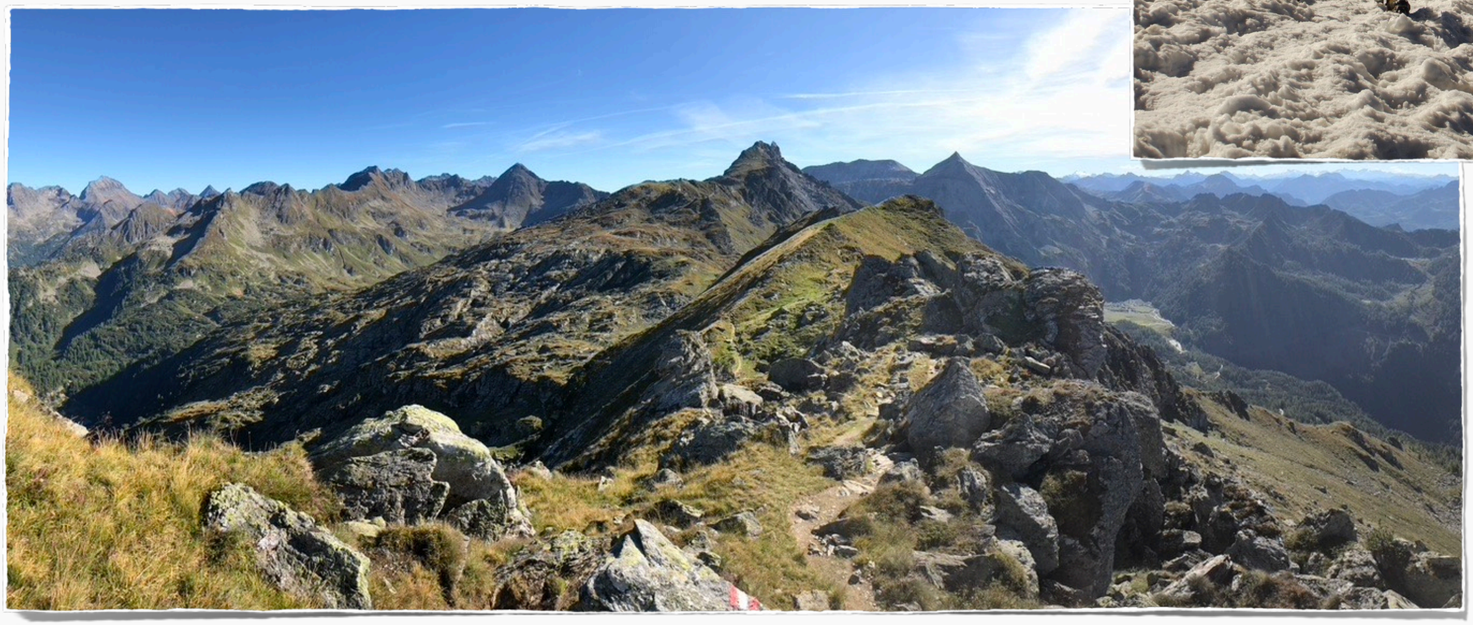
XPC exploitation on macOS

Csaba Fitzl
Twitter: @theevilbit

OFFENSIVE® security

whoami

- content developer at Offensive Security
- ex red/blue teamer
- macOS researcher
- husband, father
- hiking 🥾 🏔️
- yoga 🧘



