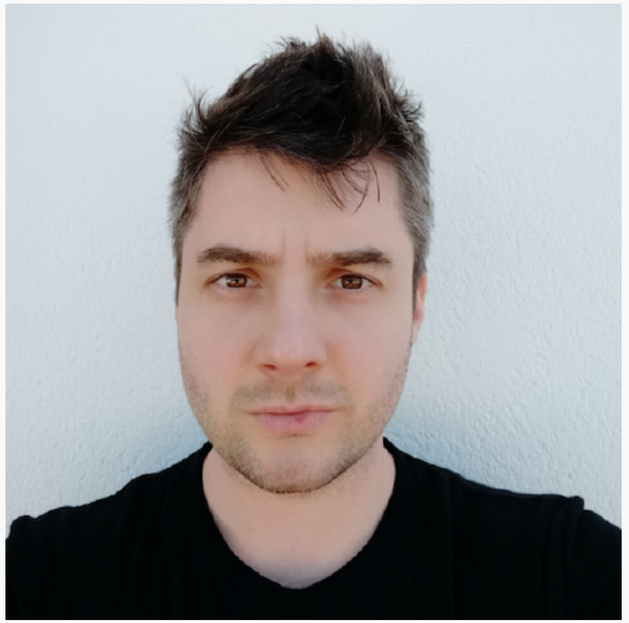
Crash One - A Starbucks Story
CVE-2025-24277

Csaba Fitzl
X: @theevilbit

Gergely Kalman
X: @gergely_kalman

whoami - Gergely

- Independent bug hunter
- ex-dev / ex-sysadmin
- 20+ 0days on macOS
- lots of file op / filesystem research
- 📖: https://gergelykalman.com



whoami - Csaba

- Principal macOS Security Researcher @Kandji 🐝
- author of EXP-312 - macOS Exploitation training (🐙) at OffSec
- macOS bug hunter (100+ CVEs)
- hiking, trail running 🥾 🏔️

agenda

1. The Wall
2. The Light
3. Sandbox Extensions
4. Consuming the Token
5. Back-channel
6. The mighty rename
7. Weaponization Strategy
8. ACL Inheritance
9. Surviving fchown
10. Putting it together
11. Sandbox Escape
12. The Fix
13. Wrap-Up

Disclaimer: Starbucks didn't sponsor this talk

ONE YEAR AGO
IN A COUNTRY
NOT SO FAR
AWAY...


STARBUCKS


THE WALL

osanalyticshelperd

- responsible for creating crash logs
  - ~/Library/Logs/DiagnosticReports - for user owned processes
  - /Library/Logs/DiagnosticReports - for root owned processes

- runs as root
  - potentially exploitable: writes to user owned directory

file system events

```
17:04:48.619044  stat64                                  /Users/pirate/Library/Logs/DiagnosticReports
17:04:48.619055  open              F=3        (R__________)  /Users/pirate/Library/Logs/DiagnosticReports

(...)

17:04:48.641041  open_dprotected   F=3        (_WC__E______)  /Users/pirate/Library/Logs/DiagnosticReports/.crash-2024-09-17-170448.ips

(...)

17:04:48.641064  fchown            F=3
17:04:48.641078  fchmod            F=3   <rw------->
17:04:48.641145  write             F=3    B=0x15b
17:04:48.641157  write             F=3    B=0x1

(...)

17:04:48.978892  rename                                 /Users/pirate/Library/Logs/DiagnosticReports/.crash-2024-09-17-170448.ips
```

problems

- the process checks if the directory is a symlink (not shown)
  - doesn't err out 🤷‍♂️
- open uses **O_EXCL | O_CREAT** -> if file exists won't be created
- sandbox profile: **/System/Library/Sandbox/Profiles/com.apple.osanalyticshelper.sb**
  - doesn't allow writing anywhere
- just trying to redirect with symlink or hardlink doesn't work


THE LIGHT

the 1 thing we missed

```
(with-filter (extension "com.apple.osanalytics-sandbox.read-write")
  (allow file-read* file-write*))
```


SANDBOX
EXTENSIONS

Sandbox extensions

- **Signed token in the form of a C string**
- Allows dynamic **expansion on the process' current privileges**
  - typically to access files or services
- flow:
  - process A with access issues a token
  - process A send the token to the sandbox process B
  - process B consumes the token
  - (process B releases the token)

Example API

```
char* sandbox_extension_issue_file_to_self(const char *sandboxEnt, const char *filePath, int flags);
char *sandbox_extension_issue_file(const char *extension_class, const char *path, uint32_t flags);
int sandbox_extension_consume(const char *token);
char* sandbox_extension_issue(const char *ext, int type, int flags, const char *subject);
char *sandbox_extension_issue_generic(const char *extension_class, uint32_t flags);
char *sandbox_extension_issue_file_to_process_by_pid(char *extension_class, const char *path, uint32_t flags, pid_t);
```

API

- We **issue a token for /** -> valid for the whole file system
- com.apple.osanalytics-sandbox.read-write
- we need osanalyticshelperd consuming it

```
// Call the sandbox extension API, here we issue a sb extension for the filesystem
    char *extension_token = sandbox_extension_issue_file_to_process_by_pid(
        extension_class,  // Type of extension (read-write)
        "/",         // Path to the file or directory
        flags,           // Additional flags (set to 0)
        target_pid       // PID of the process to grant access
    );
```


