



Haketilo Documentation

This is the documentation of Haketilo, a tool that facilitates viewing of websites while having their original JavaScript replaced by user-provided scripts. Haketilo combines the functionalities of content blocker and user script manager. It can be used with its script repository, [Hydrilla](#)  

One of Haketilo's aims is to address the issues raised in "[The JavaScript trap](#)". It is being developed with hope that it will make more user-controlled "Web" browsing possible.

Haketilo exists as an HTTP proxy and a browser extension. The latest release of Haketilo proxy is version **3.0-beta1**. It incorporates the popular [mitmproxy](#) tool and should work with any web browser that supports HTTP proxies.

Haketilo browser extension is currently in maintenance mode and is not going to have new features added. Its latest release is version **2.0.1**. WebExtension platforms supported by this version are Mozilla-based browsers (Firefox/IceCat 60 and newer) and (Ungoogled) Chromium (versions 90 and newer; older *might* also work but are untested).

Wiki sections

- [User manual](#)
- [About Haketilo](#)
- [Technical documentation](#)
- [Known limitations](#)
- [Reporting bugs](#)
- [Downloads](#)
- [Roadmap](#)

Contributing

Registrations on [hydrillabugs.koszko.org](#) are currently disabled. Please e-mail koszko@koszko.org to express interest in participating.

For information about markup used by this documentation, see [Markdown on RedMine](#).

License

This wiki is Copyright 2021, 2022 [Wojtek Koszior](#) and [Jahoti](#). It is a free cultural work made available under:

- [Creative Commons Attribution-ShareAlike 4.0 International](#)
- [GNU Free Documentation License version 1.3 or \(at your option\) any later](#)

You are free to choose which of the above licenses to use.

I, Wojtek Koszior, thereby promise not to sue for violation of this wiki's license. Although I request that you do not make use this wiki's contents in a proprietary work, I am not going to enforce this in court.

Donors

_ banner.png	_ NGI0Discovery_tag.svg
------------------------------	---

This project was funded through the [NGI0 Discovery](#) Fund, a fund established by NLnet with financial support from the European Commission's [Next Generation Internet](#) programme, under the aegis of DG Communications Networks, Content and Technology under grant agreement N° 825322.

Files

banner.png	16.7 KB	08/05/2023	koszko
NGI0Discovery_tag.svg	13.9 KB	08/05/2023	koszko

About Haketilo

Haketilo is, loosely, a combined content blocker and user script manager. As a whole it is available under GNU GPLv3, with additional permissions regarding particular files.

- [Use cases \(Support for software freedom\)](#)
- [Comparison with other tools](#)

Comparison with other tools

The functionality of Haketilo partially overlaps with that of other programs. None of those, however, seemed to aim for exactly the goals of Haketilo when the decision to start its development was made.

LibreJS

Of existing tools [LibreJS](#) seems to be the only one with a software-freedom-oriented goal. It is a browser extension which inspects sites' scripts in search of license notices in a specified format. If it decides a given script is libre software or otherwise trivial, it allows it to run.

While LibreJS shares many of its social aims with Haketilo, its scope is different. At the time of this writing it facilitates script blocking (and not replacement or injection) on HTTP(s) pages in Mozilla-derived browsers.

More fundamentally, there is also a difference in the underlying assumptions about webmasters. Haketilo takes an adversarial view, refusing to trust or expect co-operation by default. LibreJS more charitably gives webmasters greater freedom to choose what scripts run in users' browsers. This happens at the cost of more burdensome requirement of maintaining license notice in scripts that are use on website. It should be noted that Haketilo's script blocking can be disabled, allowing for example LibreJS to be used for script blocking instead.

NoScript

[NoScript](#) is an extension often used as a content blocker. However, it actually does a bit more and is more properly described as a security suite. It can be used to block scripts on per-site basis and works with both Firefox- and Chromium-derived browsers. NoScript author, Giorgio Maone, has also worked on LibreJS as an FSF contractor.

NoScript, while sometimes useful for the task of blocking nonfree JavaScript, is not by itself fully freedom-oriented and for example has youtube.com in its default site whitelist.

uBlock Origin

Often abbreviated as "UBO", [uBlock Origin](#) is a selective content blocker. It gives quite fine-grained control over what kinds of elements are allowed to load, including the possibility of blocking third-party resources on a per-domain basis. Firefox and Chrome and at least 2 proprietary browsers are supported.

UBO is able to load and apply adware and spyware blacklists from several sources which makes it able to function as an ad-blocker. However, what's also relevant for people who want to hack on the software they use, is that UBO's codebase - although big - is rather clean and readable.

uMatrix

The [uMatrix extension](#) is UBO's twin. Developed by the same author, these 2 share part of their codebase. While UBO can be used rather easily, uMatrix was a content blocker aimed at more advanced users.

Greasemonkey

The tools discussed so far have all been content blockers. [Greasemonkey](#), on the other hand, makes it possible to execute custom scripts on websites. These are usually referred to as "user scripts" and there are even sites (well, at least one) for sharing these between users. Greasemonkey only supports Firefox-derived browsers.

People often point at Greasemonkey as a possible solution when told about the need for a facility to replace sites' native JavaScript. Indeed, Greasemonkey could be used to achieve that. The fact that it doesn't block the original scripts is a small problem which gives the necessity of running some content blocker next to it. While generally suboptimal, this setup would be acceptable as a temporary solution to Haketilo's primary goal.

At the time of writing Greasemonkey doesn't execute user scripts in the context of a page but rather in the more privileged context of WebExtensions content scripts¹. This might change with policy changes to extension stores². The current approach brings in security issues, although largely mitigated by the use of sandbox. There might also be incompatibilities with scripts failing to execute the way they would in the usual context. These incompatibilities could possibly also be bypassed in some way.

Site's CSP rules cause yet another possible issue when customizing pages using Greasemonkey³. In some cases they may block custom injected elements like `<script>s` and `s`.

ViolentMonkey

[ViolentMonkey](#) is similar to and largely compatible with Greasemonkey, with the benefit of supporting a wide range of browsers. All other potential issues listed for Greasemonkey still apply, however.

JShelter

[JShelter](#) improves browser security and privacy mainly by wrapping potentially dangerous browser APIs. It's not a content blocker nor a user script manager. Although it doesn't prevent proprietary JavaScript from running, it does greatly reduce its harm. It supports Firefox, Chrome, and Opera.

Hypothesis

[Hypothesis](#) project offers facility for community-driven annotating of web sites. This idea is similar to one of Haketilo's desired use-cases and it's even possible that Haketilo will, at some point, support Hypothesis annotations. However, the general goals of these project are different.

Woob

[Woob](#) tool implements graphical (QT) interfaces and programming APIs for various websites in Python programming language. It succeeds in achieving some of the goals we set in front of Haketilo. The main difference is that our project sticks to the usual technological stack of the Web (which has both good and bad sides) and also covers creations of a repository that can allow for greater scalability.

1. <https://github.com/greasemonkey/greasemonkey/blob/efd22a93121225ada47f2fe9f021af0ab6100c21/src/bg/execute.js#L20> ↔

2. <https://developer.chrome.com/docs/extensions/mv3/intro/mv3-overview/#remotely-hosted-code> ↔

3. <https://github.com/greasemonkey/greasemonkey/issues/2046> ↔

Comparison with other extensions (Table)

This document is currently incomplete and unchecked. Do not refer to it.

	Content Blocking	User Freedom	Security	Site Compatibility	Custom Scripts
Haktilo	yes	optimal	optimal	weak	yes
LibreJS	yes	fair	none	fair	no
JShelter	no	none	configurable	very strong	no
LibreJS+JShelter	yes	fair	configurable	fair	no
NoScript	yes	weak	fair	fair	no
uBlock Origin	yes	very weak	weak	very strong	no
GreaseMonkey	no	none	none	optimal	yes

Future plans

In its current state, the extension does not yet allow for all the types of enhancements listed so far. In addition, there are some features and use-cases worth discussing more thoroughly that go beyond creation of a single browser extension.

Site translations

This is a potentially very useful feature, but unfortunately it might cause legal problems. Translated version of websites' text would still be covered by copyright of the original and could hence be impossible to distribute legally. In many cases copyright holders of text available online would be OK with someone translating it but asking for permission every time would definitely be unfeasible. A solution could be to encode translations in some way so that the actual translated text can only be reproduced with access to the original. This way we could claim it is only translation and not the copyrighted original that is distributed. Wayback Machine could be leveraged to produce translation of a site that ceases to be available.

Port to Manifest v3

Google is introducing a breaking change to its WebExtensions ecosystem. The new extensions format differs significantly from the previous one, so do the APIs provided. Mozilla follows this path and is also going to implement what is called "Manifest v3".

There's no point hiding that at least one of this change's goals is making things harder for ad blockers (and hence, also content blockers). While it is going to be a hurdle to implement proper content blocking and substitution in this scenario, it should be achievable and we still want to support Chromium-based browsers for as long as reasonably possible.

It is worth mentioning that Mozilla, unlike Google, claims to have plans to retain the `webRequest` API extensively used by current content-blocking browser extensions. In the end, this might lead us to supporting less features in Chromium-based browsers and more in Gecko-based or even WebKit-based¹ if we get to implementing an analogous functionality there.

Platform for sharing custom content

Besides being able to store pages settings locally and import/export them in JSON format, Haketilo is able to query its repository server, [Hydrilla](#), to find more custom site resources.

The ultimate goal of Hydrilla is to be a platform where anyone can contribute scripts and other works to make internet more user-controlled. We also need some means to impose guidelines and prevent spam and abuse on the platform. A scheme where long-time, credited contributors are more privileged while newcomers' contributions, however welcome, need to be approved, is likely to work best in this case. We are going to base on the practices developed by projects like Debian and Wikipedia that are known to be dealing with this issue well.

The repository software is also released under a libre license. Even though we, project authors, are running our (soon-)official Hydrilla server, the extension (or any other client-side software we develop to work with Hydrilla) is going to make it easily possible for a user instead connect to a third-party repository or even multiple ones. While it is nice and sometimes beneficial to be "the only player", it would be against the spirit of software freedom and free culture to hamper the creation of alternative repositories by those who desire so. If someone can do this better than we, the founders, can, that person should be allowed to.

It is worth mentioning we, current project members, are very concerned about software freedom. It might happen that other, third-party repositories that appear over time will have looser conditions, maybe even allowing proprietary scripts. While this is unfortunate and such repositories should not be endorsed, they should be allowed to operate freely.

Support for WebKit-based¹ browsers

Currently supported Firefox- and Chromium-based browsers are controlled by big entities and may go into the wrong direction in the future. Internet users' freedom should not depend on them and it is certainly not enough if we support just these browsers.

WebKit is a browser engine commonly used in free software projects, partially due to its easy embedding in other applications. There are several minor web browsers based on it. While in the case of Mozilla and Google browsers the most feasible approach to achieve our goal is to develop a WebExtension, in the case of WebKit it is going to be relatively easy to build the required functionality into the actual browser. This brings along a great advantage of not having to rely on some overly restrictive JavaScript APIs. In fact, if this project succeeds, it could make sense for it to start a completely new browser based on WebKit, using the format developed here for extensions.

While there also exist custom browsers based on Firefox and Chromium, they suffer from Google's great influence. Some would point out that WebKit is way behind the 2 major browser engines in terms of features. While true, this is relatively harmless for our project, since the goal is to be able to run custom JavaScript with web pages and such custom scripts can be further customized to work despite lack of some fancy features WebKit didn't implement. Additionally, a smaller number of features might be a feature of itself, as described below.

Creation of a new, simplified Web standard

Besides the issue of nonfree JavaScript and lack of user control, the "Web" has yet another issue - it is increasingly complex and is getting even worse in that matter. Someone actually [made the effort](#) to compute a word count of all "Web" standards combined. This complexity leads to web browsers becoming operating systems of their own, with even big enterprises like Microsoft being unable to keep up with the changes and resorting to using Google's implementation.

Current state of affairs leads to vendor lock-in and is generally dangerous to the entire WWW ecosystem. Some responses to this issue have appeared. One of them is the [Gemini](#) protocol which is lighter than the widespread HTTP/HTML/CSS/JavaScript stack. Such project are good and should be promoted. However, even the Gemini Project's overview states it is not going to replace the "Web". What will, then?

A feasible replacement for the "Web" needs to be something that will be appealing to both enterprises and non-technical users. It needs to be sufficient for things like content uploading and user login. One suggestion is to standardize a modest subset of HTML and CSS (without JavaScript) that would be sufficient for most of the use-cases of current "Web". Some important features have been added to HTML and CSS over the years that enable creation of good-looking, responsive content without use of JavaScript. These would be retained. At the same time some advanced use-cases like those that rely on WebSockets or WebRTC would simply be out of this new standard's scope (although a lightweight, compliant browser engine could still be embedded into standalone applications using these technologies).

Once embraced by businesses and organizations, such standard would allow for much more secure, lightweight and stable websites. At the same time, it could be used with most of existing tools. While convincing decision-makers to use it might seem like a difficult task, possible advantages are big enough that intelligent people are expected to consider it seriously. A good name is also important; "Web Zero" seems pretty catchy. Just as Coca-Cola Zero has been advertised as a drink that can help people lose weight, this standard could help "Web" become more lightweight and easier to develop tools for.

How does this all relate to this browser extension and project in general? In the case of most websites, their owners are obviously not going to instantly make them available on Web Zero. The possibility of making custom interfaces for existing websites has already been mentioned. Taking the idea one step further, it would be possible to adapt Hydrilla to also facilitate sharing code capable of proxying certain websites into Web Zero.

While this is indeed a very distant idea, it presents a perfectly valid way this project could help replace the "Web" with a lighter one.

1. Throughout this page we use name "WebKit" to refer to [Apple WebKit](#) and not [Google's fork of it](#) ↩

Social hurdles

The concept of this extension relies on custom, libre JavaScript being written for many sites. Some claim this effort is pointless as sites can change at any time. It's true that this is an issue and we're fated to play a cat-and-mouse game, but this does not mean we shouldn't try or that we are fated to lose.

Website owners' cooperation

So, how to convince website owners not to make breaking changes to their sites and to release their JavaScript as free software?

Pressuring the bad

The first good thing about our approach is that we actually have a voice. If a website is committed to keeping its js nonfree, a third-party script substitute might appear. Even if there aren't any now, the mere possibility of them showing up has a chance of making webmasters treat us seriously. The key is that a site might not like the substitutes. These could work in a different way than the site's operator wants. We, as people in control of a scripts repository, would be in a good position for negotiations: as long as the site cooperates and releases its JavaScript as libre software, we could agree to include these and not third-party scripts as *the* scripts to substitute for that site.

Doesn't this undermine the whole purpose of giving users control? No, we could still distribute alternative, third-party scripts for such cooperating sites in the repository. We would just make it so that it is the user who has to manually choose to run these and not the ones the repository marks as default. Also, we could impose further restrictions on site-donated scripts that are to be included in the repo; for example, disallowing sending of user's personal data to advertisers.

We can even do better - only distribute script substitutes that block all a site's ads *until* the site cooperates. Sure, some ad blocking fans wouldn't like the idea of allowing ads even from "good" websites and could decide to leave our platform because of that. But again, we can still allow other substitutes, including ad-blocking ones, as non-default for given site.

One could think no website owner would be afraid of any of our actions, since a change to the site can break our custom scripts. But the truth is, modifying a website requires resources and webmasters are lazy. Except for some special sites, breaking changes occur infrequently and nobody wants to put valuable resources into constant redesigns.

Besides scripts and other customizations, things like site descriptions, user comments, maybe even ratings and warnings could also be shared through the platform we propose.

Rewarding the good

Our facility shall do more than just replace JavaScript. Things like user-provided translations, custom styling or accessibility and usability enhancements can be reasonably expected to be welcomed by webmasters. In fact, if volunteer contributions prove good enough, it wouldn't be surprising to see them incorporated into websites. Not to mention that possibility of getting positive reviews on the platform could also be an incentive for cooperative website owners.

We can and should present this project as something good for website owners and not just an enemy of theirs.

Sustaining the platform

While this project is not aimed at generating profit, there are nevertheless some ways in which it and its contributors can possibly gain money.

NLnet fund

Among funds operated by the Dutch [NLnet Foundation](#) is the [User-Operated Internet Fund](#). It will support initiatives to "help create an open, trustworthy and reliable internet for all". The description of the fund matches the goals of our project *exactly* and we have applied. We hope such proper funding could help us pay for developers' time needed to make advancement and start fixing the "Web".

Donations

There are internet users who care about their freedom and there are also those who would be convinced by other improvements our project could bring. We hope they will appreciate our work and once the platform becomes notable, some of them will become donors. While we do not count on donations being sufficient to fund a salary for everyone who contributes, they might be just enough to keep any necessary infrastructure running and encourage those who commit their free time to the cause to commit even more.

Once a proper script substitutes repository is set up, we also hope for independent organizations and individuals to come up with mirrors of it.

Being paid directly for some tasks

Someone who wants custom scripts or features to be developed for some site could contract one of the developers or contributors of this platform for that. In fact, even an owner of some website who for some reason wants it to be supported might decide to go this way. In the end, the platform could also become a job-finding tool for programmers and web developers similar to Stack Overflow.

Deals and ads

As already mentioned, in case of uncooperative websites it can be required that all applicable script substitutes in the repository block their ads. When a site starts cooperating and releases its JavaScript as libre software, we could request it to make a donation before support for displaying the ads is

restored in script substitutes.

This idea by itself is not perfect. Use of major ad systems is likely to be impossible in a surveillance-free way, in which case even libre scripts interacting with those ad systems deserve rejecting. Perhaps a better option would be for the platform to allow users to opt in for some privacy-friendly ads.

One thing is certain: users who don't want to see ads, should not be forced to. While it might make sense to make some trade-offs in various fields, neither user control nor freedom should be one of these and we must be ready to sacrifice possible profits.

Volunteer contributions

Despite some possible ways of funding the work as described above, community is still going to form the core of this platform. Many excellent initiatives, from Wikipedia through GNU/Linux distributions to existing ad blocking databases, are being worked on by volunteers. The more useful this project is to the general public, the more individuals are going to decide to support it.

