xepor Documentation

Release 0.6.0.post1.dev1+g146e39e

ttimasdf

Jul 06, 2023

CONTENTS

Xepor (pronounced */zf/*, zephyr) is a web routing framework for reverse engineers and security researchers. It provides a Flask-like API for hackers to intercept and modify HTTP requests and/or HTTP responses in a human-friendly coding style.

This project is meant to be used with mitmproxy. Users write scripts with `xepor`, and run the script *inside* mitmproxy with `mitmproxy -s your-script.py`.

If you want to step from PoC to production, from demo(e.g. http-reply-from-proxy.py, http-trailers.py, http-stream-modify.py) to something you could take out with your WiFi Pineapple, then Xepor is for you!

CHAPTER

ONE

FEATURES

1. Code everything with `@api.route()`, just like Flask! Write everything in *one* script and no `if..else` any more.
2. Handle multiple URL routes, even multiple hosts in one `InterceptedAPI` instance.
3. For each route, you can choose to modify the request *before* connecting to server (or even return a fake response without connection to upstream), or modify the response *before* forwarding to user.
4. Blacklist mode or whitelist mode. Only allow URL endpoints defined in scripts to connect to upstream, blocking everything else (in specific domains) with HTTP 404. Suitable for transparent proxying.
5. Human readable URL path definition and matching powered by parse
6. Host remapping. define rules to redirect to genuine upstream from your fake hosts. Regex matching is supported. **Best for SSL stripping and server-side license cracking**!
7. Plus all the bests from mitmproxy! **ALL** operation modes (`mitmproxy` / `mitmweb` + `regular` / `transparent` / `socks5` / `reverse:SPEC` / `upstream:SPEC`) are fully supported.

CHAPTER

TWO

USE CASE

1. Evil AP and phishing through MITM.
2. Sniffing traffic from target device by iptables + transparent proxy, modify the payload with xepor on the fly.
3. Cracking cloud-based software license. See examples/krisp/ as an example.
4. Write a complicated web crawler in **~100 lines of code**. See examples/polyv_scrapper/ as an example.
5. … and many more.

SSL stripping is NOT provided by this project.

CHAPTER

THREE

INSTALLATION

```
pip install xepor
```

CHAPTER

FOUR

QUICK START

Take the script from examples/httpbin as an example.

```
mitmweb -s example/httpbin/httpbin.py
```

Set your Browser HTTP Proxy to `http://127.0.0.1:8080`, and access the web interface at http://127.0.0.1:8081/.

Send a GET request from http://httpbin.org/#/HTTP_Methods/get_get , Then you could see the modification made by Xepor in mitmweb interface, browser dev tools or Wireshark.

The `httpbin.py` do two things.

1. When user access http://httpbin.org/get, inject a query string parameter `payload=evil_param` inside HTTP request.
2. When user access http://httpbin.org/basic-auth/xx/xx/ (we just pretend we don't know the password), sniff `Authorization` headers from HTTP requests and print the password to the attacker.

Just what mitmproxy always does, but with code written in *xepor* way.

```python
# https://github.com/xepor/xepor-examples/tree/main/httpbin/httpbin.py
from mitmproxy.http import HTTPFlow
from xepor import InterceptedAPI, RouteType


HOST_HTTPBIN = "httpbin.org"

api = InterceptedAPI(HOST_HTTPBIN)


@api.route("/get")
def change_your_request(flow: HTTPFlow):
    """
    Modify URL query param.
    Test at:
    http://httpbin.org/#/HTTP_Methods/get_get
    """
    flow.request.query["payload"] = "evil_param"


@api.route("/basic-auth/{usr}/{pwd}", rtype=RouteType.RESPONSE)
def capture_auth(flow: HTTPFlow, usr=None, pwd=None):
    """
    Sniffing password.
```

(continues on next page)

(continued from previous page)

```
    Test at:
    http://httpbin.org/#/Auth/get_basic_auth__user___passwd_
    """
    print(
        f"auth @ {usr} + {pwd}:",
        f"Captured {'successful' if flow.response.status_code < 300 else 'unsuccessful'}␣
↪login:",
        flow.request.headers.get("Authorization", ""),
    )


addons = [api]
```

4.1 Contents

4.1.1 Xepor

Xepor (pronounced */zf/*, zephyr) is a web routing framework for reverse engineers and security researchers. It provides a Flask-like API for hackers to intercept and modify HTTP requests and/or HTTP responses in a human-friendly coding style.

