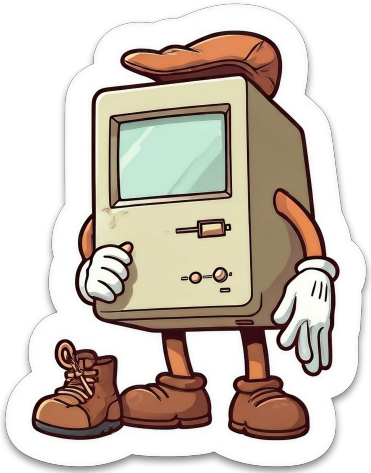


Mac, Where’s My Bootstrap?

Detecting XPC logic exploits

Csaba Fitzl
Principal Security Researcher @Kandji

Brandon Dalton
Senior Security Researcher @Microsoft

Brandon Dalton (@partyD0lphin)

Senior Security Researcher (Microsoft)

OS Internals and M365 research

Developer of Red Canary Mac Monitor

ATT&CK contributor

🏡 **Family >>**

Travel ✈️ / History 📚

Csaba Fitzl (@theevilbit)

kandji

Principal macOS Security Researcher @Kandji 🐝

author of EXP-312 - macOS Exploitation training (🐙) at OffSec

macOS bug hunter (~100 CVEs from 🍎)

husband, father

hiking, trail running 🥾 🏔️

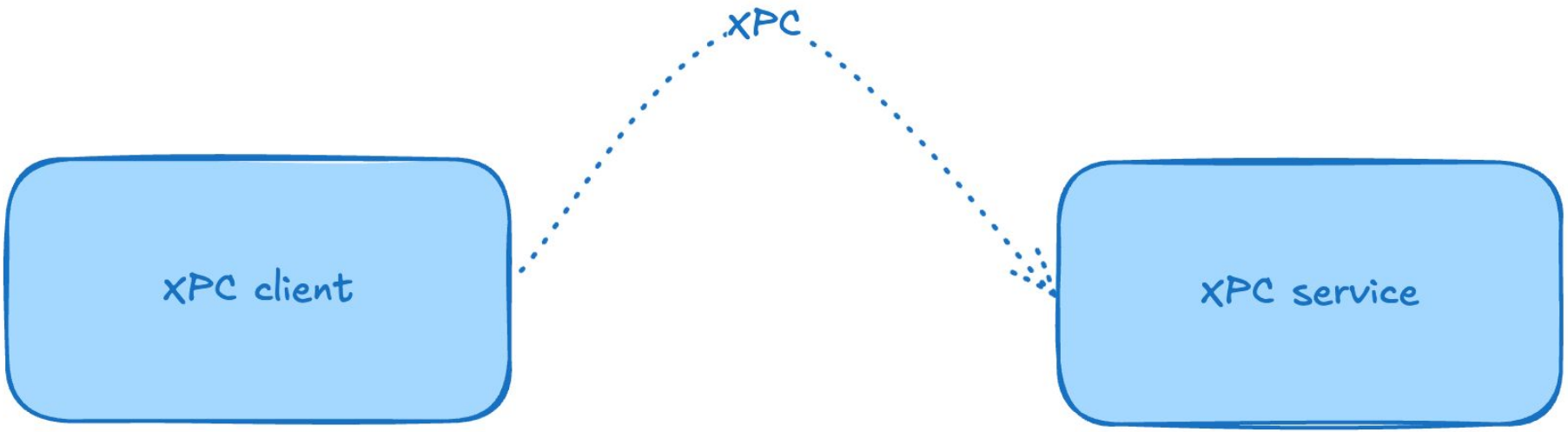
Over the next 25 minutes…

- XPC vulnerabilities and abuse
- Our idea
- `XPC_CONNECT` – what's missing? 😔
- The bootstrap server

- 🕵️ Resolving an XPC service path
- 🎁 Tool drop
- 🔥 Detecting a Twitch 0day
- ✅ Recommendations

How it started?

What is XPC?

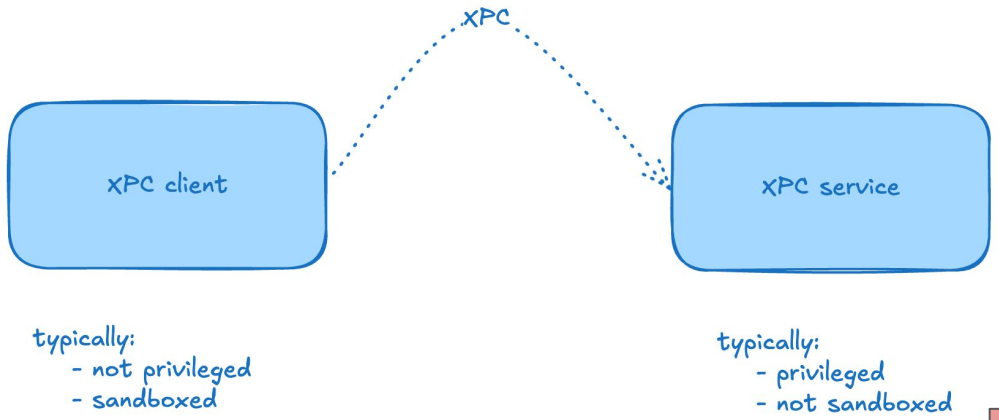


typically:
- not privileged
- sandboxed

typically:
- privileged
- not sandboxed

The XPC problem

- By default:
  - A global service reachable by every process
- If client access is not controlled ⇒ service is open for abuse
- Client validation is tricky
- Developers make mistakes…



Safe XPC client validation

Code signing validation

- Cert chain (Apple and developer) ⇒ to prevent fake root certificates
- Bundle ID(s) ⇒ to prevent other clients from the same vendor
- Team ID ⇒ to prevent other developers / apps
- App version ⇒ to prevent downgrade attacks

Verify clients entitlements

- Hardened runtime ⇒ to prevent client code injection attacks
- Library validation ⇒ to prevent client code injection attacks

Identify client process

- Use Audit token ⇒ to prevent PID reuse attacks

XPC vulnerability scope

Impact

1. Local privilege escalation
2. Data compromise

Prevalence

Relatively easy for an attacker to both **identify** and **exploit**.