Use cases

Haketilo can be used to help computer users gain control over their web browsing. For a long time, major web browsers by themselves have been giving users little choice but to view a website almost exactly as it is served to them by an HTTP server. Whether or not its JavaScript is malicious - it is executed. Whether or not the user likes its interface - it is not possible to simply substitute it with another one. Neither is it possible to provide an independent translation of a web page for others to use. Websites with accessibility problems mostly retain them until their owners make a redesign with accessibility in mind. It is also difficult to use some sites on mobile and other small devices or with slow and unstable internet connection because webmasters often don't, or didn't, account for that.

Over time many browser extensions have appeared that could help solve some of these issues. There are ad blockers that got very popular as a result of annoying advertisements becoming too pervasive. There are also tools that can be used to automatically block some kinds of spyware, selectively disable JavaScript or third-party connections on web pages and even enhance sites with custom user scripts. Some web browsers even have these functionalities built in.

While tools like those described above continue to exist and serve their users, they are usually limited to doing just one of those things and don't provide the high extent of control some people need. This extension is meant to be a basis for a bigger, browser- and extension-agnostic platform with its own public script repositories that would allow for

- JavaScript and other types of unwanted content to be blocked when browsing,
- custom JavaScript to be run on sites, for example to
 - [fix sites that stop working after their original, proprietary, spyware-encumbered JavaScript is disabled](#),
 - add features or
 - just have more control over the client-site code executing in one's browser,
- custom interfaces to be provided for websites, for example to
 - make heavy sites usable on resource-constrained devices,
 - make sites accessible,
 - be able to use them more conveniently,
 - aggregate content from many websites in one place and
 - add even better features,
- lightweight and ethical website design to be promoted.

Support for software freedom

There are quite a few ways this extension could be used, meaning people with different objectives may find it useful. However, the main motive for the authors of this extension was to be able to browse the "Web" without giving up on software freedom.

During the last 2 decades JavaScript (scripting language of the "Web", abbreviated as "js") has become very pervasive, with a majority of WWW sites utilizing it today¹. While many websites could be accused of being bloated and overusing js, there is yet another problem - the scripts that are being run in users' browsers quite often aren't [libre software](#) and cannot be legally modified by the user. We, who value software freedom, consider it an injustice and block those scripts from executing, for example using browser extensions like NoScript.

The issue was described in "[The JavaScript Trap](#)" by [Richard Stallman](#) (RMS) as early as in 2009. While suggesting blocking of nonfree scripts in the browser, RMS also advocated creation of a facility to replace these with libre scripts so that websites would be working again. While an extension called LibreJS eventually appeared that could automatically recognize properly-tagged libre scripts on pages and allow only them to execute, until 2021 there was still no convenient way for users to develop, share and run script substitutes.

This extension aims to make up for negligence of this issue in previous years. There is at least nothing to lose - with so many websites non-functional without proprietary code, it can only get better now.

1. <https://w3techs.com/technologies/details/cp-javascript> ↩

Known limitations

This page lists surprising, non-obvious issues and possible privacy shortcomings of Hydrilla & Haketilo proxy as of version 3.0-beta1. You can read about the limitations of Haketilo browser extension (now in maintenance mode) [here](#).

- The proxy only works with HTTP(s) pages and not for example FTP ones.
- Haketilo doesn't have additional features one could expect from a content blocker, e.g.
 - there is currently no option to disable Service Workers while allowing normal scripts to execute and
 - there is currently no option to disable loading of external/third-party fonts, tracking pixels or other strategies that are used by Google and similar companies to snoop on internet users.
- There is currently no facility to anonymize queries Haketilo makes to its script repositories. This means admin of a [Hydrilla](#) instance could theoretically see the sites for which a Haketilo user downloaded custom scripts.
- Haketilo might still be missing:
 - UI Translations
 - Accessibility features
 - Mobile support
- Haketilo currently tries to prevent the browser from caching HTTP responses. This is needed to make it function properly on some sites. Unfortunately, this behavior may cause a big performance hit.
- If user's browser gets to cache a page in spite of Haketilo's anti-caching measures, it might later be unable to "pick up" changes in Haketilo configuration that affect the cached page.
- If a user temporarily disables the proxy and continues to use the browser on a page that was modified by Haketilo, page's website might learn a Haketilo-generated token that ought to remain private.
- There are almost certainly other bugs.

Known limitations (browser extension)

We list surprising, non-obvious issues and possible privacy shortcomings in version 2.0.1 of Haketilo browser extension. For a list of Haketilo proxy's limitations, see [here](#).

- Currently, user scripts can only be injected to HTML pages.
- Currently, user scripts under Mozilla-based browsers may fail to run on pages loaded from file:// schema if a CSP rule is embedded in a <meta> tag early in page's HTML.
- On some (Mozilla-based) browsers Haketilo might disrupt XML preview.
- Haketilo doesn't have additional features one could expect from a content blocker:
 - There is currently no option to disable Service Workers while allowing normal scripts to execute.
 - There is currently no option to disable loading of external/third-party fonts, tracking pixels or other strategies that are used by Google and similar companies to snoop on internet users.
- There is currently no facility to anonymize queries Haketilo makes to its script repositories. This means admin of a [Hydrilla](#) instance could theoretically see the sites for which a Haketilo user tried to find custom scripts.
- Haketilo is still missing:
 - UI Translations
 - Accessibility features
 - Mobile support
- Pages that were open during Haketilo's installation/enabling might break in weird ways (under Mozilla browsers) and Haketilo will not function there properly (all browsers). Reloading each browser tab is the simplest solution to this issue.
- There are almost certainly other bugs.

Compatibility with other extensions

While it is not known to be a common problem, Haketilo may interfere with or experience interference from other extensions- especially other content blockers. Extensions where information on this is available are documented here.

LibreJS

Haketilo can be used alongside LibreJS on pages where payloads will not be injected if its default policy is set to allow pages' own scripts; otherwise LibreJS' filtering will be overridden. All configuration of script-blocking in such a setup, including whitelisting and blacklisting, should be done through LibreJS only.

Pages with payloads will have all scripts blocked anyway, as is the expected behavior for Haketilo.

NoScript

NoScript is *expected* to be mostly safe for use with Haketilo; however, on Mozilla Firefox and its derivatives, enabling NoScript's script blocking capabilities will prevent payload injection. Further testing is required to confirm no other issues exist.

LibreJS/USPS compatibility and other site fixes shipped with IceCat

These will conflict if a corresponding payload is installed in Haketilo. We advise disabling them; work is currently ongoing to port these to Haketilo.

Releases

This page lists releases of Haketilo proxy and Hydrilla which, starting with version 3, are distributed together. If you are interested in Haketilo browser extension (now in maintenance mode), take a look at [its own downloads page](#). If you are interested in Hydrilla releases prior to version 3, see [this page](#).

Public keys for verification of signatures are the ones from [koszko.org](#).

PGP key fingerprint: **E972 7060 E3C5 637C 8A4F 4B42 4BC5 221C 5A79 FD1A**
[Signify](#) public key: **RWQSF2wUdpjAtrmt7D3t9iHrHFL/GpqXOF+NxECx8ck7swrx6tNzDkM9**

You might want to read our [instructions](#) on signature verification.

Version 3.0-beta1 (pre-release version)

The binary release has been

Files

- [haketilo-and-hydrilla-bin-3.0b1-x86_64.tar.gz](#) (relocatable standalone binary release for x86-64 computers)¹
- [hydrilla-3.0b1-py3-none-any.whl](#) (Python wheel for use with pip)
- [haketilo-and-hydrilla-3.0b1.tar.gz](#) (source tarball)

Signify signatures

- [haketilo-and-hydrilla-bin-3.0b1-x86_64.tar.gz.sig](#)
- [hydrilla-3.0b1-py3-none-any.whl.sig](#)
- [haketilo-and-hydrilla-3.0b1.tar.gz.sig](#)

PGP signatures

- [haketilo-and-hydrilla-bin-3.0b1-x86_64.tar.gz.asc](#)
- [hydrilla-3.0b1-py3-none-any.whl.asc](#)
- [haketilo-and-hydrilla-3.0b1.tar.gz.asc](#)

1. Made with GNU Guix version **1.3.0-26.fd00ac7**. [↵](#)

Releases (browser extension)

This page contains download links for releases of Haketilo as a browser extension. Be aware that the extension is now in maintenance mode. Please instead look at the [Haketilo proxy releases](#) page if you are interested in trying out new features.

Public keys for verification of signatures are the ones from koszko.org.

PGP key fingerprint: **E972 7060 E3C5 637C 8A4F 4B42 4BC5 221C 5A79 FD1A**

[Signify](#) public key: **RWQSF2wUdpjAtrmt7D3t9iHrHFL/GpqXOF+NxECx8ck7swrx6tNzDkM9**

You might want to read our [instructions](#) on signature verification.

Version 2.0.1

Files

- [haketilo-2.0.1.zip](#) (Chromium build)
- [haketilo-2.0.1.xpi](#) (Mozilla build)
- [haketilo-2.0.1.tar.gz](#) (source code)

Signify signatures

- [haketilo-2.0.1.zip.sig](#)
- [haketilo-2.0.1.xpi.sig](#)
- [haketilo-2.0.1.tar.gz.sig](#)

PGP signatures

- [haketilo-2.0.1.zip.asc](#)
- [haketilo-2.0.1.xpi.asc](#)
- [haketilo-2.0.1.tar.gz.asc](#)

Version 2.0

Files

- [haketilo-2.0.zip](#) (Chromium build)
- [haketilo-2.0.xpi](#) (Mozilla build)
- [haketilo-2.0.tar.gz](#) (source code)

Signify signatures

- [haketilo-2.0.zip.sig](#)
- [haketilo-2.0.xpi.sig](#)
- [haketilo-2.0.tar.gz.sig](#)

PGP signatures

- [haketilo-2.0.zip.asc](#)
- [haketilo-2.0.xpi.asc](#)
- [haketilo-2.0.tar.gz.asc](#)

Version 2.0-beta1 (pre-release version)

Files

- [haketilo-2.0b1.zip](#) (Chromium build)
- [haketilo-2.0b1.xpi](#) (Mozilla build)
- [haketilo-2.0b1.tar.gz](#) (source code)

Signify signatures

- [haketilo-2.0b1.zip.sig](#)
- [haketilo-2.0b1.xpi.sig](#)
- [haketilo-2.0b1.tar.gz.sig](#)

PGP signatures

- [haketilo-2.0b1.zip.asc](#)
- [haketilo-2.0b1.xpi.asc](#)
- [haketilo-2.0b1.tar.gz.asc](#)

Version 1.0

Files

- [haketilo-1.0.zip](#) (Chromium build)
- [haketilo-1.0.mozilla-signed.xpi](#) (Mozilla build)
- [haketilo-1.0.tar.gz](#) (source code)

Signify signatures

- [haketilo-1.0.zip.sig](#)
- [haketilo-1.0.mozilla-signed.xpi.sig](#)
- [haketilo-1.0.tar.gz.sig](#)

PGP signatures

- [haketilo-1.0.zip.asc](#)
- [haketilo-1.0.mozilla-signed.xpi.asc](#)
- [haketilo-1.0.tar.gz.asc](#)

Version 1.0-beta3 (pre-release version)

Files

- [haketilo-1.0b3.zip](#) (Chromium build)
- [haketilo-1.0b3.xpi](#) (Mozilla build)
- [haketilo-1.0b3.tar.gz](#) (source code)

Signify signatures

- [haketilo-1.0b3.zip.sig](#)
- [haketilo-1.0b3.xpi.sig](#)
- [haketilo-1.0b3.tar.gz.sig](#)

PGP signatures

- [haketilo-1.0b3.zip.asc](#)
- [haketilo-1.0b3.xpi.asc](#)
- [haketilo-1.0b3.tar.gz.asc](#)

Version 1.0-beta2 (pre-release version)

Files

- [haketilo-1.0b2.zip](#) (Chromium build)
- [haketilo-1.0b2.xpi](#) (Mozilla build)
- [haketilo-1.0b2.tar.gz](#) (source code)

Signify signatures

- [haketilo-1.0b2.zip.sig](#)
- [haketilo-1.0b2.xpi.sig](#)
- [haketilo-1.0b2.tar.gz.sig](#)

PGP signatures

- [haketilo-1.0b2.zip.asc](#)
- [haketilo-1.0b2.xpi.asc](#)
- [haketilo-1.0b2.tar.gz.asc](#)

Version 1.0-beta1 (pre-release version)

Files

- [haketilo-1.0b1.zip](#) (Chromium build)
- [haketilo-1.0b1.xpi](#) (Mozilla build)
- [haketilo-1.0b1.tar.gz](#) (source code)

Signify signatures

- [haketilo-1.0b1.zip.sig](#)

- [haketilo-1.0b1.xpi.sig](#)
- [haketilo-1.0b1.tar.gz.sig](#)

PGP signatures

- [haketilo-1.0b1.zip.asc](#)
- [haketilo-1.0b1.xpi.asc](#)
- [haketilo-1.0b1.tar.gz.asc](#)

Version 0.1

Built from Git commit e9b6187e0c5eaa4ac5ee41aacb261ae2da208c8f.

Files

- *Chromium build is not available as all installations were supposed to contain a different secret assigned during build time.*
- [haketilo-0.1.mozilla-signed.xpi](#) (Mozilla build)
- [haketilo-0.1.tar.xz](#) (source code)

PGP signatures

- [haketilo-0.1.mozilla-signed.xpi.sig](#)
- [haketilo-0.1.tar.xz.sig](#)

Haketilo/Hydrilla Roadmap

Planned tasks

This section lists tasks on which efforts are going to concentrate. It is not said that all of those tasks are being worked on at any given point in time. They are just considered to be potentially very beneficial when completed.

Distribution of Hydrilla and Haketilo in package managers ([#106](#))

It is beneficial to have tools available in a format specific to various operating system distributions. While the process of inclusion in official repositories is often a complex and lengthy one, preparing the actual packages, as is the goal of this task, is a good first step to making that happen.

To do

- .deb packaging of Haketilo and Hydrilla¹
- Nix packaging of Hydrilla
- Pacman PKGBUILDs for Haketilo and Hydrilla
- ~~Guix packaging of Haketilo and Hydrilla~~
- RPM packaging of Haketilo and Hydrilla

Development of Hydrilla website part ([#35](#))

A project's website makes its first impression, and therefore deserves special care. In our case the website will be part of our software Hydrilla.

To do

- planning a site structure
- designing a landing page
- cross-reference with Hydrilla to ensure uniformity of design and compatibility with the on-disk format
- crafting of text, graphics, and any other media
- assembly of website

Permissions system for Haketilo-supplied resources ([#73](#))

Custom, user-supplied resources Haketilo may deploy on viewed pages might require looser restrictions than those normally employed on pages. Or, they might allow for tighter security mechanisms to be employed.

To do

- specification of a new revision of Hydrilla API and on-disk format with permissions support²
- facility to limit domains for which a Haketilo-supplied script is allowed to perform unrestricted HTTP requests
- facility to specify what custom Content Security Policy should be used on a given pages ([#88](#))

Further means of user-controlled customization of sites ([#108](#))

Besides the initial function of replacing sites' JavaScript it is also desired to facilitate supplying additional data (e.g. images) and replacing other site components.

To do

- facility to make arbitrary bundled data files accessible to Haketilo-supplied scripts ([#69](#))
- facility to replace the entire interface of a web page with user-supplied HTML ([#70](#))
- facility to add user-supplied CSS to a web page
- facility to add user-supplied fonts to a web page

50 sample site resources for Haketilo ([#109](#))

To build the community its purpose depends on, Hydrilla must be clearly ready for use. This requires a representative, well-stocked library of packages.

To do

- guide describing how to make and contribute custom site resources to Hydrilla
- at least 5 alternative site interfaces
- JavaScript of at least 10 free/libre web tools (like Etherpad, Ethercalc) repackaged to be run in a user-controlled way from Haketilo
- at least 50 different custom site resources in total

Localization of Haketilo and Hydrilla

To truly empower to web users all over the world, Haketilo, Hydrilla, and all associated materials

must be able to support languages from across the world.

To do

- automatic content language negotiation on Hydrilla pages and the website
- language selection option on Hydrilla pages and the website
- internationalization of Haketilo ([#51](#))
- language selection option in Haketilo
- Polish translation

Tighter testing of Haketilo

Testing in multiple browser environments can help catch problems.

To do

- automated tests under each supported extension platform with at least 1 Firefox-based and Chromium-based platform
- integration tests of communication between Haketilo and a Hydrilla instance

Tooling for building of site resources

Simple scripts don't require building before distribution. Wasm modules and bigger libraries do. We could benefit from a well-defined way of accessing the sources and repeating the build process.

Hydrilla builder currently allows contents of APT packages to be reused in Haketilo packages. This already partially achieves the goal, since APT/Debian have a well-defined way of building packages. On the other hand, it might be more practical to instead use GNU Guix for the tasks as its package definitions can usually be contained inside a single file and it has a friendlier learning curve.

To do

- specification of new version of Haketilo source package format which gives ability to specify other programs the build process depends on
- Hydrilla builder functionality to automatically build a Haketilo source package ### Package signing in Haketilo and Hydrilla

Haketilo uses encrypted HTTPS connections to query Hydrilla API. However, to boost the security and enable use of mirrors, we plan to also use PGP signatures on site resources served.

To do

- specification of a new revision of Hydrilla API and on-disk format with PGP signatures support
- tool for batch signing of site resources
- Hydrilla support for serving PGP signatures
- Haketilo support for downloading and verifying PGP signatures
- facility to manage trusted public keys within Haketilo

Support for custom meta-sites in Haketilo/Hydrilla

Allowing users to modify pages loaded by their browsers is our goal. Allowing them to aggregate content from many sites on one page is a natural extension of it. Just as is allowing them to run static web apps without having to trust some website serving them.

To do

- specification of a new revision of Hydrilla API and on-disk format with meta-sites support
- support for meta-sites in Hydrilla and Haketilo ([#72](#))

REUSE specification compliance

License terms of software projects' files should be unambiguous and easy to analyze by humans and computers alike. Compliance with the REUSE specification helps ensure that.

To do

- ~~REUSE compliance in Haketilo&Hydrilla repository~~ (done)
- REUSE compliance in project website repository
- ~~REUSE compliance in custom site resources repository(ies)~~ (done)

Post/Redirect/Get in Haketilo pages

[Post/Redirect/Get](#) (PRG for short) is a common web development design pattern that makes navigation more convenient to users. It is not currently employed in Haketilo but it is something that should definitely be included in the plans.

Extra task ideas

This section lists tasks that could be considered in the future but which are not currently being worked on. In general, tasks on this list are considered to have higher amount-of-work:usefulness ratio than those in the [Planned tasks](#) section.