This project is meant to be used with mitmproxy. Users write scripts with `xepor`, and run the script *inside* mitmproxy with `mitmproxy -s your-script.py`.

If you want to step from PoC to production, from demo(e.g. http-reply-from-proxy.py, http-trailers.py, http-stream-modify.py) to something you could take out with your WiFi Pineapple, then Xepor is for you!

Features

1. Code everything with `@api.route()`, just like Flask! Write everything in *one* script and no `if..else` any more.
2. Handle multiple URL routes, even multiple hosts in one `InterceptedAPI` instance.
3. For each route, you can choose to modify the request *before* connecting to server (or even return a fake response without connection to upstream), or modify the response *before* forwarding to user.
4. Blacklist mode or whitelist mode. Only allow URL endpoints defined in scripts to connect to upstream, blocking everything else (in specific domains) with HTTP 404. Suitable for transparent proxying.
5. Human readable URL path definition and matching powered by parse
6. Host remapping. define rules to redirect to genuine upstream from your fake hosts. Regex matching is supported. **Best for SSL stripping and server-side license cracking**!
7. Plus all the bests from mitmproxy! **ALL** operation modes (`mitmproxy` / `mitmweb` + `regular` / `transparent` / `socks5` / `reverse:SPEC` / `upstream:SPEC`) are fully supported.

Use Case

1. Evil AP and phishing through MITM.
2. Sniffing traffic from target device by iptables + transparent proxy, modify the payload with xepor on the fly.
3. Cracking cloud-based software license. See examples/krisp/ as an example.
4. Write a complicated web crawler in **~100 lines of code**. See examples/polyv_scrapper/ as an example.
5. ... and many more.

SSL stripping is NOT provided by this project.

4.1.2 Installation

```
pip install xepor
```

4.1.3 Quick start

Take the script from examples/httpbin as an example.

```
mitmweb -s example/httpbin/httpbin.py
```

Set your Browser HTTP Proxy to `http://127.0.0.1:8080`, and access the web interface at http://127.0.0.1:8081/.

Send a GET request from http://httpbin.org/#/HTTP_Methods/get_get , Then you could see the modification made by Xepor in mitmweb interface, browser dev tools or Wireshark.

The `httpbin.py` do two things.

1. When user access http://httpbin.org/get, inject a query string parameter `payload=evil_param` inside HTTP request.
2. When user access http://httpbin.org/basic-auth/xx/xx/ (we just pretend we don't know the password), sniff `Authorization` headers from HTTP requests and print the password to the attacker.

Just what mitmproxy always does, but with code written in *xepor* way.

```
# https://github.com/xepor/xepor-examples/tree/main/httpbin/httpbin.py
from mitmproxy.http import HTTPFlow
from xepor import InterceptedAPI, RouteType


HOST_HTTPBIN = "httpbin.org"

api = InterceptedAPI(HOST_HTTPBIN)


@api.route("/get")
def change_your_request(flow: HTTPFlow):
    """
    Modify URL query param.
    Test at:
    http://httpbin.org/#/HTTP_Methods/get_get
    """
```

(continues on next page)

(continued from previous page)

```
    flow.request.query["payload"] = "evil_param"


@api.route("/basic-auth/{usr}/{pwd}", rtype=RouteType.RESPONSE)
def capture_auth(flow: HTTPFlow, usr=None, pwd=None):
    """
    Sniffing password.
    Test at:
    http://httpbin.org/#/Auth/get_basic_auth__user___passwd_
    """
    print(
        f"auth @ {usr} + {pwd}:",
        f"Captured {'successful' if flow.response.status_code < 300 else 'unsuccessful'}␣
↪login:",
        flow.request.headers.get("Authorization", ""),
    )


addons = [api]
```

