Introduction

An open source anti-detect browser with robust fingerprint injection. 🦊

Warning

Camoufox is in active development! Releases are avaliable, but may not be suitable for production use.

Features

- **Invisible to all anti-bot systems** 🎭
  - Camoufox performs better than most commercial anti-bot browsers.
- **Fingerprint injection & rotation (without JS injection!)**
  - All navigator properties (device, OS, hardware, browser, etc.) ✅
  - Screen size, resolution, window, & viewport properties ✅
  - Geolocation, timezone, locale, & Intl spoofing ✅
  - WebRTC IP spoofing at the protocol level ✅
  - Voices, speech playback rate, etc. ✅
  - And much, much more!
- **Anti Graphical fingerprinting**
  - Canvas anti-fingerprinting ✅
  - WebGL parameters, supported extensions, context attributes, & shader precision formats ✅
  - Font spoofing & anti-fingerprinting ✅
- **Quality of life features**
  - Human-like mouse movement 🖱️
  - Blocks & circumvents ads 🛡️
  - No CSS animations 💨
- Debloated & optimized for memory efficiency ⚡
- PyPi package for updates & auto fingerprint injection 📦
- Stays up to date with the latest Firefox version 🕓

Get started

python/

Design and Implementation

The goal of Camoufox is to provide a robust, undetectable anti-fingerprinting solution that blends in with regular user traffic.

1. Fingerprinting Protection

2. Crowdblending

3. Stealth

Why Firefox and not Chromium?

Camoufox is built on top of Firefox instead of Chromium for these main reasons:

1. Chrome is bundled with certain features that Chromium does not have. Anti-bot providers can detect if you are using Chromium rather than Chrome. Since Chrome is closed source, patching Chrome is significantly more difficult.
2. CDP is more widely used and known, so it's a more common target for bot detection
3. Juggler operates on a lower level than CDP, making it less prone to JS leaks.
4. Firefox is more ideal for fingerprint rotation. More research has been made on Firefox for fingerprinting resistance than on Chromium.

Is Camoufox open source?

Almost all of Camoufox's code is publicly avaliable on GitHub. However, certain features like the Canvas fingerprint rotation patch are closed source to prevent anti-bot providers from reverse engineering Camoufox.

If you'd like to build Camoufox without the closed source features, please refer to the build guide.

What's planned?

- Continue research on potential leaks
- AI powered selector location
- Remote hosting Camoufox as a Playwright server

Important Notice

Hello! In March 2025, I had a medical emergency and have been hospitalized since. I haven't been able to continue my work for some time.

I was previously updating Camoufox daily, and plan to return to that schedule as soon as I'm able. My goal is to have Camoufox back to its usual performance and be fully caught up by the end of 2025.

Thank you for your patience and understanding.

Features List

Below is a list of patches and features implemented in Camoufox.

Fingerprint spoofing

- Navigator properties spoofing (device, browser, locale, etc.)
- Support for emulating screen size, resolution, etc.
- Spoof WebGL parameters, supported extensions, context attributes, and shader precision formats.
- Spoof the CPU Canvas fingerprint by manipulating the anti-aliasing logic (closed source)
- Spoof inner and outer window viewport sizes
- Spoof AudioContext sample rate, output latency, and max channel count
- Spoof device voices & playback rates
- Spoof the amount of microphones, webcams, and speakers avaliable.
- Network headers (Accept-Languages and User-Agent) are spoofed to match the navigator properties
- WebRTC IP spoofing at the protocol level
- Geolocation, timezone, and locale spoofing
- Battery API spoofing
- etc.

Stealth patches

- Avoids main world execution leaks. All page agent javascript is sandboxed
- Avoids frame execution context leaks
- Fixes `navigator.webdriver` detection
- Fixes Firefox headless detection via pointer type (#26)
- Removed potentially leaking anti-zoom/meta viewport handling patches
- Uses non-default screen & window sizes
- Re-enable fission content isolations

- Re-enable PDF.js
- Other leaking config properties changed
- Human-like cursor movement

Anti font fingerprinting

- Automatically uses the correct system fonts for your User Agent
- Bundled with Windows, Mac, and Linux system fonts
- Prevents font metrics fingerprinting by randomly offsetting letter spacing

Playwright support

- Custom implementation of Playwright for the latest Firefox
- Various config patches to evade bot detection

Debloat/Optimizations

- Stripped out/disabled *many, many* Mozilla services. Runs faster than the original Mozilla Firefox, and uses less memory (200mb)
- Patches from LibreWolf & Ghostery to help remove telemetry & bloat
- Debloat config from PeskyFox, LibreWolf, and others
- Speed & network optimizations from FastFox
- Removed all CSS animations
- Minimalistic theming
- etc.

Addons

- Load Firefox addons without a debug server by passing a list of paths to the `addons` property
- Added uBlock Origin with custom privacy filters
- Addons are not allowed to open tabs
- Addons are automatically enabled in Private Browsing mode
- Addons are automatically pinned to the toolbar
- Fixes DNS leaks with uBO prefetching

Python Interface

- Automatically generates & injects unique device characteristics into Camoufox based on their real-world distribution
- WebGL fingerprint injection & rotation
- Uses the correct system fonts and subpixel antialiasing & hinting based on your target OS
- Avoid proxy detection by calculating your target geolocation, timezone, & locale from your proxy"s target region
- Calculate and spoof the browser"s language based on the distribution of language speakers in the proxy"s target region
- Remote server hosting to use Camoufox with other languages that support Playwright
- Built-in virtual display buffer to run Camoufox headfully on a headless server
- Toggle image loading, WebRTC, and WebGL
- etc.

Thanks ❤️

The design and implementation of Camoufox was inspired by the following projects, in no particular order:

Stealth Performance

In Camoufox, all of Playwright's internal Page Agent Javascript is sandboxed and isolated. This makes it **impossible** for a page to detect the presence of Playwright through Javascript inspection.

Tests

Camoufox performs well against every major WAF I've tested. (Original test sites from Botright)

Test	Status	Notes
CreepJS	✔️	71.5%. Successfully spoofs all OS predictions.
Rebrowser Bot Detector	✔️	All tests pass.
BrowserScan	✔️	100%. Spoofs all geolocation & locale proxy detection.

reCaptcha Score	✔️	
‣ nopecha.com	✔️	
‣ recaptcha-demo.appspot.com	✔️	0.9
‣ berstend.github.io	✔️	0.9
DataDome	✔️	
‣ DataDome bot bounty	✔️	All test sites pass.
‣ hermes.com	✔️	
Imperva	✔️	
‣ ticketmaster.es	✔️	
Cloudflare	✔️	
‣ Turnstile	✔️	
‣ Interstitial	✔️	
WebRTC IP Spoofing	✔️	
‣ Browserleaks WebRTC	✔️	Spoofs public IP correctly.
‣ CreepJS WebRTC	✔️	Spoofs Host & STUN IP correctly.
‣ BrowserScan WebRTC	✔️	Spoofs Host & STUN IP correctly.
Font Fingerprinting	✔️	
‣ Browserleaks Fonts	✔️	Rotates all metrics.
‣ CreepJS TextMetrics	✔️	Rotates all metrics.
Incolumitas	✔️	0.8-1.0
SannySoft	✔️	
Fingerprint.com	✔️	
IpHey	✔️	
Bet365	✔️	

Warning

Camoufox does **not** support injecting Chromium fingerprints. Some WAFs (such as Interstitial) test for Spidermonkey engine behavior, which is impossible to spoof.