The problem…

Exploiting XPC in AntiVirus

Home > Techniques > Enterprise > Inter-Process Communication > XPC Services

Inter-Process Communication: XPC Services

Other sub-techniques of Inter-Process Communication (3)

Exploiting GOG Galaxy XPC service

for Mac Privilege Escalation via a Legacy Package

OFFENSIVECON BY Binary Gecko

REGISTER SPEAKERS CFP AGENDA VENUE TRAININGS SPONSORS ARCHIVE CONTACT

TYLER BOHAN

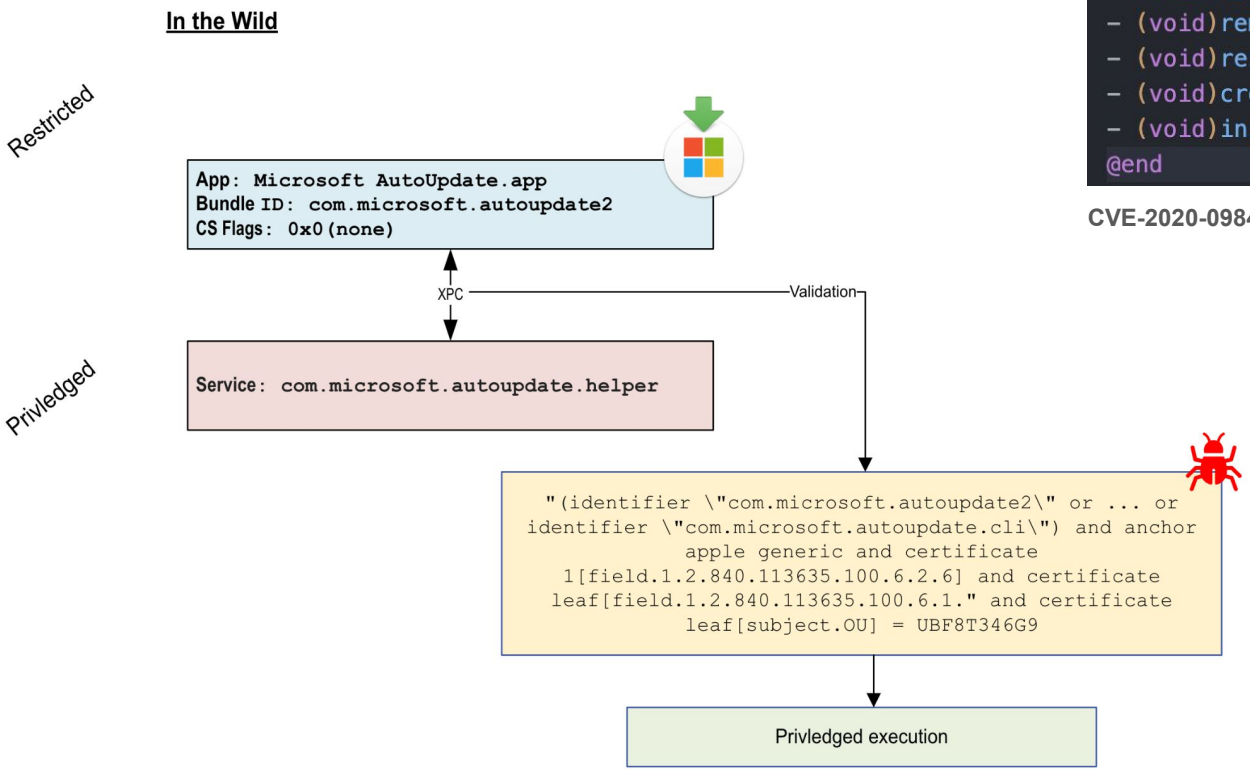
OSX XPC REVISITED - 3RD PARTY APPLICATION FLAWS

Author: Zhipeng Huo(@R3dF09) of Tencent Security Xuanwu Lab

Validation Bypass

In `/Library/PrivilegedHelperTools/com.microsoft.autoupdate.helper` there's a XPC service `com.microsoft.autoupdate.helper`.

For example…




```
@protocol MAUHelperToolProtocol <NSObject>
- (void)removeInstallLogFile:(NSString *)ar
- (void)logString:(NSString *)arg1 atLevel:
- (void)removeClone:(NSString *)arg1 withRe
- (void)restoreCloneToAppInstallLocation:(N
- (void)createCloneFromApp:(NSString *)arg1
- (void)installUpdateWithPackage:(NSString
@end
```

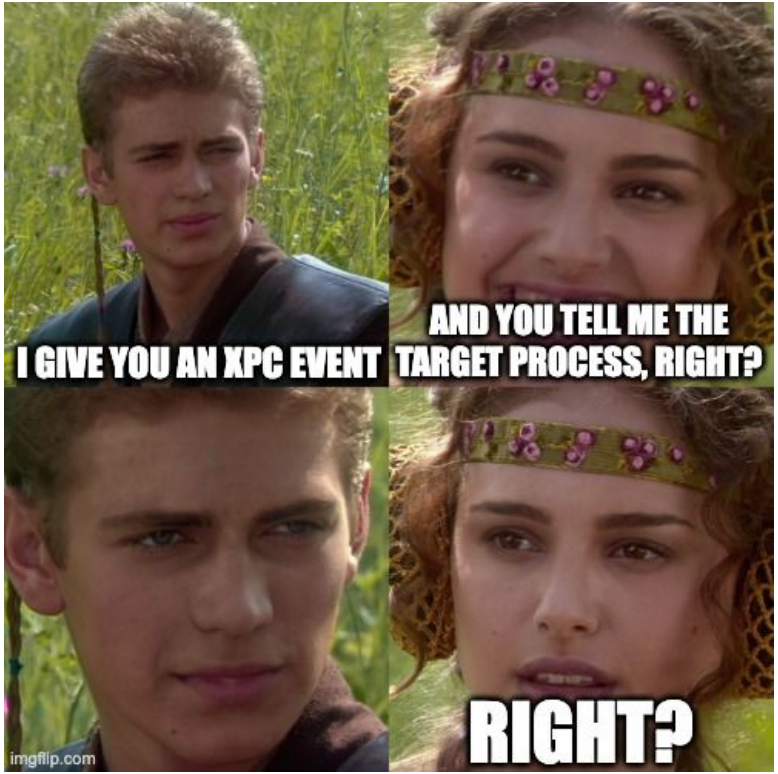
CVE-2020-0984 – Microsoft AutoUpdate

The Idea 

Detect XPC Attacks using Endpoint Security

- 💡 We have an XPC ES event (thanks Sonoma)!
- 💡 Let’s monitor client - service validation with an ES client, e.g.:
  - 💡 Verify TeamID/etc… for third party

