Lifting the veil - how to test browsers for spyware

Introduction

Privacy support is one of the chief criteria upon which users pick (or should, anyway) a web browser. Often, a person's perception of a browser's privacy is manufactured by assumptions, marketing talk, or its privacy policy (which can be hard to read and understand, **omit information or outright lie**). Wouldn't it be great if we had a way to **prove** whether a browser actually cares about our privacy, or just bullshits its way through? Fortunately, there is a powerful tool to see exactly what a browser does behind your back, and I'm going to present it

to you right now. **Let's lift the veils!**

Installing mitmproxy

To accomplish this feat, we will be using *mitmproxy* - a local proxy server to which you can point your browser and see the connections it makes (it has many more features, but in this guide, that's our only focus). As of 2023, mitmproxy has portable binaries - which is what I recommend these days, instead of pulling all the python deps that are going to be used only for mitmproxy. This way, you can do your testing, and then trash it after finishing - without burdening your system with rotting python deps. Or even throw the mitmproxy binary in */usr/bin* and keep it there, since it's only 37mb. I can't overstate how much of a relief this is, compared to fiddling with pip, possible errors, and litter in your OS - which was the only option when I originally wrote this article and for years after. Moving on...

Hooking up a browser

If you threw the mitmproxy binary in */usr/bin* as recommended above, then you can just run mitmproxy by typing the terminal command *mitmproxy -p 3128*. Otherwise, you're going to have to *cd* into the folder the binary is in, then type *./mitmproxy -p 3128*. The *3128* is the port which the proxy will listen on. Now go to your browser's proxy settings, and put in *127.0.0.1* for the IP, and *3128* for the port (make sure to fill both the HTTP and HTTPS fields). Ignore everything about SOCKS proxies, since mitmproxy is a HTTP(s) proxy. Chrome-based browsers have it slightly harder, since they don't support GUI proxy settings. You have to run them from the command line like this: *name of browser --proxy-server= "127.0.0.1:3128"*. Replace *“name of browser”* with the executable name, for example *iridium-browser --proxy-*

server= "127.0.0.1:3128".

What if a browser doesn't support proxy settings?

You will need to run it through proxychains with this config file put into */etc/proxychains.conf*. Then, type *proxychains4 name-of-browser* into terminal. If it worked, the proxychains output should be something like this (this is for the suckless Surf browser):

```
proxychains4 surf digdeeper.club
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib64/libproxychains4.so
[proxychains] DLL init: proxychains-ng 4.14
[proxychains] DLL init: proxychains-ng 4.14
[proxychains] DLL init: proxychains-ng 4.14
[proxychains] Dynamic chain ... 127.0.0.1:3128 ... digdeeper.club:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:3128 ... digdeeper.club:443 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:3128 ... digdeeper.club:443 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:3128 ... digdeeper.club:443 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:3128 ... digdeeper.club:443 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:3128 ... digdeeper.club:443 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:3128 ... digdeeper.club:443 ... OK
```

And of course, you should see all the requests in the mitmproxy terminal window. This method also works for Firefox-based browsers, **but will error out in Chrome-based ones**. It has one more advantage in that it will prevent the attempts of malicious browser developers to try to

sneak some requests past the proxy. So, you might want to use proxychains regardless of whether your browser supports proxy settings.

Enabling SSL decryption

Now, if your browser is of the spyware kind, you should already be seeing some requests in your terminal window - but wait, setting up mitmproxy isn't over yet. By default, it only shows pure HTTP requests, since browsers won't let it decrypt SSL. Fortunately it has an easy way to add a root certificate to your browser, which will allow doing just that (this is what the "mitm" part in mitmproxy refers to). Switch to your browser window (this has to be the same browser that's hooked up to mitmproxy, just to be clear). Then...

For Pale Moon, go to *Tools* -> *Preferences* -> *Advanced* -> *Certificates* -> *View Certificates* -> *Authorities* -> *Import*. Now navigate to */home/username/.mitmproxy*, of course replacing *"username"* with your actual username. If you already ran mitmproxy, this folder will contain the certificates, and so you just double-click *mitmproxy-ca-cert.pem* (no more need to visit *mitm.it*). Then, trust it to identify websites.

For Firefox and derivatives, go to *Settings* -> *Privacy and Security*, scroll way down, click *View Certificates*. Ensure you're in the *Authorities* tab, which should be the default; then click *Import*, and finally import the cert the same way as above and trust it to identify websites. These instructions might differ slightly between different browsers and versions, e.g *Settings* might be called something like *Preferences*, etc. But I'm sure you can figure it out.

