defaultnamehere / **cookie_crimes** Public

master

32 Commits

LICENSE
Makefile
README.md
cookie_crimes.py
cookie_crimes_macos.sh
format_for_editthiscookie.sh
requirements.txt

README MIT license

Chrome Cookie Extraction Without Root

This will print out a user's Chrome cookies. You don't need to have their password or be root to use it. nice nice nice nice nice.

If you are not the kind of person who regularly gets the ability to execute code on other people's computers, you probably don't care about this.

Features

- Prints all Chrome cookies in sweet sweet JSON
- Works without root or the user's password
- Works on Windows, Linux, macOS
- Actually also works on the Microsoft Edge browser
- Get cookies fom any Chrome Profile
- Never leaves you on read
- Cooks a mean lasagna
- Compiles to a single binary

Metasploit module

For ezmode #ethical #hacking, please direct your meterpreter session to https://github.com/rapid7/metasploit-framework/blob/9616a9f79de0b22bfd142f12affd74cecbbd4413/documentation/modules/post/multi/gather/chrome_cookies.md

Blog post

Read the full details at https://mango.pdf.zone/stealing-chrome-cookies-without-a-password

About

Read local Chrome cookies without root or decrypting

mango.pdf.zone/stealing-chrome-cookies...

security cookies security-tools osx-security

Readme
MIT license
Activity
611 stars
22 watching
79 forks
Report repository

Releases

No releases published

Packages

No packages published

Contributors 5

Languages

Python 76.9% Shell 21.5% Makefile 1.6%

Installation

Requires Python3.6+ to run locally, but the binary it compiles to works anywhere.

```
pip3 install -r requirements.txt
```

Usage

Windows and Linux

To run it locally:

```
python cookie_crimes.py
```

This will print your Chrome cookies as JSON for the default profile. They're conveniently in the right format to be loaded into the EditThisCookie Chrome Extension

To compile to a single binary:

```
make
```

Note that the binary created will be for the OS you run `make` on. There's no fancy cross-compiling magic going on here. You'll have to build this on the same OS as you're running it on.

macOS

Why is it different on macOS?

For whatever reason, running Chrome with `--headless` has allowed reading of cookies from normal ("headful") Chrome on-and-off over the last few years as changes to Chrome are made. Seriously there are so many commits every *day* that it's become difficult to say "Chrome does not have this feature". This has caused the `headless` method to *sometimes* not work on macOS.

How to do it

Instead, you can run:

```
./cookie_crimes_macos.sh
```

Formatting for EditThisCookie

Chrome's cookie format stores domains with leading dots (e.g. `.google.com`), and so to import *all* cookies into Chrome via the EditThisCookie Chrome Extension, you'll need to remove the leading dots. You can do this via the following Enterprise Grade and completely unnecessary bash script:

```
cat cookies.json | ./format_for_editthiscookie.sh
```

How it works

On macOS, remote debugging is enabled by quickly killing and restarting Chrome, and attaching remote debugging to the new Chrome session with `--restore-last-session` (Just like clicking "restore tabs" in Chrome). This does have the downside of making the Chrome window look like it crashed for about 0.5s (it did lol) and reloading all tabs. But hey, the user will probably just assume their Chrome crashed and restored itself.

Extra crispy thanks to @IAmMandatory for sharing this trick `<3`

`cookie_crimes_macos.sh` will also download, execute, and delete a websocat binary to make the websocket request.

Microsoft Edge

Listen I know that's not Chrome, but hear me out. Because Edge is based on Chromium, the same trick works. Here's a blog post by @wunderwuzzi23 with all the details.

Multiple Profiles

If you want to extract the Chrome cookies for a profile other than the Default profile, just edit the `PROFILE` variable in `cookie_crimes.py`. This uses some sneaky "writing to `/tmp`" tricks to trick Chrome into reading the cookies for us.

How it works

Headless Chrome and `user-data-dir`

Headless (no window is rendered) Chrome is allowed to specify a `user-data-dir`. This directory contains cookies, history, preferences, etc. By creating a new headless Chrome instance, and specifying the `user-data-dir` to be the same as the victim's, your headless Chrome instance will authenticate as the vicitm.

Remote debugging

From here, we just use a normal (but extremely forbidden and undocumented) feature of Chrome: the Remote Debugging protocol. This is how Chrome Developer Tools communicate with Chrome. Once your headless Chrome (with remote debugging enabled) instance is running, this code just executes remote debugging commands to print the user's cookies for all websites in plaintext.

You can fully control Chrome at this point, taking any action the user could take.

closing ceremony

don't do crimes with this please