EDR INTERNALS

for macOS and Linux

Kyle Avery

October 2024

OUTFLANK

clear advice with a hacker mindset

WHOAMI

Kyle Avery – @kyleavery_

- Offensive R&D @ Outflank
- Former BHIS/Antisyphon
- C/C++ developer for Windows+macOS+Linux

OUTFLANK

- Outflank Security Tooling (OST)
- Red Teaming Services

AGENDA

Background

- Notable capabilities
- Previous research
- Getting started

EDR Internals

- Telemetry sources
- Sandbox detection

Case Study: Attacking EDR

- Spoofing Linux syscalls

EDR BACKGROUND – NOTABLE CAPABILITIES

Endpoint detection and response

- Collects host data to inform defenders and their tools
- Often bundled with antivirus

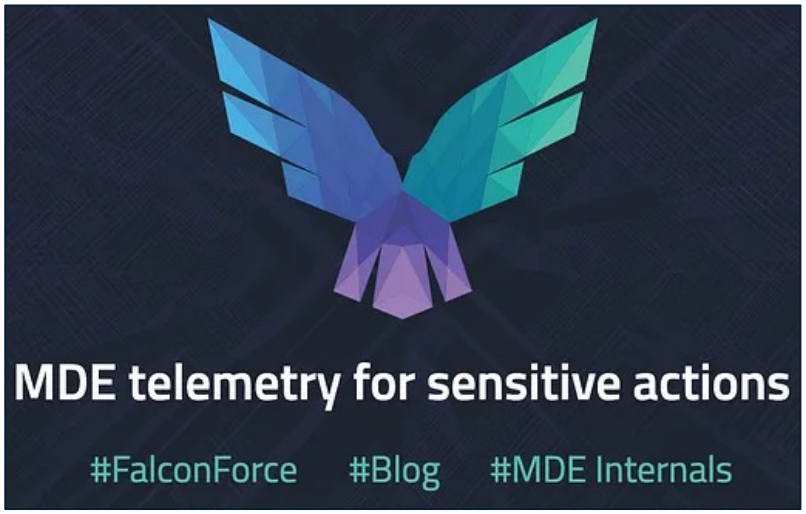
Common event types

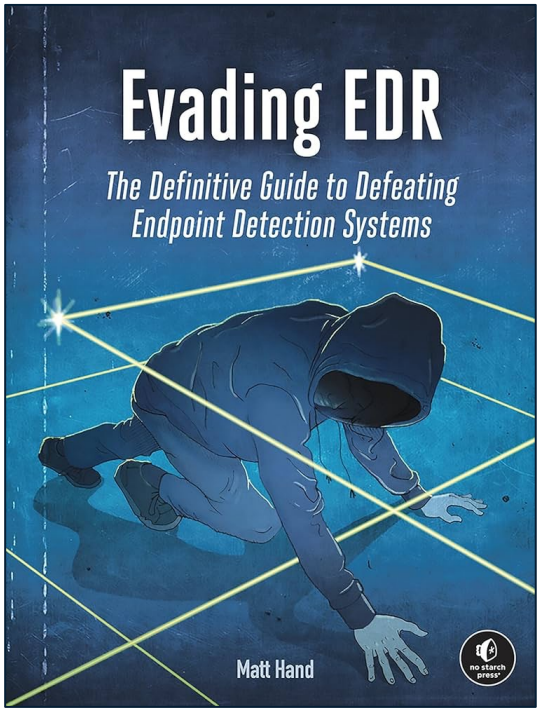
- Authentication attempts
- Process creation and termination
- File access, modification, creation, and deletion
- Network traffic

EDR BACKGROUND – PREVIOUS RESEARCH

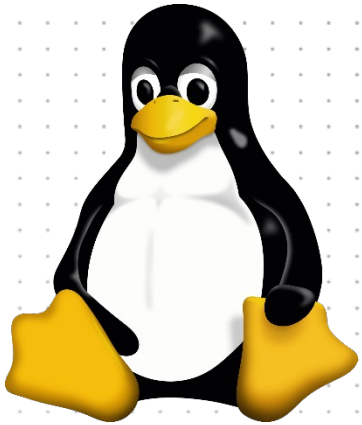
Windows

- Many blogs and presentations on specific components or bypass methods
- No Starch Press Book – Evading EDR
- FalconForce Blog Series – MDE Internals





