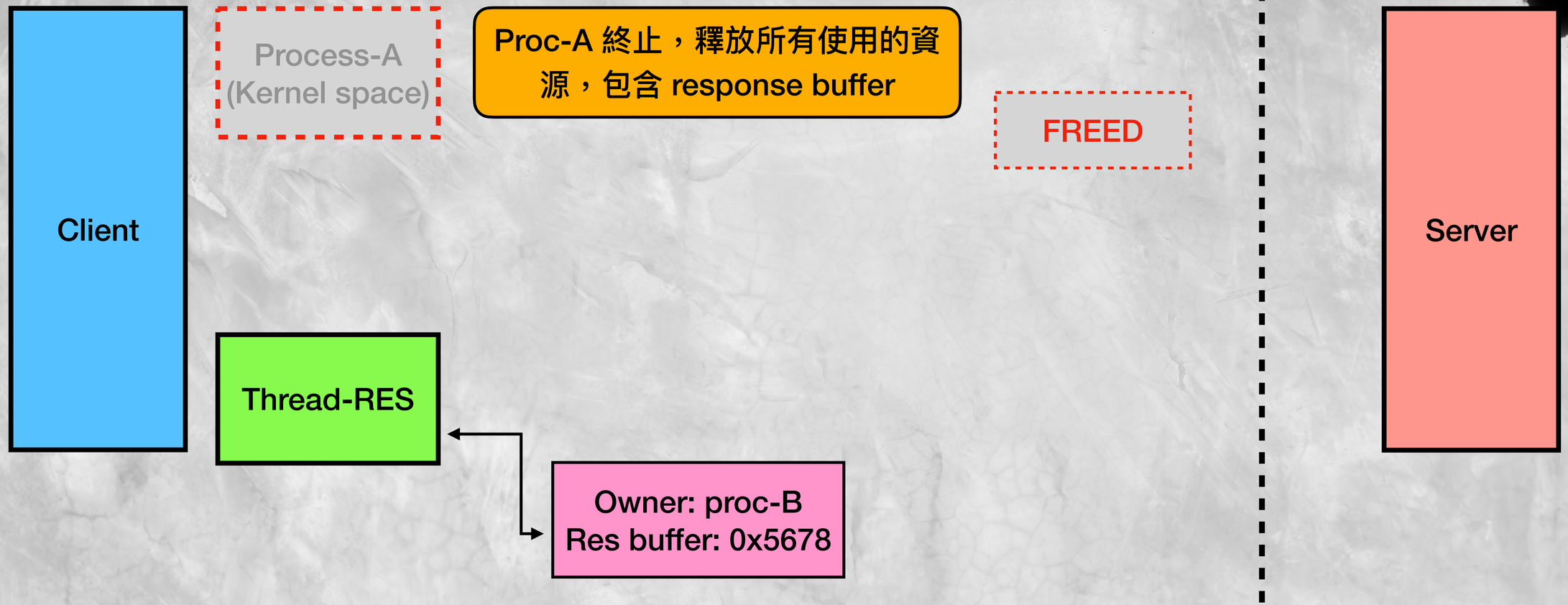
CVE-2023-1192



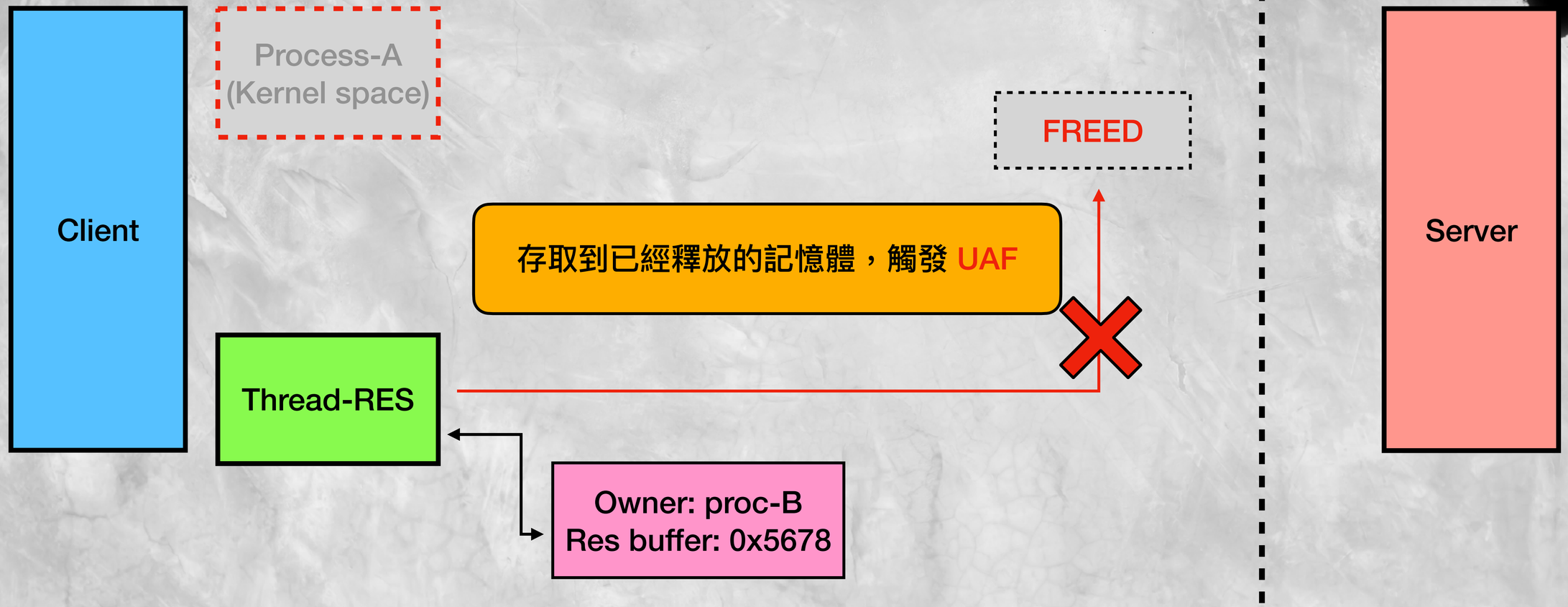
CVE-2023-1192



CVE-2023-1192



CVE-2023-1192



CVE-2023-1192

- 目前觀察到 thread-RES 存取該記憶體只用來檢查是否超時
- 只能讓用戶端與伺服器重新建立連線

Conclusion

Limitation

- 在對 syzkaller 有一定的認識後，配合微調就能測試到預設不支援的目標
- 對於遠端檔案系統，syzkaller 找不太到記憶體越界存取相關的漏洞
- 研究如何接上 fuzzer 也有時間成本，需要審慎評估

Experience

- 配合工具挖掘漏洞能事半功倍
- 遠端檔案系統仍潛藏許多問題
- 模糊測試的使用只會與日俱增