CONSUMING
THE TOKEN

consume

```
__int64 OSASandboxConsumeExtension()
{
  return _OSASandboxConsumeExtension();
}
```

```
__int64 __fastcall OSASandboxConsumeExtension(id a1, __int64 a2)
{
  __int64 v2; // rbx
  id v3; // rax
  __int64 v4; // rax
  double v5; // xmm0_8
  __int64 v6; // rax
  __int64 v7; // r14

  v2 = MEMORY[0x7FF94007AC88](a2, a2);
  v3 = j__objc_retainAutorelease_2(a1);
  v5 = MEMORY[0x7FF94007AC20](v3, "UTF8String");
  if ( !v4 )
  {
    if ( j__os_log_type_enabled_5(MEMORY[0x7FF940000998], OS_LOG_TYPE_ERROR) )
      initHardwareInfo_cold_1_60(v5);
    goto LABEL_10;
  }
  v6 = j__sandbox_extension_consume_0(v4);
  if ( v6 < 0 )
  {
    if ( j__os_log_type_enabled_5(MEMORY[0x7FF940000998], OS_LOG_TYPE_ERROR) )
      OSASandboxConsumeExtension_cold_2();
LABEL_10:
    (*(void (__fastcall **)(__int64, double))(v2 + 16))(v2, v5);
    return MEMORY[0x7FF94007AC80](v2);
  }
  v7 = v6;
  (*(void (__fastcall **)(__int64, double))(v2 + 16))(v2, v5);
  if ( (int)j__sandbox_extension_release_0(v7) < 0
    && j__os_log_type_enabled_5(MEMORY[0x7FF940000998], OS_LOG_TYPE_ERROR) )
  {
    OSASandboxConsumeExtension_cold_3();
  }
  return MEMORY[0x7FF94007AC80](v2);
}
```

```
void __fastcall func_handle_createForSubmissionWithXPCRequest(__int64 a1)
{

(...)

  if ( *(_QWORD *)(a1 + 32) )
  {
    v2 = kOSALogMetadataBugType;
    string = xpc_dictionary_get_string(
               *(xpc_object_t *)(a1 + 40),
               (const char *)objc_msgSend(kOSALogMetadataBugType, "UTF8String"));

(...)

    v5 = xpc_dictionary_get_string(*(xpc_object_t *)(a1 + 40), "caller");

(...)

      ((void (__fastcall *)(void *, __int64 *))OSASandboxConsumeExtension)(v24, v43);

(...)

}
```