Upstream proxy configurable through Haketilo UI

It is currently possible to use [proxychains-ng](#) to tunnel Haketilo traffic through yet another proxy. This can be for example a [Tor](#) SOCKS proxy. Chaining Haketilo with other proxies could be made more convenient, especially for non-technical users. It can be achieved for example by integrating proxychains into Haketilo as a dependency.

Haketilo site resources available as GreaseMonkey user scripts (when applicable)

Haketilo in general aims to do something different than GreaseMonkey does. Despite that, some scripts distributed through Hydrilla could probably be also useful to GreaseMonkey users.

More thorough documentation of Haketilo and Hydrilla internals

With codebase refactored and stabilized, a worthy thing is to have it properly described for others to hack on.

To do

- graphical diagram(s) describing execution contexts in Haketilo and the way scripts running in various context communicate
- graphical diagram(s) describing the algorithm for querying by Haketilo URL patterns
- comprehensive description of strategies employed and APIs used for replacing scripts and CSP in Haketilo
- graphical diagram describing how entities (resources, mappings, licenses) depend on each another
- docstring documentation of every Python function
- HTML documentation generated from Python source code
- ~~JSDoc description of every Haketilo JavaScript function exported from file~~ (not applicable to Haketilo proxy)
- ~~HTML documentation generated from JavaScript source code~~ (not applicable to Haketilo proxy)

Sample meta-sites for Haketilo/Hydrilla

Running a static webapp like litewrite by visiting its website relies on the security of TLS and network connectivity. Having it packaged as a separate browser extension requires giving it excessive permissions. Running it from an HTML file is inconvenient.

To do

- at least 5 existing webapps packaged as meta-sites
- at least 5 meta-sites aggregating content from various client websites

User upload of custom site resources to Hydrilla website

To be able to easier gather and share custom site resources within the community, we need a user-friendly platform.

To do

- registrations on a Hydrilla instance
- upload of custom site resources to a Hydrilla instance
- facility to easily and efficiently moderate the content uploaded by users

Facility for setting up Hydrilla repository mirrors

While allowing users to set up independent instances of Hydrilla gives them greater control over site content they use, it does not by itself increase the robustness and maximum throughput of Hydrilla platform. Enabling the use of mirrors does.

To do

- support for setting up and automatically synchronizing Hydrilla mirrors
- support for announcing available mirrors in Hydrilla
- support for fetching repository mirrors list in Haketilo
- support for distributing requests over multiple repository mirrors in Haketilo
- documentation

Self-documented Haketilo

Now matter how user-friendly the graphical interface is, an explanation of some of the concepts might be needed. The next step, after having the documentation available on the project website, is bundling it with the extension itself.

To do

- ~~Haketilo popup self documented inline~~ (not applicable to Haketilo proxy)
- Haketilo settings page self-documented inline
- documentation included as extension-bundled HTML pages

Automatic generation of independent browser extensions from Haketilo site resources

Haketilo's rich feature set might also be an inconvenience. It may be overwhelming or irritating to some users and has a higher risk of breaking with newer browser versions than a simple extension would have. Thus, an option to install just a single Haketilo resource in the browser would be useful.

To do

- automatic generation of Firefox WebExtensions from Haketilo site resources
- automatic generation of Chromium ManifestV3 WebExtensions from Haketilo site resources

Facility to automatically convert page's "native" scripts to a Haketilo resource (#6)

Haketilo gives users control over scripts being executed on a given web page. The scripts to be used need to be defined in Haketilo as a resource. Doing this manually might be time-consuming for a user who aims to use mostly the same JavaScript a website normally serves, but served from within Haketilo.

To do

- automatic conversion of page's inline scripts in a Haketilo resource
- inclusion of page's external scripts in generated resource
- inclusion of page's intrinsic JavaScript events in generated resource (#7)
- displaying warnings when a site's JavaScript is known to use mechanisms that might stop such automatic package from working properly

Support for building Hydrilla and Haketilo using Autotools

The specificity of Haketilo and Hydrilla means a complex build system like Autotools is not necessary. It could, however, be added as optional to supplement the Python build system used.

To do

- Haketilo&Hydrilla buildable with Autotools
- Haketilo&Hydrilla out-of-source builds possible
- Haketilo&Hydrilla tarball producible with a make rule

Completed tasks

Section title leaves no need for additional explanation.

Haketilo and Hydrilla 1.0 pre-release (#103)

Some big code changes to land in Haketilo and Hydrilla 1.0 will be available in a pre-release. The pre-release will be made before delivery of several other side artifacts planned for 1.0.

To do

- [project plan](#)³
- [tentative software bill of materials](#)⁴⁵
- [use of registerContentScript API in Firefox Haketilo port](#) (#92)⁶
- [move to the new Hydrilla JSON API prototyped at \[https://hydrillabugs.koszko.org/projects/hydrilla/wiki/Repository_API\]\(https://hydrillabugs.koszko.org/projects/hydrilla/wiki/Repository_API\)](#)⁶
- [most WebExtension storage.local uses replaced with IndexedDB](#) (#98)⁶
- [Python implementation of Hydrilla](#)⁷

Haketilo and Hydrilla 1.0 release (#104)

This will be the first release since receiving the NLnet grant and the first non-demo release, hence it includes many improvements in various fields.

To do

- [basic automated Haketilo tests using Selenium and a Firefox based web browser](#) (#66)
- [JSON schemas describing Hydrilla on-disk resource format, Hydrilla HTTP API and other JSON interfaces in use](#)⁸
- [validation of all external JSON data in Haketilo and Hydrilla using included JSON schemas](#) (#105)⁹
- [sample Apache2 configuration file for use with Hydrilla](#) (#55)¹⁰
- [detailed documentation for installation and running of Hydrilla](#) (#55)¹¹
- [manpage for Hydrilla](#) (#55)¹²

Development of a user-controlled captcha client (#107)

Haketilo's goal is to give internet users control over their browsing. Replacing proprietary, privacy-hostile client-side programs is part of that. A tool similar to the librecaptcha Python program is needed, but in the form of a JavaScript library.

To do

- ~~facility for Haketilo-supplied scripts to bypass CORS¹³~~
- ~~free/libre JavaScript library for solving reCAPTCHA challenges¹⁴~~
- ~~sample Haketilo resource making use of the library on a chosen website¹⁵~~

Haketilo LibrePlanet presentation (#110)

LibrePlanet is a conference organized by the Free Software Foundation (FSF). It is "an opportunity to meet and interact with other people with both a technical and non technical background" and to share experience.

To do

- ~~applied to LibrePlanet 2022~~
- ~~prepared presentation about giving users back the control over web browsing~~
- ~~made the presentation at LibrePlanet 2022 (if accepted there) or posted a video presentation on Haketilo website (as a fallback case)¹⁶~~

Integrity constraints in Haketilo

One Haketilo custom site resource may depend on another, but initial versions of Haketilo did not verify that dependencies are present. This and other sanity checks can be employed.

To do

- ~~dependency checks when "installing" or upgrading a custom resource in Haketilo~~
- ~~dependency checks when removing a custom resource from Haketilo~~
- ~~facility for cascade removal~~
- ~~validation of Haketilo URL patterns and other values typed in by the user~~

Tasks that have been put aside

This section describes tasks that were once in the roadmap but which will not be specifically worked on. Tasks might have landed here for various reasons. It might be that they are too complicated to complete, too far-reaching or that their completion relied on actions of some other party. Regardless of the cause, the tasks are listed here for documentation purposes.

Security vetting of Haketilo and Hydrilla

As NLNet-funded projects, Haketilo and Hydrilla have the privilege of a security review from Radically Open Security. To make use of this opportunity, we will ensure any findings provided are properly addressed.

To do

- action on any recommendations or other findings
- report of how each finding from the vetting was addressed, and why
- note of any key issues in the developer documentation, in order to avoid repetition in the future

Accessibility vetting of Haketilo and Hydrilla

To empower every web user, Haketilo and Hydrilla must support the interfaces they need.

To do

- action on any recommendations or other findings
- report of how each finding from the vetting was addressed, and why
- note of any key issues in the developer documentation, in order to avoid repetition in the future
- certified WCAG accessible

Manifest V3 Haketilo port

Although highly controversial, the Manifest V3 extension format seems unavoidable.

To do

- background page replaced with Service Workers
- blocking webRequest operations replaced with declarativeNetRequest
- Haketilo working under a Chromium-based browser as a Manifest V3 extension

Easier content management and editing within Haketilo (I)

Easy configuring and editing of site resource bundles is Haketilo's raison d'être. To definitively meet this expectation, any shortcomings must be identified and rethought.

To do

- testing with untrained users/consultation with "UX experts"
- identified annoying quirks/problems
- comparison with UIs of similar extensions
- designed alternatives to identified problems
- user interface mock
- a compiled plan for UI changes

Easier content management and editing within Haketilo (II)

The previously compiled plan and carefully-prepared user interface mocks will direct the implementation efforts.

To do

- new Haketilo settings page interface implementation following the plan
- new Haketilo popup page implementation following the plan
- automated Haketilo GUI tests

Haketilo build system runnable from the browser

For portability of Haketilo's POSIX shell-based build system we avoided depending on Node.js, NPM and similar tools. However, an even more portable alternative exists - to contain the build system inside a standalone HTML page.

To do

- JavaScript-based build system in an HTML page ([#47](#))
- facility to run the JavaScript-based build system from the command line

Further development of Hydrilla platform

Users should be able to share not only custom site resources but also their opinions about them.

To do

- support for user comments
- support for user ratings
- support for flagging site resources that are broken or have other issues
- development of comment quality control systems and policies

150 sample site resources for Haketilo

To maintain community growth and participation, Hydrilla's collection must be visibly alive and evolve with Haketilo's feature set.

To do

- at least 20 alternative site interfaces
- at least 20 existing webapps packaged as meta-sites
- at least 150 custom site resources in total

200 sample site resources for Haketilo

To maintain community growth and participation, Hydrilla's collection must be visibly alive and evolve with Haketilo's feature set.

To do

- at least 20 accessibility-improving site changes
- at least 10 meta-sites aggregating content from various client websites
- at least 200 custom site resources in total

Automated building of Haketilo source packages uploaded to Hydrilla

Requiring packagers to upload compiled code places an extra burden on them, and complicates reproducibility. Hydrilla should be able to build from source packages.

To do

- Hydrilla automated resource builds feature
- security consultation of the feature

Displaying Hypothesis annotations for given site

Haketilo makes site resources for websites you visit available in only a few clicks. It would be useful to have the same capacity for comments. The established, libre <https://hypothes.is/> provides a framework for this.

To do

- support for displaying current site's Hypothesis annotations in the popup
- support for adding adding Hypothesis annotations in Haketilo

Use of a standalone JavaScript engine to perform unit tests in Haketilo

A Selenium-driven web browser is currently used to test parts of Haketilo. Those tests that don't rely on browser APIs could as well be run outside of browser which would save time during tests.

To do

- selected the JavaScript engine to use for testing
- facilitated writing Haketilo tests against the chosen engine
- applicable existing tests modified to be run without a web browser

Supplemental anti-bot measures in Hydrilla

Limiting the number of allowed registrations and content uploads is our planned basic way to prevent Hydrilla instances from being harmed by automated requests. Another measures can be added to further improve platform's resilience.

To do

- email-verified registrations
- selected an ethical, privacy-friendly captcha solution
- implementation of the chosen captcha solution

Support for external user authentication mechanisms in Hydrilla

It should be possible to run Hydrilla as part of a bigger web service. Users should be able to use the same set of credentials for logging in in various parts of such service.

To do

- selected an authentication mechanism to support
- implementation of the feature

Evaluation of non-WebExtension platforms for the purpose of porting Haketilo

WebExtensions are really a convenient platform for developing software that empowers users. But this platform is also tightly controlled by big organizations and has some serious limitations and shortcomings.

To do

- evaluation of existing Webkit-based browsers
- evaluation of XUL extensions platform still used in some Firefox forks
- prepared evaluation report

Development of the first non-WebExtension Haketilo port

Users suffer a vendor lock-in with few mainstream web browsers. Lack of their favorite extensions is what stops them from switching to more user-controlled alternatives. Haketilo should not contribute to that problem.

To do

- selection of a target platform based on previous evaluation
- specification of tasks
- development roadmap
- prototype
- automated tests
- developer documentation
- user documentation

1. [APT repository](#) and debian package git branches ([Hydrilla](#) and [Hydrilla builder](#)) ↔

2. [commit 7206db45f277c10c34d1b7ed9bd35343ac742d30](#) ↔

3. [this very document](#) ↩
4. [Haketilo Software Bill of Materials](#) ↩
5. [Hydrilla Software Bill of Materials](#) ↩
6. [commit 4c6a2323d90e9321ec2b78e226167b3013ea69ab](#) ↩
7. [Hydrilla](#) and [Hydrilla builder](#) repositories ↩
8. [JSON schemas](#) repository ↩
9. [commit 57ce414ca81682a71288018a4d9001604002ec23](#) ↩
10. [commit ea6afb92048c835752fe1c72ad52f424e2df88a8](#) ↩
11. [User manual](#) ↩
12. [commit 1cb6aaae2055283d04aa0aa581e82addb8049ce4](#) and [commit 363cbbb6a9fac49a377d8fa13ffede1483feabd5](#) ↩
13. [Haketilo release v2.0-beta1](#) ↩
14. [Hacktcha release 2022.6.21](#) ↩
15. [Hacktcha demo script](#) ↩
16. <https://libreplanet.org/2022/speakers/#5790> ↩

Technical documentation

- [URL patterns](#)
- [Repository API](#)
- [Haketilo Software Bill of Materials](#)

Building the browser extension

There're currently 2 ways to build Haketilo.

1. Simple stupid way - build.sh script

You only need a POSIX-compliant environment for this (shell, awk, etc.). It is a viable option if you don't need to run the automated test suite. From project's root directory, using a POSIX shell, you type either:

```
./build.sh mozilla # to build for Firefox-based browsers
```

or:

```
./build.sh chromium # to build for Chromium-based browsers
```

The unpacked extension shall be generated under `./mozilla-unpacked/` or `./chromium-unpacked/`, respectively. You can then load it into your browser as a temporary extension or pack it into an `xpi/crx/zip` archive manually, e.g.:

```
7z a -tzip haketilo.xpi -w mozilla-unpacked/.
```

2. configure-based build

This method assumes you have not only a POSIX environment but also a working Make tool and the zip command. From project's root directory, run the shell commands:

```
./configure --host=mozilla # or analogically with --host=chromium  
make
```

This would generate the unpacked extension under `./mozilla-unpacked/` and its zipped version under `./mozilla_build.zip` (which you can rename to `.xpi` if you want).

You can also perform an out-of-source build, for example:

```
mkdir /tmp/haketilo-build && cd /tmp/haketilo-build  
/path/to/haketilo/sources/configure --host=chromium  
make all # will generate both ./mozilla-build.zip and ./chromium-build.zip
```

Code modularity

Note: this page needs to be updated to reflect changes made in Haketilo 1.0-beta1

Complex functionality has to be split into multiple JavaScript files. However, managing these by hand can get very cumbersome as project grows. There exist various ways of bundling JavaScript. Some tools commonly used for this purpose are [Browserify](#), [jspm](#), [Webpack](#), [RequireJS](#) and [Babel](#). Yet another option is to use [JavaScript modules](#) - a native feature of the language.

Native modules considerations

JavaScript modules are rather convenient to use. Unfortunately, support varies between browsers and in some cases there is simply no facility to load a module. An example of such case are content scripts. They are loaded as listed from manifest.json file and need to be traditional script files. Chromium already offers a way to load a module dynamically from within a traditional script, but all modules loaded this way (including those pulled by the initial one) need to be listed as web-accessible resources in the manifest.

Bundler considerations

If we decided to use a bundler, it would create an additional dependency for the project. Haketilo clearly has no need for many of the features bundlers provide and as long as the functionality needed from them can be provided by a simple script, this is preferred.

It is also worth mentioning some of the bundlers are part of the Node.js/NPM ecosystem which, although consisting mostly of libre software, is problematic from software freedom point of view due to requiring people to rely on a third-party package manager and a non FSDG-compliant software repository. If any bundler is to be used, it is preferred to choose one that doesn't rely on Node.js or at least is packaged for an FSDG-compliant GNU/Linux distribution.

Current approach

At some point early on, this extension utilized native JavaScript modules. Due to problems with these, sources were slightly modified, so that the import/export semantics of modules were mostly retained, but import and export statements were replaced with assignments and accesses to keys of the window object. Each script file's contents are wrapped into a function call so as not to clobber the global namespace. Properly ordered lists of scripts to load in each context were compiled by hand and placed in manifest.json and relevant HTML files.

The build.sh script is now used to automatically write out such assignments and accesses to keys of the window.killtheweb object, wrap scripts, and compile those lists. Imports are performed using comments of the form

```
/*
 * IMPORTS_START
 * IMPORT first_import
 * ...
 * IMPORT last_import
 * IMPORTS_END
 */
```

placed before the beginning of the code, while exports use a similar structure after the code with EXPORT replacing IMPORT throughout.

Code structure

Note: this page needs to be updated to reflect changes made in Haketilo 1.0-beta1

The idea of arranging script files based on their execution context might not be the best one. Suggestions are welcome.

background/

Contains scripts that are to be run exclusively in the context of the background page.

build.sh

Builds the project, creating a build_mozilla or build_chromium folder in the working directory containing the unpacked extension. Run as /path/to/build.sh mozilla|chromium.

common/

Contains scripts that are to be run in more than one context. E.g. script that gets evaluated in contexts of both settings page and background page is going to reside in this directory.

content/

Contains scripts that execute exclusively in the context of content scripts.

copyright

Contains copyright information in [Debian's machine-readable format](#) (not all license texts are included; these reside under licenses/).

default_settings.json

The default settings for the extension, in the format that settings are exported from the extension. This will be removed at some point, as per [#48](#).

html/

Contains JavaScript, HTML and CSS files for settings and popup pages.

icons/

Contains icons graphics used by the extension.

licenses/

Contains full legal texts of licenses used.

manifest.json

Is the manifest file with the most important extension information for the browser. For now this is a version 2 manifest (version 3 is to require some more changes in the extension). This file is not a complete manifest - that one gets generated from manifest.json when build.sh script gets run.

README.txt

Contains some general information about the extension.

