mitmproxy-HTTPolice Documentation

Vasiliy Faronov

Jun 27, 2019

Contents

mitmproxy is an advanced HTTP debugging tool that can intercept TLS-encrypted connections, supports HTTP/2, and much more.

mitmproxy-HTTPolice is an addon for mitmproxy that will check intercepted exchanges and produce HTTPolice reports.

CHAPTER 1

Contents

1.1 Walkthrough

This walkthrough explains how to use mitmproxy-HTTPolice in various cases, but it does not explain mitmproxy itself. For best results, familiarize yourself with mitmproxy and its docs first.

1.1.1 Installation

Do this in a Python 3.6+ environment:

```
$ pip3 install mitmproxy-HTTPolice
```

If this is giving you trouble, see mitmproxy docs and HTTPolice docs for more detailed instructions.

Note: **Do not use** mitmproxy's pre-built self-contained binaries. mitmproxy and HTTPolice need to live in the same Python environment, and this is only possible if you install mitmproxy from source via pip. See the "Installation via pip3" sections in mitmproxy docs.

1.1.2 Startup

Get the path to the mitmproxy-HTTPolice addon file with the following command:

```
$ python3 -m mitmproxy_httpolice
/home/vasiliy/.local/lib/python3.6/site-packages/mitmproxy_httpolice.py
```

Tell mitmproxy to load this file with the `-s` (`--scripts`) option, like this (note the backticks):

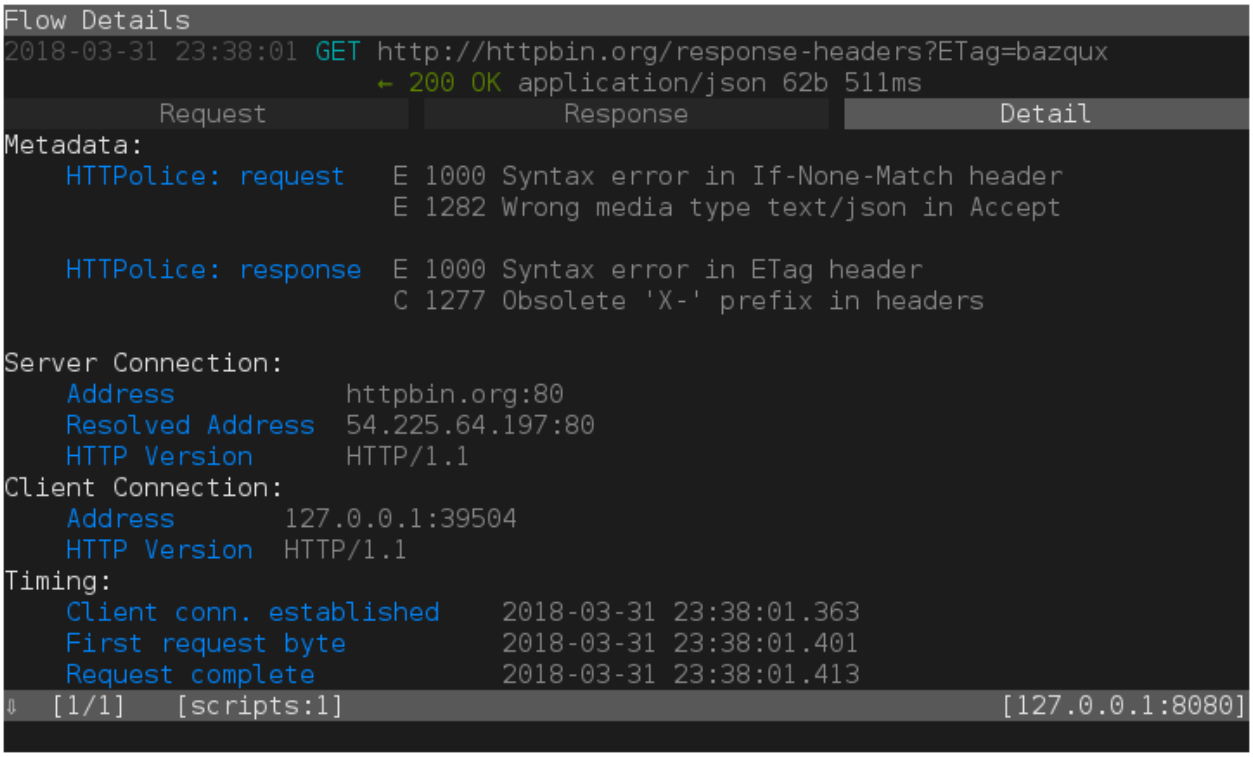
```
$ mitmproxy -s "`python3 -m mitmproxy_httpolice`"
```