XPC

- massive function
- lots of entries
- would take a lot to reverse
- let's crash something and sniff xpc

```
int main(int argc, const char * argv[]) {
    char* a = 0;
    *a = 0x41;
}
```

```
<OS_xpc_dictionary: dictionary[0x62e0ec000]: { refcnt = 1, xrefcnt = 1, subtype = 1, count = 6, transport = 0, dest port = 0x280b, dest
msg id = 0x280b, transaction = 1, voucher = 0x10066f0b0 } <dictionary: 0x62e0ec000> { count = 6, transaction: 1, voucher = 0x10066f0b0,
contents =
    "options" => <dictionary: 0x62e058660> { count = 9, transaction: 0, voucher = 0x0, contents =
        "capture-time" => <int64: 0x9490188b51478dff>: 748511703
        "LogType" => <string: 0x62e0282d0> { length = 9, contents = "309_crash" }
        "OSASandboxExtensionKey" => <string: 0x62e028330> { length = 227, contents =
"5b9d563288ed30916e04895f6208da02f0312497a30e3218718267010173e4fbb;00;00000000;00000000;00000000;0000000000000028;com.apple.osanalytics-
sandbox.read-write;01;01000016;0000000000131e3f;01;/users/tree/library/logs/diagnosticreports" }
        "observer_info" => <dictionary: 0x62e0586c0> { count = 7, transaction: 0, voucher = 0x0, contents =
            "frames" => <array: 0x62e0282a0> { count = 2, capacity = 2, contents =
                0: <dictionary: 0x62e058600> { count = 4, transaction: 0, voucher = 0x0, contents =
                    "imageIndex" => <int64: 0x9490188a35ac8347>: 0
                    "symbol" => <string: 0x62e0289c0> { length = 4, contents = "main" }
                    "imageOffset" => <int64: 0x9490188a35ad7f87>: 16280
                    "symbolLocation" => <int64: 0x9490188a35ac8387>: 24
                }
                1: <dictionary: 0x62e0585a0> { count = 4, transaction: 0, voucher = 0x0, contents =
                    "imageIndex" => <int64: 0x9490188a35ac834f>: 1
                    "symbol" => <string: 0x62e0289f0> { length = 5, contents = "start" }
                    "imageOffset" => <int64: 0x9490188a35af90e7>: 25204
                    "symbolLocation" => <int64: 0x9490188a35acdb87>: 2840
                }
            }
            "time" => <int64: 0x9490188b51478dff>: 748511703
            "images" => <array: 0x62e028240> { count = 5, capacity = 5, contents =
                0: <dictionary: 0x62e058540> { count = 7, transaction: 0, voucher = 0x0, contents =
                    "source" => <string: 0x62e028270> { length = 1, contents = "P" }
                    "arch" => <string: 0x62e028390> { length = 5, contents = "arm64" }
                    "base" => <int64: 0x94901882313c8347>: 4304535552
                    "name" => <string: 0x62e028360> { length = 5, contents = "crash" }
                    "size" => <int64: 0x9490188a35ae8347>: 16384
                    "path" => <string: 0x62e028540> { length = 17, contents = "/Users/USER/crash" }
                    "uuid" => <string: 0x62e0283c0> { length = 36, contents = "0e85d332-d459-362a-a6f2-0cdc6a99066f" }
                }
                1: <dictionary: 0x62e0584e0> { count = 7, transaction: 0, voucher = 0x0, contents =
                    "source" => <string: 0x62e0285d0> { length = 1, contents = "P" }
                    "arch" => <string: 0x62e0288a0> { length = 6, contents = "arm64e" }
                    "base" => <int64: 0x9490188654038347>: 6647308288
                    "name" => <string: 0x62e028570> { length = 4, contents = "dyld" }
                    "size" => <int64: 0x9490188a35edb607>: 53418
                    "path" => <string: 0x62e0285a0> { length = 13, contents = "/usr/lib/dyld" }
                    "uuid" => <string: 0x62e028600> { length = 36, contents = "68cc64d1-738b-35fd-968d-0fbd8938819f" }
                }
                2: <dictionary: 0x62e058480> { count = 4, transaction: 0, voucher = 0x0, contents =
                    "source" => <string: 0x62e028690> { length = 1, contents = "A" }
                    "base" => <int64: 0x9490188a35ac8347>: 0
                    "size" => <int64: 0x9490188a35ac8347>: 0
                    "uuid" => <string: 0x62e028630> { length = 36, contents = "00000000-0000-0000-0000-000000000000" }
                }
                3: <dictionary: 0x62e058420> { count = 7, transaction: 0, voucher = 0x0, contents =
                    "source" => <string: 0x62e028420> { length = 1, contents = "P" }
                    "arch" => <string: 0x62e028660> { length = 6, contents = "arm64e" }
                    "base" => <int64: 0x94901886e5000347>: 6880071680
                    "name" => <string: 0x62e0286c0> { length = 17, contents = "libSystem.B.dylib" }
                    "size" => <int64: 0x9490188a35ac7ca7>: 8188
                    "path" => <string: 0x62e028510> { length = 26, contents = "/usr/lib/libSystem.B.dylib" }
                    "uuid" => <string: 0x62e0284b0> { length = 36, contents = "2066bca7-03d8-3c61-a4d7-a64e9e25ab6d" }
                }
                4: <dictionary: 0x62e0583c0> { count = 7, transaction: 0, voucher = 0x0, contents =
                    "source" => <string: 0x62e028480> { length = 1, contents = "P" }
                    "arch" => <string: 0x62e028450> { length = 6, contents = "arm64e" }
                    "base" => <int64: 0x9490188656208347>: 6651215872
                    "name" => <string: 0x62e028990> { length = 24, contents = "libsystem_platform.dylib" }
                    "size" => <int64: 0x9490188a35af7c67>: 32740
                    "path" => <string: 0x62e0284e0> { length = 40, contents = "/usr/lib/system/libsystem_platform.dylib" }
                    "uuid" => <string: 0x62e0283f0> { length = 36, contents = "0b09ae47-f8c6-3a6d-80ae-d25708beaf3d" }
                }
            }
            "name" => <string: 0x62e028960> { length = 5, contents = "crash" }
            "isSimulated" => <int64: 0x9490188a35ac8347>: 0
            "pid" => <int64: 0x9490188a35ac9b77>: 774
            "bug_type" => <string: 0x62e028780> { length = 3, contents = "309" }
        }
        "Signature" => <string: 0x62e0280c0> { length = 40, contents = "c3ba8770e26e4c56600e8c339aebbc944bbd03d9" }
        "override-filePrefix" => <string: 0x62e028870> { length = 5, contents = "crash" }
        "file-owner" => <string: 0x62e028210> { length = 4, contents = "tree" }
        "file-owner-uid" => <int64: 0x9490188a35ac8cef>: 501
        "SubmissionPolicy" => <string: 0x62e0286f0> { length = 9, contents = "Alternate" }
    }
    "operation" => <uint64: 0x9410188a35ac8377>: 6
    "datawriter_endpoint" => <endpoint>
    "caller" => <string: 0x62e028c30> { length = 11, contents = "ReportCrash" }
    "additionalHeaders" => <dictionary: 0x62e058360> { count = 9, transaction: 0, voucher = 0x0, contents =
        "app_version" => <string: 0x62e028120> { length = 0, contents = "" }
        "incident_id" => <string: 0x62e0288d0> { length = 36, contents = "904943AE-43BF-42B6-829C-4D6BF52872E4" }
        "is_first_party" => <int64: 0x9490188a35ac834f>: 1
        "app_name" => <string: 0x62e0280f0> { length = 5, contents = "crash" }
        "name" => <string: 0x62e028930> { length = 5, contents = "crash" }
        "slice_uuid" => <string: 0x62e028c60> { length = 36, contents = "0e85d332-d459-362a-a6f2-0cdc6a99066f" }
        "platform" => <int64: 0x9490188a35ac834f>: 1
        "build_version" => <string: 0x62e028bd0> { length = 0, contents = "" }
        "share_with_app_devs" => <int64: 0x9490188a35ac8347>: 0
    }
    "bug_type" => <string: 0x62e028c00> { length = 3, contents = "309" }
}>
```