Python Interface

Lightweight wrapper around the Playwright API to help launch Camoufox. 🚀

Camoufox's Python library wraps around Playwright's API to help automatically generate & inject unique device characteristics into Camoufox such as the OS, device info, navigator, fonts, headers, screen/viewport size, addons, etc.

It uses BrowserForge under the hood to generate fingerprints that mimic the statistical distribution of device characteristics in real-world traffic.

In addition, it will also calculate your target geolocation, timezone, & locale to avoid proxy detection.

Installation

First, install the `camoufox` package:

```
pip install -U camoufox[geoip]
```

The `geoip` parameter is optional, but heavily recommended if you are using proxies. It will download an extra dataset to determine the user's longitude, latitude, timezone, country, & locale.

Download the browser

```
camoufox fetch
```

To uninstall, run `camoufox remove`.

Camoufox CLI

Options

```
Usage: python -m camoufox [OPTIONS] COMMAND [ARGS]...  Options:   --help  Show this message and exit.
Commands:   fetch    Fetch the latest version of Camoufox   path     Display the path to the Camoufox
executable   remove   Remove all downloaded files   server   Launch a Playwright server   test     Open the
Playwright inspector   version  Display the current version
```

Usage

Camoufox is fully compatible with your existing Playwright code. You only have to change your browser initialization:

```
from camoufox.sync_api import Camoufox  with Camoufox() as browser:    page = browser.new_page()    page.goto("https://example.com")
```

Parameters List

All Playwright Firefox launch options are accepted, along with the following:

Device Rotation

Camoufox will generate device information for you based on the following parameters.

os

Type: `Optional[ListOrString]`

Operating system to use for the fingerprint generation. Can be `"windows"`, `"macos"`, `"linux"`, or a list to randomly choose from. By default, Camoufox will randomly choose from a list of all three.

```
# Use a specific OS with Camoufox(os="windows") as browser:    ...  # Randomly choose from a list of OS with Camoufox(os=["windows", "macos", "linux"]) as browser:    ...
```

fonts

Type: `Optional[List[str]]`

Fonts to load into Camoufox, in addition to the default fonts for the target `os`. Takes a list of font family names that are installed on the system.

Fonts & font fingerprinting

../../fingerprint/fonts/

```
custom_fonts = ["Arial", "Helvetica", "Times New Roman"] with Camoufox(fonts=custom_fonts) as browser:    ...
```

screen

Type: `Optional[Screen]`

Constrains the screen dimensions of the generated fingerprint. Takes a `browserforge.fingerprints.Screen` instance.

While this sets the screen dimensions, it has very light impact on the size of the window. Sometimes the window will be larger, sometimes smaller.

```
from browserforge.fingerprints import Screen  constrains = Screen(max_width=1920, max_height=1080)  with Camoufox(screen=constrains) as browser: ...
```

webgl_config

Type: `Optional[Tuple[str, str]]`

Use a specific WebGL vendor/renderer pair. Passed as a tuple of (vendor, renderer). The vendor & renderer combination must be supported for the target `os` or this will cause leaks.

Check here for a list of Camoufox"s supported vendor & renderer combinations.

```
with Camoufox(    webgl_config=("Apple", "Apple M1, or similar"),    os="macos", ) as browser:    ...
```

config

Type: `Optional[Dict[str, Any]]`

If needed, individual Camoufox config properties can be overridden by passing them as a dictionary to the `config` parameter. This can be used to enable features that have not yet been implemented into the Python library.

warning

This is an advanced feature and should be used with caution. The Camoufox Python library is designed to populate these properties for you.

Passing config

../config/

Configuration

Extra feature configuration and quality of life options.

humanize

Type: `Optional[Union[bool, float]]`

Humanize the cursor movement. Takes either `True`, or the MAX duration in seconds of the cursor movement. The cursor typically takes up to 1.5 seconds to move across the window.

Cursor movement info & demo

../../fingerprint/cursor-movement/

```
# Enable humanization with default settings with Camoufox(humanize=True) as browser:    ...  # Set a custom max duration for cursor movement with Camoufox(humanize=2.0) as browser:    ...
```

headless

Type: `Optional[Union[bool, Literal["virtual"]]]`

Whether to run the browser in headless mode. Defaults to False. If you are running linux, passing "virtual" will use Xvfb.

Virtual Display

../virtual-display/

```
# Run in headless mode with Camoufox(headless=True) as browser:    ... # Run in headless mode on linux
with Camoufox(headless="virtual") as browser:    ...
```

addons

Type: `Optional[List[str]]`

List of Firefox addons to use. Must be paths to extracted addons.

To load an `.xpi` file, rename it to a `.zip` file, extract it, and pass the extracted folder.

```
addons = ["/path/to/addon1", "/path/to/addon2"] with Camoufox(addons=addons) as browser:    ...
```

exclude_addons

Type: `Optional[List[DefaultAddons]]`

Exclude the default addons. Passed as a list of `camoufox.DefaultAddons` enums.

Default addons

../../fingerprint/addons/#default-addons

window

Type: `Optional[Tuple[int, int]]`

Set the window size in (width, height) pixels. This will also set the `window.screenX` and `window.screenY` properties to position the window at the center of the generated screen.

Camoufox will automatically generate a window size for you. Using a fixed window size can lead to fingerprinting. Do not set the window size to a fixed value unless for debugging purposes.

```
with Camoufox(window=(1282, 955)) as browser:    ...
```

main_world_eval

Type: `Optional[bool]`

Whether to inject scripts into the main world when prefixed with `mw:`

```
with Camoufox(main_world_eval=True) as browser:    page = browser.new_page()
page.goto("https://example.com")    # Modify the DOM
page.evaluate("mw:document.querySelector(\'h1\').remove()")
```

See more info

../main-world-eval/

enable_cache

Type: `Optional[bool]`

Whether to cache previous pages, requests, etc. Disabled by default as it uses more memory.

