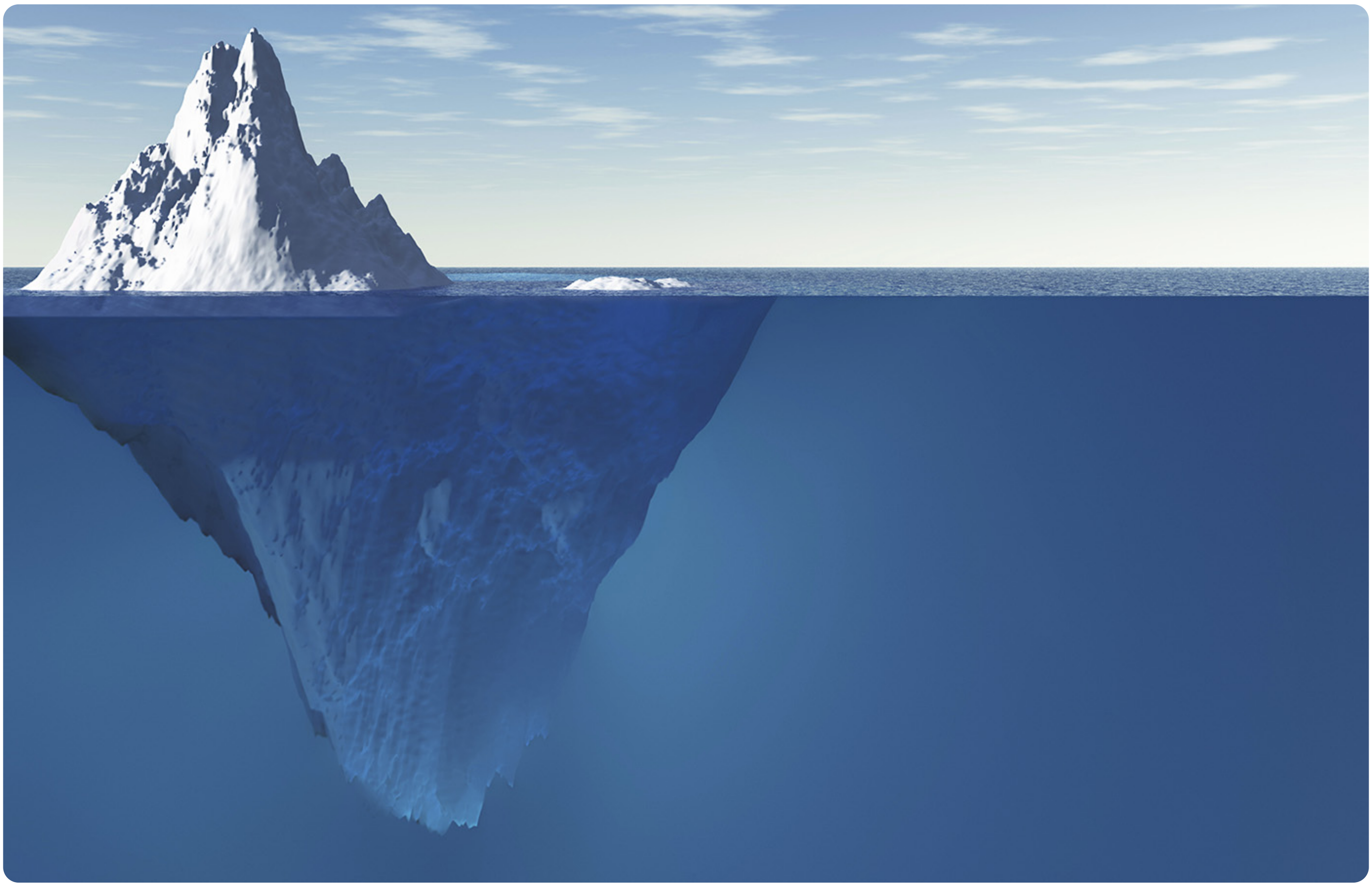
WhoisXMLAPI

Amplify a blacklist with the Typosquatting Data Feed. A technical blog

Posted on January 31, 2020



The Typosquatting Data Feed list groups of domains that have been registered on the same day, and whose names are similar to each other within the group. A question might be: why buy such data. Here we illustrate the power of the data set through a very efficient application to detect malicious domains. A simple Python code will be presented to illustrate how it works. Then we will illustrate its efficiency by applying it to the PhishTank data feed, demonstrating that it is capable of revealing a tremendous amount of additional domains.

Detection of malicious domains is an important and hard task in IT security. It is the major ingredient of protection against phishing, malware, botnet activity, etc. The most reliable approach to the problem is the use of blacklists such as PhishTank or URLhaus, where a community or a specialized group of experts publish a list of domains or URLs that are confirmed to be malicious. PhishTank, for instance, is community operated: a number of benevolent activists do a great favor to all of us by checking suspicious domains and reveal their phishing activity.

A blacklist of domains is not only useful for direct use in firewalls or spam filters though. It can also serve as an input for methods that can find additional domains strongly related to the blacklisted ones, thus being suspicious. By "amplification" of a blacklist we mean its extension with such a method. With WhoisXML API's recently introduced Typosquatting Data Feed such an amplification can be easily achieved. Some of the domains in the original blacklist will turn out to be the "top of the iceberg": we shall find a relevant set of related domains.

1. Principles

The key idea is the following. Take a domain from a blacklist. It is a habit of miscreants, observed by many investigations, that they tend to register multiple domains in bulk to achieve their bad goals. These groups are frequently registered in bursts, i.e., on the same day. In addition, they are often similar to each other. And these are the very domain groups the typosquatting feed reports on a daily basis. Hence, if we find the domain in such a group, the other members of the group are almost surely related to the confirmedly malicious one, thereby they need attention; typically the best action is to blacklist all of them. Now we demonstrate how simple it is do it in practice.

2. Implementation

We shall use Python as the programming environment for the goal, as it is commonly used and available on virtually any platform. Further, we use the pandas data analysis library for Python to easily read the data files and process them. So our code starts like this:

```
#!/usr/bin/env python3

import sys
import os
import datetime
import re
import pandas as pd
```

A key ingredient is the data from the Typosquatting Data Feed. Using your subscription, we download those files to a local directory, say, "typosquatting_feed", in the working directory of the script. We define this location as a variable, so we write

```
TYPOSQUATTING_DIRECTORY = "./typosquatting_feed"
```

(We use a Linux system when writing the present blog, and the directory is below the working directory of the script. Hence the ./ before the directory name.) The directory has the csv files, named according to the feed's convention, e.g., groups_2020_01_04.csv has the data of the feed provided on 4 January 2020. In our example we will have these files from January 2020, but the more files one has in the directory, the more efficient the algorithm will be, as it will have a broader time range to look for the additional domains. With your subscription you can use any download utility like "wget" to download these data in bulk.

Next we read all these data into a pandas DataFrame, a spreadsheet-like representation of the data in the program. This we do by the following code:

```
typosquatting_data = pd.DataFrame(columns = ('groupno',
        'dno','nmembers',
      'domain', 'date'))
directory = os.fsencode(TYPOSQUATTING_DIRECTORY)
for file in os.listdir(directory):
     filename = os.fsdecode(file)
     if filename.endswith(".csv"):
  this_df = pd.read_csv(TYPOSQUATTING_DIRECTORY + \
          os.sep + \
          filename, names=('groupno',
      'dno',
      'nmembers',
      'domain'))
  the_date = datetime.datetime.strptime(re.search(r'[0-9]..._[0-9]._[0-9].', filename).group(), '%Y_%m_
  this_df['date'] = the_date
  typosquatting_data = typosquatting_data.append(this_df)
     else:
  continue
```

What it does is that it loops through all the files in the data feed directory ending with ".csv", reads the fields of the lines: the ordinal number of the group, the number of the domain within the group, the total number of domains in the group, and the domain name itself, into the DataFrame. Then it finds out the date from the file name (which we will need for reasons to be detailed later), and adds a "date" column with this value to the DataFrame. During the loop, all these DataFrames are merged into a single one, named typosquatting_data.

Next we read our input file, specified as the first command-line argument:

```
blacklist_data = pd.read_csv(sys.argv[1], names = ('domain',))
```

Here we assume that the file has a single column, one domain name in each line. The resulting DataFrame has thus a single domain name column.

We have all the input data at hand now, so we need to identify all domains which are there in both DataFrames. This is the key step, and it is as simple as this:

```
common_data = set(blacklist_data['domain']).intersection(
    set(typosquatting_data['domain']))
```