Plan

- 1. Create a token
- 2. Have osanalyticshelperd consume it
- 3. Have osanalyticshelperd create a normal crash file
- **failed badly at step 3**
  - back-channel errors


BACK CHANNEL

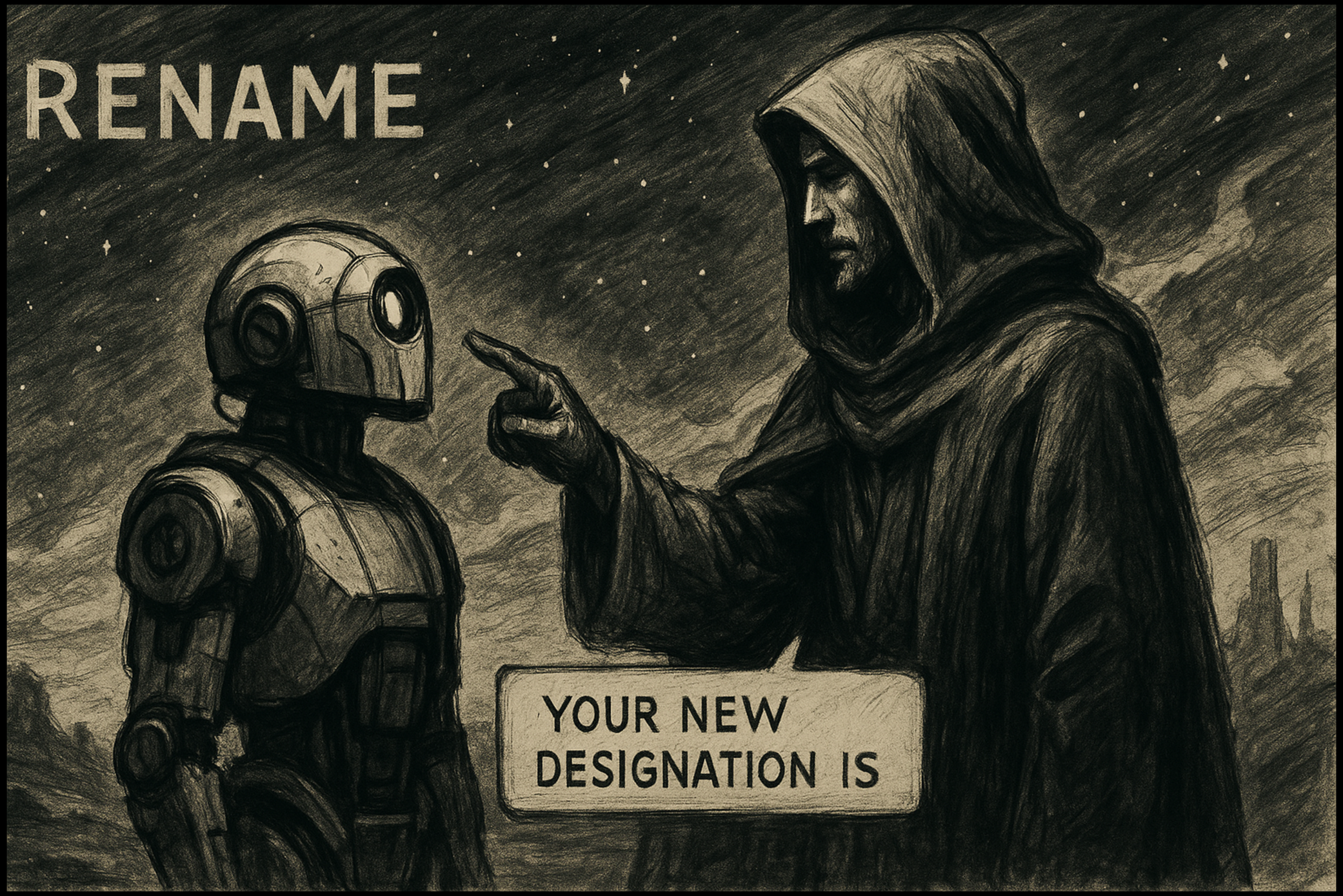
```
bool __fastcall func_backchannel(__int64 a1, int a2, _QWORD *a3)
{

  if ( os_log_type_enabled((os_log_t)&_os_log_default, OS_LOG_TYPE_DEFAULT) )
  {
    *(_WORD *)buf = 0;
    _os_log_impl(
      (void *)&_mh_execute_header,
      (os_log_t)&_os_log_default,
      OS_LOG_TYPE_DEFAULT,
      "S3. helper service utilizing back-channel with file descriptor for payload",
      buf,
      2u);
  }
  v5 = objc_retainAutoreleasedReturnValue(xpc_dictionary_get_value(*(xpc_object_t *)(a1 + 32), "datawriter_endpoint"));
  v6 = xpc_connection_create_from_endpoint(v5);
  v7 = v6;
  *(_QWORD *)buf = 0;
  v36 = buf;
  v37 = 0x3032000000LL;
  v38 = (__int64 (__fastcall *)())sub_100010DE0;
  v39 = (__int64 (__fastcall *)())sub_100010DF0;
  v40 = 0;
  if ( v6 )
  {
    handler[0] = (__int64)_NSConcreteStackBlock;
    handler[1] = 3254779904LL;
    handler[2] = (__int64)sub_100011538;
    handler[3] = (__int64)&unk_100021108;
    handler[4] = (__int64)buf;
    xpc_connection_set_event_handler(v6, handler);
    xpc_connection_resume(v7);
    v8 = (NSDictionary *)xpc_dictionary_create(0, 0, 0);
    xpc_dictionary_set_fd(v8, "fileDesc", a2);
    v9 = xpc_connection_send_message_with_reply_sync(v7, v8);
    v10 = v9;
    if ( v9 )
      v11 = xpc_dictionary_get_bool(v9, "result");

  (...)

}
```