Keeping this disabled will not allow you to use the `page.go_back()` or `page.go_forward()` methods.

```
with Camoufox(enable_cache=True) as browser:    ...
```

persistent_context

Type: `Optional[bool]`

Whether to create a persistent context. Requires `user_data_dir`.

```
with Camoufox(    persistent_context=True,    user_data_dir=\"/path/to/profile/dir\", ) as context:    ...
```

Location & Language

Prevent proxy detection by matching your geolocation & locale with your target IP. This will populate the Geolocation & Intl properties for you.

geoip

Type: `Optional[Union[str, bool]]`

Calculates longitude, latitude, timezone, country, & locale based on the IP address. Pass the target IP address to use, or `True` to find the IP address automatically.

Geolocation & Proxies

../geoip/

```
# Use a specific IP address with Camoufox(geoip="203.0.113.0", proxy=...) as browser:    ... # Automatically find the IP address with Camoufox(geoip=True, proxy=...) as browser:    ...
```

locale

Type: `Optional[Union[str, List[str]]]`

Locale(s) to use in Camoufox. Can be a list of strings, or a single string separated by a comma. The first locale in the list will be used for the Intl API.

```
# Use a single locale with Camoufox(locale="en-US") as browser:    ... # Generate a language based on the distribution of speakers the US with Camoufox(locale="US") as browser:    ... # Use multiple accepted locales with Camoufox(locale=["en-US", "fr-FR", "de-DE"]) as browser:    ... with Camoufox(locale="en-US,fr-FR,de-DE") as browser:    ...
```

Toggles

Shortcuts for common Firefox preferences and security toggles.

block_images

Type: `Optional[bool]`

Blocks all requests to images. This can help save your proxy usage.

```
with Camoufox(block_images=True) as browser:    ...
```

block_webrtc

Type: `Optional[bool]`

Blocks WebRTC entirely.

```
with Camoufox(block_webrtc=True) as browser:    ...
```

block_webgl

Type: `Optional[bool]`

Whether to block WebGL. To prevent leaks, only use this for special cases.

```
with Camoufox(block_webgl=True) as browser:    ...
```

disable_coop

Type: `Optional[bool]`

Disables the Cross-Origin-Opener-Policy (COOP). This allows elements in `cross-origin` iframes, such as the Turnstile checkbox, to be clicked.

```
# Cloudflare turnstile example with Camoufox(disable_coop=True, window=(1280, 720)) as browser:
page = browser.new_page()    page.goto(\'https://nopecha.com/demo/cloudflare\')
page.wait_for_load_state(state="domcontentloaded")    page.wait_for_load_state(\'networkidle\')
page.wait_for_timeout(5000)  # 5 seconds    page.mouse.click(210, 290)    ...
```

Camoufox will warn you if your passed configuration might cause leaks.

More Usage Docs

Geolocation & Proxies

../geoip/

Virtual Display

../virtual-display/

Remote Server

../remote-server/

GeoIP & Proxy Support

By passing `geoip=True`, or passing in a target IP address, Camoufox will automatically use the target IP"s longitude, latitude, timezone, country, locale, & spoof the WebRTC IP address.

It will also calculate and spoof the browser"s language based on the distribution of language speakers in the target region.

Installation

Install Camoufox with the `geoip` extra:

```
pip install -U "camoufox[geoip]"
```

Usage

Pass in `geoip=True` with Playwright"s `proxy` parameter:

```
with Camoufox(    geoip=True,    proxy={        \"server\": \"http://example.com:8080\",        \"username\": \"username\",        \"password\": \"password\"    } ) as browser:    page = browser.new_page()    page.goto(\"https://www.browserscan.net\")
```

Main World Execution

Reading the DOM - Isolated World

By default, all JavaScript execution is ran in an isolated scope, invisible to the page.

This makes it impossible for the page to detect JavaScript reading the DOM:

```
>>> page.goto("https://example.com/") >>> page.evaluate("document.querySelector(\'h1\').innerText") \'Example Domain\'
```

However, your JavaScript will **not** be able to modify the DOM:

```
>>> page.evaluate("document.querySelector(\'h1\').remove()") # Will not work!
```

Modifying the DOM - Main World

To be able to modify the DOM, run JavaScript in the **main world**—the non-isolated scope.

Leak Warning

All code executed in the main world can be detected by the target website. Only execute JavaScript in the main world if absolutely necessary.

To enable main world execution, set the `main_world_eval` parameter to `True`:

```
with Camoufox(main_world_eval=True) as browser:    page = browser.new_page()    page.goto("https://example.com/")
```

Now, you can inject JavaScript in the main world by prefixing the code with `mw:`

```
>>> page.evaluate("mw:document.querySelector(\'h1\').remove()") # h1 is now removed!
```

You can also return JSON serializable data from the main world:

```
>>> page.evaluate("mw:{key: \'value\'}") {\'key\': \'value\'}
```

Note

Returning references to nodes/elements from the main world is not supported, since Playwright"s utilities do not run in the main world.

To retrieve node references, use the isolated world instead.

Remote Server

Warning

Warning! This feature is experimental. It uses a hacky workaround to gain access to undocumented Playwright methods.

Camoufox can be ran as a remote websocket server. It can be accessed from other devices, and languages other than Python supporting the Playwright API.

Launching

To launch a remote server, run the following CLI command:

```
python -m camoufox server
```

Or, configure the server with a launch script:

```
from camoufox.server import launch_server  launch_server(    headless=True,    geoip=True,    proxy={        \"server\": \"http://example.com:8080\",        \"username\": \"username\",        \"password\": \"password\"    },    port=1234,    ws_path=\'hello\' )
```

The server"s port and URL path will be defined using the `port` and `ws_path` parameters, or will be random if not defined.

```
Websocket endpoint: ws://localhost:1234/hello
```

All of the following parameters are also supported:

See parameters list

../usage/#parameters-list

Connecting

To connect to the remote server, use Playwright"s `connect` method:

```
from playwright.sync_api import sync_playwright  with sync_playwright() as p:    # Example endpoint    browser = p.firefox.connect(\'ws://localhost:1234/hello\')    page = browser.new_page()    ...
```

Note

Because servers only use **one browser instance**, fingerprints will not rotate between sessions. If you plan on using Camoufox at scale, consider rotating the server between sessions.

Virtual Display

While Camoufox includes patches to prevent headless detection, running in headless mode may still be detectable in the future. It"s recommended to use a virtual display buffer to run Camoufox headlessly.

If you are running Linux, and would like to run Camoufox headlessly in a virtual display, install `xvfb`:

```
sudo apt-get install xvfb
```

Confirm `Xvfb` is installed:

```
$ which Xvfb /usr/bin/Xvfb
```

Usage

Passing `headless=\"virtual\"` will spawn a new lightweight virtual display in the background for Camoufox to run in.

BrowserForge Integration

Camoufox is compatible with BrowserForge fingerprints.