EDR BACKGROUND – PREVIOUS RESEARCH



macOS and Linux internals

- Apple documentation
- Linux kernel source

Open-source projects

- Objective-See – FileMonitor, ProcessMonitor, DNSMonitor
- Microsoft SysmonForLinux, Elastic eBPF
- Linux Runtime Security – Falco, tracee
- OSSEC, Wazuh

Objective-See

EDR BACKGROUND – GETTING STARTED

Public information

- Vendor blogs and docs
- Open-source projects

CROWDSTRIKE | BLOG

Mapping It Out: Analyzing the Security of eBPF Maps

Microsoft Defender

- Defender for Endpoint on macOS
  - Deploy Defender for Endpoint on macOS
  - Configure Defender for Endpoint on macOS
- Defender for Endpoint on Linux
  - Deploy Defender for Endpoint on Linux
  - Configure Defender for Endpoint on Linux

red canary

Contain yourself: An intro to Linux EDR

SentinelOne blog

Why Apple's Silent Approach to Endpoint Security Should be a Wake-Up Call

Detection Rules

Detection Rules is the home for rules used by Elastic Security. This repository is used for the development, maintenance, testing, validation, and release of rules for Elastic Security's Detection Engine.

EDR BACKGROUND – GETTING STARTED

Private information

- Customer docs
- Log data
- Reverse engineering

MACOS TELEMETRY

OUTFLANK

clear advice with a hacker mindset

TELEMETRY SOURCES – MACOS

Kernel extensions (KEXT)

- Once considered a de facto endpoint security sensor
- Unsupported since Catalina (2019)

Unified logging

- Introduced in Sierra (2016)
- Great source of info for debugging, forensics/IR
- Only accessible with **`log`** CLI tool or **`Console`** GUI app

TELEMETRY SOURCES – MACOS

Endpoint Security API

- Authentication attempts
- Process creation and termination
- File access, modification, and creation