back-channel

- purpose:
  - **send a file descriptor back to the caller** (ReportCrash)
  - caller can add more info - stack trace, vm map, etc...
  - **caller has to return "1"**, otherwise the crash log creation will bail out
- we created an anonymous XPC endpoint
  - which returned 1

where are we?

- we can create a token, which is consumed by osanalyticshelperd
- we can create a crash log via xpc request
- if the directory is symlinked, we can drop that file anywhere (due to the token allowing broad access)


RENAME
YOUR NEW
DESIGNATION IS

Remember this?

```
17:04:48.619044  stat64                                  /Users/pirate/Library/Logs/DiagnosticReports
17:04:48.619055  open              F=3       (R__________)  /Users/pirate/Library/Logs/DiagnosticReports

(...)

17:04:48.641041  open_dprotected   F=3       (_WC__E_____)  /Users/pirate/Library/Logs/DiagnosticReports/.crash-2024-09-17-170448.ips

(...)

17:04:48.641064  fchown            F=3
17:04:48.641078  fchmod            F=3   <rw------->
17:04:48.641145  write             F=3    B=0x15b
17:04:48.641157  write             F=3    B=0x1

(...)

17:04:48.978892  rename                                /Users/pirate/Library/Logs/DiagnosticReports/.crash-2024-09-17-170448.ips
```

Remember this?

- This is an “in-place” rename()
- Equivalent to: rename(“/a/b/c/**x**”, “/a/b/c/**y**”)
- A rename within the same directory, so most assume this is ok
- Most assume the kernel is “clever” (caching, etc…)
- **It’s not**

```
17:04:48.978892  rename                                  /Users/pirate/Library/Logs/DiagnosticReports/.crash-2024-09-17-170448.ips
```


ME
BAD RENAME
SOME
OTHER PROJECT
imgflip.com

the rename() pitfall

- In any rename(src, dst) the kernel will look up src and dst **separately**
- It **has to** - things might move around mid-syscall
- This is POSIX behavior
- Unintuitive, but useful: **every rename() is racy!**

the rename() pitfall

- I found 0days with this before
- **Prerequisite**: we must be able to control one of the path components
- **Nice to have**: control over the final filename (in dst)
- We match the prerequisite but have no filename control…
- But notice: the target runs as **root**

Winning the race

- To win a tight race like this you must: **be clever** or **use bruteforce**
- You can ask me later about being clever 🤪
- Bruteforce is easier but only feasible if the race:
  - has no serious side-effects
  - is quick to run
- Both is true in our case

Winning the race

- If we atomically swap the parent directory with a symlink in a loop
- We will eventually end up with a rename that is equivalent to this:
  - rename(“/original/**x**“, “/attacker_controlled/**y**”)
- This is an **almost fully controlled file (over)write**
  - The only thing we don’t control is the file’s name
- A pretty powerful primitive, but lack of filename control sucks


WEAPONIZATION STRATEGY

what to target?

- We need a vector to turn a file write into a privilege escalation
- We control everything about the file, **except it’s name**
- A couple things might come to mind (cron, scripts, etc…)
- But this is **macOS**
  - cron is TCC protected
  - No scripts (not that we control the file’s name)

what to target?

- But there are: **system services** and **sudo**
- Generally speaking, any /etc/*.d/ directory is a good target
  - because file names don’t matter here
- We ended up attacking **sudo**

sudo’s requirements

- **sudo** does require:
  - the file to be root-owned
  - restricted permissions (no write bits for “other”)
- We can solve this any number of ways: open fd, hardlink, symlinks, etc…
- But I used ACLs because they’re:
  - **Easy:** No racing required
  - **Flexible:** I retain access to the file forever


ACL Inheritance
ACCESS
GRANTED

What are ACLs?