By default, Camoufox will generate an use a random BrowserForge fingerprint based on the target `os` & `screen` constraints.

```
from camoufox.sync_api import Camoufox from browserforge.fingerprints import Screen  with Camoufox(
 os=(\'windows\', \'macos\', \'linux\'),    screen=Screen(max_width=1920, max_height=1080), ) as browser:
page = browser.new_page()    page.goto(\"https://example.com/\")
```

If Camoufox is being ran in headful mode, the max screen size will be generated based on your monitor"s dimensions unless otherwise specified.

Note

To prevent UA mismatch detection, Camoufox only generates fingerprints with the same browser version as the current Camoufox version by default. If rotating the Firefox version is absolutely necessary, it would be more advisable to rotate between older versions of Camoufox instead.

Injecting custom Fingerprint objects...

Passing Config

If needed, Camoufox `config` can be overridden/passed as a dictionary to the `config` parameter. This can be used to enable features that have not yet been implemented into the Python library.

Warning

This isn"t recommended, as Camoufox will populate this data for you automatically. See the parameters for more proper usage.

```
from camoufox.sync_api import Camoufox  with Camoufox(    config={      \"webrtc:ipv4\":
\"123.45.67.89\",      \"webrtc:ipv6\": \"e791:d37a:88f6:48d1:2cad:2667:4582:1d6d\",    } ) as browser:
page = browser.new_page()    page.goto(\"https://www.browserscan.net/webrtc\")
```

Camoufox will warn you if you are manually setting properties that the Python library handles internally.

Fingerprint Injection

In Camoufox, data is intercepted at the C++ implementation level, making the changes undetectable through JavaScript inspection.

To spoof fingerprint properties, pass a JSON containing properties to spoof to the Python interface:

```
>>> with Camoufox(config={\"property\": \"value\"}) as browser:
```

Config data not set by the user will be automatically populated using BrowserForge fingerprints, which mimic the statistical distribution of device characteristics in real-world traffic.

Getting started

Camoufox accepts the following properties:

Navigator

navigator/

Cursor Movement

cursor-movement/

Fonts

fonts/

Screen

screen/

Window

window/

Document

document/

HTTP Headers

headers/

Geolocation & Intl

geolocation/

WebRTC IP

webrtc/

WebGL

webgl/

Media & Audio

media-audio/

Voices

voices/

Addons

addons/

Miscellaneous

miscellaneous/

Navigator

Navigator properties can be fully spoofed to other Firefox fingerprints, and it is **completely safe**!

Properties

Note

- **navigator.webdriver** is set to false at all times.
- `navigator.language` & `navigator.languages` will fall back to the `locale:language` / `locale:region` values if not set.
- When spoofing Chrome fingerprints, the following may leak:
  - navigator.userAgentData missing.
  - navigator.deviceMemory missing.
- Changing the presented Firefox version can be detected by some testing websites, but typically will not flag production WAFs.

Cursor Movement

Human-like Cursor movement

Camoufox has built-in support for human-like cursor movement. The natural motion algorithm was originally from rifosnake"s HumanCursor, but has been rewritten in C++ and modified for more distance-aware trajectories.

Properties

Property	Type	Description
`humanize`	bool	Enable/disable human-like cursor movement. Defaults to False.

`humanize:maxTime`	double	Maximum time in seconds for the cursor movement. Defaults to `1.5`.
`humanize:minTime`	double	Minimum time in seconds for the cursor movement.
`showcursor`	bool	Toggles the cursor highlighter. Defaults to True.

Note

The cursor highlighter is **not** ran in the page context. It will not be visible to the page. You don"t have to worry about it leaking.

Demo

Here is a demo of the cursor highlighter. The testing page for this can be found here.

Fonts

Fonts can be passed to be used in Camoufox through the `fonts` config property.

By default, Camoufox is bundled with the default Windows 11 22H2 fonts, macOS Sonma fonts, and Linux fonts used in the TOR bundle. Camoufox will automatically add the default fonts associated your spoofed User-Agent OS (the value passed in `navigator.userAgent`):

Mac OS fonts (from macOS Sonma):

