How to Make an HTTP Proxy in Python

Learn how to use mitmproxy framework to build HTTP proxies using Python

Redouane Niboucha · 6 min read · Updated Jan 2021 · Ethical Hacking

A network proxy server is an intermediary network service that users can connect to, and that relies their traffic to other servers, proxy servers can be of different types, to list a few, there are:

- **Reverse Proxies**: proxies that hide the address of servers you are trying to connect to, apart from the obvious security use case, they are often used to perform load-balancing tasks, where the reverse proxy decides to which server it should forward the request, and caching. Popular reverse proxies are HAProxy, Nginx and Squid.
- **Transparent proxies**: these are proxies that forward your data to the server, without offering any kind of anonymity, they still change the source IP of the packets with the proxy's IP address. They can be useful for implementing antivirus or internet filtering on entreprise networks, they can also be used to evade simple bans based on the source IP.
- **Anonymous proxies**: these are proxies that hide your identity from the target server, they are mostly used for anonymity.

- **HTTP Proxies**: The HTTP protocol supports proxy servers, the `CONNECT` method is used to ask the proxy server to establish a tunnel with a remote server.
- **Socks Proxies**: The Socks protocol, which uses Kerberos for authentication, is also widely used for proxies.

Related: *How to Use Proxies to Rotate IP Addresses in Python.*

Mitmproxy is a modern, open source HTTP/HTTPS proxy, it offers a wide range of features, a command line utility, a web interface, and a Python API for scripting. In this tutorial, we will use it to implement a proxy that adds HTML and Javascript code to specific websites we visit, we also make it work with both HTTP and HTTPS.

First, we need to install `mitmproxy`, it can be easily done with the following command on Debian-based systems:

```
$ sudo apt install mitmproxy
```

Although it's highly suggested you follow along with a Linux machine, you can also install `mitmproxy` on Windows in the official mitmproxy website.

For this tutorial, we will write a simple proxy that adds an overlay to some pages we visit, preventing the user from clicking anything on the page by adding an overlay HTML code to the HTTP response.

Below is the code for the proxy:

Tags

Machine Learning
Ethical Hacking
General Python Topics
Web Scraping
Computer Vision
Python Standard Library
Application Programming Interfaces
Database
Finance
Packet Manipulation Using Scapy
Natural Language Processing
Healthcare

```
OVERLAY_HTML = b"<img style='z-
index:10000;width:100%;height:100%;top:0;left:0;position:fixed;opaci
 src='https://cdn.winknews.com/wp-content/uploads/2019/01/Police-
lights.-Photo-via-CBS-News..jpg' />"
OVERLAY_JS = b"<script>alert('You can\'t click anything on this
page');</script>"

def remove_header(response, header_name):
    if header_name in response.headers:
        del response.headers[header_name]


def response(flow):
    # remove security headers in case they're present
    remove_header(flow.response, "Content-Security-Policy")
    remove_header(flow.response, "Strict-Transport-Security")
    # if content-type type isn't available, ignore
    if "content-type" not in flow.response.headers:
        return
    # if it's HTML & response code is 200 OK, then inject the
overlay snippet (HTML & JS)
    if "text/html" in flow.response.headers["content-type"] and
flow.response.status_code == 200:
        flow.response.content += OVERLAY_HTML
        flow.response.content += OVERLAY_JS
```

The script checks if the response contains HTML data, and the response code is `200 OK`, if that's the case, it adds the HTML and Javascript code to the page.

Content Security Policy (CSP) is a header that instructs the browser to only load scripts from specific origins, we remove it to be able to inject inline scripts, or to load scripts from different sources.

The HTTP Strict Transport Security (HSTS) header tells the browser to only connect to this website via HTTPS in the future, if the browser gets this header, no man-in-the-middle will be possible when it will be acessing this website, until the HSTS rule expires.

We save the above script under the name proxy.py and execute it via mitmproxy command:

```
$ mitmproxy --ignore '^(?!duckduckgo\.com)' -s proxy.py
```

The `--ignore` flag tells mitmproxy to not proxy any domains other than `duckduckgo.com` (otherwise, when fetching any cross-domain resource, the certificate will be invalid, and this might break the webpage), the

Now the proxy listens on the address `localhost:8080`, we must tell our browser to use it, or redirect traffic to it transparently using an `iptables` tool in Linux.

In Firefox browser, it can be done from the network settings:

But if we want to make the proxy work for every application in our system, we will have to use `iptables` (in case of Linux system) to redirect all TCP traffic to our proxy:

```
$ iptables -t nat -A OUTPUT -p tcp --match multiport --dports
80,443 -j REDIRECT --to-ports 8080
```

Now go to your browser and visit `duckduckgo.com`. Of course, if the website is using HTTPS (and it is), we will get a certificate warning, because mitmproxy generates its own certificate to be able to modify the HTML code:

But if the site is not already preloaded in the HSTS preload list (if it's the case, the browser will not allow bypassing the warning), we can proceed and visit the page:

As you can see, we will not be able to do anything in the website, as the injected HTML overlay is preventing us, you can also check if your proxy is indeed working by running the following `curl` command:

```
$ curl -x http://127.0.0.1:8080/ -k https://duckduckgo.com/
```

If it's working properly, you'll see the injected code in the end as shown in the following image:

Conclusion

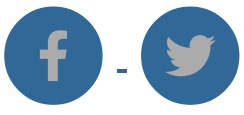
Note that the script can be used in the different proxy modes that mitmproxy supports, including regular, transparent, socks5, reverse and upstream proxying.

Mitmproxy is not limited of being an HTTP proxy, we can also proxy websocket data, or even raw TCP data in a very similar way.

Check the full code here.

Learn also: *How to Use Proxies to Rotate IP Addresses in Python.*

Happy Hacking ♥



Read Also

How to Use Proxies to Rotate IP Addresses in Python

Learn how to perform web scraping at scale by preventing websites to ban your ip address while scraping them using different proxy methods in Python.

VISIT →

Logging in Python

Learn how to use Python's built-in logging module to log your Python applications, changing default format, level, and learning in a concrete example, as well as using logging handlers.

VISIT →

How to Make a Chat Application in Python

Learn how to build a simple chat room application that accepts multiple

VISIT →

Follow @ThePythonCode

Comment panel

Your email address will not be published.

Enter your name

Enter Email

Write a comment...

Subscribe for our newsletter

COMMENT

Subscribe

Newsletter

JOIN OUR NEWSLETTER THAT IS FOR PYTHON DEVELOPERS & ENTHUSIASTS LIKE YOU !

Enter your Email Address

GET PYTHON TUTORIALS →