- “man acl”

ACL speedrun any%

- **POSIX** IEEE 1003.1e draft 17
- a **revoked** standard
- Got implemented on **Linux** and **BSD** (**macOS**) anyway (in different ways)
- On **macOS** it allows us to have inheriting ACLs:
  - chmod +a **attacker_user** allow read,write,…,file_inherit,directory_inherit **my_dir**
  - Anything placed in **my_dir**, **attacker_user** will be able to manipulate forever

ACL speedrun any%

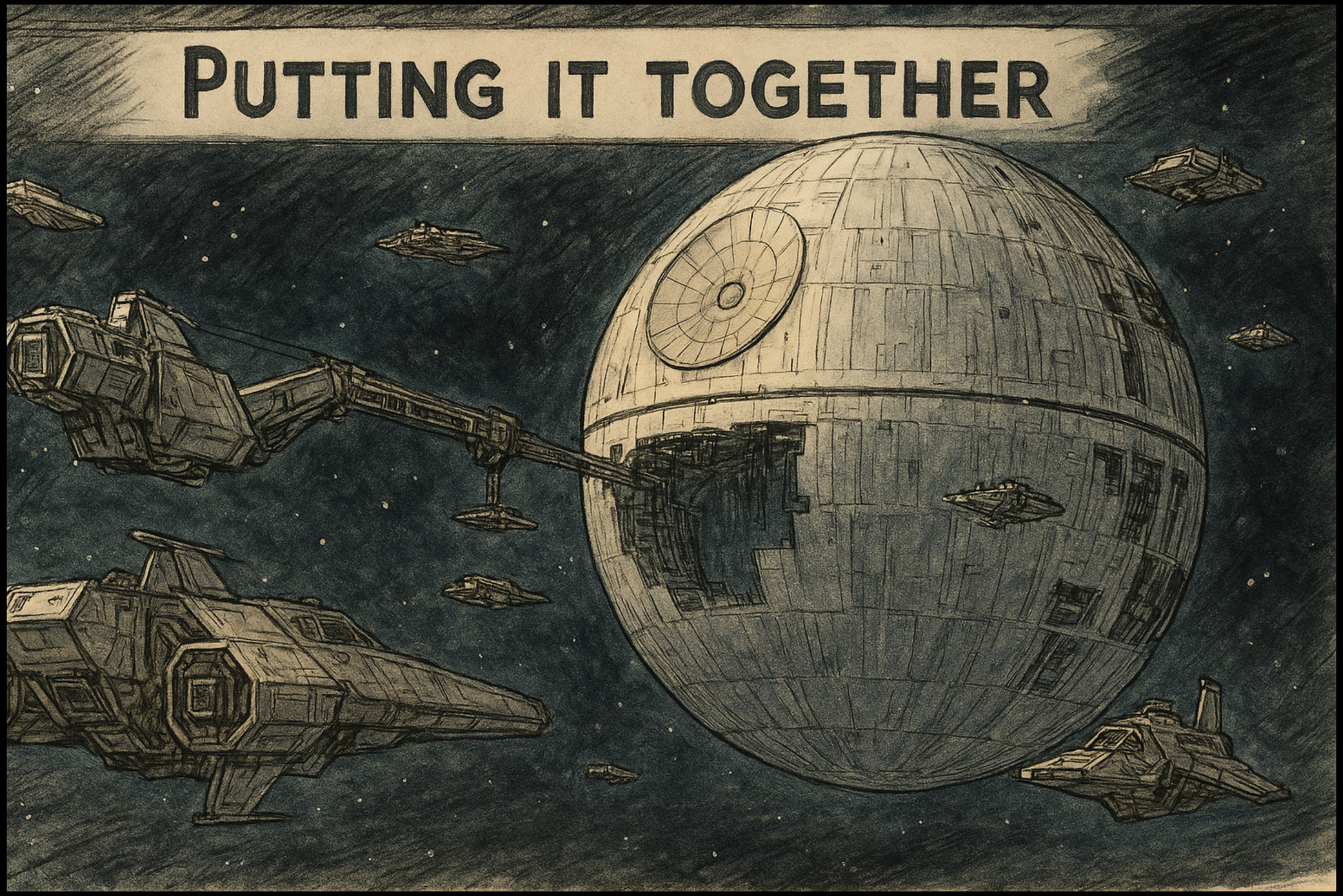
- Bonus: This is invisible to most programs 👀
- You need to call libc functions to get the ACLs
  - and most programs don’t do this
- **sudo** is one of these


SURVIVING CHOWN
YOUR FILES ARE MINE
NO

fchown

- osanalyticshelperd sets the user to be the owner
- but! we want to retain root as the owner
- ACLs would solve this problem, but we found a way to sidestep this entirely
- simply:

```
xpc_dictionary_set_int64(options, "file-owner-uid", 0);
```