Hydrilla&Haketilo Software Bill of Materials

[Software Bill of Materials \(SBOM\)](#) lists external components used or included in a given software product.

This is the SBOM of Hydrilla and Haketilo proxy versions 3.x. For the SBOM of Haketilo browser extension (currently in maintenance mode), see [here](#).

You may also want to download this SBOM as a [text file](#).

Incorporated code

Currently none

External dependencies

Python3

field	value
name	cpython
version	>=3.7 (>=3.9 for Haketilo proxy)
copyright	2001-now Python Software Foundation; 2000 BeOpen.com; 1995-2001 Corporation for National Research Initiatives; 1991-1995 Stichting Mathematisch Centrum
license	PSF 2; BeOpen.com License Agreement for Python 2.0; CNRI License Agreement for Python 1.6.1; CWI License Agreement for Python 0.9.0 through 1.2
upstream url	https://www.python.org
dependency type	runtime

Pytest (Python library/application)

field	value
name	pytest
version	no known constraints
copyright	2004-now Holger Krekel and others
license	MIT (Expat)
upstream url	https://pytest.org
dependency type	development/build-time

Setuptools (Python library)

field	value
name	setuptools
version	>=44
copyright	Jason R. Coombs
license	MIT (Expat)
upstream url	https://setuptools.pypa.io/en/latest/
dependency type	development/build-time

setuptools_scm (Python library)

field	value
name	setuptools_scm
version	>=5.0
copyright	Ronny Pfannschmidt < opensource@ronnypfannschmidt.de > and contributors
license	MIT (Expat)

field	value
upstream url	https://github.com/pypa/setuptools_scm
dependency type	development/build-time

wheel (Python library)

field	value
name	wheel
version	no known constraints
copyright	2012-now Daniel Holth, Alex Grönholm and contributors
license	MIT (Expat)
upstream_url	https://github.com/pypa/wheel
dependency type	development/build-time

Babel (Python library)

field	value
name	babel
version	no known constraints
copyright	2013-2019 the Babel Team
license	BSD-3-Clause
upstream url	http://babel.pocoo.org/
dependency type	development/build-time

jsonschema (Python library)

field	value
name	jsonschema
version	>=3.0
copyright	2011-now Julian Berman and contributors
license	MIT (Expat)
upstream url	https://github.com/Julian/jsonschema
dependency type	runtime

Flask (Python library)

field	value
name	flask
version	>=1.1
copyright	2010-now Pallets
license	BSD-3-Clause
upstream url	https://flask.palletsprojects.com
dependency type	runtime

Click (Python library)

field	value
name	click
version	no known constraints
copyright	2014-now Pallets
license	BSD-3-Clause

field	value
upstream url	https://click.palletsprojects.com
dependency type	runtime

ItsDangerous (Python library)

field	value
name	itsdangerous
version	no known constraints
copyright	2011-now Pallets
license	BSD-3-Clause
upstream url	https://itsdangerous.palletsprojects.com
dependency type	runtime

immutables (Python library)

field	value
name	immutables
version	>=0.16
copyright	2018-now Contributors to the immutables project.
license	Apache-2.0; MIT (Expat)
upstream url	https://github.com/MagicStack/immutables
dependency type	runtime

gnupg (Python library)

field	value
name	gnupg
version	no known constraints
copyright	2013-now Isis Lovecruft isis@leap.se ; 2013 Andrej B.; 2013 LEAP Encryption Access Project; 2008-2012 Vinay Sajip; 2005 Steve Traugott; 2004 A.M. Kuchling; contributors
license	GPL-3.0-or-later
upstream url	https://github.com/isislovecruft/python-gnupg
dependency type	runtime

reuse (Python library/application)

field	value
name	reuse
version	no known constraints
copyright	Carmen Bianca Bakker carmenbianca@fsfe.org ; contributors
license	Apache-2.0; CC-BY-SA-4.0; CC0-1.0; GPL-3.0-or-later
upstream url	https://reuse.software/
dependency type	runtime

Beautiful Soup (Python library)

field	value
name	beautifulsoup4
version	no known constraints
copyright	2004-now Leonard Richardson

field	value
license	MIT (Expat)
upstream url	https://www.crummy.com/software/BeautifulSoup/
dependency type	runtime

html5lib (Python library)

field	value
name	html5lib
version	no known constraints
copyright	2006-now James Graham and other contributors
license	MIT (Expat)
upstream url	https://github.com/html5lib/html5lib-python
dependency type	runtime

mitmproxy (Python library/application)

field	value
name	mitmproxy
version	>=8.0; <9.0
copyright	2013 Aldo Cortesi; contributors
license	MIT (Expat)
upstream url	mitmproxy.org/
dependency type	runtime

Optional dependencies

GNU Guix

field	value
name	guix
version	no known constraints
copyright	Ludovic Courtès; Guix contributors; Eelco Dolstra; Nix contributors
license	GPL-3.0-or-later
upstream url	https://guix.gnu.org/

Guix is used to produce the standalone, relocatable release tarball of Haketilo&Hydrilla.

Software Bill of Materials - Haketilo (browser extension)

[Software Bill of Materials \(SBoM\)](#) lists external components used or included in a given software product.

This SBoM corresponds to Haketilo browser extension versions 2.x. The browser extension is currently in maintenance mode. You might instead want to look at the [SBoM of Hydrilla and Haketilo proxy](#).

Incorporated code

Software parts that have been copied over to Haketilo source tree with only slight or no modification (code that has been mostly rewritten and non-software artworks are not mentioned here).

js-sha256

field	value
name	js-sha256
version	0.9.0
copyright	2014-2017 Chen, Yi-Cyuan < emn178@gmail.com >
license	MIT (Expat)
upstream url	https://github.com/emn178/js-sha256

This JavaScript implementation of SHA256 is included in the browser extension itself. It is used to derive nonces used internally by Haketilo. This library is only used in contexts where synchronous computation of SHA256 digest is required. In other cases (e.g. verification of integrity of downloaded files), the asynchronous `crypto.subtle` JavaScript API is used.

Reset CSS

field	value
name	Reset CSS
version	2.0
copyright	2008,2011 Eric A. Meyer
license	public domain
upstream url	https://meyerweb.com/eric/tools/css/reset/

The CSS Reset style sheet is used on pages displayed by Haketilo.

jsonschema (JavaScript library)

field	value
name	jsonschema
version	1.4.0
copyright	2012-2021 Tom de Grunt < tom@degrunt.nl > and contributors
license	MIT (Expat)
upstream url	https://github.com/tdegrunt/jsonschema

This library shall be used to validate external JSON documents (e.g. those downloaded from Hydrilla repository).

External dependencies

POSIX environment

Standard UNIX tools (sh, awk, etc.) are needed to **build** Haketilo. There's no known dependency on specific implementations of those (e.g. gawk should work just as well as nawk).

Make

Make build system is an **optional requirement for building** Haketilo and a **strict requirement for running the test suite**. There's no known dependency on specific Make implementation.

Python3

Python in at least version 3.7 is needed to run the automated test suite.

Pytest

field	value
name	pytest
version	no known constraints (6.0.2 used in development)
copyright	2004-2021 Holger Krekel and others
license	MIT (Expat)
upstream url	https://pytest.org

Pytest library is used in automated tests of the extension.

Selenium webdriver (Python)

field	value
name	selenium
version	no known constraints (3.141.0 used in development)
copyright	2011-2021 Software Freedom Conservancy; 2004-2011 Selenium committers
license	Apache-2.0
upstream url	https://www.selenium.dev/

Selenium Python library is used in automated tests of the extension.

Web browser

A Firefox-derived web browser with at least version 60 **or** a Chromium-derived browser with at least version 90 (although older Chromium versions are likely to work as well) is needed to use the extensions.

In addition, a Firefox-derived web browser with at least version 60 is needed to run the automated test suite.

geckodriver

field	value
name	geckodriver
version	no known constraints (0.30.0 used in development)
copyright	???
license	MPL-2.0
upstream url	https://firefox-source-docs.mozilla.org/testing/geckodriver/index.html

Geckodriver compatible with the Firefox-derived browser used is needed to run the automated test suite.

Inkscape

field	value
name	Inkscape
version	no known constraints (0.92.4 used in development)
copyright	Inkscape Authors
license	GPL-3.0-only
upstream url	https://inkscape.org/

Inkscape is an optional build dependency used to generate png icons from an svg file.

Site whitelisting

Note: this page needs to be updated to reflect changes made in Haketilo 1.0-beta1

Blocking of scripts is tricky. LibreJS rewrites page's HTML as it is downloaded. For this it uses `webRequest` along with some other, Mozilla-specific API. That only works for HTTP(s) documents and is generally not an option for us, as we wanted to support Chromium too.

The proper way of blocking scripts, even for non-HTTP documents (like those opened from `file://`) is with a Content Security Policy (CSP). From WebExtension's content script it is possible to add an appropriate `<meta>` tag under `<head>` to cause all scripts but those later injected by us to be blocked through CSP.

It is worth noting that for HTTP(s) documents we're additionally using the `webRequest` API to inject an HTTP response header containing the rule. The problems described below are therefore only relevant for other protocols like `file://`.

Whitelisting of sites turned out to be problematic in this case. Extension's content script injected to pages needs to decide immediately whether to block their scripts or not. Removing a CSP `<meta>` tag after it is added does not cause its CSP rule to be lifted. Re-injecting site's own scripts is also likely to fail under some browser/site configurations. The possibility of site relying on synchronous scripts using the `document.write()` API function makes most hacks we could think of unreliable. Hence, we decided a whitelisted page should not have its scripts blocked in the first place. It seems easy but the content script that injects the CSP rule has to know whether the page it is running on is whitelisted or not. It could get that information from local storage or from background scripts but that would happen asynchronously and it would then be too late to inject a CSP rule if the page is not to be whitelisted.

We came up with 2 solutions. The first was to redirect any whitelisted URL to itself with a "#something" added to it. That "something" (not literal "something", btw) is a value that indicates the page should be whitelisted. Content script then synchronously checks the URL for presence of that whitelist indicator and based on that either injects the CSP rule tag or not. Content script is then also responsible for quickly removing the "#something" from the URL bar so as not to annoy the user.

The approach described above is now used for `file://` and `ftp://` (in those few browsers that still support it) protocol pages. The redirection is performed by content script once it learns it applied a script-blocking rule where it is not wanted. For HTTP urls such redirection was at some point performed using `WebRequest` but it lead to injected string not being removed from browser's URL bar in cases where page failed to load and Haketilo's content scripts were not executed.

A second approach, now used for HTTP(s) pages, is smuggling a similar value in a cookie. This is performed by injecting an HTTP response `Set-Cookie` header using `webRequest` API. The cookie, itself having a short TTL value set (30 seconds) is immediately deleted by content script and an additional `webRequest` listener also checks outgoing HTTP requests for cookies that might have slipped through in order to stop them from leaking to sites' servers.

Now, the challenge was to make the smuggled "something" unpredictable for website owners to avoid tricking the extension into whitelisting a page. The solution was to derive "something" from a value that's unique to a given browser instance and that can be retrieved in a content script synchronously. In Mozilla browsers, each extension is assigned a unique per-session id that happens to be part of `getURL()`'s return value. This was sufficient and we used it. In Chromium browsers there is no per-session id (extension's main id is used everywhere instead), so we resorted to using the "key" value from extension's manifest (we're also considering using a synchronous AJAX call to fetch a file bundled with the extension), assuming it will be different in every browser instance (which would be possible to automate when for example installing the extension from a GNU/Linux distro's repo).

Additionally, for HTTP(s) pages, `webRequest` API is used to inject CSP headers as a supplemental script-blocking method and a safety measure.

Couldn't we look into other extensions (e.g. NoScript) to see how they solve this problem? We could and we did, but NoScript's solution of "freezing" page's elements on non-HTTP(s) pages and "unfreezing" them afterwards did not present satisfactory reliability.

In the upcoming Manifest v3 port there is another way to solve this - we can dynamically inject content scripts at runtime. We could simply set up a new content script for every whitelisted URL pattern and that script would immediately know it has to allow the scripts there.

URL patterns

We want to be able to apply different rules and custom scripts for different websites. However, merely specifying "do this for https://example.com" is not enough. Single site's pages might differ strongly and require different custom scripts to be loaded. However, always matching against a full URL like https://example.com/something/somethingelse doesn't allow us to properly handle a site that serves similar pages for multiple values substituted for somethingelse.

Currently employed solution

Wildcards are being used to address the problem. Each payload and rule in Haketilo has a URL pattern that specifies to which internet pages it applies. A URL pattern can be as simple as literal URL in which case it only matches itself. It can also contain wildcards in the form of one or more asterisks (*) that correspond to multiple possible strings occurring in that place.

Wildcards can appear in URL's domain and path that follows it. These 2 types of wildcards are handled separately.

Domain wildcards

A domain wildcard takes the form of one, two or three asterisks occurring in place of a single domain name segment at the beginning (left). Depending on the number of asterisks, the meaning is as follows:

- no asterisks (e.g. example.com) - match domain name exactly (e.g. example.com)
- one asterisk (e.g. *.example.com) - match all domains resulting from substituting * with a **single** segment (e.g. banana.example.com or pineapple.example.com but **not** pineapple.pen.example.com nor example.com)
- two asterisks (e.g. **.example.com) - match all domains resulting from substituting ** with **two or more** segments (e.g. monad.breakfast.example.com or pure.monad.breakfast.example.com but **not** cabahell.example.com nor example.com)
- three asterisks (e.g. ***.example.com) - match all domains resulting from substituting *** with **zero or more** segments (e.g. hello.parkmeter.example.com or ilikettrains.example.com or example.com)

Path wildcards

A path wildcard takes the form of one, two or three asterisks occurring in place of a single path segment at the end of path (right). Depending on the number of asterisks, the meaning is as follows:

- no asterisks (e.g. /joke/clowns) - match path exactly (e.g. /joke/clowns)
- one asterisk (e.g. /itscalled/*) - match all paths resulting from substituting * with a **single** segment (e.g. /itscalled/gnulinux or /itscalled/glamp but **not** /itscalled/ nor /itscalled/gnu/linux)
- two asterisks (e.g. /another/**) - match all paths resulting from substituting ** with **two or more** segments (e.g. /another/nsa/backdoor or /another/best/programming/language but **not** /another/apibreak nor /another)
- three asterisks (e.g. /mail/dmarc/***) - match all paths resulting from substituting *** with **zero or more** segments (e.g. /mail/dmarc/spf, /mail/dmarc or /mail/dmarc/dkim/failure but **not** /mail/)

If pattern ends **without** a trailing slash, it matches paths with any number of trailing slashes, including zero. If pattern ends **with** a trailing slash, it only matches paths with one or more trailing slashes. For example, /itscalled/* matches /itscalled/gnulinux, /itscalled/gnulinux/ and /itscalled/gnulinux// while /itscalled/*/ only matches /itscalled/gnulinux/ and /itscalled/gnulinux// out of those three.

If two patterns only differ by the presence of a trailing slash, pattern **with** a trailing slash is considered **more specific**.

Additionally, any path with literal trailing asterisks is matched by itself, even if such pattern would otherwise be treated as wildcard (e.g. /gobacktoxul/** matches /gobacktoxul/**). This is likely to change in the future and would best not be relied upon. Appending three additional asterisks to path pattern to represent literal asterisks is being considered.

Ideas for future additions

Right now a URL's query string is being completely disregarded for the purpose of matching the URL patterns. In the future, support for matching URLs with specific query parameters and maybe even HTTP(s) requests with specific POST parameters or specific cookies might be added.

Currently, protocols in the URL are matched exactly. However, as of Haketilo 3.0 support for protocol wildcard of http*:// is being developed which will match both "http://" and "https://".

The wildcards that have been added so far were designed to allow a reasonable level of flexibility, considering some common ways websites are served. However, only practice can show what works best, and so wildcard semantics are subject to change as the project matures.

Wildcard priorities and querying

In case multiple patterns match some URL, the more specific one is preferred. Specificity is considered as follows:

- If patterns only differ in the final path segment, the one with least wildcard asterisks in that segment is preferred.
- If patterns, besides the above, only differ in path length, one with longer path is preferred. Neither final wildcard segment nor trailing dashes account for path length.
- If patterns, besides the above, only differ in the initial domain segment, one with least wildcard asterisks in that segment is preferred.
- If patterns differ in domain length, one with longer domain is preferred. Initial wildcard segment does not account for domain length.

As an example, consider the URL http://settings.query.example.com/google/tries/destroy/adblockers//. Patterns matching it are, in the following order:


```
http://***.query.example.com/google/***
http://***.query.example.com/**/
http://***.query.example.com/**
http://***.query.example.com/****/
http://***.query.example.com/****
http://***.example.com/google/tries/destroy/adblockers/
http://***.example.com/google/tries/destroy/adblockers
http://***.example.com/google/tries/destroy/adblockers/****/
http://***.example.com/google/tries/destroy/adblockers/****
http://***.example.com/google/tries/destroy/*/
http://***.example.com/google/tries/destroy/**
http://***.example.com/google/tries/destroy/****/
http://***.example.com/google/tries/destroy/****
http://***.example.com/google/tries/**/
http://***.example.com/google/tries/**
http://***.example.com/google/tries/****/
http://***.example.com/google/tries/****
http://***.example.com/google/**/
http://***.example.com/google/**
http://***.example.com/google/****/
http://***.example.com/google/****
http://***.example.com/**/
http://***.example.com/**
http://***.example.com/****/
http://***.example.com/****
http://***.example.com/google/tries/destroy/adblockers/
http://***.example.com/google/tries/destroy/adblockers
http://***.example.com/google/tries/destroy/adblockers/****/
http://***.example.com/google/tries/destroy/adblockers/****
http://***.example.com/google/tries/destroy/*/
http://***.example.com/google/tries/destroy/**
http://***.example.com/google/tries/destroy/****/
http://***.example.com/google/tries/destroy/****
http://***.example.com/google/tries/**/
http://***.example.com/google/tries/**
http://***.example.com/google/tries/****/
http://***.example.com/google/tries/****
http://***.example.com/google/**/
http://***.example.com/google/**
http://***.example.com/google/****/
http://***.example.com/google/****
http://***.example.com/**/
http://***.example.com/**
http://***.example.com/****/
http://***.example.com/****
```

For a simpler URL like `https://example.com` the patterns would be:

```
https://example.com
https://example.com/****
https://***.example.com
https://***.example.com/****
```

Variants of those patterns with a trailing dash added would **not** match the URL.