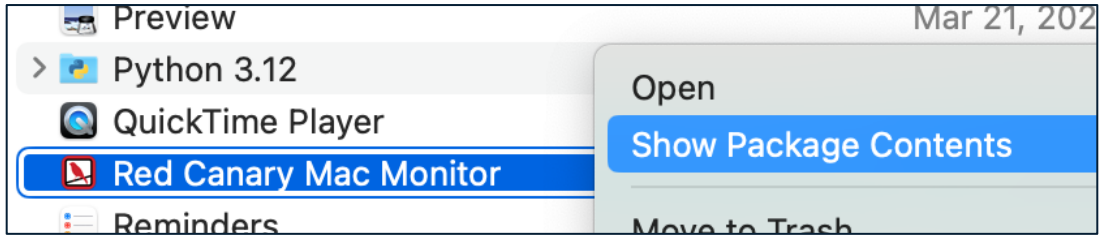
Event types

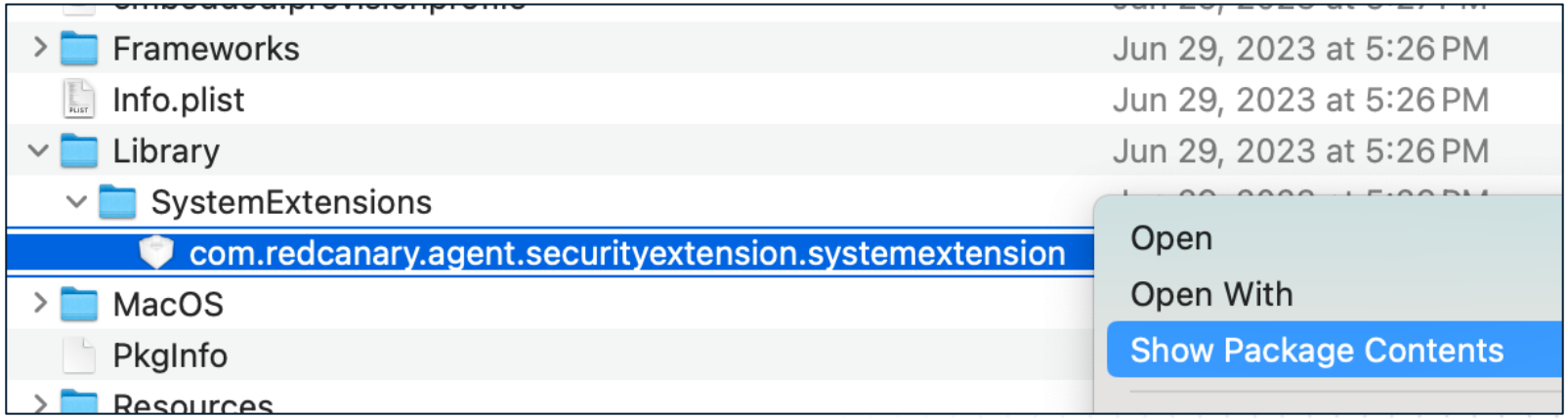
- Follows a standard format: **ES_EVENT_TYPE_<RESPONSE>_<NAME>**
- Response: NOTIFY or AUTH
- Names: EXEC, OPEN, LOGIN, etc.

TELEMETRY SOURCES – MACOS

Analyzing Endpoint Security clients

- Clients are typically system extensions
- Check app for **`Contents/Library/SystemExtensions`**





TELEMETRY SOURCES – MACOS

- Every client must use the **es_subscribe** function to monitor events
- Two interesting parameters: **events** and **event_count**

```
const module = Process.getModuleByName("libEndpointSecurity.dylib");
const func = module.getExportByName("es_subscribe");

Interceptor.attach(func, {
  onEnter: function (args) {
    const event_count = parseInt(Number(args[2]), 10);
    const events = args[1];
    console.log(`[+] Subscribed to ${event_count} event types:`);
    for (let i = 0; i < event_count; i++) {
      const event_id = events.add(i * 4).readU32();
      // Lookup and print event name
    }
  }
});
```

Terminal Shell Edit View Window Help

user — -zsh — 75×25

```
user@vm ~ %
```

TELEMETRY SOURCES – MACOS

Event types are enough, right?

- Some event types are clearer than others…

LW_SESSION_LOGIN

```
Notification that LoginWindow has logged in a user.

username               Short username of the user.
graphical_session_id   Graphical session id of the session.
```

AUTHENTICATION

```
Notification that an authentication was performed.

success            True iff authentication was successful.
type               The type of authentication.
data               Type-specific data describing the authentication.
```

LOGIN_LOGIN

```
Notification for authenticated login event from /usr/bin/login.

success                True iff login was successful.
failure_message        Optional. Failure message generated.
username               Username used for login.
has_uid                Describes whether or not the uid of the user logged in
uid                    Union that is valid when `has_uid` is set to `true`
uid.uid                uid of user that was logged in.
```

TELEMETRY SOURCES – MACOS

- ESDump – Subscribe to any event types, output raw data
  - Full JSON-formatted event data
  - Converts enum values to string representation
  - Resolves username, process path, and PID for any audit tokens

EDR Internals

Tools for analyzing EDR agents. For details, see our blog post.

- ESDump - macOS Endpoint Security client that dumps events to `stdout`

```
constexpr es_event_type_t TARGET_EVENTS[] = {
    ES_EVENT_TYPE_NOTIFY_LW_SESSION_LOGIN,
    ES_EVENT_TYPE_NOTIFY_AUTHENTICATION,
    ES_EVENT_TYPE_NOTIFY_LOGIN_LOGIN
};
```