Or just put it into your `~/.mitmproxy/config.yaml`:

```
scripts:
  - /home/vasiliy/.local/lib/python3.6/site-packages/mitmproxy_httpolice.py
```

1.1.3 Inspecting traffic on the fly

mitmproxy-HTTPolice checks every flow (exchange) and prints any results on the flow's *Details* pane, under *Metadata*:



1.1.4 Marking flows with problems

From the flow list, how do you know which flows have any interesting notices on them?

HTTPolice can *mark* them for you if you set the option `httpolice_mark`. There are several ways to set it:

- From inside mitmproxy: with the options editor (by typing `O`). The new value you set there will **only apply to newly captured flows**.
- On the command line:

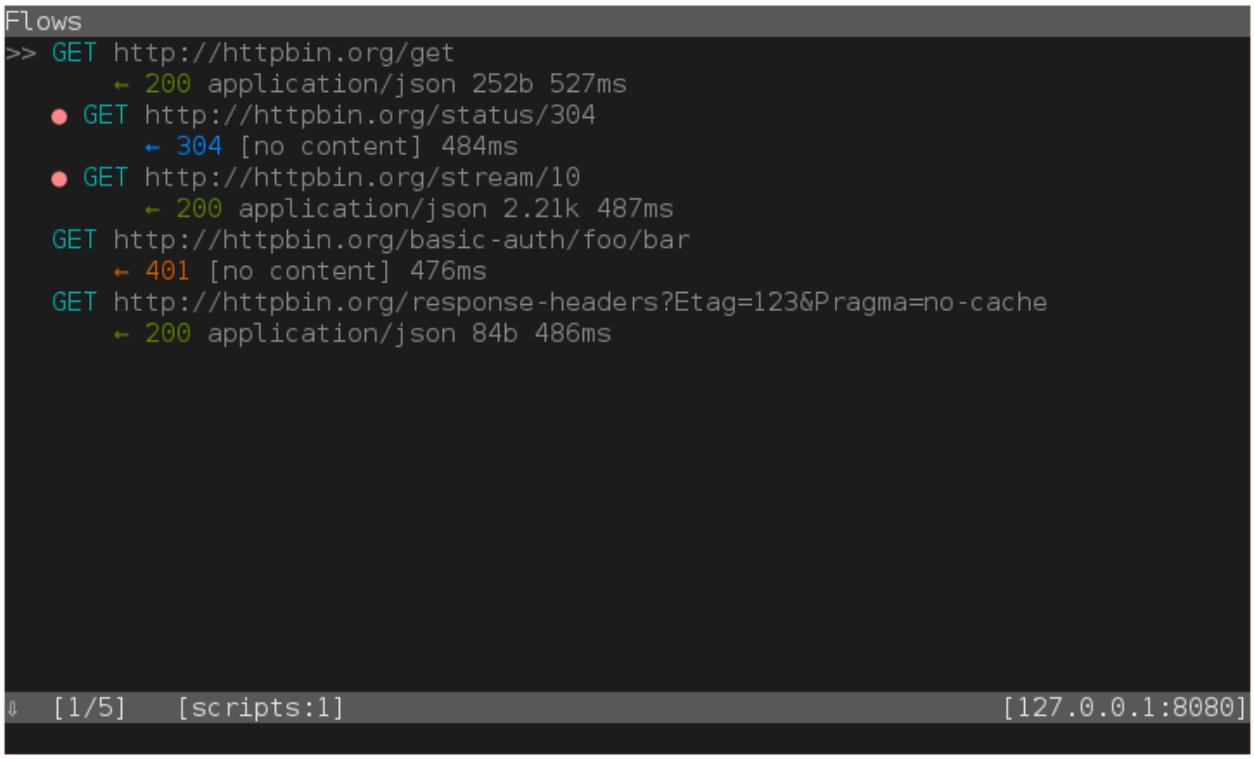
  ```
  $ mitmproxy --set httpolice_mark=comment
  ```

