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 HTTP~~s~~ 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 :)