TELEMETRY SOURCES – MACOS

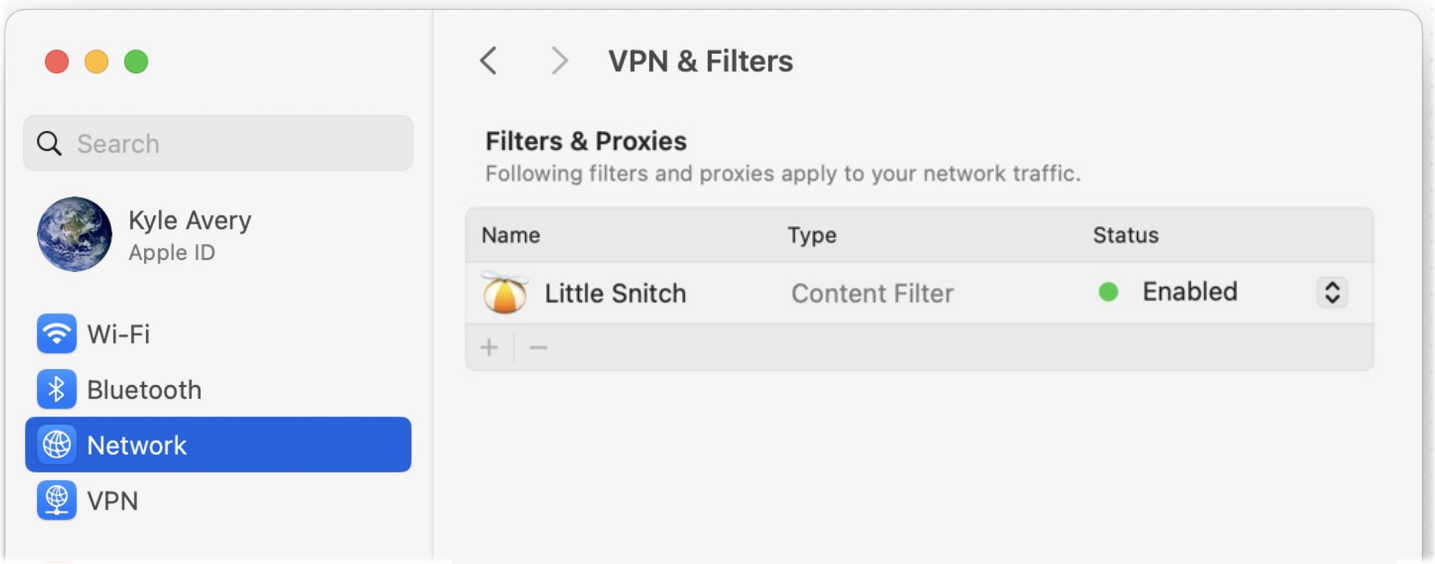
Trimmed ESDump output:

```
"type": "ES_EVENT_TYPE_NOTIFY_EXEC",
"path": "/usr/bin/whoami",
"cwd": "/Users/user",
"audit_token": {
  "path": "",
  "pid": 1313,
  "uid": 501,
  "username": "user"
},
"responsible_audit_token": {
  "pid": 390,
  "path": "/System/Applications/Utilities/Terminal.app/Contents/MacOS/Terminal"
  "uid": 501,
  "username": "user"
}
```

TELEMETRY SOURCES – MACOS

NetworkExtension framework

- Implements a provider:
  - DNS Proxy – Great for simple examples
  - Content Filter – Used by every reviewed agent
  - Packet Tunnel – Used by some agents



TELEMETRY SOURCES – MACOS

Analyzing network extensions

- Clients must be system extensions
- Extension plist contains provider type and class name