`[".Al Bayan PUA", ".Al Nile PUA", ".Al Tarikh PUA", ".Apple Color Emoji UI", ".Apple SD Gothic NeoI", ".Aqua Kana", ".Aqua Kana Bold", ".Aqua かな", ".Aqua かな ボールド", ".Arial Hebrew Desk Interface", ".Baghdad PUA", ".Beirut PUA", ".Damascus PUA", ".DecoType Naskh PUA", ".Diwan Kufi PUA", ".Farah PUA", ".Geeza Pro Interface", ".Geeza Pro PUA", ".Helvetica LT MM", ".Hiragino Kaku Gothic Interface", ".Hiragino Sans GB Interface", ".Keyboard", ".KufiStandardGK PUA", ".LastResort", ".Lucida Grande UI", ".Muna PUA", ".Nadeem PUA", ".New Peninim MT", ".Noto Nastaliq Urdu UI", ".PingFang HK", ".PingFang SC", ".PingFang TC", ".SF Arabic", ".SF Arabic Rounded", ".SF Compact", ".SF Compact Rounded", ".SF NS", ".SF NS Mono", ".SF NS Rounded", ".Sana PUA", ".Savoye LET CC.", ".ThonburiUI", ".ThonburiUIWatch", ".苹方-港", ".苹方-简", ".苹方-繁", ".蘋方-港", ".蘋方-簡", ".蘋方-繁", "Academy Engraved LET", "Al Bayan", "Al Nile", "Al Tarikh", "American Typewriter", "Andale Mono", "Apple Braille", "Apple Chancery", "Apple Color Emoji", "Apple SD Gothic Neo", "Apple SD 산돌고딕 Neo", "Apple Symbols", "AppleGothic", "AppleMyungjo", "Arial", "Arial Black", "Arial Hebrew", "Arial Hebrew Scholar", "Arial Narrow", "Arial Rounded MT Bold", "Arial Unicode MS", "Athelas", "Avenir", "Avenir Black", "Avenir Black Oblique", "Avenir Book", "Avenir Heavy", "Avenir Light", "Avenir Medium", "Avenir Next", "Avenir Next Condensed", "Avenir Next Condensed Demi Bold", "Avenir Next Condensed Heavy", "Avenir Next Condensed Medium", "Avenir Next Condensed Ultra Light", "Avenir Next`

```
Demi Bold", "Avenir Next Heavy", "Avenir Next Medium", "Avenir Next Ultra Light", "Ayuthaya", "Baghdad",
"Bangla MN", "Bangla Sangam MN", "Baskerville", "Beirut", "Big Caslon", "Bodoni 72", "Bodoni 72 Oldstyle",
"Bodoni 72 Smallcaps", "Bodoni Ornaments", "Bradley Hand", "Brush Script MT", "Chalkboard",
"Chalkboard SE", "Chalkduster", "Charter", "Charter Black", "Cochin", "Comic Sans MS", "Copperplate",
"Corsiva Hebrew", "Courier", "Courier New", "Czcionka systemowa", "DIN Alternate", "DIN Condensed",
"Damascus", "DecoType Naskh", "Devanagari MT", "Devanagari Sangam MN", "Didot", "Diwan Kufi", "Diwan
Thuluth", "Euphemia UCAS", "Farah", "Farisi", "Font Sistem", "Font de sistem", "Font di sistema", "Font
sustava", "Fonte do Sistema", "Futura", "GB18030 Bitmap", "Galvji", "Geeza Pro", "Geneva", "Georgia", "Gill
Sans", "Grantha Sangam MN", "Gujarati MT", "Gujarati Sangam MN", "Gurmukhi MN", "Gurmukhi MT",
"Gurmukhi Sangam MN", "Heiti SC", "Heiti TC", "Heiti-간체", "Heiti-번체", "Helvetica", "Helvetica Neue",
"Herculanum", "Hiragino Kaku Gothic Pro", "Hiragino Kaku Gothic Pro W3", "Hiragino Kaku Gothic Pro W6",
"Hiragino Kaku Gothic ProN", "Hiragino Kaku Gothic ProN W3", "Hiragino Kaku Gothic ProN W6", "Hiragino
Kaku Gothic Std", "Hiragino Kaku Gothic Std W8", "Hiragino Kaku Gothic StdN", "Hiragino Kaku Gothic StdN
W8", "Hiragino Maru Gothic Pro", "Hiragino Maru Gothic Pro W4", "Hiragino Maru Gothic ProN", "Hiragino
Maru Gothic ProN W4", "Hiragino Mincho Pro", "Hiragino Mincho Pro W3", "Hiragino Mincho Pro W6",
"Hiragino Mincho ProN", "Hiragino Mincho ProN W3", "Hiragino Mincho ProN W6", "Hiragino Sans",
"Hiragino Sans GB", "Hiragino Sans GB W3", "Hiragino Sans GB W6", "Hiragino Sans W0", "Hiragino Sans W1",
"Hiragino Sans W2", "Hiragino Sans W3", "Hiragino Sans W4", "Hiragino Sans W5", "Hiragino Sans W6",
"Hiragino Sans W7", "Hiragino Sans W8", "Hiragino Sans W9", "Hoefler Text", "Hoefler Text Ornaments", "ITF
Devanagari", "ITF Devanagari Marathi", "Impact", "InaiMathi", "Iowan Old Style", "Iowan Old Style Black",
"Järjestelmäfontti", "Kailasa", "Kannada MN", "Kannada Sangam MN", "Kefa", "Khmer MN", "Khmer Sangam
MN", "Kohinoor Bangla", "Kohinoor Devanagari", "Kohinoor Gujarati", "Kohinoor Telugu", "Kokonor",
"Krungthep", "KufiStandardGK", "Lao MN", "Lao Sangam MN", "Lucida Grande", "Luminari", "Malayalam
MN", "Malayalam Sangam MN", "Marion", "Marker Felt", "Menlo", "Microsoft Sans Serif", "Mishafi", "Mishafi
Gold", "Monaco", "Mshtakan", "Mukta Mahee", "MuktaMahee Bold", "MuktaMahee ExtraBold", "MuktaMahee
ExtraLight", "MuktaMahee Light", "MuktaMahee Medium", "MuktaMahee Regular", "MuktaMahee SemiBold",
"Muna", "Myanmar MN", "Myanmar Sangam MN", "Nadeem", "New Peninim MT", "Noteworthy", "Noto
Nastaliq Urdu", "Noto Sans Adlam", "Noto Sans Armenian", "Noto Sans Armenian Blk", "Noto Sans Armenian
ExtBd", "Noto Sans Armenian ExtLt", "Noto Sans Armenian Light", "Noto Sans Armenian Med", "Noto Sans
Armenian SemBd", "Noto Sans Armenian Thin", "Noto Sans Avestan", "Noto Sans Bamum", "Noto Sans
Bassa Vah", "Noto Sans Batak", "Noto Sans Bhaiksuki", "Noto Sans Brahmi", "Noto Sans Buginese", "Noto
Sans Buhid", "Noto Sans CanAborig", "Noto Sans Canadian Aboriginal", "Noto Sans Carian", "Noto Sans
CaucAlban", "Noto Sans Caucasian Albanian", "Noto Sans Chakma", "Noto Sans Cham", "Noto Sans Coptic",
"Noto Sans Cuneiform", "Noto Sans Cypriot", "Noto Sans Duployan", "Noto Sans EgyptHiero", "Noto Sans
Egyptian Hieroglyphs", "Noto Sans Elbasan", "Noto Sans Glagolitic", "Noto Sans Gothic", "Noto Sans Gunjala
Gondi", "Noto Sans Hanifi Rohingya", "Noto Sans HanifiRohg", "Noto Sans Hanunoo", "Noto Sans Hatran",
"Noto Sans ImpAramaic", "Noto Sans Imperial Aramaic", "Noto Sans InsPahlavi", "Noto Sans InsParthi",
"Noto Sans Inscriptional Pahlavi", "Noto Sans Inscriptional Parthian", "Noto Sans Javanese", "Noto Sans
Kaithi", "Noto Sans Kannada", "Noto Sans Kannada Black", "Noto Sans Kannada ExtraBold", "Noto Sans
Kannada ExtraLight", "Noto Sans Kannada Light", "Noto Sans Kannada Medium", "Noto Sans Kannada
```

```
SemiBold", "Noto Sans Kannada Thin", "Noto Sans Kayah Li", "Noto Sans Kharoshthi", "Noto Sans Khojki", "Noto Sans Khudawadi", "Noto Sans Lepcha", "Noto Sans Limbu", "Noto Sans Linear A", "Noto Sans Linear B", "Noto Sans Lisu", "Noto Sans Lycian", "Noto Sans Lydian", "Noto Sans Mahajani", "Noto Sans Mandaic", "Noto Sans Manichaean", "Noto Sans Marchen", "Noto Sans Masaram Gondi", "Noto Sans Meetei Mayek", "Noto Sans Mende Kikakui", "Noto Sans Meroitic", "Noto Sans Miao", "Noto Sans Modi", "Noto Sans Mongolian", "Noto Sans Mro", "Noto Sans Multani", "Noto Sans Myanmar", "Noto Sans Myanmar Blk", "Noto Sans Myanmar ExtBd", "Noto Sans Myanmar ExtLt", "Noto Sans Myanmar Light", "Noto Sans Myanmar Med", "Noto Sans Myanmar SemBd", "Noto Sans Myanmar Thin", "Noto Sans NKo", "Noto Sans Nabataean", "Noto Sans New Tai Lue", "Noto Sans Newa", "Noto Sans Ol Chiki", "Noto Sans Old Hungarian", "Noto Sans Old Italic", "Noto Sans Old North Arabian", "Noto Sans Old Permic", "Noto Sans Old Persian", "Noto Sans Old South Arabian", "Noto Sans Old Turkic", "Noto Sans OldHung", "Noto Sans OldNorArab", "Noto Sans OldSouArab", "Noto Sans Oriya", "Noto Sans Osage", "Noto Sans Osmanya", "Noto Sans Pahawh Hmong", "Noto Sans Palmyrene", "Noto Sans Pau Cin Hau", "Noto Sans PhagsPa", "Noto Sans Phoenician", "Noto Sans PsaPahlavi", "Noto Sans Psalter Pahlavi", "Noto Sans Rejang", "Noto Sans Samaritan", "Noto Sans Saurashtra", "Noto Sans Sharada", "Noto Sans Siddham", "Noto Sans Sora Sompeng", "Noto Sans SoraSomp", "Noto Sans Sundanese", "Noto Sans Syloti Nagri", "Noto Sans Syriac", "Noto Sans Tagalog", "Noto Sans Tagbanwa", "Noto Sans Tai Le", "Noto Sans Tai Tham", "Noto Sans Tai Viet", "Noto Sans Takri", "Noto Sans Thaana", "Noto Sans Tifinagh", "Noto Sans Tirhuta", "Noto Sans Ugaritic", "Noto Sans Vai", "Noto Sans Wancho", "Noto Sans Warang Citi", "Noto Sans Yi", "Noto Sans Zawgyi", "Noto Sans Zawgyi Blk", "Noto Sans Zawgyi ExtBd", "Noto Sans Zawgyi ...4872 bytes truncated...
```