⇒ requires code signing validation of both sides of the connection


I GIVE YOU AN XPC EVENT
AND YOU TELL ME THE TARGET PROCESS, RIGHT?
RIGHT?
imgflip.com

Endpoint Security… what do we have?


```
/**
 * @brief Notification for an XPC connection being established to a named service.
 *
 * @field service_name        Service name of the named service.
 * @field service_domain_type The type of XPC domain in which the service resides in.
 *
 * @note This event type does not support caching (notify-only).
 */
typedef struct {
    es_string_token_t service_name;
    es_xpc_domain_type_t service_domain_type;
} es_event_xpc_connect_t;
```

```
/**
 * @brief This enum describes the types of XPC service domains.
 */
typedef enum {
    ES_XPC_DOMAIN_TYPE_SYSTEM = 1,
    ES_XPC_DOMAIN_TYPE_USER,
    ES_XPC_DOMAIN_TYPE_USER_LOGIN,
    ES_XPC_DOMAIN_TYPE_SESSION,
    ES_XPC_DOMAIN_TYPE_PID,
    ES_XPC_DOMAIN_TYPE_MANAGER,
    ES_XPC_DOMAIN_TYPE_PORT,
    ES_XPC_DOMAIN_TYPE_GUI,
} es_xpc_domain_type_t;
```

XPC connection details

- **Service name:** `com.apple.audio.SandboxHelper`
- **Domain type:** `PID`

Mac Monitor

```
"event": {
  "xpc_connect": {
    "service_name": "com.apple.system.opendirectoryd.api",
    "service_domain_type": 1
  }
},
```

eslogger

 Let’s take it out for a test drive…

What will not trigger an XPC connect event?

`bootstrap_look_up(...)`

Create a Mach service and register w/launchd

```c
#include <mach/mach.h>
#include <servers/bootstrap.h>

int main() {
    mach_port_t port;
    mach_port_allocate(mach_task_self(), MACH_PORT_RIGHT_RECEIVE, &port);
    mach_port_insert_right(mach_task_self(), port, port, MACH_MSG_TYPE_MAKE_SEND);
    bootstrap_register(bootstrap_port, "com.microsoft.domain-example", port);
```

What launchd knows...

```
user/501 = {
  type = user
  handle = 501
  ...
  creator = bluetoothd[397]
  creator euid = 0
  session = Background
  gui asid = 100016
  security context = {…
  }

  bringup time = 112 ms
  death port = 0x0
  subdomains = {
    gui/501
  }

  services = {…
  }

  unmanaged processes = {
  }

  endpoints = {
    0x32703    M   D   com.apple.lsd.openurl
    0x133b87   U   A   com.microsoft.domain-example
    0xa8e23    M   A   2BUA8C4S2C.com.1password.browser-helper
    0xb0c5f    U   A   cs (Apple)_OpenStep
  }
```

Looking up a Mach service name will not trigger an ES event

```c
#include <mach/mach.h>
#include <servers/bootstrap.h>

int main() {
    mach_port_t port;
    bootstrap_look_up(bootstrap_port, "com.microsoft.domain-example", &port);
}
```

Endpoint Security (ES) event

```
"xpc_connect": {
    "service_domain_type": 2,
    "service_name": "com.apple.bg.system.task"
}
```

1. Create an launchd (XPC) service

```c
#include <xpc/xpc.h>

int main() {
    xpc_connection_t service = xpc_connection_create_mach_service(
        "com.microsoft.xpc-domain-example",
        NULL,
        XPC_CONNECTION_MACH_SERVICE_LISTENER
    );
```

launchd service plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.microsoft.xpc-domain-example</string>
    <key>MachServices</key>
    <dict>
        <key>com.microsoft.xpc-domain-example</key>
        <true/>
    </dict>
```

2. Register

What launchd knows...

```
gui/501 = {
    type = login
    handle = 100016
    active count = 431
    service count = 430
    active service count = 257
    creator = loginwindow[407]
    creator euid = 0
    session = Aqua
    endpoint destination = com.apple.xpc.launchd.domain.user.501
    auxiliary bootstrapper = com.apple.xpc.otherbsd (complete)
    security context = {
        uid = 501
        asid = 100016
    }

    bringup time = 174 ms
    death port = 0x10d03

    environment = {
        SSH_AUTH_SOCK => /private/tmp/com.apple.launchd.7hOFG3GZg3/Listeners
    }

    services = {
            30309      -     com.microsoft.xpc-domain-example
            18128      -     application.com.apple.Preview.115292150031189733 2.
                0      0     com.google.GoogleUpdater.wake
    }

    unmanaged processes = {-
    }

    endpoints = {
    }

    externally-hosted endpoints = {
        0x3ce9b    M   A   com.microsoft.xpc-domain-example
        0xb0c5f    U   A   Multilingual (Apple)_OpenStep
        0xb0c5f    U   A   vi (Apple)_OpenStep
        0xb0c5f    U   A   ko (Apple)_OpenStep
    }
```

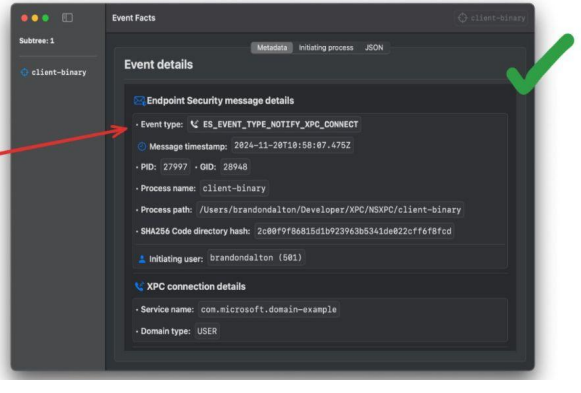
3. Client request

Looking up the service will trigger the ES event

