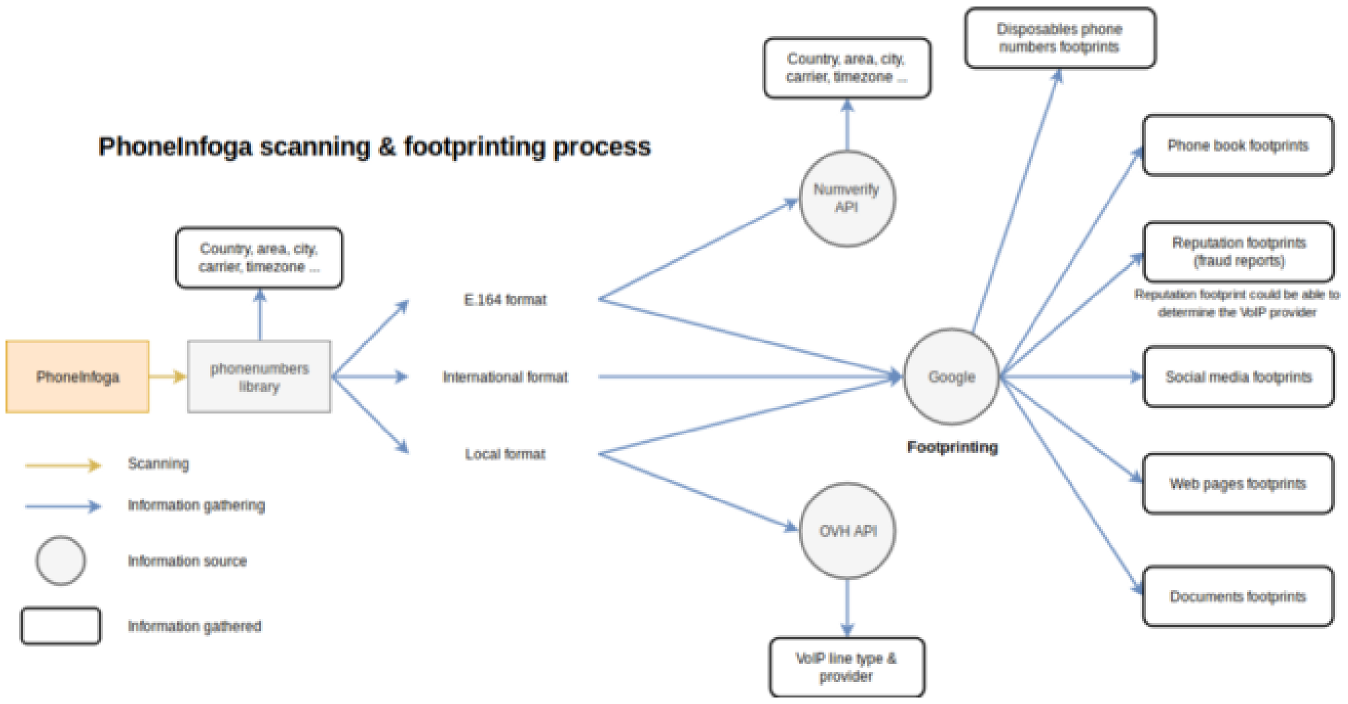

PhoneInfoga scanning & footprinting process
PhoneInfoga
phonenumbers library
Country, area, city, carrier, timezone ...
E.164 format
International format
Local format
Numverify API
Country, area, city, carrier, timezone ...
OVH API
VoIP line type & provider
Google
Footprinting
Disposables phone numbers footprints
Phone book footprints
Reputation footprints (fraud reports)
Reputation footprint could be able to determine the VoIP provider
Social media footprints
Web pages footprints
Documents footprints
Scanning
Information gathering
Information source
Information gathered

Building an OSINT Reconnaissance Tool from Scratch

Raphaël Follow
Dec 9, 2018 · 9 min read

Everyone has a phone, using at least one phone number. Phone numbers are a very common resource for Social Engineering. It’s something we use almost every day to communicate and sometimes we may have to deal with unsolicited phone calls or messages. We may have to gather information about a phone number we found about a company or an individual. Basic information such as line type and carrier can be very useful for a Social Engineer.

Supposing I know your name and your phone number, I could send you a phishing threat using your carrier’s mail template. Or I may call your carrier’s support service to gather as much information as I can on you. Another example, if the number is a land line, some of the digits will tell me the area where it comes from. These information are very simple to get without using a tool, but what about going further ?

The goal is to gather as much information as possible on the given phone number, including the ITSP or the owner.

> *An Internet telephony service provider (ITSP) offers digital telecommunications services based on Voice over Internet Protocol (VoIP) that*

> *are provisioned via the Internet. Wikipedia*

Getting technical

First I have to understand the composition of a phone number and how to handle it. A phone number has different formats :

- E.164: +3396360XXXX
- International: +33 9 63 60 XX XX
- National: 09 63 60 XX XX
- RFC3966: tel:+33–9–63–60-XX-XX
- Out-of-country format from US: 011 33 9 63 60 XX XX

E.164 formatting for phone numbers entails the following:

- A + (plus) sign
- International Country Calling code
- Local Area code

- Local Phone number

For example, here’s a US-based number in standard local formatting: (415) 555–2671

© Twilio

Here’s the same phone number in E.164 formatting: +14155552671

© Twilio

In the UK, and many other countries internationally, local dialing may require the addition of a ‘0’ in front of the subscriber number. With E.164

formatting, this ‘0’ must usually be removed.

Another example, here’s a UK-based number in standard local formatting: 020 7183 8750

© Twilio

Here’s the same phone number in E.164 formatting: +442071838750

The country code is essential. Without it, I can’t scan the phone number and determine the country. So the tool will only support E.164 and International formats.

But wait.. what if there was a library to automatically parse information from the number ? Meanwhile searching for resources about phone numbers, I found this magical Google’s repository which is a Java, C++ and JavaScript library for parsing, formatting, and validating international phone numbers. The library also exists in Go, PHP, Ruby, Rust and Python. Hooray! I don't have to do all the job by myself.

To identify basics information, I select some lookup sites I can use for free, even if I have to use an "hack" to use them. Because some websites allow a reverse search for free using the web page but requires an API key as soon as you want to use their API. For example, I can trick the Ajax call to make an API call in my tool. I want my tool to be usable without any API registration.

Identify the carrier is quit simple because each carrier has number ranges. For example, if we know the number +33679368314 is from Orange (french carrier), it's easy to understand that +3367936XXXX number range is owned by Orange as well. Google, like a lot of other services, has a huge database of these number ranges associated to their carriers. However, in some case, people change carrier but keep their phone number so the information about the number range becomes invalid.

Using Open Source Intelligence & open data

On my way learning about security, I discovered months ago Open Source Intelligence (OSINT). OSINT is the collection of information from publicly available and open data sources to be used in an intelligence context.

> *In the intelligence community, the term "open" refers to overt, publicly available sources (as opposed to covert or clandestine sources). It is not related to open-source software or public intelligence. -Wikipedia*

Open Source Intelligence (OSINT) takes three forms: Passive, Semi-passive, and Active. There are several way to deal with information in an intelligence context. Especially when it comes to footprinting. I'm gonna practice **Passive Information Gathering** (or Passive Reconnaissance), it means I will not store data gathered and mostly use third party sources. But I'll gather information from many sources and filter the results to find the owner or the ITSP.

Learn more about OSINT reconnaissance techniques and footprinting here.

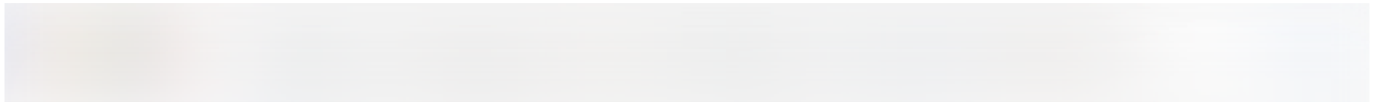
. . .

First of all, I want my OSINT tool to check for :

- Phone number reputation (phone fraud reports)
- Footprints on VoIP and temporary number providers websites
- Social media pages (facebook, twitter, linkedin, instagram) and phone books results

Most of my resources are from this open source OSINT framework and IntelTechniques.

To find documents and web pages related to the phone number, I use Google Dork requests. I make a list of every disposable number providers I found. Some of them exposes their numbers and all of this is indexed by search engines. For example if I ask Google for a web page on one of these sites with the number included in content and found a result, this usually means the number is part of their number range. As my OSINT reconnaissance is pretty basic, I'll never consider a result as a success.