Limits

In order to prevent some easy-to-conduct DoS attacks, older versions of Haketilo and Hydrilla limited the lengths of domain and path parts of processed URLs. This is no longer the case.

Alternative solution: mimicking web server mechanics

While wildcard patterns as presented give a lot of flexibility, they are not the only viable approach to specifying what URLs given settings of custom scripts should be applied to. In fact, wildcards are different from how the server side of a typical website decides what to return for a given URL request.

In a typical scenario, an HTTP server like Apache reads configuration files provided by its administrator and uses various virtual host, redirect, request rewrite, CGI, etc. instructions to decide how to handle given URL. It is possible using a schema that mimics the configuration options typically used

with web servers would give more efficiency in specifying what page settings to apply when.

This approach may be considered in the future.

User manual

This page documents the usage of Haketilo proxy. For a documentation of Haketilo extension's usage (in maintenance mode), see [here](#).

Installation

The recommended way of installing Haketilo is by using the "relocatable standalone binary release for x86-64 computers" from the [Releases](#) page. Installation from Python wheel is also possible.

Using binary release

You'll be using a .tar.gz archive that was prepared using [GNU Guix](#) package manager as described in Haketilo's [README.md](#) file. This binary pack can be used on top of most GNU/Linux systems, even those that don't have GNU Guix installed.

Once the .tar.gz release archive finishes downloading, create a new directory and extract the tarball there. You'll see a few executables named haketilo, hydrilla-builder, etc. They are ready to use. The release is *relocatable*, so it is also possible to move the executables to another place in the filesystem (together with the gnu/ directory that was extracted with them) and run the programs from there.

Using Python wheel

Haketilo can be installed using the pip command. The functionality of Hydrilla server, builder and Haketilo proxy is, since version 3.0-beta1, contained in a single hydrilla Python package available on [PyPI](#). While PyPI repository does not offer the same reproducibility and software freedom standards as GNU Guix, it might currently be the only installation option available to some of the users. The package is available in 4 flavors

- hydrilla[server] - besides package itself, all dependencies of Hydrilla server will be installed but not those of the proxy or builder
- hydrilla[builder] - besides package itself, only dependencies of Hydrilla builder will be installed
- hydrilla[haketilo] - besides package itself, only dependencies of Haketilo proxy will be installed
- hydrilla[all] - the package will be installed with dependencies of all its components

So, once you have Python 3 and pip installed, you can run

```
python3 -m pip install hydrilla[all]==3.0-b1
```

This should make the hydrilla, hydrilla-builder and haketilo commands available.

Some features of Hydrilla builder might additionally rely on [GnuPG](#) being available. GnuPG cannot be simply pulled from PyPI as a dependency. If you encounter problems, please make sure it is installed via some other method.

Running

You can start the proxy by running the executable named haketilo. Upon the first run, it will create a .haketilo/ directory inside your user home directory. It will store its data (user scripts, cryptographic certificates, etc.) there.

If you're familiar with the command line, you can optionally tell Haketilo to listen on a different port number than the default of 8080 (--port option) or to use a different data directory than the default of ~/.haketilo (--directory option).

Configuring the browser

Once Haketilo is running, your web browser needs to be told to connect to the internet through it. You'll want to configure it to use the proxy at address 127.0.0.1 and port 8080 for both HTTP and HTTPS connections.

Under Firefox and derived browsers, the relevant settings can be accessed by navigating to about:preferences, scrolling all the way down to "Network Settings" and clicking on the "Settings..." button. You'll want to choose the "Manual proxy configuration" option, enter address and port in the fields next to the "HTTP proxy" label and tick the "Also use this proxy for HTTPS" checkbox beneath.

If you performed all the steps correctly, you can now access the locally-served configuration page of Haketilo at <http://hkt.mitm.it>. The browser does not yet know the security certificate of our proxy, so it might present to you a warning about the HTTPS version of the page being unavailable. In *this particular case* the warning is safe to ignore.

At this point the browser does not yet allow Haketilo to modify HTTPS pages. For that, you need to visit <http://mitm.it> which is yet another page hosted locally by Haketilo. Once there, follow the instructions to install the certificate in your operating system, your browser or both.

All HTTP and most HTTPS websites should now load correctly in the browser. However, some user agents pin the certificates of certain sites. E.g. Mozilla Firefox by default pins the certificate used by <https://mozilla.org>. This security feature makes it impossible to access the site through Haketilo proxy. If you want to do that nevertheless, you might consider disabling the feature. For example, in Firefox-derived browsers this can be done by visiting the about:config page, looking up the security.cert_pinning.enforcement_level preference and setting its value to 0.

Managing the proxy

Haketilo can be configured from its locally-hosted meta-site <https://hkt.mitm.it>. Take a while to experiment a bit with the interface so that you can make yourself familiar it.

Using together with other proxy (e.g. Tor SOCKS proxy)

As of version 3.0-beta1 Haketilo does not offer direct upstream proxy support. Instead, the recommended way of making it talk to another proxy server is by means of the [proxychains-ng](#) tool. Proxychains-ng is a fork of unmaintained [Proxychains](#). It works under many UNIX-like systems and is available in the repositories of various GNU/Linux distributions, including [Debian](#), [GNU Guix](#), [Arch](#) and [Fedora](#).

Assuming you've successfully installed Proxychains-ng and Haketilo and you want to route your traffic through a SOCKSv5 proxy at port 9050 on localhost (default for Tor), write the following minimal configuration to a file of choice. Let's call the file `./my-proxychains.conf` for the purpose of this manual.

```
strict_chain
quiet_mode
proxy_dns
tcp_read_time_out 15000
tcp_connect_time_out 8000
```

```
[ProxyList]
socks5 127.0.0.1 9050
```

Now, you can start Haketilo from your terminal with the following command

```
proxychains4 -f ./my-proxychains.conf haketilo
```

Haketilo's traffic should now be additionally routed through the second proxy.

Understanding the concepts

Script blocking and injection

Haketilo combines features of a user script manager and a content blocker. Out of the box, it can be used to block site's JavaScript, similarly to how NoScript (for example) does it. Once you import custom scripts into Haketilo (either from a Hydrilla repository server or by typing code in a form on the import page), it can also inject them into pages.

Script blocking and injection is configured using [URL patterns](#). Patterns have different specificity. More specific patterns will override the settings of the less specific ones. In short, for every visited page, Haketilo performs the following steps:

1. Try to find a script blocking/allowing/injection rule with a pattern matching page's URL. If found, apply the rule and don't perform the next step.
 - If multiple rules match, pick the one with the most specific pattern.
2. If no rules matched, check whether the default policy is to block scripts or allow them. Act accordingly.

In the end, Haketilo's action shall be one of the following:

- block page's own scripts
- allow page's own scripts to execute (i.e. don't do anything)
- inject the supplied user script **and** block page's own scripts

We can see that Haketilo's concepts are different from those of most user script managers. GreaseMonkey, for instance, executes user scripts alongside page's original scripts. Haketilo instead **replaces** page's scripts with the user-supplied ones.

Packages and Libraries

To make mapping custom JavaScript applications and their dependencies to web pages more manageable, we introduced our own concept of packages. Currently, Haketilo understands 2 different types of items

- library - Also referred to as *resource*. Defines a set of scripts that can be injected together into a page. It can also name other libraries as its dependencies. When injecting scripts of a given library into some page, Haketilo will first inject scripts of all libraries depended on. Installed libraries are only viewable in Haketilo UI if advanced interface features are enabled.
- package - It associates URL patterns with libraries. If pattern `https://example.com/**` is associated with library `my-sample-lib` it means the scripts from `my-sample-res` should be injected into all HTTPs pages under the `example.com` domain.

For simple cases, this may be overly complex. Because of that, Haketilo's interface contains a simple form that can be used to quickly define a script payload for a set of URL patterns.

Packages and libraries can also be installed from Hydrilla repository which serves both simple and complex (i.e. multi-library) payloads. Defining more complex packages and libraries within Haketilo itself is not yet supported.

What to do next

Please [REPORT BUGS](#). This is incredibly important. If nobody reports them, they likely won't get fixed.

You might want to learn about [current limitations](#) of Haketilo. If despite these you like the tool, please spread the word. We'll be also happy to receive some feedback or - if you're a programmer - code contributions. Consider [creating an account](#) on our issue tracker or writing to koszko@koszko.org :)

User manual (browser extension)

This page documents the usage of Haketilo WebExtension. For a documentation of Haketilo proxy's usage, see [here](#).

Installation

Instructions for different browsers have been put on their respective pages:

- [Installation instructions \(browser extension, Mozilla\)](#)
- [Installation instructions \(browser extension, Chromium\)](#)

Users who want to install Haketilo from source can also visit [Building the extension](#) page.

Understanding the concepts

Script blocking and injection

Haketilo combines features of a user script manager and a content blocker. Out of the box, it can be used to block site's JavaScript, similarly to how NoScript (for example) does it. Once you import custom scripts into Haketilo (either from a Hydrilla repository server or by typing code in a form in the settings page), it can also inject them into pages.

Script blocking and injection is configured using [URL patterns](#). Patterns have different specificity. More specific patterns will override the settings of the less specific ones. In short, for every visited page, Haketilo performs the following steps:

1. Check if the page is privileged (e.g. it is its own settings page or a directory listing of local filesystem). If it is, don't attempt to block nor inject anything on this page and don't perform the remaining steps.
2. Try to find a script blocking/allowing/injection rule with a pattern matching page's URL. If found, apply the rule and don't perform the next step.
 - If multiple rules match, pick the one with the most specific pattern.
3. Check whether the default policy is to block scripts or allow them. Act accordingly.

In the end, Haketilo's action shall be one of the following:

- block page's own scripts
- allow page's own scripts to execute (i.e. don't do anything)
- inject the supplied user script **and** block page's own scripts

We can see that Haketilo's concepts are different from those of most user script managers. GreaseMonkey, for instance, executes user scripts alongside page's original scripts. Haketilo effectively **replaces** page's scripts with the user-supplied ones.

Mappings and Resources

To make handling JavaScript libraries more manageable, we introduced our own concept of packages. Currently, Haketilo understands 2 different types of items (i.e. packages):

- Resource - It defines a set of scripts that can be injected together into a page. It can also name other resources as its dependencies. When injecting scripts of a given resource into some page, Haketilo will first inject scripts of all resources depended on.
- Mapping - It associates URL patterns with resources in a 1:1 fashion. If pattern `https://example.com/**` is associated with resource `my-sample-res` it means the scripts from `my-sample-res` should be injected into all HTTPS pages under the `example.com` domain.

For simple cases, this may be overly complex. Because of that, Haketilo's settings page contains a simple form that can be used to quickly define a script payload for a set of URL patterns.

Mappings and resources can also be installed from Hydrilla repository which serves both simple and complex (i.e. multi-resource) payloads. Defining more complex mappings and resources within Haketilo itself is not (yet) supported.

Operating the popup window

While browsing with Haketilo installed, a small Haketilo icon will be present in the extensions panel (usually located to the right of the url bar). Clicking on it will open the extension's popup.

haketilo_popup.png

At the very bottom of the popup there is a convenience shortcut - a button which, when clicked, opens Haketilo's [settings page](#). At the top is page's status report, containing information on how Haketilo has modified the currently viewed page. For the above website there is no custom policy set. As a result Haketilo simply applies the default policy of blocking scripts. In cases a policy is set for a matching [URL pattern](#), it is used to modify the page and its details can be viewed in the popup.

haketilo_popup_payload.png

Sometimes you might wish to look for custom scripts to use on sites or for ethical fixes others made to enable browsing without running nonfree JavaScript sites serve. Such search can be performed from the popup. Clicking the "Search for custom resources" button below status report will bring up the repository query view.

haketilo_popup_search_for_custom.png

Repository view contains a list of all installed repositories. Each entry has a "Show results" button which, when clicked, will make Haketilo query the repository for payloads that can be used on the viewed website. Haketilo comes with https://hydrilla.koszko.org/api_v1/ configured as the default repository to use. This can of course be changed in the settings page.

haketilo_popup_show_results.png

Currently, there are very few custom scripts listed in the main Hydrilla instance; if you are lucky enough to find a match, however, click the "More..." button on the right to install the policy.

haketilo_popup_more.png

In the view that appears you can see the components of the policy you want to install. You'll often see 2 entries with the same name. One represents a resource which defines a set of scripts. The other one represents a mapping which associates resource(s) with URL patterns. If, by chance, some components are already installed (for example because they were required by some other policy you installed earlier), they won't be listed. There is an exception - components with a newer version available. Those will be listed and proceeding with the installation will cause them to be updated.

When you're sure you want to commit the installation, just click "Install".

haketilo_popup_install.png

Voilà! The installed items (mapping and resource(s)) will be used the next time you load a matching page. They can also be viewed and managed under the appropriate tabs on the [settings page](#).

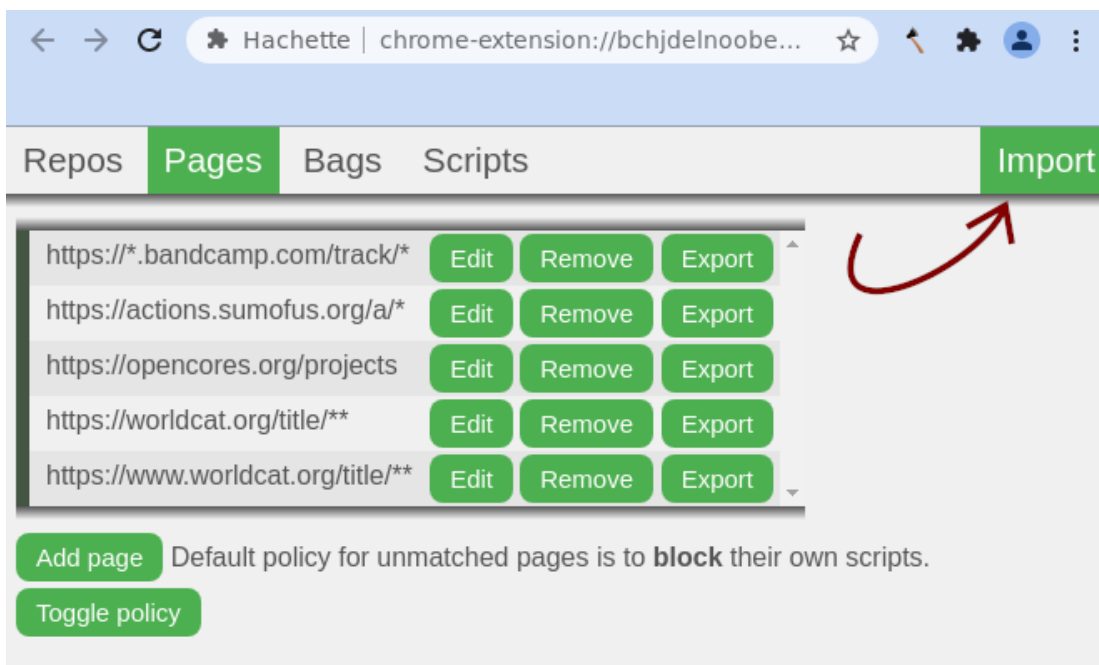
Above all else, Haketilo is designed to put you in control of your web browsing. Like Haketilo itself, all scripts distributed as defaults with the extension and published in the Hydrilla repository are free software. This means you can read, modify and distribute them without undue restrictions in either compiled form or as source code.

Manually importing custom scripts

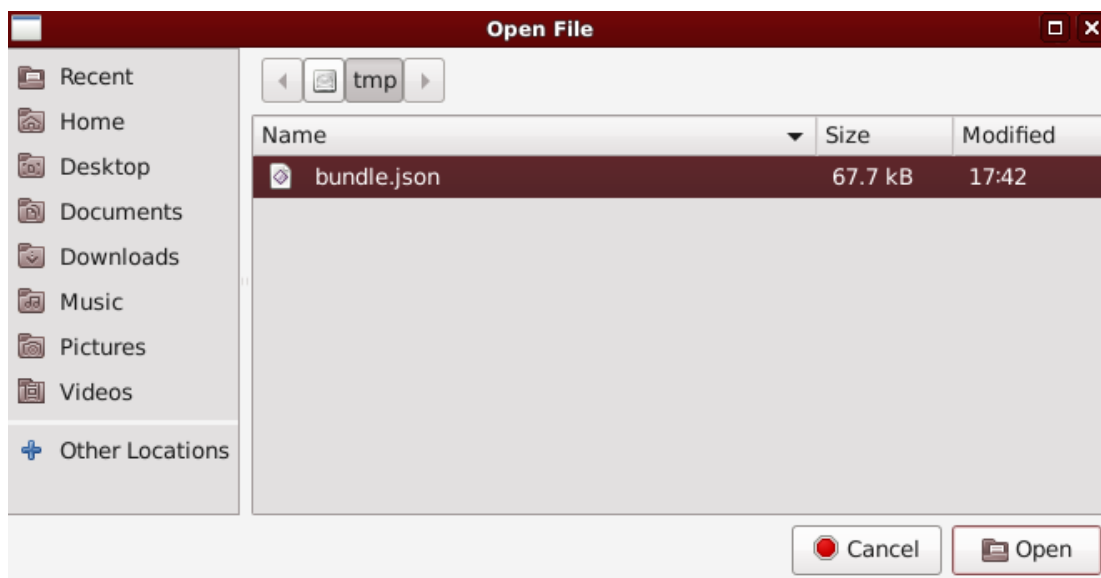
Note: this section of the manual corresponds to old Haketilo version 0.1 and is awaiting an update

Although installation of site fixes and custom content is meant to be convenient through the use of a repository, one can also export and import such payloads to and from JSON files. In fact, all the scripts currently served by the default Hydrilla repository can be downloaded in that format from <https://hachette-hydrilla.org/>. Most are fixes for js-encumbered websites, but there are also some alternative interfaces for already-functional sites. You can download and install particular scripts one by one or go crazy and just import it all at once from the "All-in-one bundle".