- In your `~/.mitmproxy/config.yaml`:

  ```
  httpolice_mark: error
  ```

`httpolice_mark=comment` means "mark any flows where HTTPolice has found at least one comment or error". `httpolice_mark=error` limits this to errors.

Marking flows is a general concept in mitmproxy. Marked flows have a big fat dot next to them in the flow list:



You can quickly run commands on "all marked flows", for instance, to save them to a file:

```
: save.file @marked /path/to/marked.flows
```

You can also manually toggle the mark on any flow by typing `m`.

1.1.5 Silencing unwanted notices

With the `httpolice_silence` option, you can tell HTTPolice which notice IDs to silence. They will disappear from flow details, and so on.

When editing this option in mitmproxy's interactive options editor, type `a` to add a new item, then type the notice ID, then Esc to commit. Type `d` on an item to delete it. The new value you set will apply to newly captured flows and newly produced reports.

In `~/.mitmproxy/config.yaml`, notice IDs must be quoted so they are treated as strings, **not** numbers:

```
httpolice_silence:
  - "1234"
  - "1256"
```

1.1.6 Full reports

If you prefer to see HTTPolice's full HTML report, you can create one with the `httpolice.report.html` command. For example:

```
: httpolice.report.html @all /path/to/report.html
```

Here, `@all` means "all flows". You can replace it with any of mitmproxy's filter expressions, among them `@marked` for flows that have been previously *marked* by HTTPolice.

There's also the `httpolice.report.text` command if you want the plain text report.

1.1.7 In-memory reports

In fact, you don't have to go through a file to see the report. You can put a dash (`-`) in place of the filename:

```
: httpolice.report.html @all -
```

The report will be stored in memory, and you can view it by visiting `/+httpolice/` *via mitmproxy*. If mitmproxy is running on `localhost:8080` in reverse proxy mode, then just go to `http://localhost:8080/+httpolice/` in your Web browser. When in regular proxy mode, visit something like `http://example.com/+httpolice/` through the proxy (the domain doesn't matter, only the path).

1.1.8 Key bindings

Because `httpolice.report.html` is a normal mitmproxy command, you can set key bindings for it. For example, if you put the following into your `~/.mitmproxy/keys.yaml`:

```
- key: W
  cmd: httpolice.report.html @focus ~/report.html
```

then typing `W` (that is, Shift+W) will produce an HTML report on the currently focused flow in `~/report.html`.

1.1.9 Example workflow

Here's one workflow that can arise from the features explained so far.

Let's say you're a developer (or tester) iterating on a piece of software that sends or serves HTTP requests, and you want that software to implement the HTTP protocol correctly.

First, *set option* `httpolice_mark` to `comment`, and *set a keybinding* like this:

```
- key: f5
  cmd: httpolice.report.html @marked -
```

Then:

1. Do something with your software, capturing a bunch of flows into mitmproxy.
2. In mitmproxy, type F5. HTTPolice produces a report on all the problems found so far and stores it in memory.
3. Open (or refresh) `http://localhost:8080/+httpolice/` in your Web browser (tweak the URL *as necessary*) to read the report.
4. Fix the problems in your software, or *silence them* in HTTPolice.
5. Now that you are done with that particular batch of flows, type `z` in mitmproxy to clear the flow list.
6. Rinse, repeat.