Labels | Proc. | Str

DNSProxyProvider

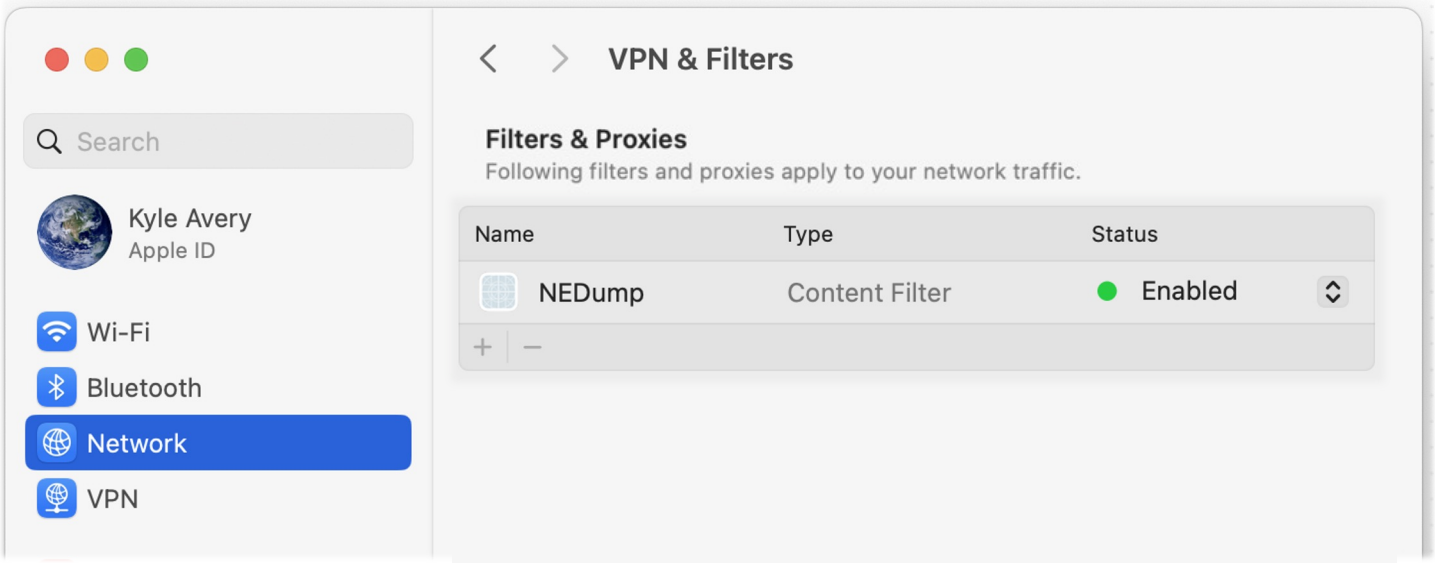
Tag Scope

Idx	Name
8	-[DNSProxyProvider startProxyWithOptions
9	-[DNSProxyProvider loadBlockList]
10	-[DNSProxyProvider stopProxyWithReason:
11	-[DNSProxyProvider handleNewFlow:]
19	-[DNSProxyProvider flowInUDP:connection
	DNSProxyProvider flowOutUDP:]
	DNSProxyProvider flowInTCP:connection
	DNSProxyProvider flowOutTCP:connectio
	DNSProxyProvider shouldBlock:]

```
<key>NetworkExtension</key>
<dict>
    <key>NEMachServiceName</key>
    <string>VBG97UB4TA.com.objective-see.dnsmonitor</string>
    <key>NEProviderClasses</key>
    <dict>
        <key>com.apple.networkextension.dns-proxy</key>
        <string>DNSProxyProvider</string>
    </dict>
</dict>
```

TELEMETRY SOURCES – MACOS

- NEDump – Content filter provider, output raw data
  - Full JSON-formatted event data
  - Converts enum values to string representation
  - Resolves username, process path, and PID for any audit tokens



Applications — -zsh — 75×24

/Applications — -zsh

/Applications — -zsh

```
user@vm /Applications %
```

LINUX TELEMETRY

OUTFLANK

clear advice with a hacker mindset

TELEMETRY SOURCES – LINUX

Auditd

- Great source of data
- Sometimes used for addon or backup sensors
  - Performance concerns
  - Compatibility issues

Auditd module

The `auditd` module collects and parses logs from the audit daemon (`auditd`).

How eBPF works

With eBPF, events previously obtained from the auditd event provider now flow from the eBPF sensor.