Windows fonts (from Windows 11 22H2):

```
["Bahnschrift", "Calibri", "Cambria", "Cambria Math", "Candara", "Cascadia Code", "Cascadia Mono", "Comic Sans MS", "Consolas", "Constantia", "Corbel", "Courier New", "Ebrima", "Franklin Gothic Medium", "Gabriola", "Gadugi", "Georgia", "HoloLens MDL2 Assets", "Impact", "Jokerman", "Jokerman Solid", "Leelawadee UI", "Lucida Console", "Lucida Sans Unicode", "Malgun Gothic", "Marlett", "Meiryo", "Microsoft Himalaya", "Microsoft JhengHei", "Microsoft New Tai Lue", "Microsoft PhagsPa", "Microsoft Sans Serif", "Microsoft Tai Le", "Microsoft YaHei", "Microsoft Yi Baiti", "MingLiU-ExtB", "Mongolian Baiti", "MS Gothic", "MV Boli", "Myanmar Text", "Nirmala UI", "Palatino Linotype", "Segoe Fluent Icons", "Segoe MDL2 Assets", "Segoe Print", "Segoe Script", "Segoe UI", "Segoe UI Black", "Segoe UI Emoji", "Segoe UI Historic", "Segoe UI Symbol", "SimSun", "Sitka Banner", "Sitka Display", "Sitka Heading", "Sitka Small", "Sitka Text", "Sylfaen", "Symbol", "Tahoma", "Times New Roman", "Trebuchet MS", "Verdana", "Webdings", "Wingdings", "Yu Gothic", "Yu Gothic UI"]
```

Linux fonts (from TOR bundle):

```
["DejaVu Sans", "DejaVu Serif", "Liberation Sans", "Liberation Serif", "Nimbus Sans L", "Nimbus Roman No9 L", "Standard Symbols PS", "URW Bookman L", "URW Chancery L", "URW Gothic L", "URW Palladio L"]
```

New fonts

If you want to add new fonts, you can pass a list of font family names to the `fonts` config property:

```
with Camoufox(fonts=["Arial", "Helvetica", "Times New Roman"]) as browser:    ...
```

Font Metrics

Camoufox prevents font metrics fingerprinting by randomly offsetting letter spacing.

To disable, set `fonts:spacing_seed` to `0`:

```
with Camoufox(config={"fonts:spacing_seed": 0}) as browser:    ...
```

Screen

Spoof the screen size, pixel depth, and position of the window.

Properties

Property	Type	Description
`screen.availHeight`	uint	Available height of the screen
`screen.availWidth`	uint	Available width of the screen
`screen.availTop`	uint	Y-coordinate of the top of the available area
`screen.availLeft`	uint	X-coordinate of the left of the available area
`screen.height`	uint	Total height of the screen
`screen.width`	uint	Total width of the screen
`screen.colorDepth`	uint	Color depth of the screen
`screen.pixelDepth`	uint	Pixel depth of the screen
`screen.pageXOffset`	double	Horizontal scroll position
`screen.pageYOffset`	double	Vertical scroll position

Note

`screen.colorDepth` and `screen.pixelDepth` are synonymous.

Window

Spoof the inner/outer window size, scroll offsets, window position, history length, and more.

Properties

Property	Type	Description
`window.scrollMinX`	int	Minimum horizontal scroll offset
`window.scrollMinY`	int	Minimum vertical scroll offset
`window.scrollMaxX`	int	Maximum horizontal scroll offset
`window.scrollMaxY`	int	Maximum vertical scroll offset
`window.outerHeight`	uint	Total height of the browser window
`window.outerWidth`	uint	Total width of the browser window
`window.innerHeight`	uint	Height of the browser viewport
`window.innerWidth`	uint	Width of the browser viewport
`window.screenX`	int	Horizontal distance from the left border
`window.screenY`	int	Vertical distance from the top border
`window.history.length`	uint	Number of entries in the session history
`window.devicePixelRatio`	double	Ratio of physical pixels to CSS pixels

Note

Setting the outer window viewport will cause some cosmetic defects to the Camoufox window if the user attempts to manually resize it. Under no circumstances will Camoufox allow the outer window viewport to be resized.

Document

Spoof the size of the HTML body element.

Warning

Spoofing document.body has been implemented, but it is more advisable to set `window.innerWidth` and `window.innerHeight` instead.

Properties

Property	Type	Description
`document.body.clientWidth`	uint	Width of the document"s body
`document.body.clientHeight`	uint	Height of the document"s body
`document.body.clientTop`	uint	Width of the top border of the document"s body
`document.body.clientLeft`	uint	Width of the left border of the document"s body

Canvas

What most anti-detect browsers do

Nearly all commerical anti-detect solutions add noise to the canvas to avoid canvas fingerprinting, even on shapes that render the same on every device. This technique is easily detectable by drawing a rectangle and comparing it to an expected result.

What Camoufox does

Camoufox implements Canvas anti-fingerprinting by using a patched build of Skia with modified subpixel rendering logic, making it impossible to tell your device apart from a device that renders anti-aliasing in a different way.

Properties

This patch is closed source. Releases with this patch is only avaliable on the GitHub releases page.