```c
#include <xpc/xpc.h>

int main() {
    xpc_connection_t connection = xpc_connection_create_mach_service(
        "com.microsoft.xpc-domain-example",
        NULL,
        0
    );
```

4. ES event



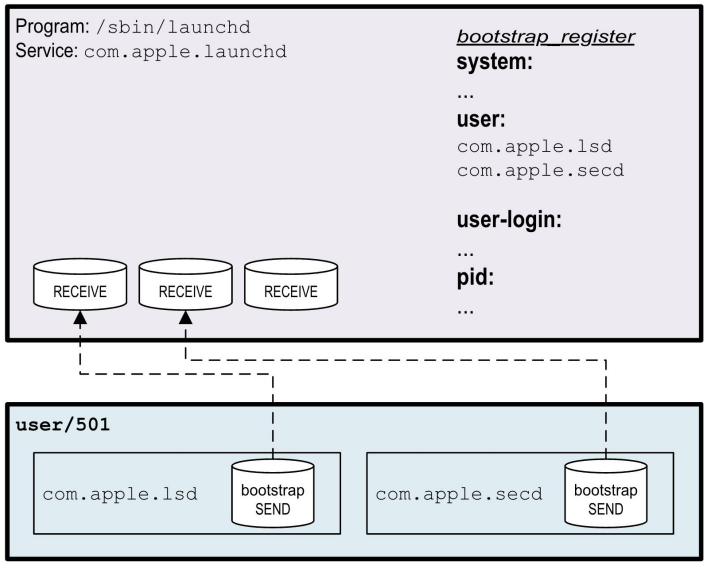
Great! Now where's the path...?

 … the bootstrap server

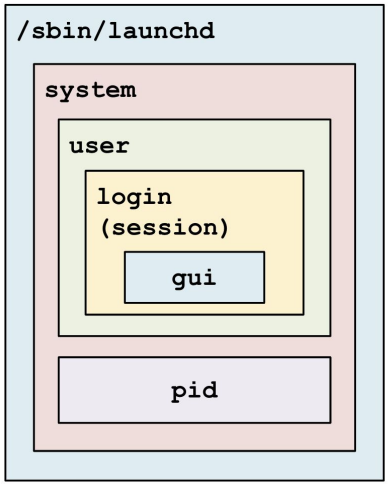
/sbin/launchd

"Domains are, in effect, nothing more than figments of launchd's twisted imagination." - MOXiL Vol 1 pg.438