OFFENSIVE® security

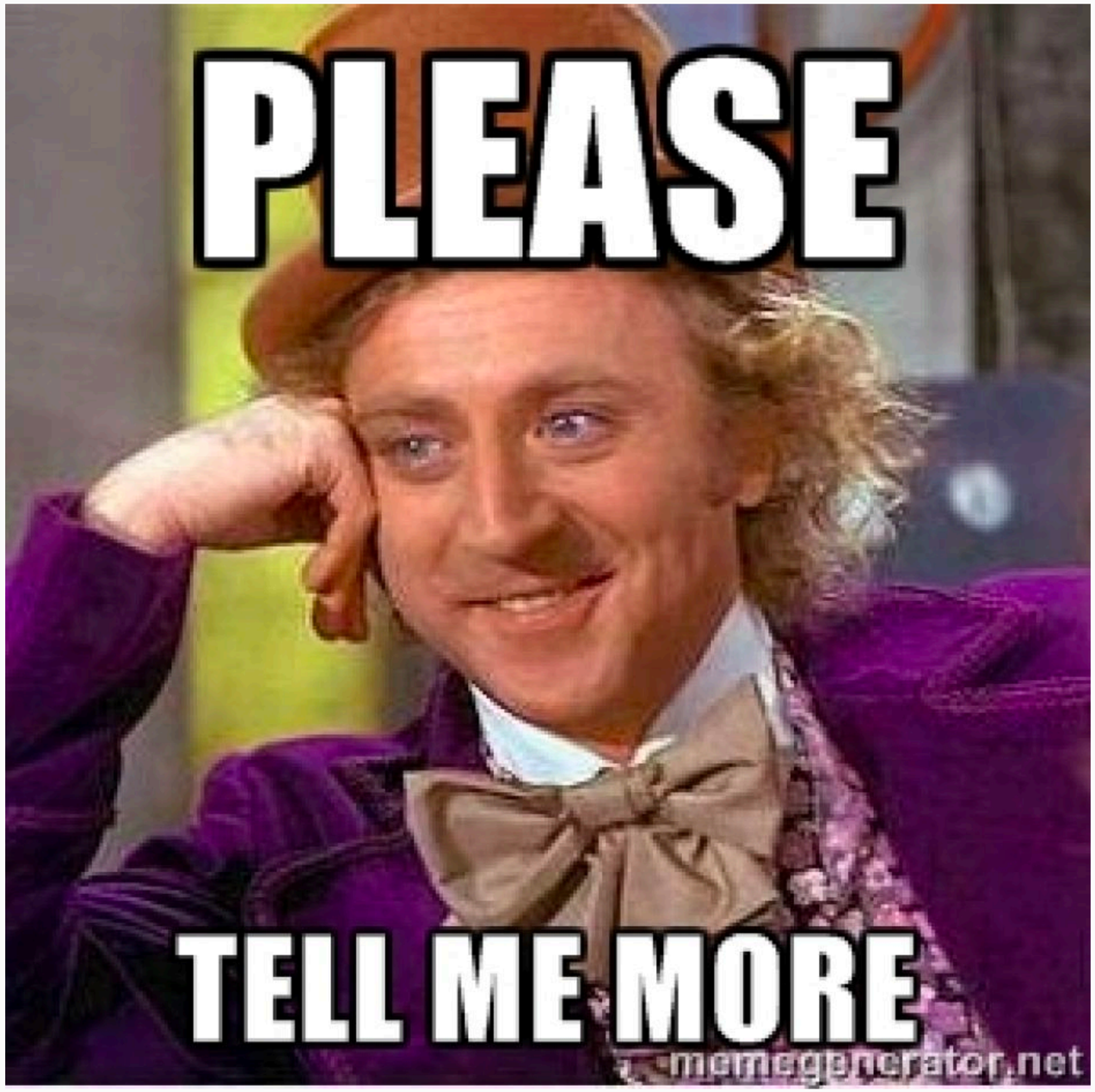
agenda

- intro
- what is xpc?
- xpc APIs
- exploit scenarios
  - Apple's sample code
  - vulnerable clients
  - incorrect client verification
  - PID reuse
- preventing attacks

