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 Hydrilla[1]
- 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 support[2]
- 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~~[4][5]
- ~~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 [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 CORS~~[13]
- ~~free/libre JavaScript library for solving reCAPTCHA challenges~~[14]
- ~~sample Haketilo resource making use of the library on a chosen website~~[15]

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 ↩