- Manager for jobs and XPC / Mach service
- **… 👀 do you know where the program is?**



Bootstrap namespaces / domains




LAUNCHD KNOWS
ALL THE SERVICE PATHS
IT'S EASY TO
LOOK IT UP, RIGHT?
RIGHT?
imgflip.com

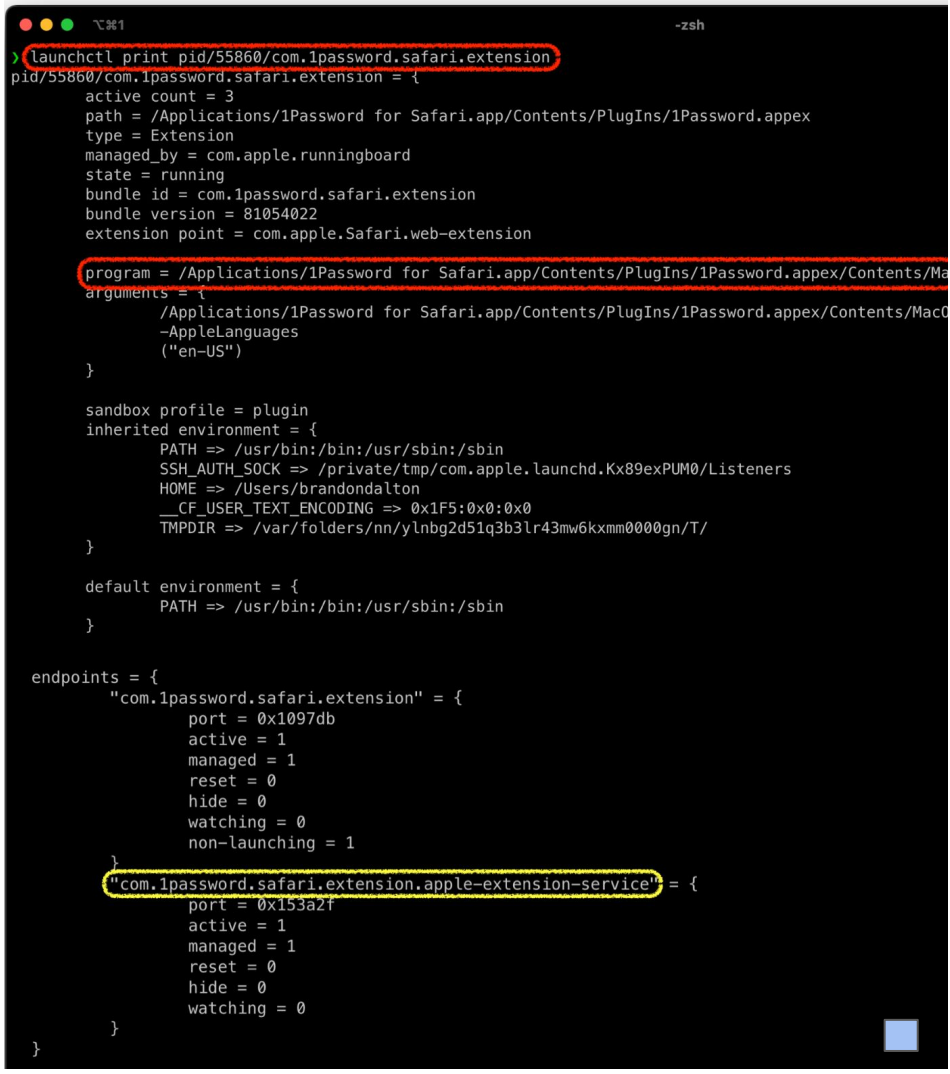
The idea

`/usr/bin/launchctl`
Seems to know how to get all the information we need…

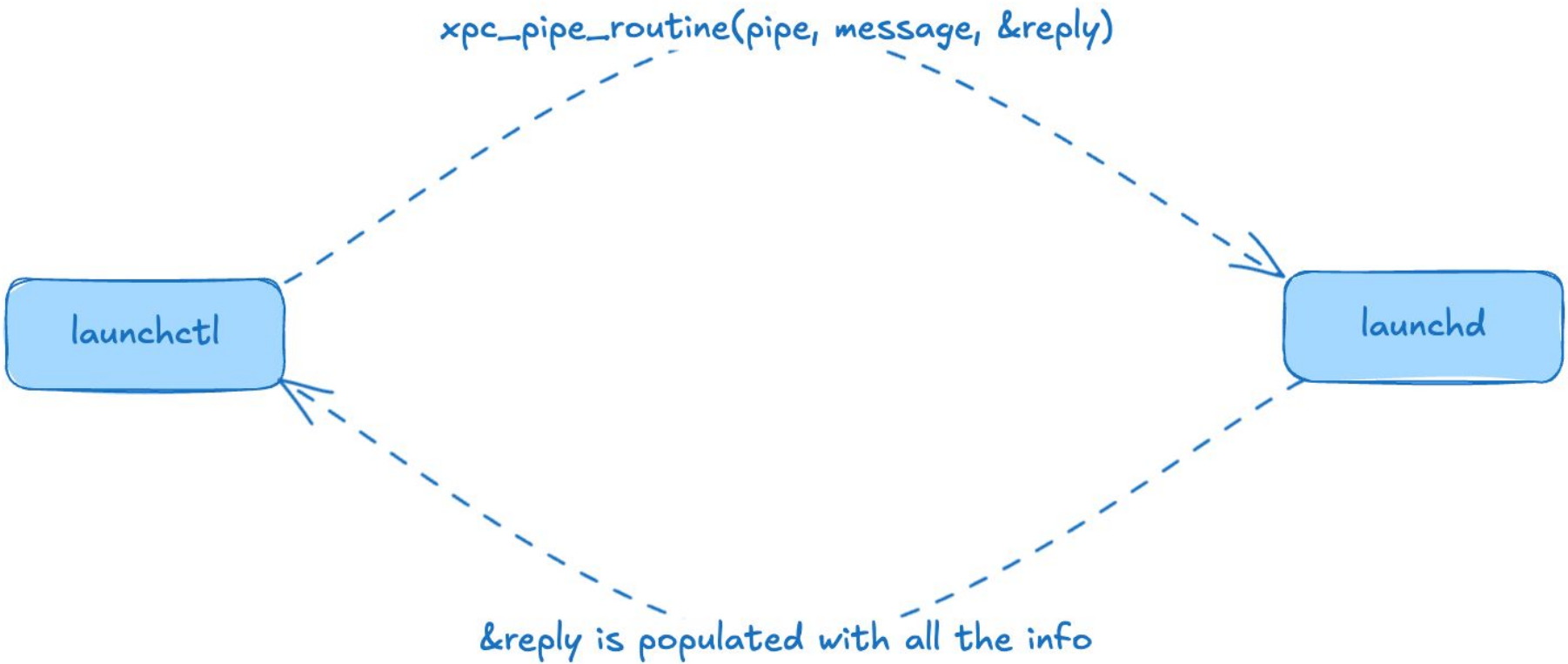
What used to help?

Jonathan Levin's launjctl - https://newosxbook.com/articles/jlaunchctl.html

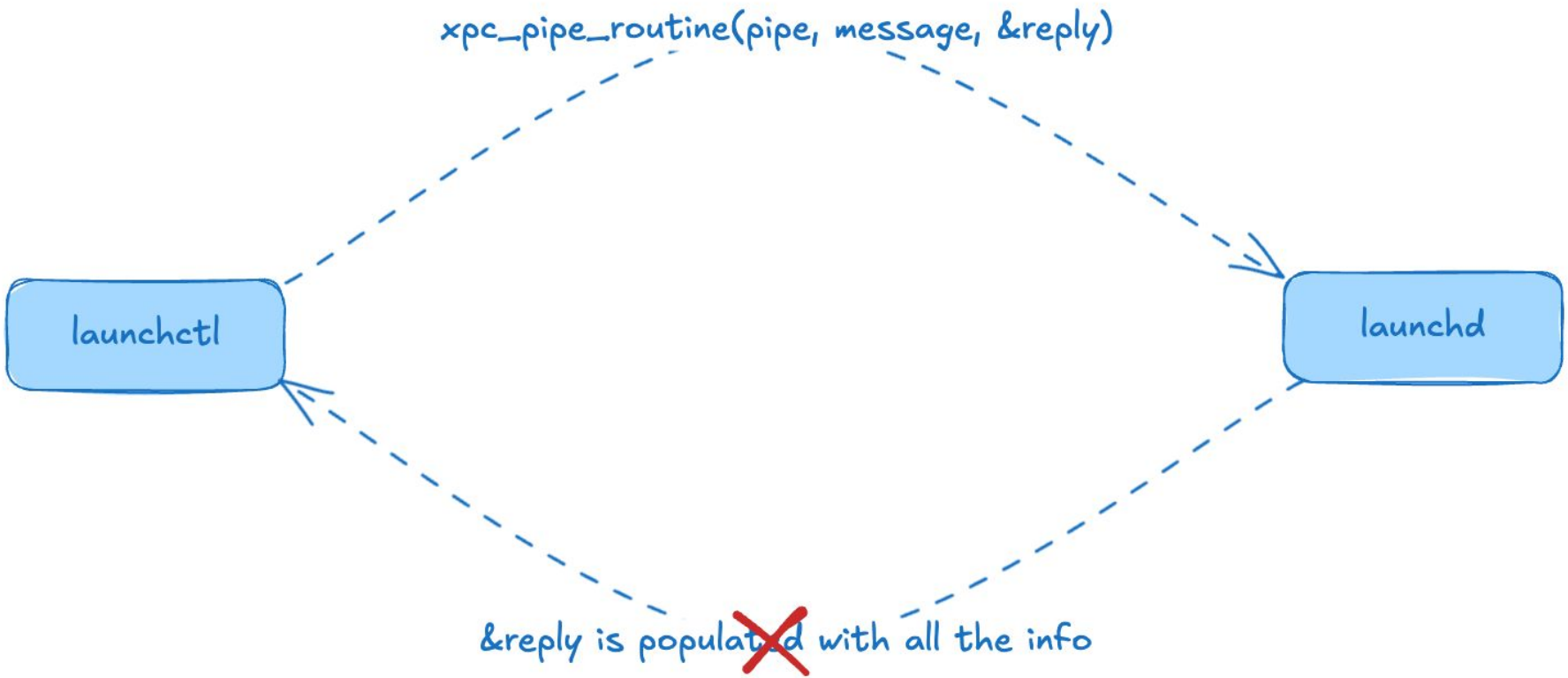
Apple broke it :(



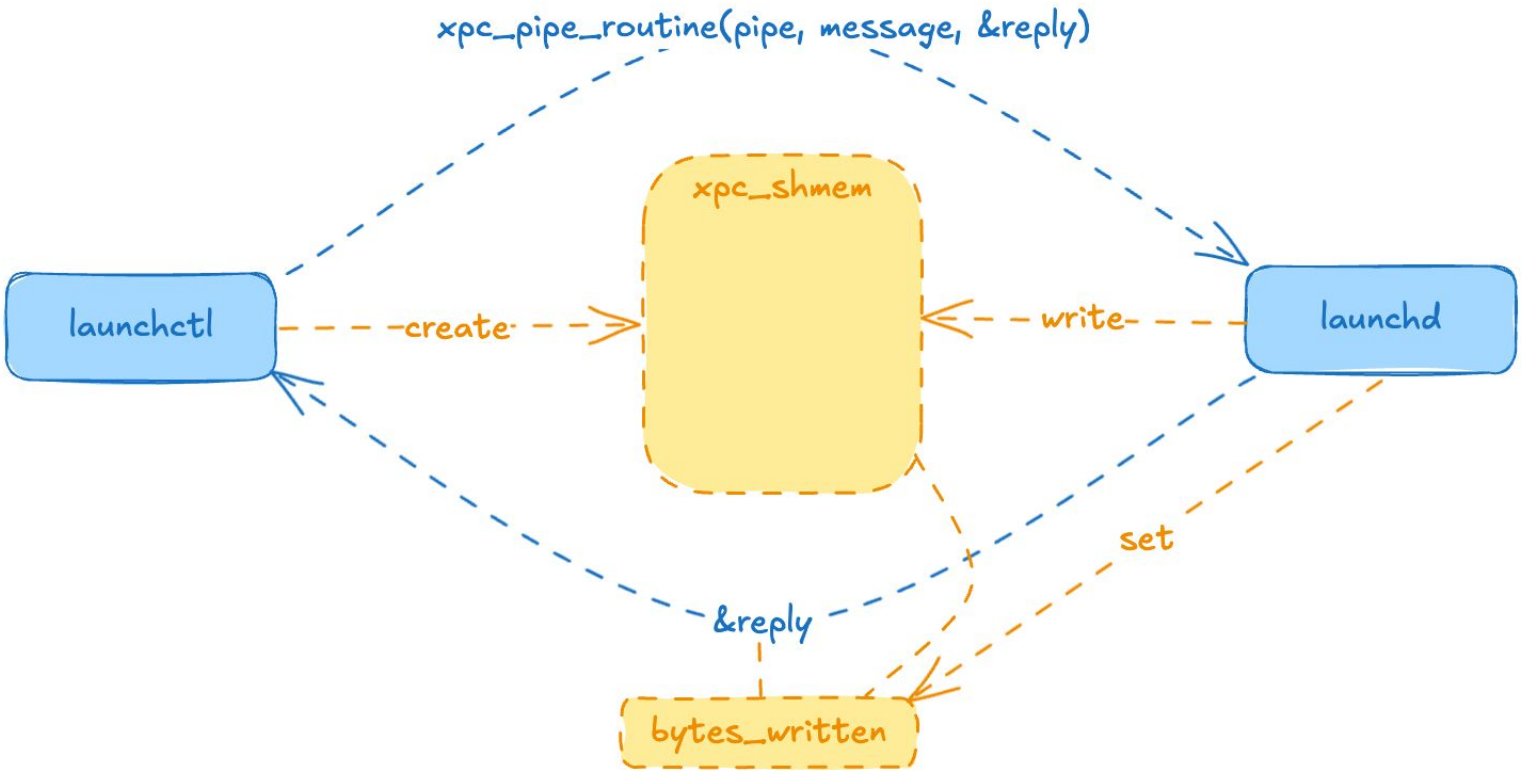
Before


xpc_pipe_routine(pipe, message, &reply)
launchctl
launchd
&reply is populated with all the info

We have an issue…



Then how does `launchctl` do it?



Incoming request:
1) print service domain target
2) print domain target

Create an XPC dictionary using known keys and Create a shared memory region

```
let message = xpc_dictionary_create(nil, nil, 0)
xpc_dictionary_set_uint64(message, "handle", handle)
xpc_dictionary_set_uint64(message, "type", type)
xpc_dictionary_set_uint64(message, "routine", routine)
xpc_dictionary_set_uint64(message, "subsystem", subsystem)
if let name = name {
    xpc_dictionary_set_string(message, "name", name)
}

allocateSharedMemory(for: message)
```

Send the message and parse the response

```
let pipe = xpc_pipe_create_from_port(bootstrapPort, 0)
var reply: xpc_object_t?
let error = xpc_pipe_routine(pipe, message, &reply)
// ...
return parseResponse(from: reply ...)
```

Why do this to us? Apple must know it's a barrier to entry…

Parse... "this"... 😅

```
gui/501 = {
        type = login
        handle = 100016
        active count = 435
        service count = 434
        active service count = 259
        creator = loginwindow[407]
        creator euid = 0
        session = Aqua
        // ...

        services = {
                    65033      -         application.com.vector35.binaryninja.119897822.128734987
                      672      -         com.apple.syncdefaultsd
                      745      -         com.apple.assistantd
                        0      0         com.apple.DataDetectorsLocalSources
                        0      0         com.apple.unmountassistant.useragent
                     9903   (pe)         com.apple.mlhostd
                        0      -         com.apple.SafariHistoryS
```

What are those special keys?

MOXiI Vol 1 Table 13-17

Using the **bootstrap port** we can request info from `launchd` using **XPC dictionaries.**

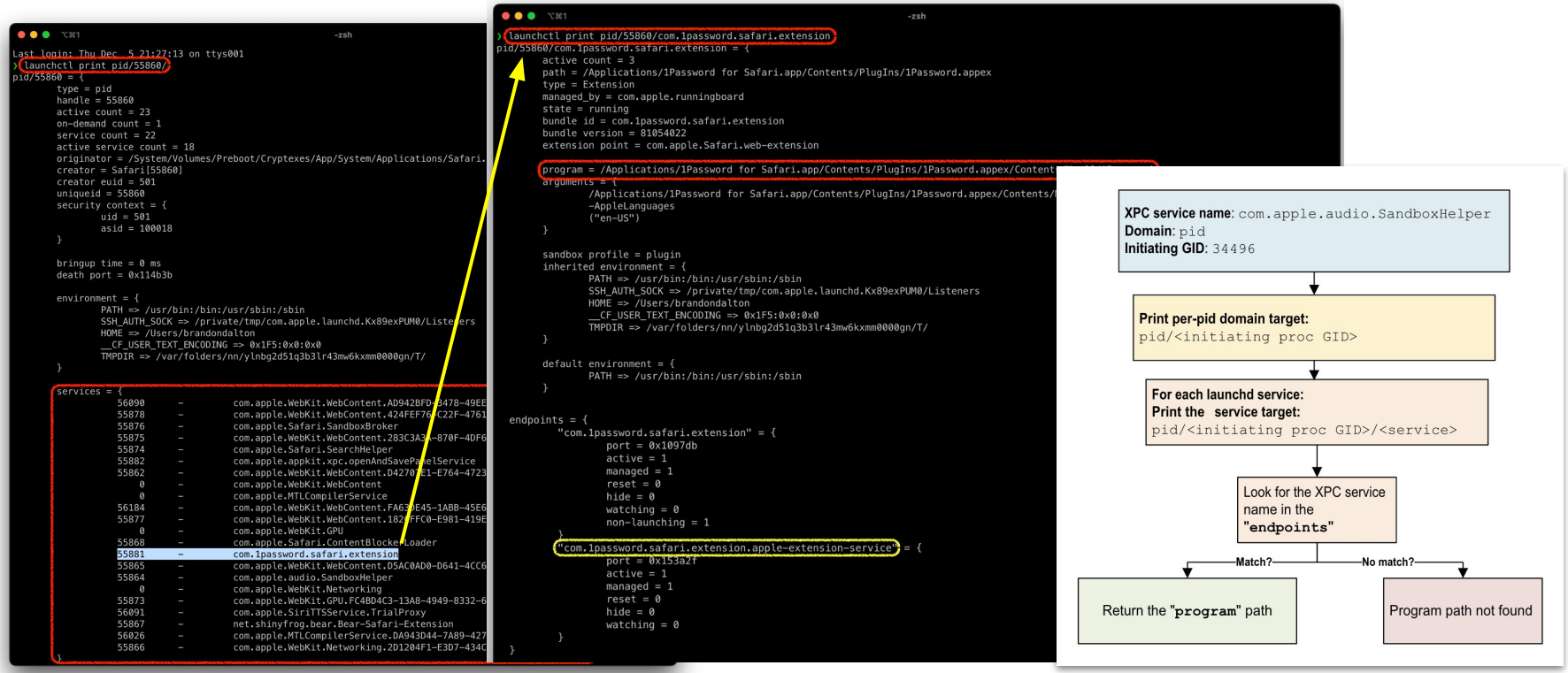
- **`type`**: The domain we're targeting.
  - `system`, `user`, `login`, `pid`, `gui`
- **`subsystem`**: Service or domain targets
- **`handle`**: The domain specifier
  - E.g. UID, ASID, or PID
- **`routine`**: A specific command in subsystem
- and **`name`**: The service name if service target

```
xpc_pipe_create_from_port(bootstrap,;)