Footprint recon using a Google Hacking request targeting a "Phone Fraud" website

Listing phone number fraud sources

- scamcallfighters.com
- signal-arnaques.com

Listing disposable number providers

- receive-sms-online.com
- receive-sms-now.com
- hs3x.com

- **twilio.com**
- freesmsverification.com
- freeonlinephone.org
- sms-receive.net
- **smsreceivefree.com**
- receive-a-sms.com
- receivefreesms.com
- freephonenum.com
- receive-smss.com
- receivetxt.com
- **temp-mails.com**
- receive-sms.com
- receivesmsonline.net
- receivefreesms.com
- sms-receive.net

- **pinger.com (=> textnow.com)**
- receive-a-sms.com
- **k7.net**
- **kall8.com**
- **faxaway.com**
- receivesmsonline.com
- receive-sms-online.info
- sellaite.com
- getfreesmsnumber.com
- smsreceiving.com
- smstibo.com
- catchsms.com
- freesmscode.com
- smsreceiveonline.com
- smslisten.com

- sms.sellaite.com

Bold ones means they don’t put their phone numbers on their site (so no footprints) or seem to not exist anymore.

Yeah. There’s a lot of disposable number providers. Using Google search engine, I can search for footprints specifying the domain name and phone number using different formats.

> site:”domain” intext:”international_num” | “local_num”

There are plenty of open data sources online that could’ve been very useful for this project. While searching for data sources, I found a JSON database of country associated to their ISO & dial codes. Next I found a database of area codes for every country. Unfortunately, this will not be used since *phonenumbers* Google’s library already check for these information as well as the carrier and the location.

ITSPs (VoiP number providers) list — via http://directory.didww.com/top-providers

Information gathering on individuals

However, find information about individuals is a bit easier than expected. There are plenty of open data sources (with API or just reverse search feature) that allows me to search about a phone number. According to Intel Techniques, this is the average workflow using phone number search tools :

© IntelTechniques

List of owner information gathering open sources

- Facebook
- True People
- Fast People
- Background Check
- Pipl
- Spytox
- Makelia
- IvyCall
- PhoneSearch
- 411
- USPhone
- WP Plus
- Thats Them
- True Caller

- Sync.me
- WhoCallsMe
- ZabaSearch
- DexKnows
- WeLeakInfo
- OK Caller
- SearchBug
- numinfo.net

One day someone I didn’t know sent me an email. I was curious so I typed his email address on Google. I discovered that a profile on numinfo.net was associated to his email address and from here, I was able to know his phone number, location (including country, state and city) his name and even his physical address! I didn’t know if information were correct or not, but still. To verify, I asked him few questions about his location and guess what… everything matched.

The problems

At this point we are able to gather a lot of information about almost every phone number in the world and that's amazing! But I'm facing two major problems while building my tool.

First, Google blacklist the client IP and ask you to complete a captcha after a few amount of requests. When you practice complex custom requests, Google ask you to complete the captcha. After some investigations, I discover how Google handle captcha and bot detection.

When you search on Google using custom requests (Google Dorks), you get very easily blacklisted. So Google shows up a page where you have to complete a captcha to continue. As soon as the captcha is completed, Google create a cookie named "**GOOGLE_ABUSE_EXEMPTION**" which is used to whitelist your browser and IP address for some minutes. This temporary whitelist is enough to let you gather a lot of information from many sources. So I decided to add a simple user manipulation to bypass this bot detection. First I make a user-agent list to make the request as random as possible. I'll not use proxies since Google blacklists free proxies instantly. I'll use the exact same headers as a normal user that uses a browser and, of course, that

“GOOGLE_ABUSE_EXEMPTION” cookie. Of course, I can’t generate a new token as Google generate it by time and IP address. So I’ll just try make requests and wait until I get a 503 error, which means I got blacklisted. Then I ask the user to follow an URL to manually complete the captcha and copy the whitelist token to paste it in the CLI. The tool is now able to continue to scan! At this point this is not really a bypass, but rather a workaround.

Second, ITSPs doesn’t share their phone number ranges. At the moment this is way too difficult to gather phone number of every single providers in the world, or maybe this is just another level of OSINT. Also telephone numbers change too often. It’s very hard to find the ITSPs using only footprinting. But I noticed that some lookup websites (such as 411.com) were able to recover the ITSP that owns the number.