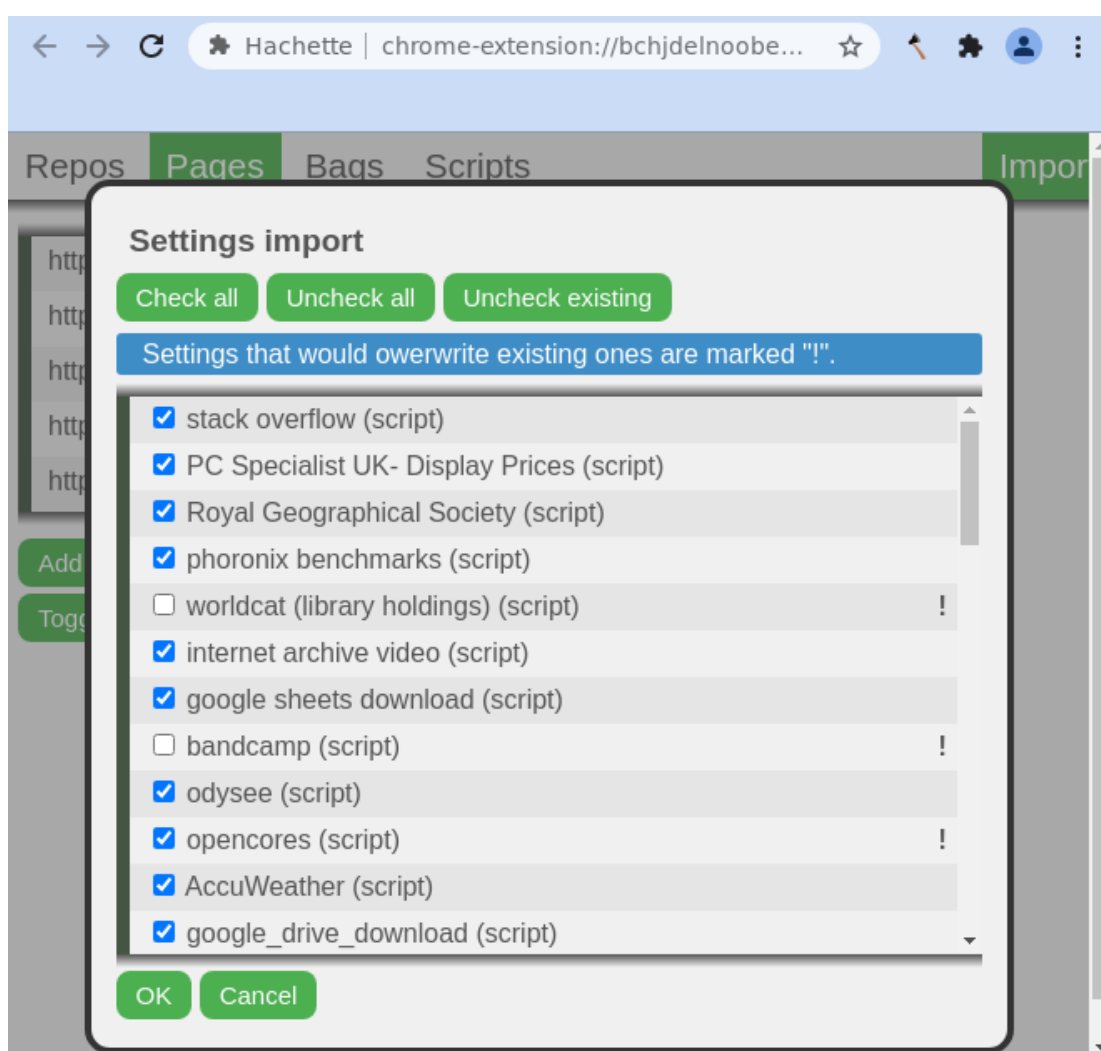
After you've downloaded the right .json file, go to Haketilo's settings page (reachable by clicking the button at the bottom of the popup window) and click "Import".



Now, find and choose the .json file with scripts.



In the frame that appears you can select which components you want to install. Note that components might behave improperly if you disable other ones they depend on. As of version 0.1, Haketilo will do nothing to stop you from messing things up ;)



If you click "Ok", scripts and settings are imported. Voilà! They can be viewed and managed under the appropriate tabs on the [settings page](#).

Using the settings

Note: this section of the manual corresponds to old Haketilo version 0.1 and is awaiting an update

As well as allowing you to [manually import scripts from JSON files](#), the settings page provides access to all configurable options available in Haketilo. These are arranged under four tabs, one for each type of item- repository, page (pattern), bag, and script.

haketilo_bags_tab.png

Each tab is arranged similarly, with a list of items followed by an "Add ..." button (which opens an empty new item for editing). Besides every item is an edit button, a remove button, and- except for on repositories- an export button, which exports the item and its dependencies to JSON format for download.

Editing an item will open up a form with appropriate fields. **Changes are not autosaved**; they must be manually committed using the "save" button at the bottom of each form.

haketilo_repo_edit.png

Repositories and scripts currently employ relatively simple forms. However, to allow ample space for editing the source code of scripts, the form overflows the page, which requires scrolling down to find the "save" button and avoid losing changes.

haketilo_script_edit.png

Bags can contain scripts and other bags, and opening one to edit it will present alongside the name a list of items contained within. To add new items, click on the "Add scripts" button towards the bottom of the form, select which new items, and click "Ok".

haketilo_bag_edit.png

haketilo_bag_popup.png

Editing a page is slightly different to other items, even if the form works the same way. Instead of a name, pages have a field for a [URL pattern](#) that they apply to and a field for a payload to inject into them. This payload can be changed by clicking on the "Choose payload" button, which will bring up a popup similar to the one for bags except that only one item is selected at a time. Injecting a payload **will block all scripts that are sent with a page natively**.

haketilo_page_edit.png

haketilo_page_popup.png

As well as a list of all available scripts and bags, at the very bottom of the payloads popup there is the option "(None)" for no payload. When selected, this enables an additional checkbox in the editing form, which controls whether or not a the scripts sent with a page natively should be blocked. Script-blocking behavior on URLs without a corresponding page policy listed can also be set, at any time, using the "Toggle policy" button at the bottom of the pages tab.

haketilo_edit_script_blocking.png

What to do next

Please [REPORT BUGS](#). This is incredibly important. If nobody reports them, they likely won't get fixed.

You might want to learn about [current limitations](#) of Haketilo. If despite these you like the extension, please spread the word. We'll be also happy to receive some feedback or - if you're a programmer - code contributions. Consider [creating an account](#) on our issue tracker or writing to koszko@koszko.org :)

Files

hachette_popup_injected_scripts.png	68.9 KB	09/10/2021	koszko
hachette_popup_install_scripts.png	67.5 KB	09/10/2021	koszko
hachette_popup_install_scripts_import_frame.png	72 KB	09/10/2021	koszko
hachette_popup_opencores.png	105 KB	09/10/2021	koszko
hachette_popup_possible_patterns.png	114 KB	09/10/2021	koszko
hachette_popup.png	54.5 KB	09/11/2021	koszko
hachette_settings_import_but.png	63.3 KB	09/11/2021	koszko
navigate_scripts_bundle.png	33.4 KB	09/11/2021	koszko
hachette_settings_import_frame.png	84.8 KB	09/11/2021	koszko
haketilo_bag_edit.png	39.2 KB	09/15/2021	jahoti
haketilo_bag_popup.png	57.8 KB	09/15/2021	jahoti
haketilo_bags_tab.png	30.4 KB	09/15/2021	jahoti
haketilo_bags_tab.png	30.4 KB	09/15/2021	jahoti
haketilo_edit_script_blocking.png	67.7 KB	09/15/2021	jahoti
haketilo_page_edit.png	70.5 KB	09/15/2021	jahoti
haketilo_page_popup.png	76.3 KB	09/15/2021	jahoti
haketilo_repo_edit.png	27.4 KB	09/15/2021	jahoti

haketilo_script_edit.png	62.8 KB	09/15/2021	jahoti
haketilo_popup_payload.png	83.4 KB	03/12/2022	koszko
haketilo_popup_search_for_custom.png	81.3 KB	03/12/2022	koszko
haketilo_popup_show_results.png	61.3 KB	03/12/2022	koszko
haketilo_popup_more.png	74.2 KB	03/12/2022	koszko
haketilo_popup_install.png	62.2 KB	03/12/2022	koszko
haketilo_popup.png	59.3 KB	03/12/2022	koszko

Installation instructions (browser extension, (Ungoogled) Chromium)

Note: This guide only applies to Chromium-based browsers that support Manifest V2 type of WebExtensions. If you are using a newer browser that only supports Manifest V3 extensions, there's unfortunately no way to install Haketilo in it.

Here we'll describe how to install Haketilo as an *Unpacked Extension*. Chromium extensions are most often installed from Chrome Web Store but it has serious privacy and freedom problems.

Although installation as Unpacked Extension is meant mainly for developers during testing, it is also simple enough for casual users.

Note: If you want, you can also follow [this](#) third-party guide to generate a .crx file yourself and install from it.

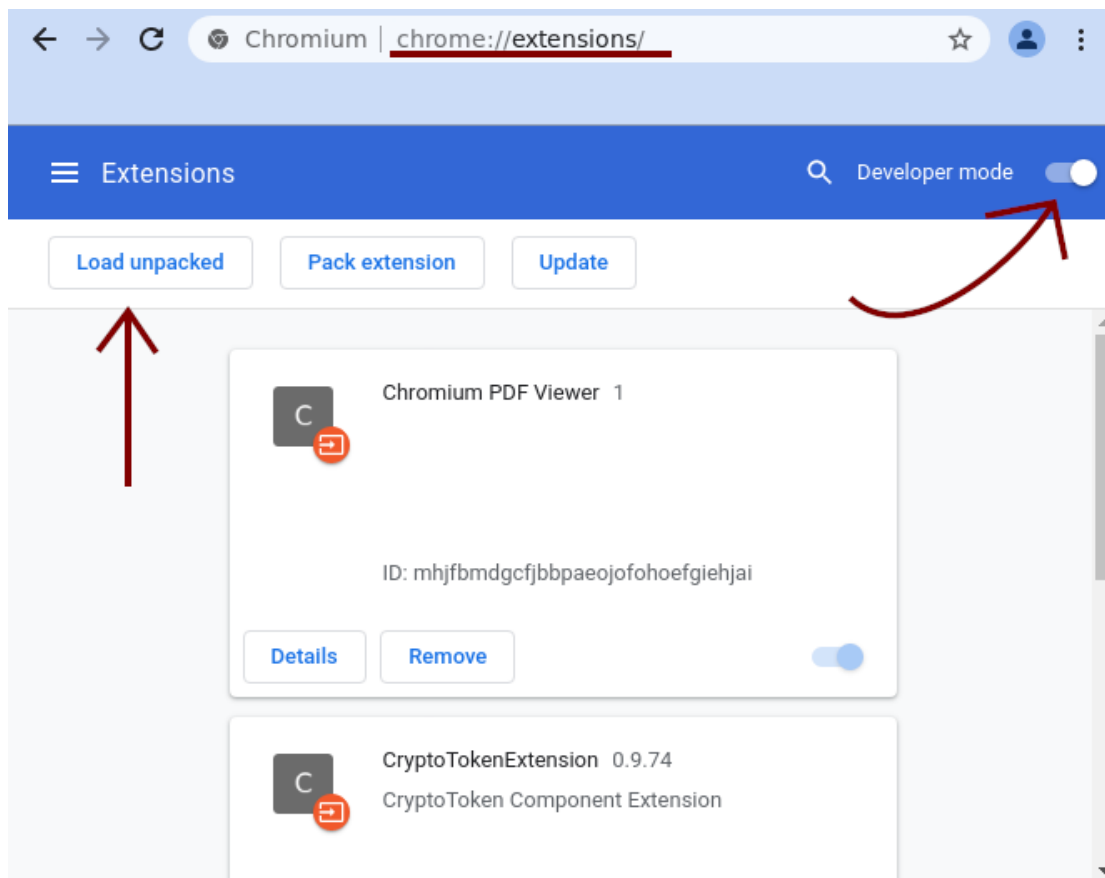
Browser compatibility

Haketilo is currently compatible with recent (versions 90+) versions of Ungoogled Chromium and derivatives. Older versions might work as well but were not tested.

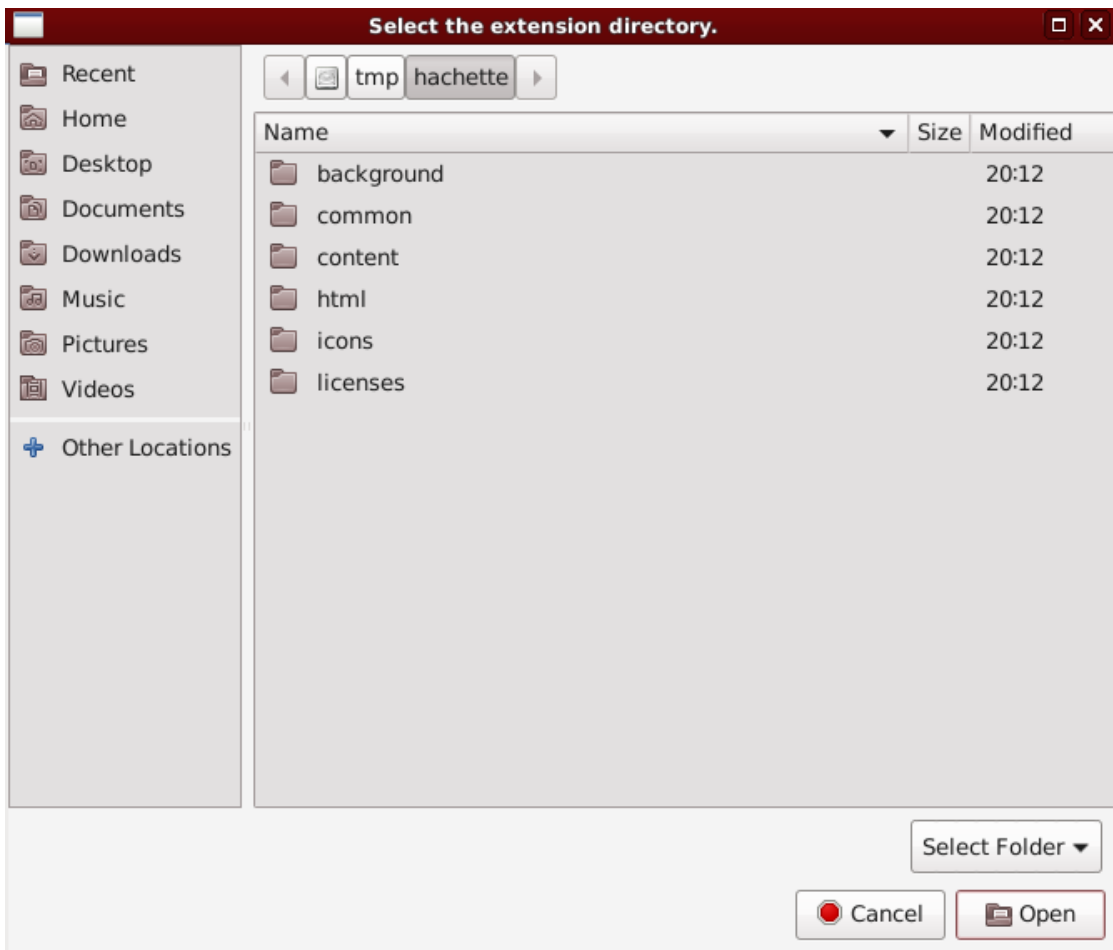
Although Haketilo should also run fine under the standard, non-Ungoogled Chromium and Google Chrome, these browsers are **malware** unrecommended due to serious privacy and freedom violations.

Steps

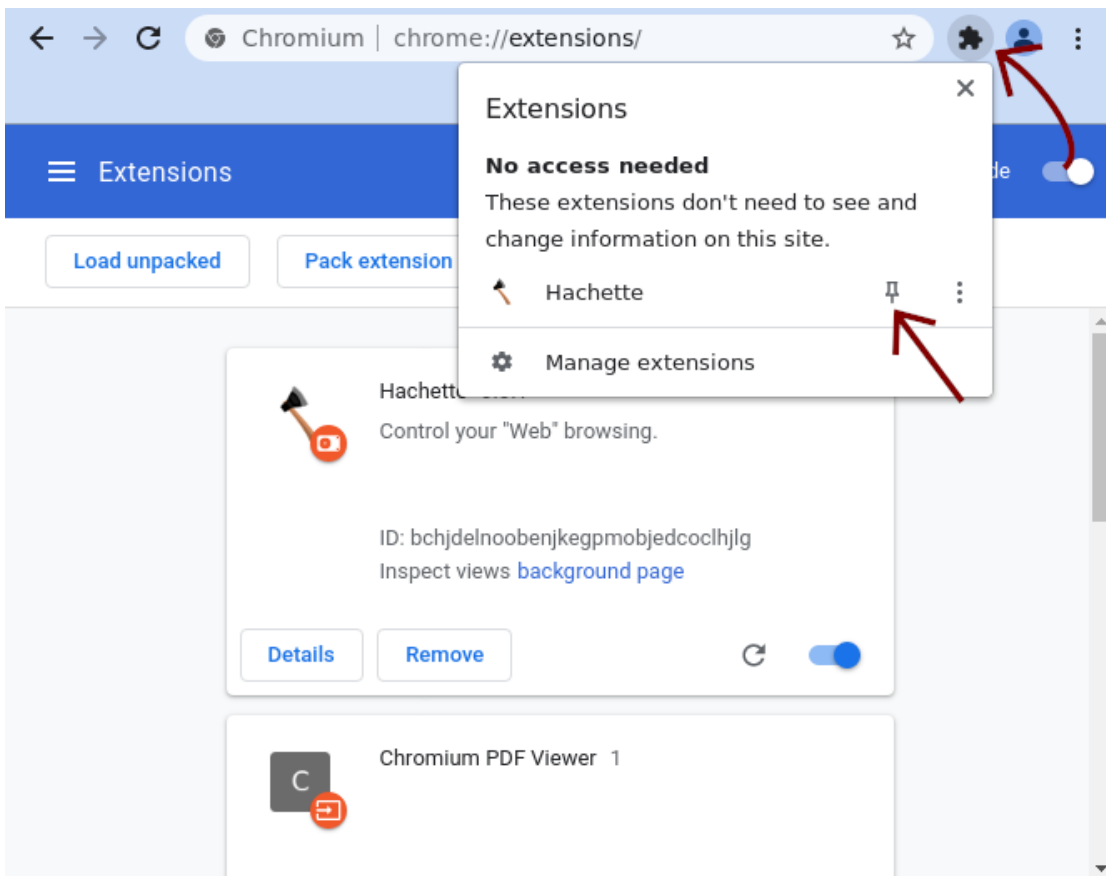
First, download the extension's .zip file from the [extension releases page](#). You are encouraged to also download the provided PGP and Signify signatures of the file and verify them and described [here](#). Once you made sure the zip file hasn't been tampered with, extract its contents to some directory. Here we use /tmp/hachette. Then, go to the chrome://extensions URL. In the extensions management page that shows up, you need to enable developer mode using toggle in the top right of the page. Once you do so, you should see a "Load unpacked" button. Click it.