4.1.4 xepor package

class xepor.**InterceptedAPI**(*default_host=None*, *host_mapping={}*, *blacklist_domain=[]*, *request_passthrough=True*, *response_passthrough=True*, *respect_proxy_headers=False*)

Bases: `object`

the InterceptedAPI object is the central registry of your view functions. Users should use a function decorator *route()* to define and register URL and host mapping to the view functions. Just like flask's `flask.Flask.route()`.

```
from xepor import InterceptedAPI, RouteType

HOST_HTTPBIN = "httpbin.org"
api = InterceptedAPI(HOST_HTTPBIN)
```

Defining a constant for your target (victim) domain name is not mandatory (even the *default_host* parameter itself is optional) but recommanded as a best practise. If you have multiple hosts to inject (see an example at xepor/xepor-examples/polyv_scrapper/polyv.py), you would have to specify the domain name multiple times in each *route()* in *host* parameter, (especially for domains other than *default_host*). So it's better to have a variable for that.

Add route via function call similar to Flask `flask.Flask.add_url_rule()` is not yet implemented.

Parameters

- **default_host** (*str | None*) – The default host to forward requests to.
- **host_mapping** (*List[Tuple[str | Pattern, str]]*) – A list of tuples of the form (regex, host) where regex is a regular expression to match against the request host and host is the host to redirect the request to.
- **blacklist_domain** (*List[str]*) – A list of domains to not forward requests to. The requests and response from hosts in this list will not respect `request_passthrough` and

response_passthrough setting.

- **request_passthrough** (*bool*) – Whether to forward the request to upstream server if no route is found. If `request_passthrough = False`, all requests not matching any route will be responded with *default_response()* without connecting to upstream.
- **response_passthrough** (*bool*) – Whether to forward the response to the user if no route is found. If `response_passthrough = False`, all responses not matching any route will be replaced with the Response object generated by *default_response()*.
- **respect_proxy_headers** (*bool*) – Set to *True* only when you use Xepor as a web server behind a reverse proxy. Typical use case is to set up an mitmproxy in `reverse` mode to bypass some online license checks. Xepor will respect the following headers and strip them from requests to upstream.
  - *X-Forwarded-For*
  - *X-Forwarded-Host*
  - *X-Forwarded-Port*
  - *X-Forwarded-Proto*
  - *X-Forwarded-Server*
  - *X-Real-Ip*

load(*loader*)

This function is called by the mitmproxy framework *before* proxy server started. Currently it's used to set a must-have mitmproxy option for Xepor to work: `connection_strategy=lazy`. If you want to override this method, remember to call `super().load(loader)` in your code.

User can also import and use `mitmproxy.ctx` object to configure other options for mitmproxy when overriding this function.

```
from mitmproxy import ctx

ctx.options.connection_strategy = "lazy"
```

Parameters
loader (*Loader*) – a `mitmproxy.addonmanager.Loader` which can be used to add custom options.

Returns
None

request(*flow*)

This function is called by the mitmproxy framework whenever a request is made.

Parameters
flow (*HTTPFlow*) – The `mitmproxy.http.HTTPFlow` object from client request.

Returns
None

response(*flow*)

This function is called by the mitmproxy when a response is returned the server.

Parameters
flow (*HTTPFlow*) – The `mitmproxy.http.HTTPFlow` object from server response.

Returns
None

route(*path*, *host=None*, *rtype=RouteType.REQUEST*, *catch_error=True*, *return_error=False*)

This is the main API used by end users. It decorate a view function to register it with given host and URL.

Typical usage (taken from official example: httpbin.py):

```
@api.route("/get")
def change_your_request(flow: HTTPFlow):
    flow.request.query["payload"] = "evil_param"

@api.route("/basic-auth/{usr}/{pwd}", rtype=RouteType.RESPONSE)
def capture_auth(flow: HTTPFlow, usr=None, pwd=None):
    print(
        f"auth @ {usr} + {pwd}:",
        f"Captured {'successful' if flow.response.status_code < 300 else
↪'unsuccessful'} login:",
        flow.request.headers.get("Authorization", ""),
    )
```

See Github: xepor/xepor-examples for more examples.