xpc_pipe_routine(pipe,dictionary,&reply)
```

Resolving a path? (PID domain example)

Initiating GID: 55860 →
com.1password.safari.extension → com.1password.safari.extension.apple-extension-service

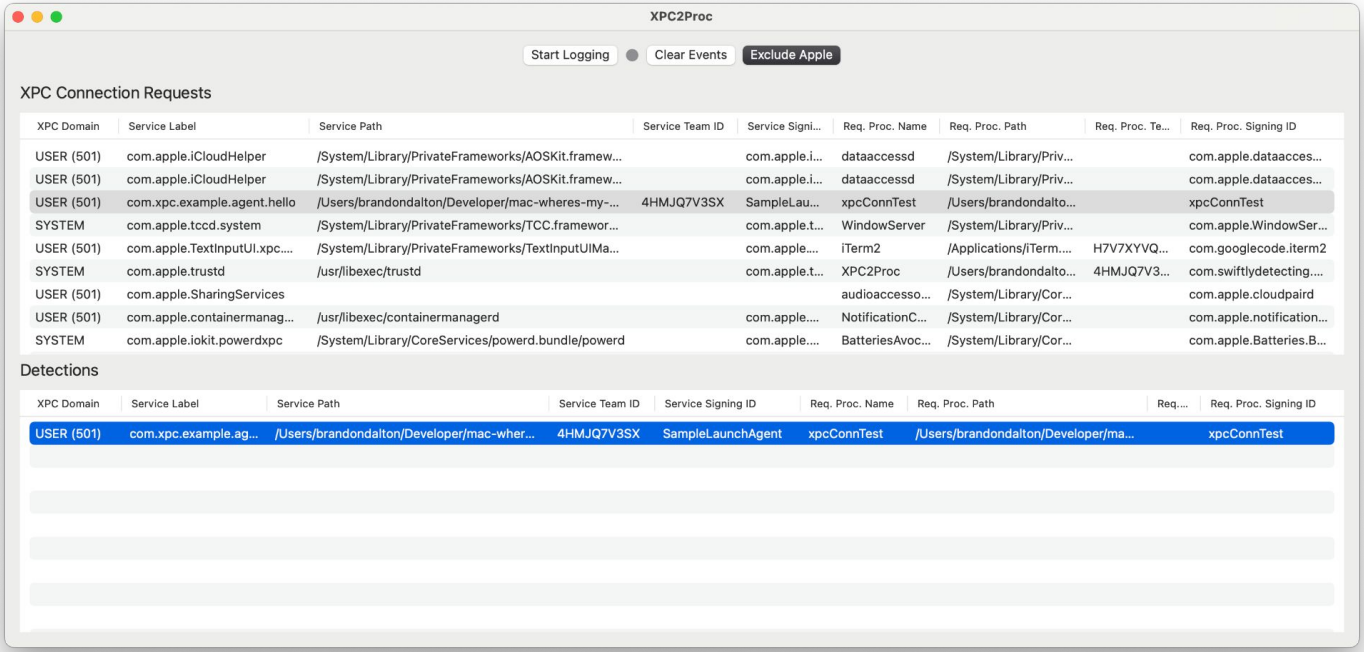


Detection not AUTH

github.com/Brandon7CC/mac-wheres-my-bootstrap/releases/

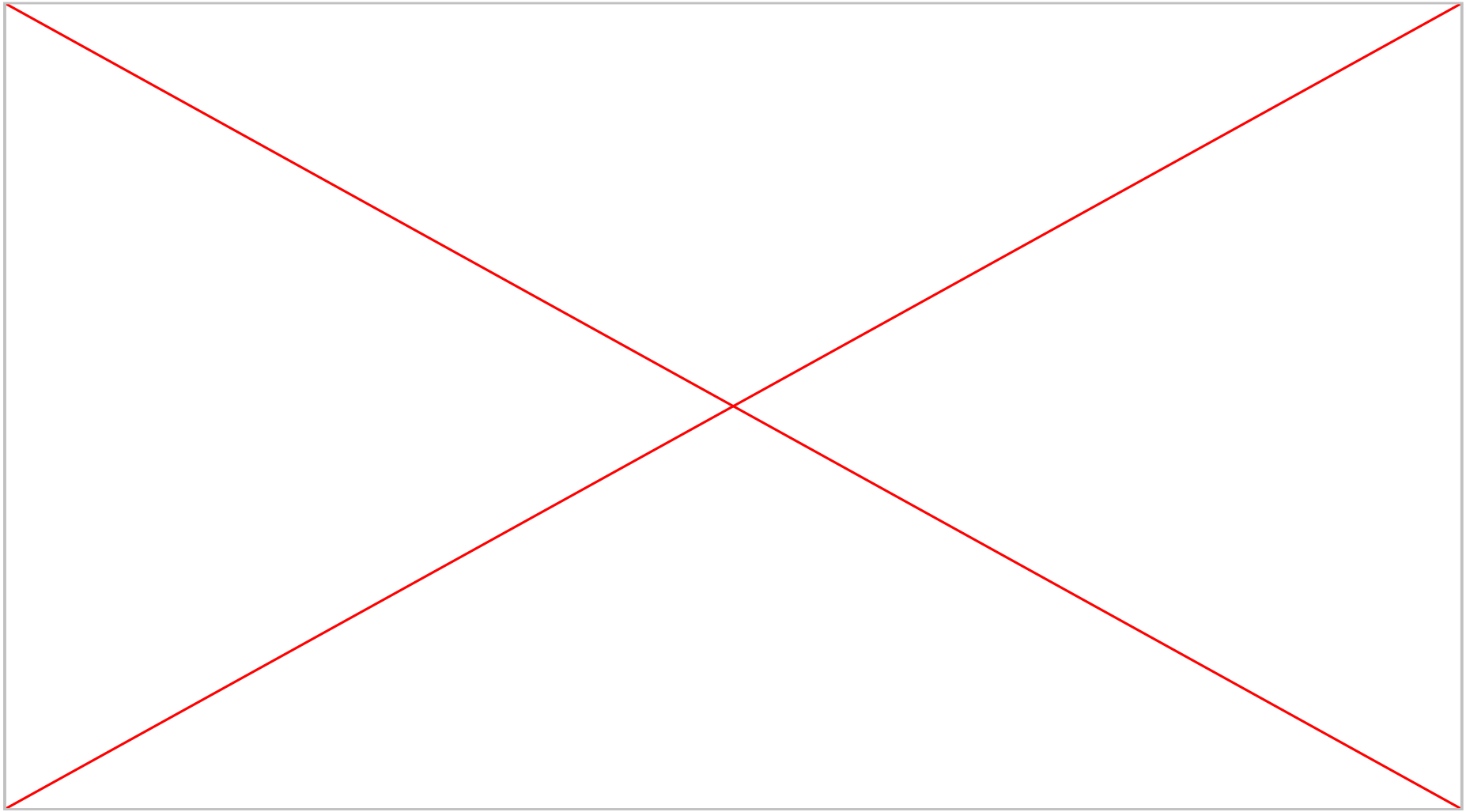
```
!service.is_apple && service.team_id != requestor.team_id
```

Distribution *entitled*



Detecting a Twitch 0day* 🔥

* https://www.kandji.io/blog/twitch-privileged-helper *by Chris Lopez* 🐝



This is not an ending note…

🙏 **Apple, help us detect XPC exploits and keep users safe!**

`es_event_xpc_connect_t`

- Provide the hosting program path of the XPC service attempting to be connected to.
- Include code signing information for the process hosting the XPC connection (Team ID, etc).

“Just one more thing”

macOS Vulnerability Research Training
by @theevilbit

Coming in
Late 2025 / Early 2026

❤️ Thank you!

github.com/Brandon7CC/**mac-wheres-my-bootstrap**/releases/