PUTTING IT TOGETHER

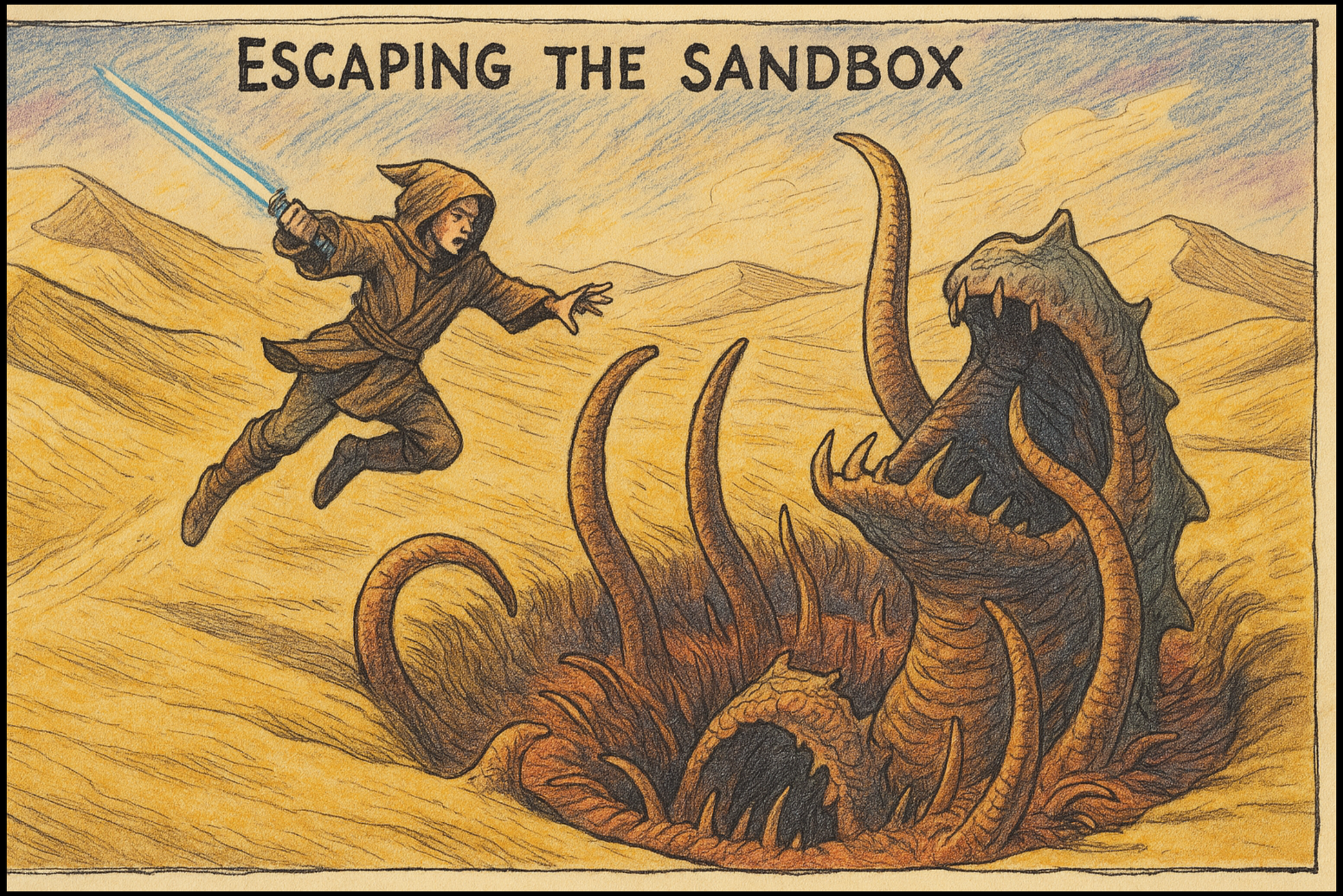
the exploit

- racer in python + trigger in C
- **trigger:**
  - handles the sandbox extension - allows target to create files anywhere
  - XPC client - issues request to trigger rename()
  - XPC server - for the backchannel / callback

the exploit

- **racer:**
  - prepares the environment - needed links, ACLs, etc…
  - executes **trigger**
  - races the rename()
  - detects success and retries