Parameters

- **path** (*str*) – The URL path to be routed. The path definition grammar is similar to Python 3 `str.format()`. Check the documentation of `parse` library: r1chardj0n3s/parse
- **host** (*str | None*) – The host to be routed. This value will be matched against the following fields of incoming flow object by order:
  1. `X-Forwarded-For` Header. (only when *respect_proxy_headers* in *InterceptedAPI* is *True*)
  2. HTTP `Host` Header, if exists.
  3. `flow.host` reported by underlying layer. In HTTP or Socks5h proxy mode, it may hopefully be a hostname, otherwise, it'll be an IP address.
- **rtype** (RouteType) – Set the route be matched on either request or response. Accepting *RouteType*.
- **catch_error** (*bool*) – If set to *True*, the exception inside the route will be handled by Xepor.

  If set to *False*, the exception will be raised and handled by mitmproxy.
- **return_error** (*bool*) – If set to *True*, the error message inside the exception (`str(exc)`) will be returned to client. This behaviour can be overrided through *error_response()*.

  If set to *False*, the exception will be printed to console, the `flow` object will be passed to mitmproxy continuely.

Note

When exception occured, the `flow` object do *not* always stay intact. This option is only a try-catch like normal Python code. If you run `modify1(flow)` and `modify2(flow)` and `modify3(flow)` and exception raised in `modify2()`, the `flow` object will be modified partially.

- **self** (*str*) –

Returns
The decorated function.

remap_host(*flow*, *overwrite=True*)

Remaps the host of the flow to the destination host.

Note

This function is used internally by Xepor. Refer to the source code for customization.

Parameters

- **flow** (*HTTPFlow*) – The flow to remap.
- **overwrite** – Whether to overwrite the host and port of the flow.

Returns
The remapped host.

get_host(*flow*)

Gets the host and port of the request. Extending from `mitmproxy.http.HTTPFlow.pretty_host` to accept values from proxy headers(`X-Forwarded-Host` and `X-Forwarded-Port`)

Note

This function is used internally by Xepor. Refer to the source code for customization.

Parameters
flow (*HTTPFlow*) – The HTTPFlow object.

Returns
A tuple of the host and port.

Return type
Tuple[str, int]

default_response()

This is the default response function for Xepor. It will be called in following conditions:

1. target host in HTTP request matches the ones in *blacklist_domain*.
2. either *request_passthrough* or *response_passthrough* is set to *False*, and no route matches the incoming flow.

Override this function if it suits your needs.

Returns
A Response object with status code 404 and HTTP header `X-Intercepted-By` set to `xepor`.

error_response(*msg='APIServer Error'*)

Returns a response with status code 502 and custom error message.

Override this function if it suits your needs.

Parameters
msg (*str*) – The message to be returned.

Returns
A Response object with status code 502 and content set to the .

`find_handler`(*host*, *path*, *rtype=RouteType.REQUEST*)

Finds the appropriate handler for the request.

Note

This function is used internally by Xepor. Refer to the source code for customization.

Parameters

- **host** – The host of the request.
- **path** – The path of the request.
- **rtype** – The type of the route. Accepting *RouteType*.

Returns
The handler and the parse result.

`class xepor.RouteType`(*value*)

Bases: Enum

This enum is an option set in route definition, specify it to be matched on either incoming request or response.

REQUEST = 1

The route will be matched on mitmproxy `request` event

RESPONSE = 2

The route will be matched on mitmproxy `response` event

`class xepor.FlowMeta`(*value*)

Bases: Enum

This class is used internally by Xepor to mark `flow` object by certain metadata. Refer to the source code for detailed usage.

REQ_PASSTHROUGH = 'xepor-request-passthrough'

RESP_PASSTHROUGH = 'xepor-response-passthrough'

REQ_URLPARSE = 'xepor-request-urlparse'

REQ_HOST = 'xepor-request-host'

4.1.5 Changelog

Version 0.6.0

- Feature: Automatically set `--set connection_strategy=lazy` when Xepor scripts are loaded. No need to manually set the options any more
- Feature: Add *xepor.InterceptedAPI.load()* API for configuration before start. Check the documentation for usage details.