intro

how it started

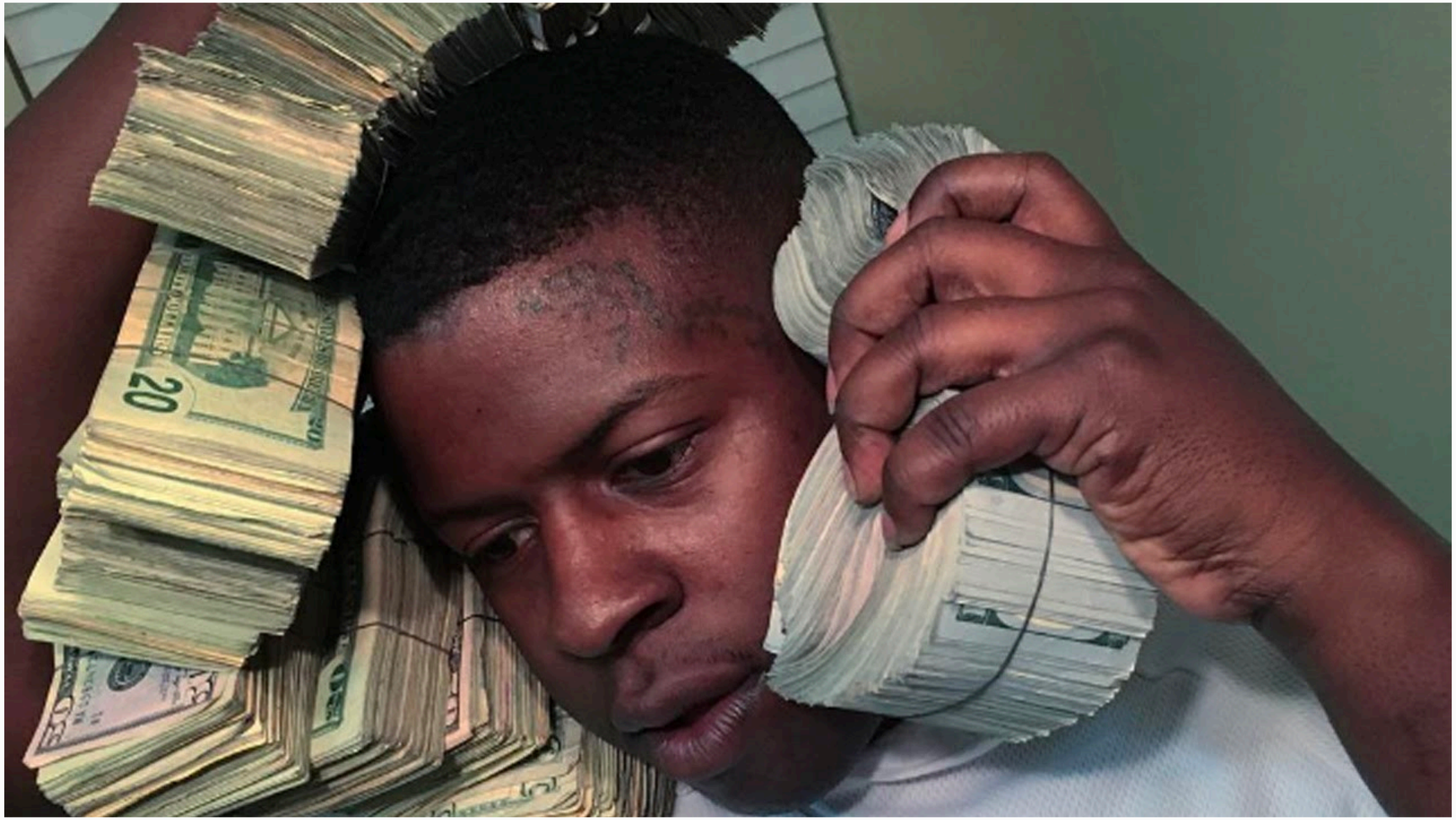
- Phil Stokes (@philofishal) : Spoofing Privileged Helpers to Gain Root
  "*Abuses of this trust mechanism between parent process and privileged helper tool are possible (CVE-2019-13013)*"


PLEASE
TELL ME MORE
memegenerator.net

exploit developers starter kit (as of 2019)

- OffensiveCon19 - Tyler Bohan - OSX XPC Revisited - 3rd Party Application Flaws
- The Story Behind CVE-2019-13013 (Little Snitch)
- Apple's EvenBetterAuthorization sample
- WWDC talks on XPC
- and then...

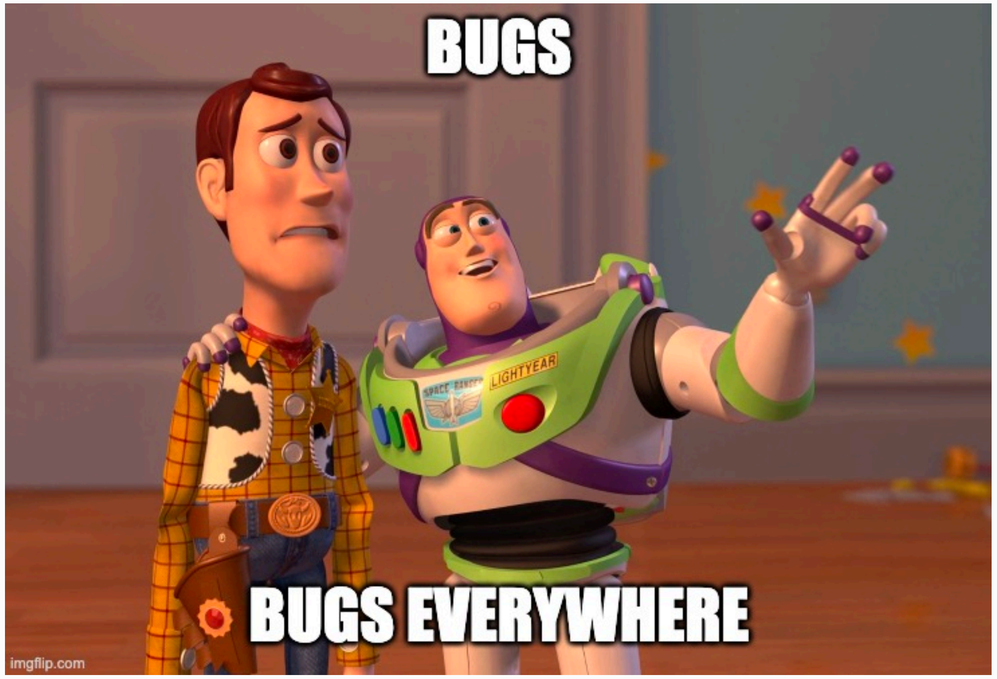
profit :-)



OFFENSIVE® security

vendors

- VMware
- Microsoft
- Adobe
- ProxyMan
- F-Secure
- and many, many more....
- ~90% of the 3rd party tools



OFFENSIVE security

why this talk?

- because of all the bugs, it's all over
- few people exploit them
- no clear guidance from Apple
- no secure public API from Apple
- very easy to make an error
- very easy to exploit

"We are in the golden ages of 3rd party XPC exploits"

-Csaba Fitzl

What is XPC?

interprocess communication

- macOS's base IPC based on Mach messages
- XPC is an IPC built on top of Mach
- introduced in 2011
- simple, easy to use API
- XPC = ? no one knows (Xenomorph Pet Cemetery or eXtended Process Communication or Cross(=X) Process Communication or ?)

core idea

- separate app components -> better security, reliability
- app components talk over XPC
- each component: least privilege (e.g.: Safari's renderer doesn't have network access)
- crash in one doesn't crash the whole app
- vulnerability in one doesn't impact the whole app
- components doesn't have to be on different privilege level

privileged helper tools

- 1 component of the app requires root (install, etc...)
- main app and privhelper talk over XPC
- privhelper performs privileged tasks
- advertised through Mach service names

the problem

- by default: any non-sandboxed app can talk to the priv helpers
  - macOS doesn't restrict the client (it does for other, non-privileged components 🙂)
- if we can talk to the helper tool => we can run privileged action
- full LPE depends on the functions offered

the XPC APIs

The C API

```
xpc_connection_t conn;
xpc_object_t msg;

msg = xpc_dictionary_create(NULL, NULL, 0);

xpc_dictionary_set_string(msg, "key", "value");
xpc_dictionary_set_string(msg, "key2", "value2");


conn = xpc_connection_create_mach_service("com.some.Helper", NULL, 0);
if (conn == NULL) {
    perror("xpc_connection_create_mach_service");
}

xpc_connection_set_event_handler(conn, ^(xpc_object_t obj) {
    printf("Received message in generic event handler: %p\n", obj);
    printf("%s\n", xpc_copy_description(obj));
});

xpc_connection_resume(conn);

xpc_connection_send_message_with_reply(conn, msg, NULL, ^(xpc_object_t resp) {
    printf("Received message: %p\n", resp);

});
```