For Chrome and derivatives, click the three dots on the top right, then enter *Settings* -> *Privacy and security* -> *Security* -> *Manage certificates* -> *Authorities* and import the file like in Pale

Moon, click the three dots near it and trust it to idenfity websites. **Edit**: there is an even better way to do this. It turns out Chrome-based browsers lift their trusted certificate list from some kind of "nss database" in your home directory. So just type this command: *sudo certutil -A -n mitmproxy -t TCu -i /home/username/.mitmproxy/mitmproxy-ca-cert.pem -d /home/username/.pki/nssdb/* which will add the mitmproxy cert to that database, **and be recognized instantly by all Chrome-based browsers**. If certutil isn't installed, do that first, of course. This works in Slackware, cannot confirm for other distros.

Some browsers (such as the aforementioned Surf) automatically trust all certificates, so you don't need to do anything to get mitmproxy to decrypt SSL in them. If you succeeded, you should be able to go to any HTTPS website and see the request in mitmproxy (which will start with *GET https://*). No idea about phones or Windows, I guess use the instructions provided by *mitm.it*. Okay, we've got SSL decryption enabled - what now?

Using mitmproxy

Just wait! Yes, that's it. The whole point is to wait and see what requests the browser makes without your input. You can scroll through requests with the arrow keys and inspect them in detail by pressing Enter. This will show ALL the data that the browser is sending, as well as receiving. Of course, understanding it takes a lot of experience, but at least you now have the opportunity to lift the veil, if you want to! You might be surprised to see that **common web browsers make hundreds of requests without your knowledge**, even ones that are generally considered "respectful of your privacy". For example, Waterfox used to score at exactly 109 unsolicited requests when I wrote this article - just from turning it on! And it was (and still is) advertised as privacy-based - but thanks to mitmproxy, you could have lifted the veil, and

exposed the claim for the lie it was. Isn't that empowering? No more relying on popular opinion, deceptive advertising, or shitty privacy policies (which should be called spy policies). Now it's all there for you to check.

Of course, you won't see all the spyware by just waiting. Some of it can hide in places such as the new tab pages or require visiting a website (Opera's collection of browsing history for example). Also, many requests are only made the first time you run a browser, when it's not going through mitmproxy yet, which will prevent you from seeing them. How to bypass this?

How to see first run requests in Chrome

If you have added the mitmproxy cert to the NSS database - as explained in Enabling SSL decryption - then you don't need to do anything here, as all requests will have been shown already. If you have used the settings method instead, close your browser. Now, go to the browser's config directory (for example */home/hackerman/.config/vivaldi/*) and remove the *First Run* file. That should do it. Press the *Z* key in mitmproxy to delete all previous requests, so that it is easier to see only the new ones. Run the browser again.

How to see first run requests in Firefox

A little tougher. You also have to go to the browser's config directory - which will not be *.config* but *.mozilla* (for FF and IceCat), *.waterfox* (for Waterfox), *.librewolf* (for LibreWolf) or *.moonchild productions* (for Pale Moon). Now, **close your browser** (if you have it open), enter a folder like *x9823yhhnkj.default* (the string before *.default* will be different), find the file *prefs.js*, and in it, the line *“user_pref("network.proxy.http", "127.0.0.1");”*. Now copy

everything starting from that until *“user_pref("network.proxy.type", 1);”*. Delete everything else in the file and save. Your *prefs.js* should look like this (now probably not everything has to necessarily be deleted, but let's be safe...). **If using the proxychains method, you can safely delete all of the file's contents**, as it's now proxychains handling the proxy settings. Also remove all other files except *prefs.js* and either *cert8.db* or *cert9.db*, depending on browser version (these two contain mitmproxy certificate info) - this will leave your Firefox-based browser with ONLY the proxy settings changed from the default clean install.

Mitmproxy vs Wireshark

For the purposes of spyware investigation in browsers, **mitmproxy is the best tool available**. It is a lot simpler to use and needs much less resources. And since mitmproxy has now got the portable binaries, way easier installation also comes into play. More importantly, **Wireshark cannot split traffic by application**. By default, it shows you all the traffic from your chosen network interface (e.g wlan0 or eth0). So you see all the DNS requests, NTP requests, etc. Even if you filter by HTTP, that would still show you **all** the HTTP traffic, instead of just the application you want. To inspect a single application in Wireshark, you'd have to install a separate operating system and ensure it's not sending any other requests. Even then, the format would still be different. However, it can inspect protocols other than HTTP, where mitmproxy is useless.

Back to the front page