Version 0.5.1

- Feature: Add compatibility with mitmproxy 9.x and Python 3.11~3.12.
  - Since mitmproxy 8.x is not compatible with Python 3.11, therefore updating to mitmproxy 9.x + xepor 0.5.1 is recommended especially for Kali users with rolling updates.

Version 0.5.0

Breaking Change: Incompatible API changes.

- Change: Move some constants inside *InterceptedAPI* to seperate enums: *xepor.RouteType()*, *xepor.FlowMeta()*. (xepor/xepor@83128e8)
- Change: rename `reqtype` in *xepor.InterceptedAPI.route()* to `rtype`. The name there before is for historical reason. xepor/xepor@c6c7e83
- All exposed API is now type hinted and well documented! check the latest Documentation. example scripts are also updated with API change.

Version 0.4.0

- Feature: Add compatibility with mitmproxy 8.x.

Version 0.3.0

- Feature: Add `return_error` option for *xepor.InterceptedAPI.route()* (xepor/xepor@fddad13)
- Add more docs and examples. Also some tests to ensure the script stay up-to-date with version changes. (xepor/xepor@3f96bf1)

Version 0.2.0

Breaking Change: mitmproxy 7.0 contains backward-incompatible API change comparing with <=6.x, update to 7.x may need to modify all of your scripts (including those not written with Xepor). Just a change on some methods' signatures.

- Feature: Add `blacklist_domain` and `respect_proxy_headers` options for *InterceptedAPI*. Refer to API Doc for their usage. (xepor/xepor@aa2517d)
- Feature: Add a debug header when client request hitting *default_response()* (xepor/xepor@d57be79)
- Fix: adopt API change in mitmproxy 7.x (xepor/xepor@beedeab)
- Fix: Request is forwarded to wrong destination host when mitmproxy is start in reverse/upstream mode. (xepor/xepor@aa2517d) **Be sure to start mitmproxy with `--set connection_strategy=lazy` under ANY circumstances or the upstream connection may still be wrong.**
- Update example scripts.

Version 0.1 (development)

This is **the only version** compatible with mitmproxy <= 6.x

- Implement most features.

4.1.6 Contributors

- ttimasdf https://github.com/ttimasdf

4.1.7 License

(continues on next page)

(continued from previous page)

```
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions
   to that Work or Derivative Works thereof, that is intentionally
   submitted to Licensor for inclusion in the Work by the copyright owner
   or by an individual or Legal Entity authorized to submit on behalf of
   the copyright owner. For the purposes of this definition, "submitted"
   means any form of electronic, verbal, or written communication sent
   to the Licensor or its representatives, including but not limited to
   communication on electronic mailing lists, source code control systems,
   and issue tracking systems that are managed by, or on behalf of, the
   Licensor for the purpose of discussing and improving the Work, but
   excluding communication that is conspicuously marked or otherwise
   designated in writing by the copyright owner as "Not a Contribution."

   "Contributor" shall mean Licensor and any individual or Legal Entity
   on behalf of whom a Contribution has been received by Licensor and
   subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:
```

(continues on next page)