Making the tool ”smart” enough

At the moment my tool just scan everything using every sources. But since we know the country and the line type at the beginning of the scan, we should not use sources that provides results for another country. For example if I gather information about a french number (+33), it's useless to search for results in a US fraud phone number database, isn't it ? Also my OSINT reconnaissance must not return useless results such as Anonymous phone books that have several phone number footprints and choose wisely which one I'll return to the end user. So the final thing to do will be to make a quick refactor to limit the amount of API calls and only use useful sources.

Finally, I have to let the end user have more control on the results. Sometimes the phone number has footprints but is used with a different format. This is a problem because, for example if we search for *"+15417543010"*, we'll not find web pages that write it that way : *"(541) 754–3010"*. So the tool will use a (optional) custom formatting given by the user to find further and more accurate results.

Let's code!

I choose to go with Python. Using some libraries such as *phonenumbers* to parse information from phone numbers, *requests* to make API calls, *argparse* to parse cli parameters and other libraries related to formatting. The code is available on this GitHub repository.

. . .

Conclusion

OSINT is a very effective way to gather information and investigate. Especially using open sources such as search engines, which has a huge amount of indexed pages. Remember, I talked about the most basic part of OSINT, which is footprinting (or passive reconnaissance). But we could do way more using it. Also, remember that the goal of OSINT if not to automate everything but automate the information gathering. This tool might not be able to recover your name or address by scanning your number (fortunately), anyway, this is not the purpose. The goal is to provide an easy way to investigate and parse information from any phone number.

We only used free resources, if you want to go further into OSINT, keep in mind there is a lot of premium resources that probably already does the job for you. They usually have very valuable informations, but you'll have to pay for it.

Python Osint Infosec Phone Number Programming

189 claps

WRITTEN BY

Raphaël

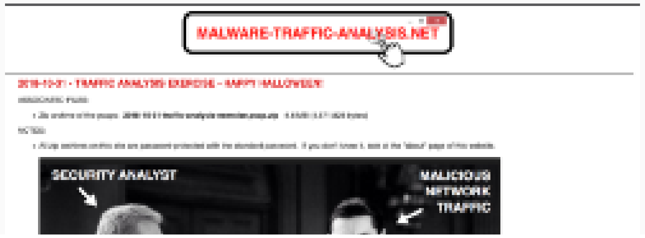
Follow

Programming, security & OSINT enthusiast. #opensource #security #infosec #osint - PGP: B64687AB97F268F43E67B97A8916203E540C65A4

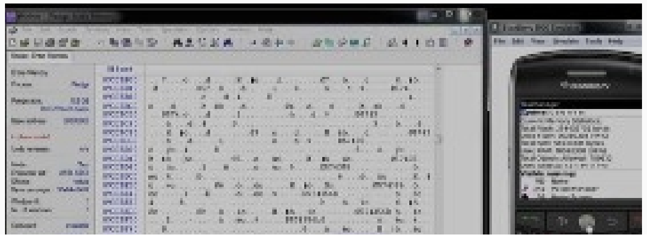
See responses (1)

More From Medium

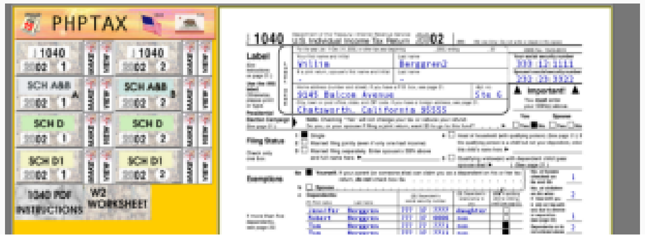
Related reads



Related reads



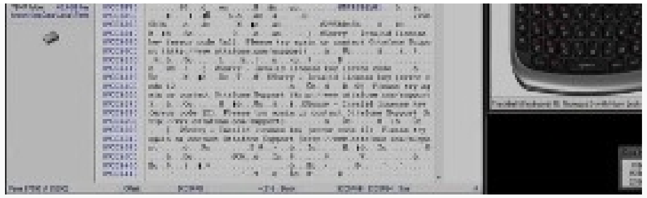
Related reads



Halloween Malware Traffic Analysis Exercise

Michael R.
Nov 2, 2018 · 6 min read

48



Mobile Device Digital Forensics

Ian Barwise
Nov 26, 2018 · 19 min...

125

VulnHub — Kioptrix: Level 5

Mike Bond
Oct 21, 2018 · 13 min...

336