The NSXPC API

```
@protocol HelperProtocol
- (void)runApp:(NSString *)arg1;
@end

int main(void) {

    NSString*  _serviceName = @"com.some.Helper";
    NSXPCConnection* con = [[NSXPCConnection alloc] initWithMachServiceName:_serviceName
options:4096];

    [con setRemoteObjectInterface:[NSXPCInterface interfaceWithProtocol:@protocol(HelperProtocol)]];
    [con resume];

    id obj = [con remoteObjectProxyWithErrorHandler:^(NSError* error) {
         (void)error;
          NSLog(@"Connection Failure");
    }];

    NSString* app = @"/bin/bash";

    [obj runApp:app];
}
```

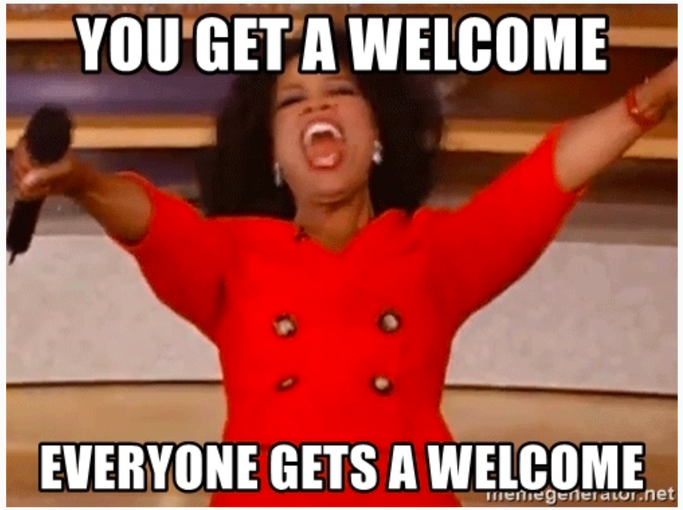
common issues

EvenBetterAuthorization

EvenBetterAuthorizationSample

- Apple's sample code for privileged helper tools
- very, very, very old
- not better, it's the worse 👎👎👎👎👎👎👎
- why?

EvenBetterAuthorizationSample



1. the XPC service accepts *every* connection
2. the service tries to limit access by authorization
    1. client sets up an empty authorization -> anyone can do that
    2. privileged helper sets up authorization based on the auth database
        1. if the auth right is: kAuthorizationRuleClassAllow -> it will be granted
        2. if kAuthorizationRuleAuthenticateAsAdmin -> user is prompted

the problems

- anyone can connect to the service
- many times the rights are set as kAuthorizationRuleClassAllow -> meaningless
- even if kAuthorizationRuleAuthenticateAsAdmin -> the user might authenticate, as the prompt comes from the legitimate helper tool

example

- ProxyMan change proxy privileged action vulnerability (CVE-2019-20057)
- used Apple's sample
- auth rights were set with kAuthorizationRuleClassAllow
- can change proxy settings
- kudos to Nghia Tran (developer) - fixed in a few days 👏

```
- (void)setProxySystemPreferencesWithAuthorization:(NSData *)arg1
enabled:(BOOL)arg2 host:(NSString *)arg3 port:(NSString *)arg4 reply:
(void (^)(NSError *, BOOL))arg5;
```

OFFENSIVE® security

demo - Proxyman

vulnerable client

vulnerable client

- some clients allow code injection
  - no hardened runtime
  - no library validation
  - hardened runtime with exception:
    - com.apple.security.cs.disable-library-validation - dylib injection
    - com.apple.security.get-task-allow - injection via task port

vulnerable client

- this is a problem
- client verification becomes pointless
- anyone can talk to the XPC service