(continued from previous page)

```
    (a) You must give any other recipients of the Work or
        Derivative Works a copy of this License; and

    (b) You must cause any modified files to carry prominent notices
        stating that You changed the files; and

    (c) You must retain, in the Source form of any Derivative Works
        that You distribute, all copyright, patent, trademark, and
        attribution notices from the Source form of the Work,
        excluding those notices that do not pertain to any part of
        the Derivative Works; and

    (d) If the Work includes a "NOTICE" text file as part of its
        distribution, then any Derivative Works that You distribute must
        include a readable copy of the attribution notices contained
        within such NOTICE file, excluding those notices that do not
        pertain to any part of the Derivative Works, in at least one
        of the following places: within a NOTICE text file distributed
        as part of the Derivative Works; within the Source form or
        documentation, if provided along with the Derivative Works; or,
        within a display generated by the Derivative Works, if and
        wherever such third-party notices normally appear. The contents
        of the NOTICE file are for informational purposes only and
        do not modify the License. You may add Your own attribution
        notices within Derivative Works that You distribute, alongside
        or as an addendum to the NOTICE text from the Work, provided
        that such additional attribution notices cannot be construed
        as modifying the License.

    You may add Your own copyright statement to Your modifications and
    may provide additional or different license terms and conditions
    for use, reproduction, or distribution of Your modifications, or
    for any such Derivative Works as a whole, provided Your use,
    reproduction, and distribution of the Work otherwise complies with
    the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
```

(continues on next page)

(continued from previous page)

```
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "{}"
   replaced with your own identifying information. (Don't include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright {yyyy} {name of copyright owner}

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0
```

(continues on next page)

(continued from previous page)

```
  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
```

4.1.8 Contributing

Welcome to `xepor` contributor's guide.

This document focuses on getting any potential contributor familiarized with the development processes, but other kinds of contributions are also appreciated.

If you are new to using git or have never collaborated in a project previously, please have a look at contribution-guide.org. Other resources are also listed in the excellent guide created by FreeCodeCamp[1].

Please notice, all users and contributors are expected to be **open, considerate, reasonable, and respectful**. When in doubt, Python Software Foundation's Code of Conduct is a good reference in terms of behavior guidelines.

Issue Reports

If you experience bugs or general issues with `xepor`, please have a look on the issue tracker. If you don't see anything useful there, please feel free to fire an issue report.

Tip: Please don't forget to include the closed issues in your search. Sometimes a solution was already reported, and the problem is considered **solved**.

New issue reports should include information about your programming environment (e.g., operating system, Python version) and steps to reproduce the problem. Please try also to simplify the reproduction steps to a very minimal example that still illustrates the problem you are facing. By removing other factors, you help us to identify the root cause of the issue.

Documentation Improvements

You can help improve `xepor` docs by making them more readable and coherent, or by adding missing information and correcting mistakes.

`xepor` documentation uses Sphinx as its main documentation compiler. This means that the docs are kept in the same repository as the project code, and that any documentation update is done in the same way was a code contribution. The document files is written in Markdown and parsed by MyST extensions.

Tip: Please notice that the GitHub web interface provides a quick way of propose changes in `xepor`'s files. While this mechanism can be tricky for normal code contributions, it works perfectly fine for contributing to the docs, and can be quite handy.

If you are interested in trying this method out, please navigate to the `docs` folder in the source repository, find which file you would like to propose changes and click in the little pencil icon at the top, to open GitHub's code editor. Once

[1] Even though, these resources focus on open source projects and communities, the general ideas behind collaborating with other developers to collectively create software are general and can be applied to all sorts of environments, including private companies and proprietary code bases.

you finish editing the file, please write a message in the form at the bottom of the page describing which changes have you made and what are the motivations behind them and submit your proposal.

When working on documentation changes in your local machine, you can compile them using `tox`:

```
tox -e docs
```

and use Python's built-in web server for a preview in your web browser (`http://localhost:8000`):

```
python3 -m http.server --directory 'docs/_build/html'
```

If you have problem previewing rst files in VSCode with reStructuredText plugin, run the following command in bash in project root:

```
ls *.rst | xargs -I% bash -c 'n="%"; b="${n::-4}"; ln -s html/${b,,}.html docs/_build/$b.
↪html'
```

Code Contributions

The core of `xepor` framework is mainly implemented in one source file, `xepor.py`. And the main API facing end users are the constructor and `api.route` function, served as a function descriptor. You should start inspecting from there.

Submit an issue

Before you work on any non-trivial code contribution it's best to first create a report in the issue tracker to start a discussion on the subject. This often provides additional considerations and avoids unnecessary work.