TELEMETRY SOURCES – LINUX

Kernel function tracing

- Sensor(s) monitor key kernel functions
- Two general hooking strategies:
  1. Syscalls
  2. Internal kernel functions

Tracing methods

- Kprobes/Return Probes – Must resolve functions
- Tracepoints – Predefined hooks
  - Raw Tracepoints – Only for non-syscall or **`sys_enter`**/**`sys_exit`**
- Function Entry / Exit Probes – Added by the compiler

TELEMETRY SOURCES – LINUX

Linux kernel modules (LKM)

- Most products have a legacy LKM agent
- Crashes may cause a kernel panic

eBPF

- Modern systems (kernel 4.x) support eBPF
- Sometimes referred to a “user-mode” agent
- Restricted so it cannot crash the kernel

root@vm:~

```
[root@vm ~]#
```

SANDBOX DETECTION

OUTFLANK

clear advice with a hacker mindset

SANDBOX DETECTION

Common strategies

- Debugger checks
- Minimal hardware resources
- Small number of installed programs

Mixed Findings

- No debugger attached*
- Most have sufficient hardware
- Most have many installed programs

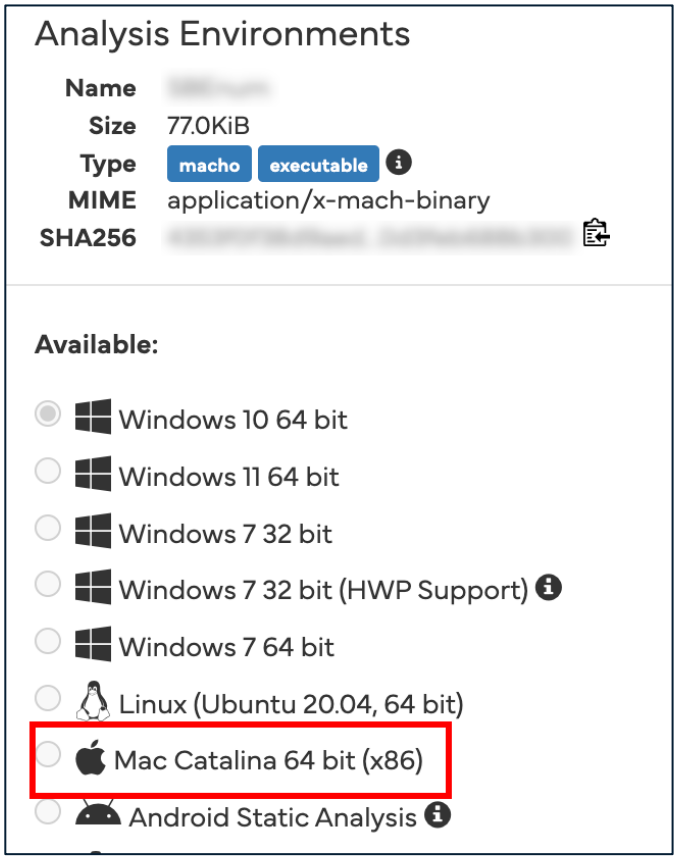
SANDBOX DETECTION

macOS Ideas:

- Look for small display
- Only target newer macOS
- Use ARM64 payloads

Linux Ideas:

- ???



CASE STUDY: SPOOFING LINUX SYSCALLS

OUTFLANK

clear advice with a hacker mindset

CASE STUDY – PREVIOUS RESEARCH

DEF CON 29 – Phantom Attack: Evading System Call Monitoring

- TOCTOU vulnerability in syscall tracing implementation
- Only targeted Falco and Tracee