incorrect client verification

incorrect client verification

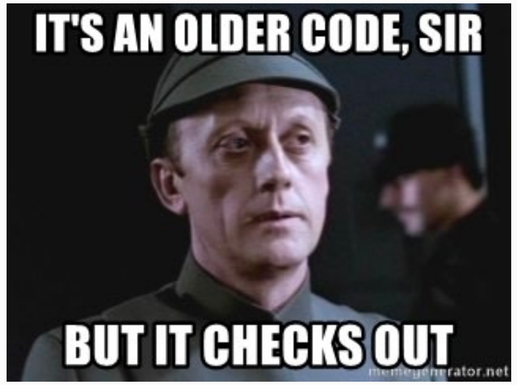
- one of the most commonly overlooked step
- XPC service verifies the signature of the client
- misses to verify if the client is secure (not injectable)

incorrect client verification

- Client verification should include all of these:
  1. Signed with a valid code signing cert (Apple signed developer cert)
  2. If it matches the expected team ID
  3. Verify if client is protected against injection - A
     A. The expected Bundle ID
     B. Minimum version of the client which is not vulnerable to injection
  4. Verify if client is protected against injection - B
     C. Client is not vulnerable to injection (hardened, no exceptions by entitlement)

incorrect client verification

- why the missing of this verification is a problem if client is hardened?
- older client might not (typically Mojave period) => we can inject into an old client, which will satisfy the code signing requirement



example

- Microsoft AutoUpdate (CVE-2020-0984)
- code signature properly verified, except client hardening
- old version not hardened - we inject into that
- priv esc via

```
/* @class MAUHelperTool */
-(char)listener:(void *)arg2 shouldAcceptNewConnection:(void
*)arg3 {

rax = SecRequirementCreateWithString(@"(identifier
\"com.microsoft.autoupdate2\" or identifier
\"com.microsoft.autoupdate.fba\" or identifier
\"com.microsoft.autoupdate.cli\") and anchor apple generic and
certificate 1[field.1.2.840.113635.100.6.2.6] and certificate
leaf[field.1.2.840.113635.100.6.1.…", 0x0, &var_48);
```

```
- (void)createCloneFromApp:(NSString *)arg1 withClonePath:(NSString
*)arg2 withReply:(void (^)(NSString *))arg3;
```

OFFENSIVE® security

demo - Microsoft AutoUpdate

use of PID for client verification

PID

- identifying clients based on PID = 👎
- problems
  1. PID range is small (up to 65k) - can be reused
  2. process can be spawned with inheriting the PID of the parent
- solution: audit token 👍
  - only available via private API 👎

audit token

- audit token is secure to identify the real client
- a.t. not vulnerable to reuse attack (since CVE-2017-7004, thanks to Ian Beer)
- a.t. only available via private API ==> App Store apps can't use it
- secure software or maintainable software ?

generic exploit

1. send a message to the XPC service
2. spawn a new process with `flags |= (POSIX_SPAWN_SETEXEC | POSIX_SPAWN_START_SUSPENDED)`
3. new process is a real client, but gets the PID of the parent
4. XPC service - checks the signature of the spawned process
    1. race condition, but can win easily

example

- VMware Fusion PID reuse vulnerability (CVE-2020-3974)
- Every priv helper was affected
- allows privileged action execution (e.g.: mount)
- full LPE

how to do it securely?

the client

- signed with hardened runtime or library validation
- doesn't have any of these entitlements
  - com.apple.security.cs.disable-library-validation
  - com.apple.security.get-task-allow
- doesn't have script files (those are not verified for code signing on every run)

the XPC service

- The client process verification in the `shouldAcceptNewConnection` call should verify the the following:
  1. The connecting process is signed by valid cert from Apple
  2. The connecting process is signed by your team ID
  3. (The connecting process is identified by your bundle ID)
  4. The connecting process has a minimum software version, where the fix has been implemented or it’s hardened against injection attacks.

- uses audit_token to identify the client

conclusion

?

Further resources

- Wojciech Reguła (@_r3ggi): Abusing and Securing XPC in macOS Apps Objective by the Sea v3
- Julia Vashchenko (@iaronskaya): Job(s) Bless Us! Privileged Operations on macOS Objective by the Sea v3
- Tyler Bohan (@1blankwall1): OSX XPC Revisited - 3rd Party Application Flaws OffensiveCon 19
- Ian Beer (@i41nbeer): A deep-dive into the many flavors of IPC available on OS X Jailbreak Security Summit 2015