Create an environment

Before you start coding, we recommend creating an isolated virtual environment to avoid any problems with your installed Python packages. This can easily be done via either `virtualenv`:

```
virtualenv <PATH TO VENV>
source <PATH TO VENV>/bin/activate
```

or Miniconda:

```
conda create -n xepor python=3 six virtualenv pytest pytest-cov
conda activate xepor
```

Clone the repository

1. Create an user account on GitHub if you do not already have one.
2. Fork the project repository: click on the *Fork* button near the top of the page. This creates a copy of the code under your account on GitHub.
3. Clone this copy to your local disk:

```
git clone git@github.com:YourLogin/xepor.git
cd xepor
```

4. You should run:

```
pip install -U pip setuptools -e .
```

   to be able to import the package under development in the Python REPL.

Implement your changes

1. Create a branch to hold your changes:

```
git checkout -b my-feature
```

   and start making changes. Never work on the main branch!
2. Start your work on this branch. Don't forget to add docstrings to new functions, modules and classes, especially if they are part of public APIs.
3. Add yourself to the list of contributors in `AUTHORS.rst`.
4. When you're done editing, do:

```
git add <MODIFIED FILES>
git commit
```

   to record your changes in git.
5. Please check that your changes don't break any unit tests with:

```
tox
```

   (after having installed `tox` with `pip install tox` or `pipx`).

   You can also use `tox` to run several other pre-configured tasks in the repository. Try `tox -av` to see a list of the available checks.

Submit your contribution

1. If everything works fine, push your local branch to GitHub with:

```
git push -u origin my-feature
```

2. Go to the web page of your fork and click "Create pull request" to send your changes for review.

Find more detailed information in creating a PR. You might also want to open the PR as a draft first and mark it as ready for review after the feedbacks from the continuous integration (CI) system or any required fixes.

Troubleshooting

The following tips can be used when facing problems to build or test the package:

1. Make sure to fetch all the tags from the upstream repository. The command `git describe --abbrev=0 --tags` should return the version you are expecting. If you are trying to run CI scripts in a fork repository, make sure to push all the tags. You can also try to remove all the egg files or the complete egg folder, i.e., `.eggs`, as well as the `*.egg-info` folders in the `src` folder or potentially in the root of your project.

2. Sometimes `tox` misses out when new dependencies are added, especially to `setup.cfg` and `docs/requirements.txt`. If you find any problems with missing dependencies when running a command with `tox`, try to recreate the `tox` environment using the `-r` flag. For example, instead of:

   ```
   tox -e docs
   ```

   Try running:

   ```
   tox -r -e docs
   ```

3. Make sure to have a reliable `tox` installation that uses the correct Python version (e.g., 3.7+). When in doubt you can run:

   ```
   tox --version
   # OR
   which tox
   ```

   If you have trouble and are seeing weird errors upon running `tox`, you can also try to create a dedicated virtual environment with a `tox` binary freshly installed. For example:

   ```
   virtualenv .venv
   source .venv/bin/activate
   .venv/bin/pip install tox
   .venv/bin/tox -e all
   ```

4. Pytest can drop you in an interactive session in the case an error occurs. In order to do that you need to pass a `--pdb` option (for example by running `tox -- -k <NAME OF THE FALLING TEST> --pdb`). You can also setup breakpoints manually instead of using the `--pdb` option.

Maintainer tasks

Releases

If you are part of the group of maintainers and have correct user permissions on PyPI, the following steps can be used to release a new version for `xepor`:

1. Make sure all unit tests are successful.
2. Tag the current commit on the main branch with a release tag, e.g., `v1.2.3`.
3. Push the new tag to the upstream repository, e.g., `git push upstream v1.2.3`
4. Clean up the `dist` and `build` folders with `tox -e clean` (or `rm -rf dist build`) to avoid confusion with old builds and Sphinx docs.
5. Run `tox -e build` and check that the files in `dist` have the correct version (no `.dirty` or git hash) according to the git tag. Also check the sizes of the distributions, if they are too big (e.g., > 500KB), unwanted clutter may have been accidentally included.

6. Run `tox -e publish -- --repository pypi` and check that everything was uploaded to PyPI correctly.

4.2 Indices and tables

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX