

Forum:[Hack Forums](#)

Thread: [Twitch Email Verifier using Gmail dot trick](#)

obby - March 2, 2017, 5:45 pm

The thing doing the actual requests is a slightly modified

<https://pawelhmh.github.io/asyncio/python/aiohttp/2016/04/22/asyncio-aiohttp.html>

NOTE: Requests for verification can't be done from a twitch banned ip, which is why the aiosocks stuff is there, I just removed the code from the ClientSession to actually use the SOCKS tho.

IMAP stuff pretty self explanatory, hope someone that writes botting stuff finds it useful.

email_verify.py

```
Code import imaplib
import verifier
```

```
from time import sleep
```

```
verificationUrls = []
```

```
mail = imaplib.IMAP4_SSL('imap.gmail.com')
mail.login('youremail', 'appspecificpassword')
mail.list()
mail.select("inbox")
```

```
result, data = mail.uid('search', None, '(HEADER Subject "Verify your Twitch")')
iterData = data[0].decode('utf-8').split(' ')
r, d = mail.uid('fetch', ','.join(iterData), '(RFC822)')
```

```
for b in d:
    try:
        b = b[1].decode('utf-8')
    except:
        continue
    verificationUrls.append(str(b).split('href="')[2].split('"')[0])
```

```
print('[+] Got {0} verification URLs'.format(len(verificationUrls)))
```

```
emailVerifier = verifier.task(verificationUrls)
verifier.py
```

```
Code import random
import asyncio
import aiosocks
from aiohttp import ClientSession
from aiosocks.connector import (
    SocksConnector, HttpProxyAddr, HttpProxyAuth
)
```

```
async def fetch(url, session):
    async with session.get(url) as response:
        return await response.read()
```

```
async def bound_fetch(sem, url, session):
    async with sem:
        await fetch(url, session)
```

```
async def run(urls):
    tasks = []
    sem = asyncio.Semaphore(1000)
    #addr = aiosocks.Socks5Addr('127.0.0.1', 43594)
    #auth = None
    #conn = SocksConnector(proxy=addr, proxy_auth=auth, remote_resolve=True)
    async with ClientSession() as session:
        for url in urls:
            task = asyncio.ensure_future(bound_fetch(sem, url, session))
            tasks.append(task)
        responses = asyncio.gather(*tasks)
        await responses
```

```
def task(urls):
    loop = asyncio.get_event_loop()
    future = asyncio.ensure_future(run(urls))
    loop.run_until_complete(future)
```

Davian - March 8, 2017, 6:36 am

Very useful.

thank you dude.