1.1.10 Non-interactive use

mitmproxy-HTTPolice is currently focused on interactive use. Of mitmproxy's three tools, only the original `mitmproxy` console UI currently supports HTTPolice. `mitmweb` lacks the necessary features, although it will probably catch up to `mitmproxy` eventually. `mitmdump` is aimed at non-interactive use and HTTPolice doesn't do anything useful under it.

That said, you can get data from `mitmdump` into HTTPolice like this:

1. Run `mitmdump` with the `--save-stream-file` option to save flows into a file.
2. Run `mitmproxy` with the `--no-server` and `--rfile` options to load flows from that file. Of course, you may run it on another system.
3. Work in `mitmproxy` as usual (`: httpolice.report.html @all ...`).

1.2 History of changes

1.2.1 0.9.0 - 2019-06-27

- You can now view full reports directly from mitmproxy, without saving them to a file first.

1.2.2 0.8.0 - 2019-03-03

This version is for mitmproxy 4.

1.2.3 0.7.1 - 2018-05-20

This version adds an explicit requirement for mitmproxy version less than 4. A new version compatible with mitmproxy 4 and/or higher will be released eventually, when time permits.

1.2.4 0.7.0 - 2018-03-31

- Overhaul for mitmproxy 3.
- Now focuses on closer integration with mitmproxy's interactive features:
  - commands to produce reports on any flows
  - options that can be changed on the fly
  - marks on flows with problems
  - better display of notices in flow details
- At least for now, the original approach of writing a report non-interactively (options `-w`, `--tail`) is not supported. You need the `mitmproxy` tool, not `mitmdump` or `mitmweb`.
- See docs for details.

1.2.5 0.6.1 - 2017-08-02

- Fixed dumping reports to non-seekable files (like `-w /dev/stdout`).
- Fixed `--tail` with small (text) reports.

1.2.6 0.6.0 - 2017-03-12

Added

- A new `--tail` option to regenerate the report on every new exchange, so you can inspect traffic as it comes (see docs).
- HTTPolice now writes brief summaries to mitmproxy's event log, like this:

```
HTTPolice found 1 errors, 2 comments in: GET /api/v1/ - 200 OK
```

  (The event log is printed to the console when you use `mitmdump`, or to the "Event log" pane when you press the 'e' key in `mitmproxy`.)
- In the `mitmproxy` console UI, you can now see a brief report for every individual exchange on its "Detail" pane (see docs).

Changed

- The output file is now specified with the `-w` option instead of just a positional argument, for example:

```
$ mitmdump -s "`python3 -m mitmproxy_httpolice` -w report.txt"
```

  This `-w` option is actually *optional*: you can omit it if you only want to view the reports in the console UI, for example.

1.2.7 0.5.1 - 2017-02-28

Fixed an error that happened on many/most HTTP/2 requests (those without a Host header).

1.2.8 0.5.0 - 2017-01-14

No interesting changes; just update docs and packaging for better compatibility with Python 3.6 and mitmproxy 1.0. This version also drops support for Python 2. If you need Python 2, use `mitmproxy==0.18.2` and `mitmproxy-HTTPolice==0.4.0`.

1.2.9 0.4.0 - 2016-10-17

This release is compatible with mitmproxy 0.18+, and **only** 0.18+ (because mitmproxy 0.18 has a new, backward-incompatible API). Note that mitmproxy (and thus mitmproxy-HTTPolice) now supports Python 3.5+.

1.2.10 0.3.0 - 2016-08-14

Technical release. No interesting changes.

1.2.11 0.2.0 - 2016-05-08

Initial release as a separate distribution.