April 9, 2019

A Pentester's Guide - Part 2 (OSINT - LinkedIn is not just for jobs)

Howdy Haxors! That's right, I am back with another banger of an article. Ready? If you read my last article on OSINT, you'll know a little bit on how to find dirt on companies, their assets, and how to do all of that passively. Well now I am going to show you a little bit about researching individuals, and specifically those who work at a specific company.

If you read the title, which I am sure you have, you'll know that we're going to utilize LinkedIn for this. LinkedIn is a very useful tool for enumerating users, and their emails.

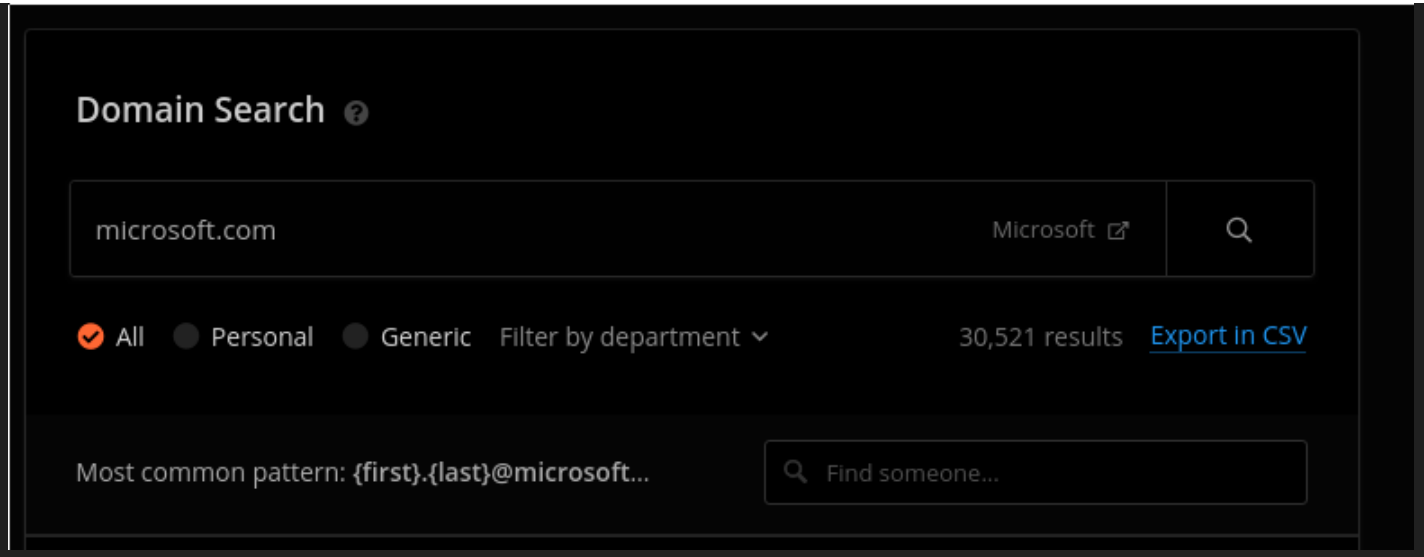
'Generating' Email Addresses

From my last article, I mentioned a tool called hunter.io, you can use it to search for company email addresses, and it even gives you a break down on what it thinks the current email structure is for a company.

Stage 1 - Figure Out the Naming Conventions

Let's use google.com as an example. Now let's assume we found a few emails with the format of hsimpson@google.com, and then we see another email, jsmith@google.com. We can soon start to get a feel for the kind of structure they're using for emails, typically, {firstinitial}{lastname}@{companyname}.com, this could vary by organization, but we can quickly come up with the naming convention, and then we can move onto stage 2.

You can do this with hunter.io. Notice the 'most common pattern' section?



Stage 2 - Get the Employees and Generate the Emails

This is where LinkedIn plays nicely with us. If we can get a list of first and last names of everybody working at a company, we can then take that information and feed it into a python script and generate email addresses from the convention. We can easily obtain these names through google dorking. Try my new tool GoogLinked, when coupled with ProxyDock (another shameless plug), you can scrape google all day with impunity.

```
~/Doc…nts/Pro…ing/Ruby/GoogLinked  master  ./GoogLinked.rb > emails.txt
_
```

One thing you'll want to do before you run this is modify the script. Now it contains Microsoft and microsoft.com as the arguments, you'll want to modify it to be "company name", (as shown on LinkedIn), and the company email domain name. In the above example, I piped all results into a file, otherwise it will just dump them onto the standard output.

```
~/Doc…nts/Pro…ing/Ruby/GoogLinked  master ?  cat emails.txt | wc -l
1490
~/Doc…nts/Pro…ing/Ruby/GoogLinked  master ?  _
```

Wow, that's a lot of email addresses, let's look and see what it came back with.

```
siya.yang@microsoft.com
brittany.zajic@microsoft.com
vahe.a.@microsoft.com
mark.mechelse@microsoft.com
tianxiao.she@microsoft.com
brent.growe@microsoft.com
mike.ramirez@microsoft.com
garrett.macleod@microsoft.com
pamela.munoz@microsoft.com
colleen.bates@microsoft.com
blanca.peña@microsoft.com
jen.gottlieb@microsoft.com
sheila.mennis@microsoft.com
byron.forte@microsoft.com
rajesh.shah,@microsoft.com
shiv.singh@microsoft.com
dwight.phd@microsoft.com
bradley.gilb@microsoft.com
charles.l.@microsoft.com
carrie.follas@microsoft.com
ron.cromer@microsoft.com
adam.halbridge@microsoft.com
sarah.rivera@microsoft.com
brian.michitti@microsoft.com
erica.layton@microsoft.com
alvin.johnson@microsoft.com
katya.kostyukova@microsoft.com
"emails.txt" 1490L, 40968C
```

So, as you can see, there are a few generated email addresses. What this has done, under the hood, is using a Google Dork, captured the titles for each person that has Microsoft listed under their portfolio as working or have worked. Then, it takes the format defined in the run_search() function.

```ruby
def run_search()
    init = search(@companyname, 0)
    pages = get_pages(init)
    results = ""
    names = []

    (0..pages).each do |pagen|
        # puts "Seaching #{pagen}"
        results += search(@companyname, pagen)

        Nokogiri::HTML(results).css(".r").css("a").each do |name|
            names.push(name.text)
        end
    end

    emails = []

    names.each do |title|
        tarr = title.split("-")[0].split("|")[0].split(" ")
        emails.push(tarr[0].downcase+"."+tarr[-1].downcase+"@"+@domain)
    end

    puts emails
```

Notice that names.each section? Modify that depending on the format you discovered for the company (as we learned how to do in the section before). In this case tarr[0] and tarr[-1] are firstname and lastname respectively. *Note, things can get a bit whacky when middle names are introduced. So just know this is not a fool proof method.*

This script is one I use personally, and so I modify it myself. If you want to make a PR to add custom functionality, feel free. Now that you have these emails, the world is your oyster. Pass them into weleaksinfo.com? (See my last article).

Bonus: Password Spraying

If you're doing a pentest, you can use these emails to your advantage. Quite often, brute-forcing a company's email with a single email and 100k passwords is your way to getting blocked by a SIEM, or even getting the account locked which impacts on production (if you're a pentester you'll know the struggle :P), which we don't want.

One very unique way to acheive this, as taught to me by atoscher, is to try each email with a single password, such as Summer2018, or Fall2018. Many companies implement quarterly password rotation policies, in the hope to make passwords more unpredictable and secure. What ends up happening is people resort to using weaker passwords as they can't remember a new password every quarter. A very common usage of this is people using passwords like Summer2018.

What else is cool about this method, is you can usually feed it through a load of VPN's, (*cough* ProxyDock *cough*), and then try legitimate services like Gmail, Microsoft lync, SharePoint, the list can go on. Your chances of getting blocked, especially if you regularly rotate IP addresses, user agents, and individuals, is very low.

That's all for this week my fellow haxors! Stay tuned, next week I am going to bring you "Active Recon - Deeper than Nmap", the name might change, we'll see. Make sure you drop a comment to provide feedback, 'like' the article if you find it useful, and please share everywhere. The more more people we educate, the better. As always, stay snappy!

Benjamin Bidmead - Offensive Security Engineer at NaviSec Delta

AUTHOR
NaviSec Delta
Read more posts by this author.