See related

Canvas fingerprint & integrity tests

https://camoufox.com/tests/canvas

2D Shape drawing tests

https://camoufox.com/tests/shapetests

Research discussion

https://github.com/daijro/camoufox/discussions/41

WebGL

Research

As more users visit this site and opt into helping Camoufox"s WebGL research, I will have a more solid profile of each GPU"s parameters, supported extensions, context attributes, & shader precision formats.

See analytics & research

../../webgl-research/

Demo site

This repository includes a demo site that prints your browser"s WebGL parameters. You can use this site to generate WebGL fingerprints for Camoufox from other devices.

See demo

https://camoufox.com/tests/webgl

Properties

Camoufox supports spoofing WebGL parameters, supported extensions, context attributes, and shader precision formats.

Warning

Do **NOT** randomly assign values to these properties. WAFs hash your WebGL fingerprint and compare it against a dataset. Randomly assigning values will lead to detection as an unknown device.

Property	Description	Example
`webGl:renderer`	Spoofs the name of the unmasked WebGL renderer.	`\"NVIDIA GeForce GTX 980, or similar\"`

`webGl:vendor`	Spoofs the name of the unmasked WebGL vendor.	`\"NVIDIA Corporation\"`
`webGl:supportedExtensions`	An array of supported WebGL extensions (full list).	`[\"ANGLE_instanced_arrays\", \"EXT_color_buffer_float\", \"EXT_disjoint_timer_query\", ...]`
`webGl2:supportedExtensions`	The same as `webGl:supportedExtensions`, but for WebGL2.	`[\"ANGLE_instanced_arrays\", \"EXT_color_buffer_float\", \"EXT_disjoint_timer_query\", ...]`
`webGl:contextAttributes`	A dictionary of WebGL context attributes.	`{\"alpha\": true, \"antialias\": true, \"depth\": true, ...}`
`webGl2:contextAttributes`	The same as `webGl:contextAttributes`, but for WebGL2.	`{\"alpha\": true, \"antialias\": true, \"depth\": true, ...}`
`webGl:parameters`	A dictionary of WebGL parameters. Keys must be GL enums, and values are the values to spoof them as.	`{\"2849\": 1, \"2884\": false, \"2928\": [0, 1], ...}`
`webGl2:parameters`	The same as `webGl:parameters`, but for WebGL2.	`{\"2849\": 1, \"2884\": false, \"2928\": [0, 1], ...}`
`webGl:parameters:blockIfNotDefined`	If set to `true`, only the parameters in `webGl:parameters` will be allowed. Can be dangerous if not used correctly.	`true` / `false`
`webGl2:parameters:blockIfNotDefined`	If set to `true`, only the parameters in `webGl2:parameters` will be allowed. Can be dangerous if not used correctly.	`true` / `false`
`webGl:shaderPrecisionFormats`	A dictionary of WebGL shader precision formats. Keys are formatted as `\"<shaderType>,<precisionType>\"`.	`{\"35633,36336\": {\"rangeMin\": 127, \"rangeMax\": 127, \"precision\": 23}, ...}`
`webGl2:shaderPrecisionFormats`	The same as `webGL:shaderPrecisionFormats`, but for WebGL2.	`{\"35633,36336\": {\"rangeMin\": 127, \"rangeMax\": 127, \"precision\": 23}, ...}`

`webGl:shaderPrecisionFormats:blockIfNotDefined`	If set to `true`, only the shader percisions in `webGl:shaderPrecisionFormats` will be allowed.	`true` / `false`
`webGl2:shaderPrecisionFormats:blockIfNotDefined`	If set to `true`, only the shader percisions in `webGl2:shaderPrecisionFormats` will be allowed.	`true` / `false`

Geolocation & Intl

Properties

Property	Type	Description	Required Keys
`geolocation:latitude`	double	Latitude to use.	`geolocation:longitude`
`geolocation:longitude`	double	Longitude to use.	`geolocation:latitude`
`geolocation:accuracy`	double	Accuracy in meters. This will be calculated automatically using the decminal percision of `geolocation:latitude` & `geolocation:longitude` if not set.	
`timezone`	str	Set a custom TZ timezone (e.g. "America/Chicago"). This will also change `Date()` to return the local time.	
`locale:language`	str	Spoof the Intl API and system language (e.g. "en")	`locale:region`
`locale:region`	str	Spoof the Intl API and system region (e.g. "US").	`locale:language`
`locale:script`	str	Set a custom script (e.g. "Latn"). Will be set automatically if not specified.	

`locale:all`	str	List of all accepted locale values (separated by comma). The first item should match the Intl API. If not passed, it will default to the locale used in the Intl API in the format `\"en-US, en\"`.	

The **Required Keys** are keys that must also be set for the property to work.

Note

- Location permission prompts will be accepted automatically if `geolocation:latitude` and `geolocation:longitude` are set.
- `timezone` **must** be set to a valid TZ identifier. See here for a list of valid timezones.
- `locale:language` & `locale:region` **must** be set to valid locale values. See here for a list of valid locale-region values.

HTTP Headers

Camoufox can override the following network headers.

Properties

Property	Type	Description
`headers.User-Agent`	str	Browser and system information
`headers.Accept-Language`	str	Acceptable languages for the response
`headers.Accept-Encoding`	str	Acceptable encodings for the response

Note

If `headers.User-Agent` is not set, it will fall back to `navigator.userAgent`.

WebRTC IP

Camoufox implements WebRTC IP spoofing at the protocol level by modifying ICE candidates and SDP before they"re sent.

Properties

Property	Type	Description
`webrtc:ipv4`	str	IPv4 address to use
`webrtc:ipv6`	str	IPv6 address to use

Note

To completely disable WebRTC, set the `media.peerconnection.enabled` preference to `false`.

Media & Audio

Media Devices

Spoof the amount of microphones, webcams, and speakers avaliable.

Property	Type	Description	Default
`mediaDevices:enabled`	bool	Whether to enable media device spoofing	`false`
`mediaDevices:micros`	uint	Amount of microphones	`3`
`mediaDevices:webcams`	uint	Amount of webcams	`1`
`mediaDevices:speakers`	uint	Amount of speakers	`1`

AudioContext

Spoof the AudioContext sample rate, output latency, and max channel count.

Property	Type	Description
`AudioContext:sampleRate`	uint	AudioContext sample rate
`AudioContext:outputLatency`	double	AudioContext output latency
`AudioContext:maxChannelCount`	uint	AudioContext max channel count

Testing site

https://audiofingerprint.openwpm.com/

Voices

Spoof the voices used in the browser"s Web Speech API.

Property	Type	Description
`voices`	array	An array of objects, one for each voice. Each map must contain the keys `isLocalService` , `isDefault` , `voiceURI` , `name` , and `lang` .
`voices:blockIfNotDefined`	bool	Whether to block all system voices. Default is `false` .
`voices:fakeCompletion`	bool	Return a successful `SpeechSynthesisUtterance` completion with a fake voice used. If disabled, an error is thrown instead (default RFP). Default is `false` .