Navigate to the extracted directory and choose it.



Your browser has just installed Haketilo. You can now delete the extracted directory and disable developer mode. If you want Haketilo's icon to be permanently visible in the top right corner of your browser window, click on the extensions icon and on Haketilo's pin icon as shown below.



That's it. You can now play a bit with the extension and [learn how it works](#). Also, make sure you realize its [limitations](#).

Files

chrome_extensions.png	40.7 KB	09/10/2021	koszko
chrome_hachette_installed.png	51.5 KB	09/10/2021	koszko
navigate_extension_dir.png	45.5 KB	09/10/2021	koszko

Installation instructions (browser extension, Mozilla)

Browser compatibility

Haketilo is currently compatible with browsers based on Mozilla Firefox 60 and upwards.

Although this makes little difference to Haketilo, please consider using a Firefox derivative that respects your freedom and privacy (i.e. does not snoop on you). Valid options are browsers in some of the ["ethical" GNU/Linux distros](#) like Parabola and Trisquel, the Tor Browser and LibreWolf.

Single-user install

Here we'll describe how to install Haketilo from a .xpi file served by our server. Extensions in Firefox and its derivatives are often installed from the Mozilla Add-ons website (AMO) but this approach has serious privacy and freedom problems.

The provided .xpi extension files for some of the releases (including 1.0) are cryptographically signed by Mozilla (this is nohow related to the signatures we provide on the [extension releases page](#)). Those have "mozilla-signed" in the filenames and are expected to install without issues on any Mozilla browser. The .xpi files of remaining releases were not signed and therefore some Mozilla browsers and even derivatives will refuse to install them. In order to have full control over your extensions, you are strongly encouraged to use a Firefox derivative like [LibreWolf](#) that allows you to manage your software and install unsigned add-ons as well.

Note: even if your browser refuses to install unsigned extensions, you can still 1) try it out by installing it as a "temporary" add-on on [about:debugging page](#) or 2) try installing it [globally](#).

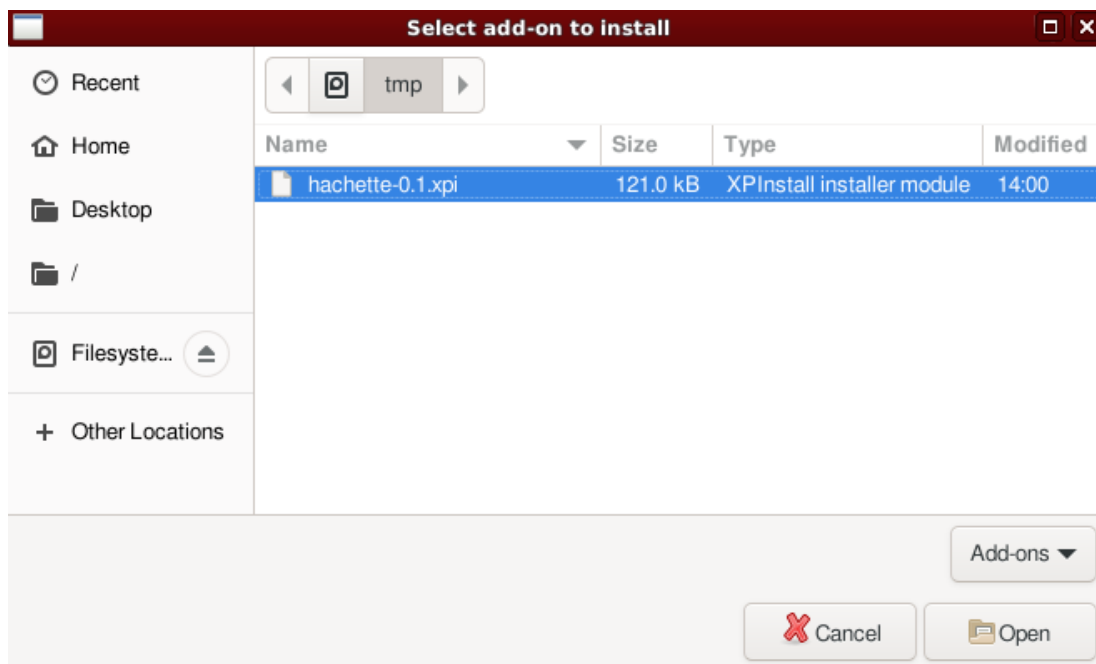
Steps

Note: depending on the versions of your web browser and Haketilo some UI elements and text messages might be slightly different on your machine than on the screenshots below

First, download the extension's .xpi file from the [extension releases page](#). You can also download the PGP and Signify cryptographic signatures made by us to verify the file hasn't been tampered with. The verification procedure is described [here](#). After downloading, go to the [about:addons](#) URL. In the extensions management page that shows up, click on the gear icon and select "Install Add-on From File...".

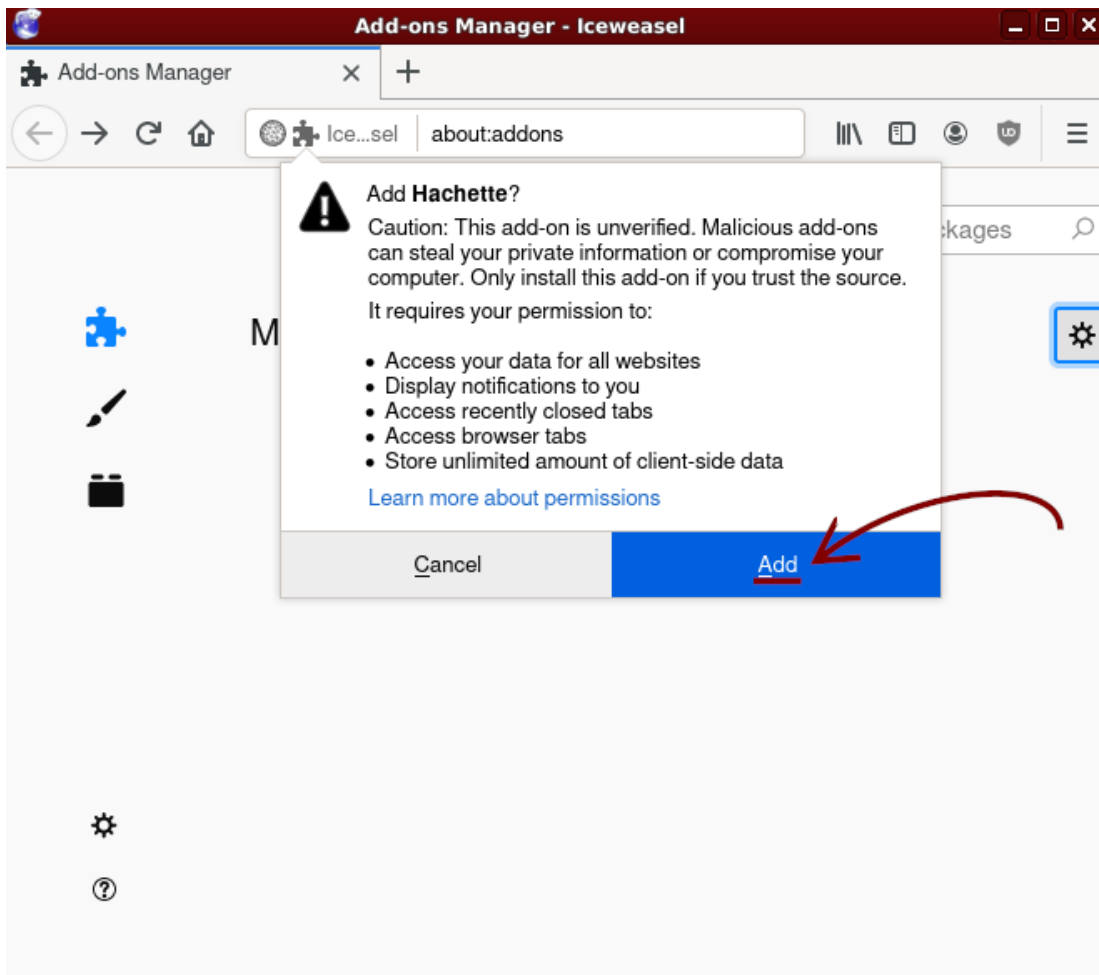
mozilla_install_from_file.png

Now, navigate to the downloaded .xpi file and select it.

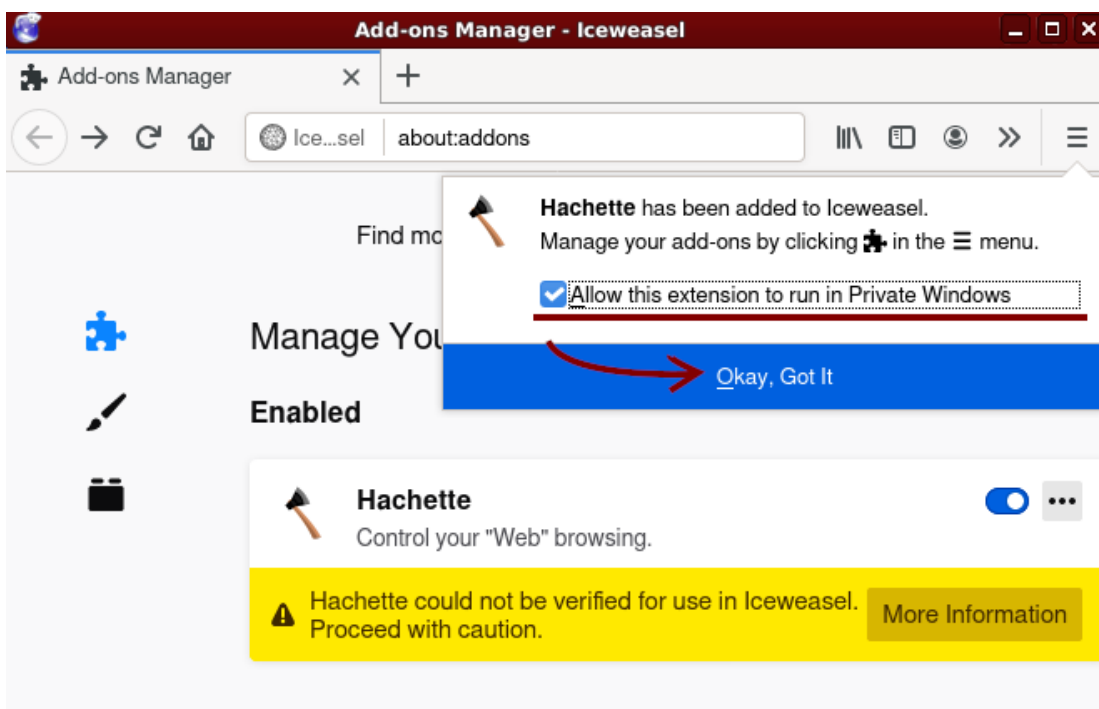


Note: if you chose to install an unsigned version of the extension, your browser might show you an error message about extension file being invalid; if you're sure the browser supports installation of unsigned extensions, go to the [about:config](#) URL, confirm the warning prompt that appears, search for the `xpinstall.signatures.required` preference and double-click it to toggle its value to "false"

You should be presented with a dialog asking whether to add Haketilo to your browser. Once you click "Add", the add-on will install.



The browser might ask whether you want to allow Haketilo to run in private windows. If you installed Haketilo for its script-blocking capabilities, you most likely also want to utilize them in Private Browsing mode and you can check this option. However, there might be some issues involved and it's recommended that you read the [related note](#).



Global install (GNU/Linux distributions)

Most Firefox-based browsers from GNU/Linux distros' package managers (including firefox-esr from Debian and abrowser from Trisquel) are configured to automatically pick up extensions placed under

```
/usr/share/mozilla/extensions/{ec8030f7-c20a-464f-9b0e-13a3a9e97384}
```

where {ec8030f7-c20a-464f-9b0e-13a3a9e97384} is the ID of Mozilla Firefox. This is what enables commands like `apt install webext-ublock-origin` to work on distros (in this case, Debian and Trisquel).

Here, we'll leverage this to install Haketilo system-wide and have it appear in the browser of every user account.

Note: you need access to root account via sudo command for the instructions below to work

Note: it is assumed you have wget and unzip commands installed

Steps

Note: depending on the versions of your web browser and Haketilo some UI elements and text messages might be slightly different on your machine than on the screenshots below

Run the following in your POSIX shell (i.e. copy+paste) to download the .xpi file and its PGP and Signify signatures made by us:

```
HAKETILO_VER="1.0b1" # replace with the version you want to have installed
HAKETILO_URL="https://hydrilla.koszko.org/downloads/haketilo-$HAKETILO_VER.xpi"
HAKETILO_DIR="/tmp/haketilo-${HAKETILO_VER}_release_files"
rm -rf $HAKETILO_DIR && mkdir -p $HAKETILO_DIR && cd $HAKETILO_DIR
wget "$HAKETILO_URL" "$HAKETILO_URL.sig" "$HAKETILO_URL.asc"
```

The files have been placed in a new directory under /tmp. You are now encouraged to verify our signatures as described [here](#). Once done, run the snippet below in **the same shell** (again, copy+paste) and type your user password when prompted:

```
FIREFOX_ID="{ec8030f7-c20a-464f-9b0e-13a3a9e97384}"
HAKETILO_ID="{6fe13369-88e9-440f-b837-5012fb3bedec}"
# You may need to use a different path for other browsers and distros.
EXT_PATH="/usr/share/mozilla/extensions/${FIREFOX_ID}/${HAKETILO_ID}/"

install_haketilo() {
    sudo rm -rf "$EXT_PATH" && sudo mkdir -p "$EXT_PATH" && cd "$EXT_PATH"
    sudo unzip $HAKETILO_DIR/haketilo-$HAKETILO_VER.xpi
    rm -rf $HAKETILO_DIR
}

install_haketilo
```

Once you have performed the installation, you might (or might not) want to give Haketilo access to private browser windows. You can do it from the about:addons page. There might be some issues involved, so it's recommended that you read the [related note](#).

Manage Your Extensions



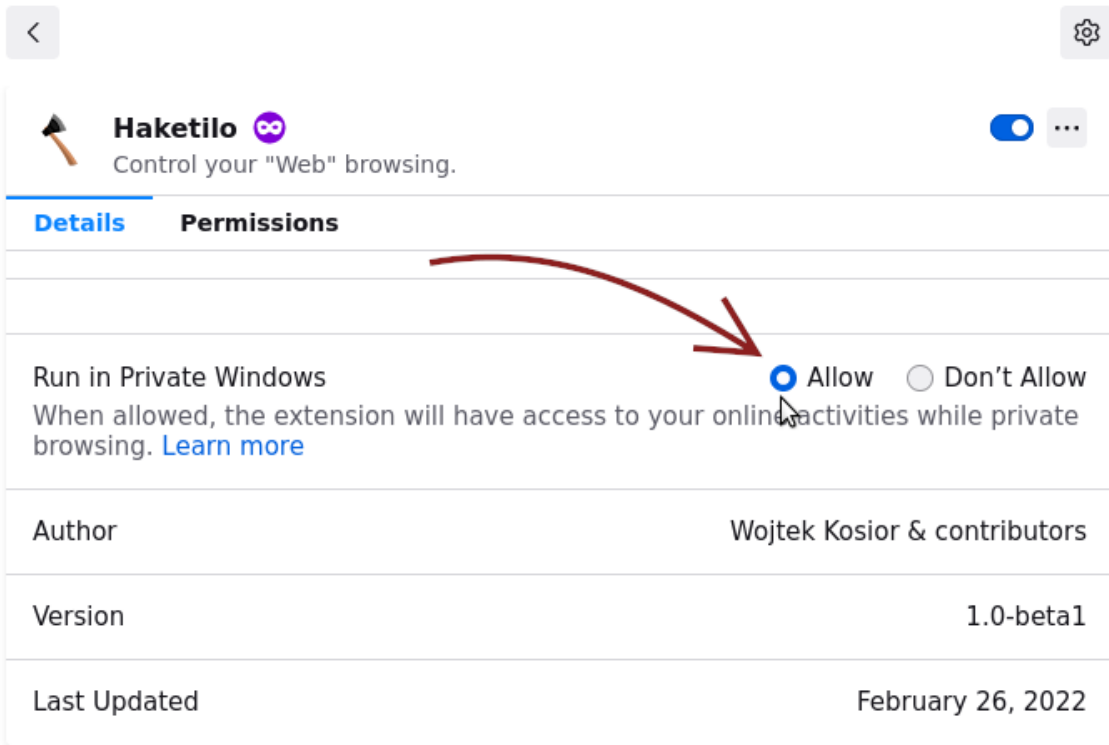
Enabled

The screenshot shows the 'Manage Your Extensions' interface. Under the 'Enabled' section, three extensions are listed:

- Haketilo** (with an infinity icon): Control your "Web" browsing. It has a blue toggle switch and a three-dot menu icon.
- Lightbeam 3.0**: Lightbeam is a browser extension that uses interactive visualizations to show you... It has a blue toggle switch and a three-dot menu icon.
- uBlock Origin**: It has a blue toggle switch and a three-dot menu icon.

A context menu is open over the Haketilo extension, showing the following options:

- Can't Be Removed [Why?](#)
- Preferences
- Report
- Manage** (highlighted by the mouse cursor)



Post-installation notes

Using Haketilo in private windows

If you're using the Private Browsing mode, you probably care about security and privacy. Those are basically unachievable with JavaScript globally allowed in the browser. Because of that, you likely want to give Haketilo permission to run in private windows as well.

Note: it is a perfectly valid approach to aim for a purely script-less experience in private browsing windows and use an add-on like NoScript or uBlock Origin there instead of Haketilo

Haketilo 1.0 uses IndexedDB in-browser database to store all its settings and data. Unfortunately, there are serious technical obstacles to using IndexedDB in Private Browsing mode (see bug [#115](#) for more details). This causes Haketilo's settings page and parts of popup to be inoperable even when Haketilo itself is allowed to run in private windows.