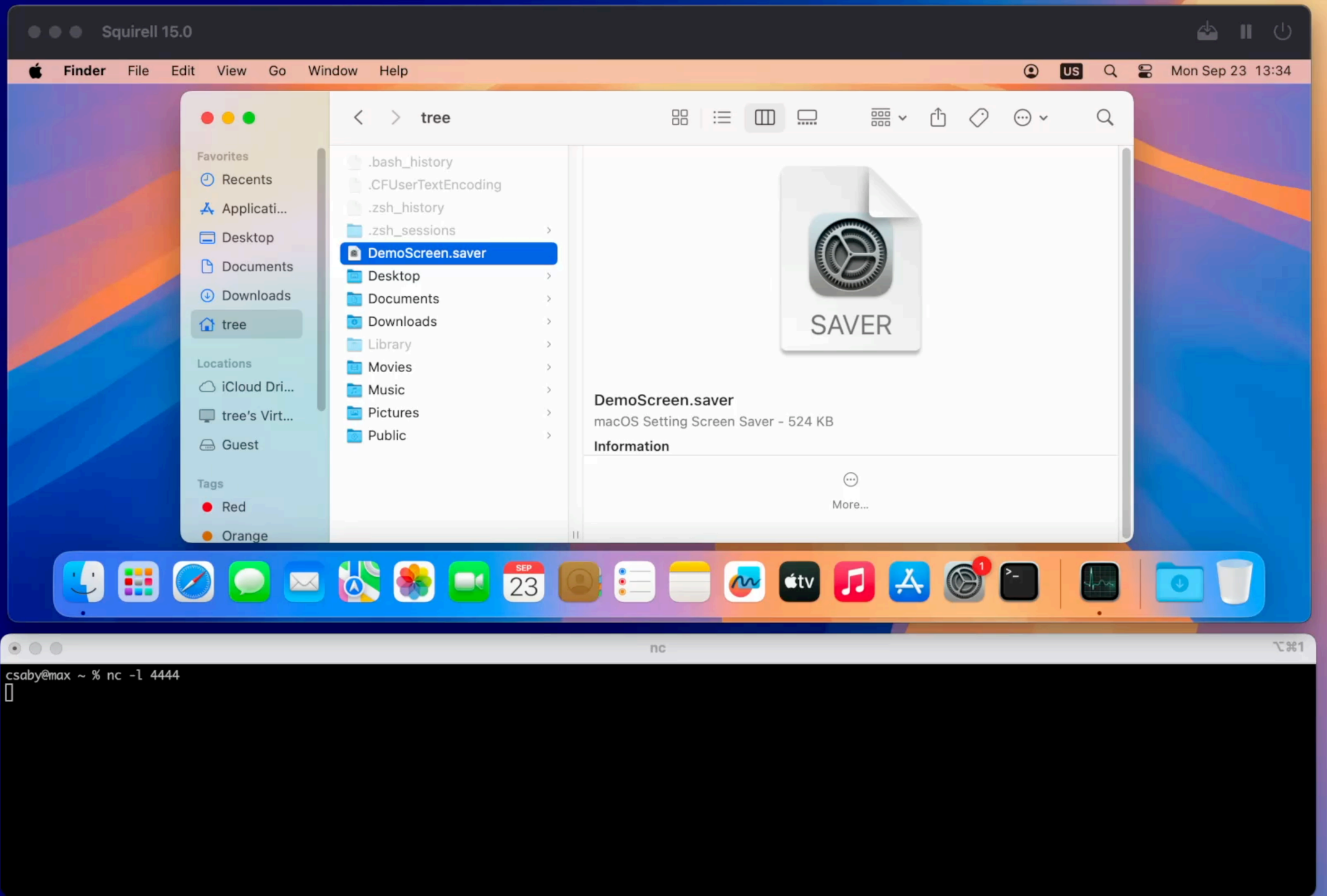
```
n00b@squirell ~ %
```


ESCAPING THE SANDBOX

sb escape how-to

- **platform binaries** can lookup the XPC service
- we can't issue sandbox extensions, but **osanalyticshelperd** can use **/Library/Logs/DiagnosticReports**
- what we do:
  - **drop a DMG** via **osanalyticshelperd** (no quarantine flag will present)
    - XPC allows **full filename control**
  - **open DMG** with embedded unsandboxed LPE

```
from system.sb:

(with-filter (process-attribute is-platform-binary)
  (allow mach-lookup (global-name "com.apple.osanalytics.osanalyticshelper")))
```


Squirell 15.0
Finder File Edit View Go Window Help
US
Mon Sep 23 13:34
tree
Favorites
Recents
Applicati...
Desktop
Documents
Downloads
tree
Locations
iCloud Dri...
tree's Virt...
Guest
Tags
Red
Orange
.bash_history
.CFUserTextEncoding
.zsh_history
.zsh_sessions
DemoScreen.saver
Desktop
Documents
Downloads
Library
Movies
Music
Pictures
Public
SAVER
DemoScreen.saver
macOS Setting Screen Saver - 524 KB
Information
More...
SEP
23
nc
⌥⌘1
csaby@max ~ % nc -l 4444


THE PATCH

```
/* @class OSALogHelper */
+(int)createForSubmissionWithXPCRequest:(int)arg2 fromConnection:(int)arg3 forReply:(int)arg4 {
    var_30 = arg0;
    var_38 = [arg2 retain];
    r12 = [arg3 retain];
    var_40 = [arg4 retain];
    r13 = &var_78;
    *r13 = 0x0;
    *(r13 + 0x8) = r13;
    *(r13 + 0x10) = 0x2020000000;
    *(int8_t *)(r13 + 0x18) = 0x0;
    xpc_connection_get_audit_token(r12, &var_C8);
    rax = xpc_copy_entitlement_for_token(0x0, &var_C8);
    rbx = rax;
    if (rax != 0x0) {
            rax = [@"com.apple.security.system-groups" UTF8String];
            rax = xpc_dictionary_get_array(rbx, rax);
            rax = [rax retain];
            r14 = rax;
            if (rax != 0x0) {
                    var_F8 = *__NSConcreteStackBlock;
                    *(&var_F8 + 0x8) = 0xfffffffffc2000000;
                    *(&var_F8 + 0x10) = sub_100010306;
                    *(&var_F8 + 0x18) = 0x10001e0d0;
                    rax = [r14 retain];
                    *(&var_F8 + 0x20) = rax;
                    *(&var_F8 + 0x28) = r13;
                    rbx = rbx;
                    xpc_array_apply(rax, &var_F8);
                    [var_D8 release];
            }
            if (xpc_dictionary_get_bool(rbx, "com.apple.private.osanalytics.write-logs.allow") != 0x0) {
                    *(int8_t *)(var_70 + 0x18) = 0x1;
            }
            [r14 release];
    }
    var_50 = rbx;
    var_58 = r12;
```

Fixes #2

- We are not allowed to call the XPC endpoint anymore
- rename() changed to renameatx_np() with the flag RENAME_NOFOLLOW_ANY
- **This was the correct way to fix!**
- Not only that, but further attacks due to rename() also got cut off

END CREDITS

Csaba Fitzl (@theevilbit) of Kandji

Gergely Kalman (@gergely_kalman)