CASE STUDY – TOCTOU

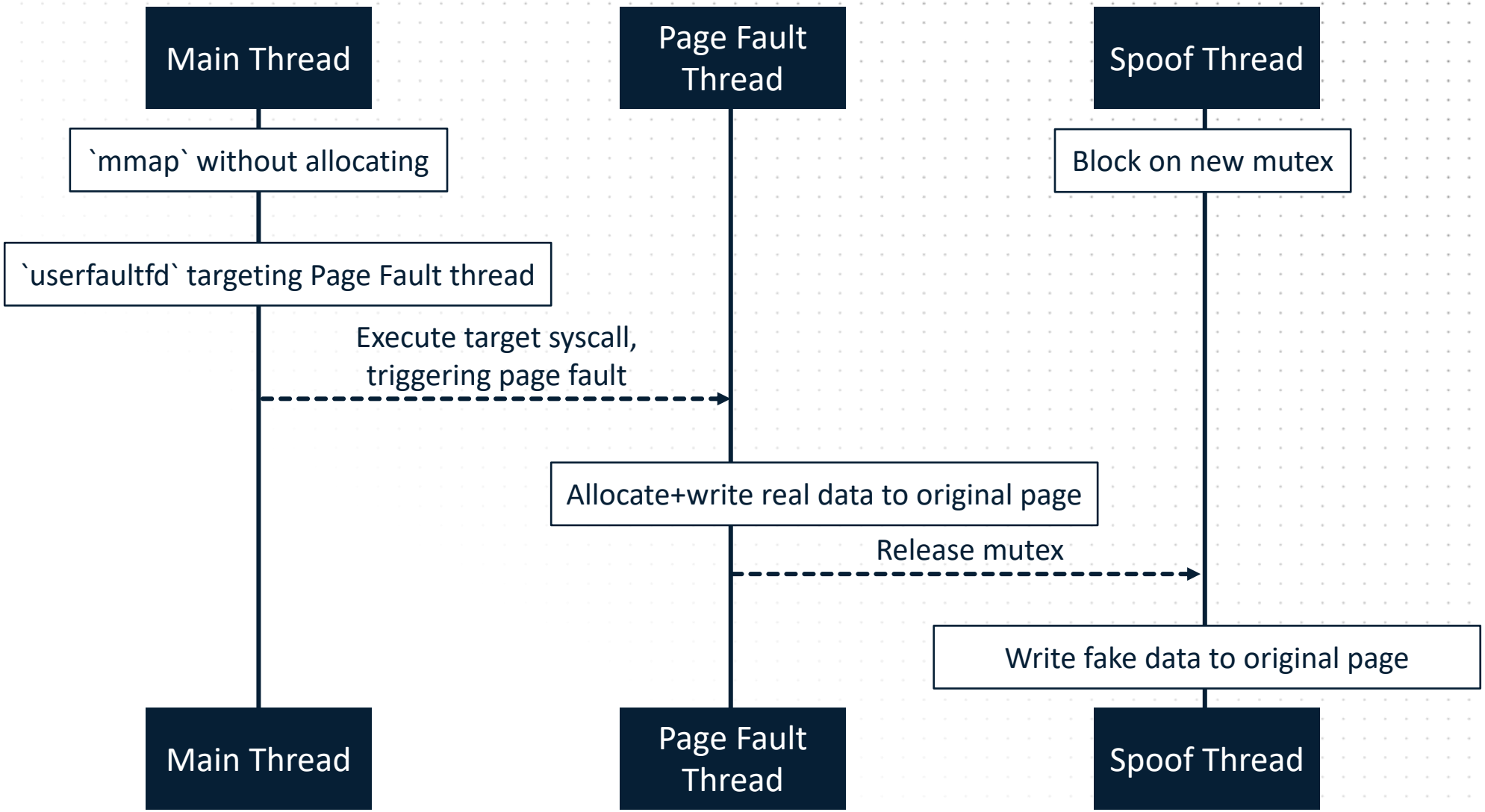
Checking something…

```
int connect(int fd, struct sockaddr __user *uservaddr, int addrlen) {
    // Entry tracing
    int ret = __sys_connect(fd, uservaddr, addrlen);
    // Exit tracing

    return ret;
}
```

```
int __sys_connect(int fd, struct sockaddr __user *uservaddr, int addrlen) {
    int ret = -EBADF;
    struct fd f;

    f = fdget(fd);
    if (f.file) {
        struct sockaddr_storage address;

        ret = move_addr_to_kernel(uservaddr, addrlen, &address);
        if (!ret)
            ret = __sys_connect_file(f.file, &address, addrlen, 0);
        fdput(f);
    }

    return ret;
}
```

Using something…

CASE STUDY – PHANTOM ATTACK



user@pwned:~/prod

```
[user@pwned prod]$ head -n 5 config.h
#pragma once

// connect
#define SPOOF_IP    "1.3.3.7"
#define REAL_IP     "142.251.116.101" // google.com
[user@pwned prod]$
```

OUTFLANK

clear advice with a hacker mindset