While work is ongoing to work around the limitation inside Haketilo, a temporary solution is to only access Haketilo's settings page and perform scripts installation when in a non-private browser window. Other parts of the extension work properly regardless of the Private Browsing mode being on. I.e. if you open Haketilo's settings page in a non-private window, configuration you make there will affect script blocking and injection in both private and non-private windows.

Congratulations

That's it. You can now play a bit with the extension and [learn how it works](#). Also, make sure you realize its [limitations](#).

Files

mozilla_install_from_file.png	50.8 KB	09/11/2021	koszko
navigate_extension_xpi_file.png	31.3 KB	09/11/2021	koszko
mozilla_add_hachette.png	57.9 KB	09/11/2021	koszko
mozilla_hachette_installed_allow_private.png	55.6 KB	09/11/2021	koszko
haketilo_manage.png	46.4 KB	02/26/2022	koszko
haketilo_manage_allow_private_tabs.png	49.7 KB	02/26/2022	koszko
about_config_accept_risk.png	37 KB	02/26/2022	koszko
about_config_restricted_domains.png	31.7 KB	02/26/2022	koszko
indexeddb_private_browsing.png	69.3 KB	03/04/2022	koszko

Verifying signatures

When downloading a release of Haketilo or Hydrilla, you might want to make sure the files are authentic and have not been tampered with. You can do this using the cryptographic signatures we provide.

New releases are signed using both [GNU Privacy Guard \(GPG\)](#) and [Signify](#) tool from the OpenBSD operating system (which can also be used on other systems, including FSF-approved ones like Trisquel). Your best bet is performing the verification using **both**, although in general Signify is considered more secure.

For the purpose of instructions below, let's make the following assumptions:

- We're verifying the integrity of Haketilo 1.0-beta1 source tarball (the procedure would be analogous when verifying other files).
- We're using a POSIX-compliant shell.
- We have access to wget command for downloading files.
- The following files have been downloaded into current directory (links here are the same as those present on Haketilo's [old releases page](#)):
 - [haketilo-1.0b1.tar.gz](#)
 - [haketilo-1.0b1.tar.gz.sig](#)
 - [haketilo-1.0b1.tar.gz.asc](#)

Verifying using Signify

First, you need to have the Signify tool installed. For Trisquel and other GNU/Linux distributions in Debian family, you can use a command like `sudo apt install signify-openbsd`. For Parabola and other Arch derivatives, you install with `sudo pacman -Syu signify`. On Guix it is `guix install signify`. On Fedora and other GNU/Linux distributions using RPM package manager you'd run `sudo dnf install signify`. An alternative command to achieve the same under RPM-powered distributions is `sudo yum install signify`.

Before you verify the signature, you need the public Signify key of Haketilo's maintainer, Wojtek Kosior. You can download it with this command:

```
wget https://koszko.org/key.pub
```

Now, ensure the downloaded key.pub file contains, besides the untrusted comment, only the **RWQsf2wUdpjAtrmt7D3t9iHrHFL/GpqXOF+NxECx8ck7swrx6tNzDkM9** string. If the string is different, it means someone's doing something nasty and you should not proceed.

You can now perform the verification with this command:

```
signify-openbsd -V -p key.pub -x haketilo-1.0b1.tar.gz.sig -m haketilo-1.0b1.tar.gz
```

Of course, you'll need to swap signify-openbsd for just signify on systems where the tool is named this way. Regardless, the output should be:

```
Signature Verified
```

If it is something else, it means verification failed and you should not trust the file you downloaded. For more information about using Signify see its [manual page](#).

Verifying using GPG

The PGP signatures we're providing can be verified using any tool that conforms to the OpenPGP standard. Nevertheless, the instructions we give here concern using the most popular GPG tool for the task. It is most likely already installed on the GNU/Linux distribution you're using. Let's assume it is available under the `gpg` command.

You need the public key of Haketilo's maintainer, Wojtek Kosior. The command below will check if `gpg`'s database already contains that key and will download and import it if not.

```
gpg --list-key koszko@koszko.org > /dev/null 2>&1 || (wget https://koszko.org/key.gpg 2>/dev/null && gpg --import key.gpg)
```

You can now perform the verification with this command:

```
gpg --verify haketilo-1.0b1.tar.gz.asc
```

It should output something like:

```
gpg: assuming signed data in 'haketilo-1.0b1.tar.gz'  
gpg: Signature made Fri 25 Feb 2022 05:17:50 PM CET  
gpg:          using EDDSA key E9727060E3C5637C8A4F4B424BC5221C5A79FD1A  
gpg: Good signature from "W. Kosior <koszko@koszko.org>" [unknown]  
gpg:          aka "Wojciech Kosior <wk@koszkonutek-tmp.pl.eu.org>" [unknown]  
gpg: WARNING: This key is not certified with a trusted signature!  
gpg:          There is no indication that the signature belongs to the owner.  
Primary key fingerprint: E972 7060 E3C5 637C 8A4F 4B42 4BC5 221C 5A79 FD1A
```

The most important thing is that the **E972 7060 E3C5 637C 8A4F 4B42 4BC5 221C 5A79 FD1A** string is present and matches. If not, it means someone's doing something nasty and you should not trust the file you downloaded.

Final considerations

The trust you can put in cryptographic signatures is at best as strong as your confidence that the public key you use for verification does indeed belong to the developer. Someone who tricks you into using a different key (for example by breaking into our server and swapping key files) can make a modified release file look like the legitimate one. For this reason it is a good idea to have public keys (fingerprints) posted in various places. The user should then make sure the key is the same as advertised in at least some of those places. Wojtek's public keys can be found in the following locations:

- the gitlab.trisquel.info [account](#)
- signed Haketilo/Hydrilla release [tags](#) (only PGP signatures and need to be extracted from the PGP data block)
- our LibrePlanet 2022 [presentation](#)

Additionally, Wojtek's GPG and Signify keys are cross-signed with each other¹.

1. <https://koszko.org/en/koszko.html#pubkeys> ↩

Administrative Documents

This is a root directory of sorts for publicly viewable paperwork-type tasks. That's it for now.

NLNet application for UOI Call August 2021

First edition of this document: https://hydrillabugs.koszko.org/projects/haketilo/wiki/NLNet_application_for_UOI_Call_August_2021/36

Please note:

[NLNet privacy statement](#).

When a project gets selected, it will legally need to retain your information for compliance purposes for at least seven years.

Note for answers:

Please be short and to the point in your answers; focus primarily on the what and how, not so much on the why. Add longer descriptions as attachments. If English isn't your first language, don't worry - reviewers don't care about spelling errors, only about great ideas. Apologies for the inconvenience of having to submit in English. On the up side, you can be as technical as you need to be (but you don't have to). Do stay concrete. Use plain text in your reply only, if you need any HTML to make your point please include this as attachment.

Attachments should only contain background information: Please make sure that the proposal without attachments is self-contained and concise. Accepted formats: HTML, PDF, OpenDocument Format and plain text files.

Abstract: Can you explain the whole project and its expected outcome(s).

No more than 1200 characters.

A browser extension, "Haketilo", (later possibly also an HTTP proxy and custom web browser) will be developed that facilitates browsing websites with custom changes (e.g. substituted page scripts, different site styling, alternative or aggregate interfaces for sites, accessibility&usability fixes, user translations, etc.) and makes it easy to edit such resources or develop them from scratch. Websites that force proprietary javascript, spyware and anti-features upon visitors will become fixable.

A project-maintained default repository, "Hydrilla", will serve as a rallying point, providing not only a comprehensive and trustworthy source of libre, privacy-respecting, secure and generally ethical site resources (including community-developed ones), but also a forum to share opinions about sites and to offer or solicit help with fixing problematic ones. Such a central hub further provides a unified body to negotiate with and pressure or advocate for particular website owners, strengthening the movement for a user-operated Internet.

All parts of the project shall be freely licensed (GPL, CC BY-SA).

Have you been involved with projects or organisations relevant to this project before? And if so, can you tell us a bit about your contributions?

Optional; this can help determine if you are the right person to undertake this effort

Our team currently consists of the following members:

- Wojtek (project maintainer) (<https://koszko.org/en/koszko.html>)
- Jahoti (<https://jahoti.tilde.team>)
- Nick (<https://nicksphere.com>)

Our experience:

- This project itself, consisting of Haketilo (<https://hydrillabugs.koszko.org/projects/haketilo>) and Hydrilla (<https://hydrillabugs.koszko.org/projects/hydrilla>), already exists as a simple yet functional setup. It is mostly written by Wojtek, with some contributions from Jahoti and Nick, and Hydrilla is entirely Wojtek's work.
- Jahoti and (especially) Wojtek have already written [free partial replacements for proprietary JavaScript](#) on numerous websites.
- Nick has studied Distributed Networks & Cybersecurity at Southern Illinois University Edwardsville, graduating with a major in the field.
- More comprehensive information for Wojtek is attached.

Additionally, we have been consulting on possible security issues of Haketilo and the repository with Richard Stallman himself for the past two months, and continue to do so.

Requested Amount (in Euro)

Explain what the requested budget will be used for? Does the project have other funding sources, both past and present?

If you want, you can in addition attach a budget at the bottom of the form. Fundable activities are (<https://nlnet.nl/useroperated/eligibility/>):

The requested budget will fund the necessary infrastructure (for 3 years) and 4 months of 2 developers' full-time work, as further detailed in the attached breakdown. This is expected to be long enough to complete the eligible tasks listed there; however, 6 months (€10988) or 8 months (€13600) would give a more polished and well-designed product.

Compare your own project with existing or historical efforts.

What is new, more thorough, otherwise different, etc.

- GNU LibreJS is the closest available comparison, as a project which also combines a browser extension with a social approach to push for greater user control of the software webpages require. Haketilo shares these ideals with LibreJS. However, the very narrow scope of LibreJS makes it unsuitable for the wider goals of Haketilo. It only supports GNU IceCat, while this project has been built from the start for both Firefox- and Chromium-based browsers with plans for more. Likewise, LibreJS only concerns itself with giving users the legal right to modify the JavaScript their browser runs, whereas Haketilo aims to provide a concrete way for anyone to modify the logic, visual layout, and other facets of what a browser presents when it loads up a webpage.
- Ad and content blockers overlap with the blocking functionality of the extension, and will likely continue to provide a source of code for this purpose as they have on previous occasions. Unfortunately, these tools only focus on trying to filter out trackers, ads or untrusted resources, giving the user passive but not active control over browsing.
- Userscript managers (e.g. GreaseMonkey and ViolentMonkey) have a long history of providing independent script injection on websites, yet differ wildly and irreconcilably from Haketilo. While they do offer some facility to source custom user scripts from online repositories and keep them up-to-date, they are designed with supplementing websites with minor tweaks in mind. As a result they chose to execute user scripts in privileged environments instead of the non-privileged page's context, thus avoiding interference with page's own scripts but also creating a security risk. A viable solution should inject scripts right into a page, making them execute in a proper sandbox. An even broader capacity to inject and maintain collections of various resources- and even to edit and develop them- is also critical.
- Hypothesis project offers facility for sharing annotations on web content. This idea is similar to one of our planned use-cases and it's even possible that Haketilo will, at some point, support Hypothesis annotations. However, the general goals of this project are significantly broader.
- Weboob tool implements graphical interfaces and programming APIs for various websites in Python programming language. It succeeds in achieving some of the goals we set in front of Haketilo. The main difference is that our project sticks to the usual technological stack of the Web, decreasing the amount of work required and that it also covers creations of a repository that will allow for greater scalability. As Weboob's code is freely licensed, it is likely some pieces of it will at some point be rewritten into javascript and uploaded to Hydrilla.

What are significant technical challenges you expect to solve during the project, if any?

Optional but recommended

- Porting to Manifest v3, especially while the standard and availability remain immature, will be a significant and important challenge.
- Developing Hydrilla as secure and robust server software that can continue working even under high load.
- Ensuring all functions of Haketilo work properly under all supported platforms.
- Testing javascript code that runs inside browser and uses a lot of browser APIs.

Describe the ecosystem of the project. How will you engage with relevant actors and promote the outcomes?

E.g. Which actors will you involve? Who should run or deploy your solution to make it a success?

End-users, particularly those with a technical inclination, will be recruited through outreach efforts in fora and locations generally sympathetic to the ideals of a free and open web. They are critical as a community to both support and expand this project and create leverage for the repository to drive change in web design practices.

Support from web developers and website owners is critical for long-term success in changing the web, as they collectively engineer its contents. Any who want our help in ethically (re)designing their creations will be offered as much support as physically possible; however, to break new ground among this group, it is expected that the leverage a strong community and influence over the repository afford will be needed.

Contributors to fill the repository and/or work on the browser extension and infrastructure obviously play an important role in attracting new users and building leverage for the movement. At least while scripts are the primary offering, sufficient capacity for this should be available among the technically knowledgeable user base just as it currently is. Further effort and experience will be sought through GNU, and appropriate organizations for other types of customization as they are added.

Thematic call

Included as a reminder- make sure to set this to **User-Operated Internet Fund**.

Attachments

- Experience- Wojtek's [CV](#)
- A [short PDF](#) with screenshots of sites broken by disabling JS and their versions fixed using Haketilo.
- [Budget Details](#) (This can probably just be exported using the "Also available in" feature of RedMine)

NLNet application for UOI Call August 2021- Budget Details

First edition of this document:

https://hydrillabugs.koszko.org/projects/haketilo/wiki/NLNet_application_for_UOI_Call_August_2021-_Budget_Details/3

Alternative Possible Budgets

We have an (effectively) fixed cost of approximately €3152 (estimated; see [#Infrastructure Costs](#)) for infrastructure critical to project management and development of Haketilo and Hydrilla, as well as wages of €653/month for two full-time developers.

Longer work means the ability to deliver a more featureful product.

Months allocated	Budget
4	€8376
6	€10988
8	€13600
10	€16212
12	€18824
16	€24048
20	€29272

Infrastructure Costs

The following are estimates, and may vary depending on applicability of VAT and exact choices of provider, as well as changes to exact project needs.

- Domain Name, another 2 years (€ ~26)
- SSL Cert, 3 years (€ ~534)
- Hosting for VCS, Project management software, website and script repo, 3 years (€ ~2592)

Developer Tasks

Project management work will also be a minor, yet important, use of time.

Social/Non-Development

- Understanding what features users most want from Haketilo
- Writing documentation for users for Haketilo and Hydrilla
- Determining effective methods to automatically aggregate already-available free JavaScript used on websites
- Studying what sites should be prioritized for fixing to deliver maximum impact
- Ensuring accessibility of Haketilo and Hydrilla for potentially underrepresented demographics
- Distribution of Haketilo in extension stores (as long as freedom and access concerns allow)
- Distribution of Haketilo and Hydrilla in GNU/Linux package managers
- Setting up and moderating the Hydrilla repository
 - Developing and writing policies for packages, packagers and (if adopted) auditors

Technical/Development

- Design and development of Haketilo and Hydrilla (available under the GPLv3)
- Writing developer documentation for Haketilo and Hydrilla
- Implementing accessibility of Haketilo and Hydrilla for potentially underrepresented demographics
- Writing and performing rigorous testing of Haketilo and Hydrilla
- Configuring a comprehensive automatic build and publishing process for Haketilo and Hydrilla
- Support for MV3 in Haketilo
- Creating and porting independent fixes and enhancements for some websites
- Security vetting on Hydrilla

Reporting bugs

Bug reporting is one of the best ways to help with the development of Haketilo and Hydrilla. If you noticed something does not work as it should, don't hesitate and write to us immediately. Bug reports are **WANTED!**

Below we describe the procedure you should follow when making a report. You may find it a bit overwhelming but it is all to make it easier for us to find and fix the bug. If you lack time, skill or knowledge to make a perfect report, you can still send us an incomplete one. This will be better than no report and we'll do what we can with it :)

Some of the following may not be relevant. E.g. switching to English locale makes no sense when the bug concerns localization itself. In such cases you can safely skip given step.

Before you send the report

- Make sure the bug can be reproduced in the newest release of the software (you can also try the version from a master git branch if you have the skills required to build it).
- Make sure there is nothing that interferes with the software.
 - If bug report concerns Haketilo or a site script, make sure the bug is reproducible after disabling all other extensions and restarting the browser.
- For the time of preparing the report, switch to using the English locale so that all messages are in English.

What to include in a bug report

Please include answers to the following questions.

- Did you follow the recommendations from the section above? If not, which point(s) did you omit?
- What is your environment?
 - If bug is in either Haketilo, a site script or project website, what browser are you using (name + version + how you installed this browser)?
 - If bug is in Haketilo proxy or in Hydrilla, what versions of dependencies are you using and how did you install them? Also, are you using virtualenv?
 - If bug is in a site script, please also include the Haketilo version and describe how you installed it.
- How did you install the misbehaving program? Did you install from git, from a source release tarball or from a repository, or did you use a prebuilt release made by us?
- Which version of the buggy software did you use? If you built from git, please provide the relevant commit hash or tag. Otherwise, just the version number is sufficient.
- What are the steps to reproduce the bug? (screenshots are not necessary but welcome)
- What is the expected behavior?
- What behavior did you experience instead? (screenshots are not necessary but welcome)
- What error/warning messages did you see (if any)?
 - If bug is in either Haketilo, a site script or project website, include the output from the Developer Tools console¹.
 - If bug is in a site script or project website, consider including the network logs exported to a HAR file². Feel free to omit this one if either it is not relevant, network logs contain confidential information or you have trouble figuring out how to do this.
 - If bug is in Haketilo browser extension, please include additional messages from extension's background page (*this is often crucial*)^{3,4}.
- Does the bug manifest itself always or only on some occasions?
- Is there anything else that could be useful for us to know?

Sending the report

There are basically 2 options.

- Make [an account](#) on this issue tracker and create a new issue.
- Write to koszko@koszko.org.

-
1. Developer Tools console can be opened with Ctrl+Shift+K in Mozilla-based browsers and Ctrl+Shift+I in Chromium-based browsers. [↪](#)
 2. HAR file can be create from Developer Tools "Network" tab, opened with Ctrl+Shift+E in Mozilla-based browsers. The "Network" tab needs to be opened *before* loading the page. [↪](#)
 3. Under modern Mozilla-based browsers, go to about:debugging#/runtime/this-firefox and click "Inspect" next to the Haketilo icon. A console with messages from the background page should show up. [↪](#)
 4. Under Chromium-based browsers, go to chrome://extensions/ and toggle "Developer mode". Click Haketilo's "Details" button and under "Inspect views" click "background page". In the DevTools window that appears switch to the "Console" tab. [↪](#)