Addons

In the Camoufox Python library, addons can be loaded with the `addons` parameter.

Camoufox takes extracted addons. To load an `.xpi` file, rename it to a `.zip` file, extract it, and pass the extracted folder.

```
from camoufox.sync_api import Camoufox  with Camoufox(addons=[\'/path/to/addon\', \'/path/to/addon2\']) as browser:     page = browser.new_page()
```

Default Addons

Camoufox will automatically download and use the latest uBlock Origin with custom privacy/adblock filters to help with ad circumvention.

You can also **exclude** default addons with the `exclude_addons` parameter:

```
from camoufox.sync_api import Camoufox from camoufox import DefaultAddons  with Camoufox(exclude_addons=[DefaultAddons.UBO]) as browser:     page = browser.new_page()
```

Miscellaneous

Spoof battery status and PDF viewer.

Property	Type	Description
`pdfViewerEnabled`	bool	Sets navigator.pdfViewerEnabled. Please keep this on though, many websites will flag a lack of pdfViewer as a headless browser.
`battery:charging`	bool	Battery charging status
`battery:chargingTime`	double	Battery charging time
`battery:dischargingTime`	double	Battery discharging time
`battery:level`	double	Battery level

Note

Please keep pdfViewer on, as many websites will flag a lack of PDF viewer as a headless browser.

Development

Note

The following content is intended for those interested in building & debugging Camoufox.

Building

Instructions on how to build Camoufox from scratch:

- Build system overview
- Build via CLI
- Build via Docker

Developing patches

How to develop patches for Camoufox and find leaks:

Build System

Here is a diagram of the build system, and its associated make commands:

This was originally based on the LibreWolf build system.

Building in CLI

Warning

Camoufox"s build system is designed to be used in Linux. WSL will not work!

First, clone this repository with Git:

```
git clone --depth 1 https://github.com/daijro/camoufox cd camoufox
```

Next, build the Camoufox source code with the following command:

```
make dir
```

After that, you have to bootstrap your system to be able to build Camoufox. You only have to do this one time. It is done by running the following command:

```
make bootstrap
```

Finally you can build and package Camoufox the following command:

```
python3 multibuild.py --target linux windows macos --arch x86_64 arm64 i686
```

CLI Parameters

```
Options:   -h, --help          show this help message and exit   --target {linux,windows,macos}
[{linux,windows,macos} ...]                    Target platforms to build   --arch {x86_64,arm64,i686}
[{x86_64,arm64,i686} ...]                    Target architectures to build for each platform   --bootstrap
Bootstrap the build system   --clean            Clean the build directory before starting  Example: $ python3
multibuild.py --target linux windows macos --arch x86_64 arm64
```

Building in Docker

To build Camoufox on a non-Linux system you can use Docker.

1. Create the Docker image containing Firefox"s source code:

```
docker build -t camoufox-builder .
```

1. Build Camoufox patches to a target platform and architecture:

```
docker run -v \"$(pwd)/dist:/app/dist\" camoufox-builder --target <os> --arch <arch>
```

How can I use my local ~/.mozbuild directory?

Docker CLI Parameters

```
Options:   -h, --help        show this help message and exit   --target {linux,windows,macos}
[{linux,windows,macos} ...]                    Target platforms to build   --arch {x86_64,arm64,i686}
[{x86_64,arm64,i686} ...]                Target architectures to build for each platform   --bootstrap
Bootstrap the build system   --clean          Clean the build directory before starting  Example: $ docker run -
v \"$(pwd)/dist:/app/dist\" camoufox-builder --target windows macos linux --arch x86_64 arm64 i686
```

Build artifacts will now appear written under the `dist/` folder.

Development Tools

This repo comes with a developer UI under scripts/developer.py:

```
make edits
```

Patches can be edited, created, removed, and managed through here.

How to make a patch

How to work on an existing patch

Leak Debugging

This is a flow chart demonstrating my process for determining leaks without deobfuscating WAF Javascript. The method incrementally reintroduces Camoufox"s features into Firefox"s source code until the testing site flags.

This process requires a Linux system and assumes you have Firefox build tools installed (see here).

Flowchart

Yes

No

Yes

No

No

Yes

Start

Does website flag in the
official Firefox?

Likely bad IP/rate-
limiting. If the website
fails on both headless and
headful mode on the official
Firefox distribution, the
issue is not with the browser.

Run make ff-dbg(1) and
build(2) a clean
distribution of Firefox.

Does the website flag in
Firefox **headless** mode(4)?

Does the website flag in
headful mode(3) AND
headless mode(4)?

Open the developer UI(5),
apply config.patch, then
rebuild(2). Does the
website still flag(3)?

Enable privacy.resistFingerprinting
in the config(6). Does the
website still flag(3)?

In the config(6), enable
FPP and start omitting
overrides until you find
the one that fixed the leak.

If you get to this point,
you may need to deobfuscate
the Javascript behind the website
to identify what it"s testing.

Open the developer UI,
apply the playwright
bootstrap patch, then
ebuild. Does it still flag?

Omit options from
camoufox.cfg(6) and
rerun(3) until you find the
one causing the leak.

Juggler needs to be
debugged to locate the leak.

The issue has nothing to do
with Playwright. Apply the
rest of the Camoufox patches
one by one until the one
causing the leak is found.

Cited Commands

Liability Disclaimer

Camoufox does NOT provide liability or warranty.

No responsibility is accepted for the use of this software. This software is intended for ethical and research purposes only. Users should use this software at their own risk. The developers of the software cannot be held liable for any damages that result from the use of this software. The software must not be used for malicious purposes. By using this software, you agree to this disclaimer and acknowledge that you are using the software responsibly and in compliance with all applicable laws and regulations.

Acceptable Use

By using Camoufox, the User agrees **NOT**:

- to violate, or encourage the violation of, the legal rights of others;
- to engage in, promote or encourage illegal activity, including child sexual exploitation, child abuse, or terrorism or violence that can cause death, serious harm, or injury to individuals or groups of individuals;
- to use Camoufox for any unlawful, invasive, infringing, defamatory or fraudulent purpose including Non-Consensual Explicit Imagery (NCEI), violating intellectual property rights of others, phishing, or creating a pyramid scheme;
- to distribute viruses, worms, Trojan horses, corrupted files, hoaxes, or other items of a destructive or deceptive nature;
- to gain unauthorized access to, disrupt, or impair the use of any service, or the equipment used to provide service, by customers, authorized resellers, or other authorized users;
- to generate, distribute, publish or facilitate unsolicited mass email, promotions, advertisements, or other solicitations ("spam"); or
- to use any service, or any interfaces provided with any service, to access any other product or service in a manner that violates the terms of service of such other product or service without authorization.