That is, we form the set of the domains occurring in both DataFrames, and take their intersection. So now we have a set of domain names which are on the initial blacklist on one hand, and they are there in the typosquatting feed, too. This means, that each of these domains belongs to a set of similarly-named ones registered on the same day. Hence, those other group members are probably not unrelated to the malicious one; these are what we are looking for. So what remains is to go through the intersection set, and list all the other elements of the group they belong to:

```
found = []
for domain in list(common_data):
    group_element = typosquatting_data[typosquatting_data['domain'] == domain]
    this_element = group_element[['date', 'groupno']]
    this_group = typosquatting_data[(
 typosquatting_data['date'] == this_element['date'].iloc[0]) & (
     typosquatting_data['groupno'] == this_element['groupno'].iloc[0])]
    found_domains = list(this_group['domain'])
    found += found_domains
    k=0
    for found_domain in found_domains:
 hadit = False
 if found_domain in common_data:
     hadit = True
 print("%d\t%s\t %s"%(k, found_domain, hadit))
 k += 1
print("Totals:")
print("Number of domains listed in both data sets:%d"%len(common_data))
print("Total number of related domains :%d"%len(set(found)))
```

Note that a group is identified by its ordinal number within the group but it is unique for a single day only. Hence, we use the date and the ordinal number to identify the group. Also note that this code does not ensure that we do not list the same group twice; but as it is just a demonstration for this blog, we do not elaborate on it to maintain simplicity. At the end of the code we provide a report on how many domains we found in the list, and how many domains we found altogether. The difference of these two numbers will tell us how many domains we have found in addition, which were not there on the blacklist but they are registered in the same burst. Now let us see our script in action. If you want to try it yourself, just copy-paste it to have the code at hand.

3. Examples

Let's pick PhishTank's list of active and confirmed phishing sites on January 12, 2020, at 21:00 GMT. As it is a list of URLs, we take the second-level domains out of it. So a part of our input file, phishtank_list.txt will look like this:

```
active.by
activehosted.com
act-secure.com
actualiza-banruralgt.com
actualplataforma.com
acumbamail.com
addergytech.com
```

While this focus on second-level domains might imply an inaccuracy, as the feed itself lists (dominantly second level) domains, and the criminal behavior it can reveal is in fact related to those. So let's run our code:

```
    ./amplify_blacklist.py phishtank_list.txt
```

which will result in the following output:

```
    0 id180724.xyz  False
    1 id180726.xyz  False
    2 id180725.xyz  True
    3 id180727.xyz  False
    4 id180721.xyz  False
    5 id180729.xyz  False
    6 id180722.xyz  False
    7 id180720.xyz  False
    8 id180728.xyz  False
    9 id180723.xyz  False
    0 pay-allegrop.xyz  True
    1 pay-allegroo.xyz  False
    2 pay-allegro.xyz  False
    0 id180713.xyz  False
    1 id180717.xyz  False
    2 id180715.xyz  False
    3 id180710.xyz  False
    4 id180712.xyz  False
    5 id180714.xyz  False
    6 id180719.xyz  False
    7 id180716.xyz  False
    8 id180711.xyz  True
    9 id180718.xyz  False
```

```
0 moon-oreoo-15.com  False
1 moon-oreoo-14.com  False
2 moon-oreoo-61.com  False
3 moon-oreoo-43.com  False

...

98 moon-oreoo-65.com  False
99 moon-oreoo-23.com  False
100 moon-oreoo-10.com  False
0 uk-item-392271283796.com  True
1 uk-item-392271283796.info  False
2 de-item-392271283796.com  False
3 de-item-392271283796.net  False
4 de-item-392271283796.info  False
Totals:
Number of domains listed in both data sets:5
Total number of related domains :129
```

(From the biggest group we have omitted here a number of domains all they have the same format though.) The labels at the end of the line show if the given domain was there in the original list. Hence, all the domains with "False" are new discoveries. Note, that from our initial seed of 5 domains in the blacklist on the given day, we ended up with a list of 129 domains, thus we have found 124 domains in addition to the original list. We did it on the basis of the typosquatting feed data of a single month. And indeed, looking at the name of the domains, it is quite plausible that they are not completely innocent…

4. Concluding remarks

We have demonstrated that the Typosquatting Data Feed can indeed be used efficiently for amplifying a blacklist in the sense of discovering additional domains related to malicious ones. In case of PhishTank data, we have carried out the following experiment.

We have made a snapshot of PhishTank's list of online and confirmed phishing URLs during January, picking their data at 21:00 GMT every day. After taking the second-level domains we had a list of 100,709 TLDs. From these there were 72 which were members of any group in the typosquatting feed in January. The amplification resulted in 1852 domains. Thus 72 confirmed phishing domains appear to imply that 1780 additional domains are also related to that phishing activity. As these were not on the original blacklist, possibly because they were not yet involved in the phishing activity. Their detection by the described procedure can prevent a successful fraud in advance, without collecting any victims.

The described technique is very simple, the code runs fast and efficient, and could be implemented very easily in many other programming environments. It can be applied to any blacklist, including the one collected by your own Security Operation Center. Got interested? Visit the webpage of the product to find out more details.