uc3m | Universidad Carlos III de Madrid

Master Degree in CyberSecurity
Academic Year 2018-2019

***Master Thesis***

# Development of novel algorithms for fraud detection in online advertising

Olaya García Fernández

Directors
Rubén Cuevas Rumín
Antonio Pastor Valles

Madrid, September 14, 2019

# Development of novel algorithms for fraud detection in online advertising

**Olaya García Fernández <ayalo83@gmail.com>**

Directors : Ruben Cuevas Rumín <rcuevas@it.uc3m.es>, Antonio Pastor <anpastor@it.uc3m.es>

September 14, 2019

**Ad fraud is one of the main problems for the Online Advertising Industry. Its annual cost is estimated to be in the tens of billions of dollars and affects well-known companies such as Google and Facebook.**

**Malicious websites have become one of the main threats regarding Cybersecurity. Most of them, with non-legitimate content, attract users with specific purposes. Later either by using phishing techniques, spam or exploits; the malicious websites scam them by stealing private information, installing malware, or getting a financial benefit.**

**The traditional approach to protect users from malicious websites is through blacklists. This method has been proved to be ineffective. Detecting malicious websites is a complicated task requiring manual effort while for an attacker is fairly easy to replicate the scheme from a different domain. Automated algorithms for detecting suspicious domains can help to update the blacklists more frequently, limiting the exposure time of the users to new attacks.**

**The first contribution of this paper is an open-source library implementing the Co-Visitation Networks algorithm proposed by Stitelman et al.(25).**

**The second contribution is a thorough analysis replicating the results of the original paper on a more recent dataset of online advertising logs.**

**We designed the library using scalable and well known Big-Data frameworks with the intention that the library can be easily used on other cybersecurity analysis where the same co-visitation patterns (or co-occurrences of events) can help to identify suspicious activity. It also gets us closer to using Graphs for classification problems; and proposes new techniques like Graph Embedding, for future studies.**

**It also gets us closer to using Graphs for classification problems; and proposes new techniques like Graph Embedding, for future studies.**

Fraud Detection | Advertising Exchanges | Internet Advertising | Graphs | GraphFrames | PySpark | Python | iGraph

## 1. Introduction

Web Advertising makes money, and in less than 1 second, as users, we are profiled from our previous searches in web browsers, to offer us the correct advert to arouse interest.

Today in the webpages, we can see neverending choices of ads managed by advertisers who need real visits and quality traffic for their advertising campaigns to succeed. Scammers take advantage by trying to turn 'invalid traffic' into something that looks like legitimate traffic (24).

Several studies estimate advertising fraud losses will reach 42 billion dollars this year, and that these losses will increase year on year, reaching more than 100 billion in 2023 (18; 26). Despite these losses Web Advertising is a growing business (10).

But, **What is Programmatic Advertising?** In a short answer, it is the use of software to buy digital advertising. Lets explain this with an example.
Google is not only a search engine. As a company, its main source of profit is Google Ads (35), a dynamic advertising payment method for the customer. This means that the advertiser pays or charges for the traffic that is generated in one direction or the opposite. For example, advertisers will only pay for the Ads in which the users interested in their products clicked on (Pay-Per-Click) (36); website owners will charge based on the number of clicks that the Ads generate on their page.

Many Google Ads, like those which offer loans or insurance, can cost up to more than 50 dollars per click to the advertiser, whereas the malicious websites only pay 0,2 dollars per click to their workers (17). Due to this significant difference, it's normal that there is a big illegal trade that represents a considerable threat for the Cybersecurity, giving rise to massive online fraud platforms.

The Web traffic could comprise of visits from users that are not real ones, here the terms 'legitimate traffic' and 'invalid traffic' come up, although not all traffic considered invalid can be regarded as fraud.

The owners of malicious websites have the possibility of buying this type of cheap and poor quality traffic, with the objective of improving their SEO positioning over their competitor websites, or deliberately redirect the traffic to known websites to get benefits per click.

Furthermore, there is the term **Non-Intentional Traffic (NIT)**, which fits better than the traditional term Non-Human Traffic (25); that arises from the traffic that can directly involve real users but leads them to websites that they did not intend to visit (websites that open when closing another website or websites that are loaded on background).

Even though there are large differences between the web traffic generated by a human and the automatic one, it is becoming increasingly difficult to differentiate them due to new Machine Learning Adversarial Attacks and their capacity of rapid evolution and expansion. Ad Network defenses evolve generating liveness-proof to distinguish non-human traffic but attackers are intent on mimicing their actions to create credible 'invalid traffic'. Many bots are used to automate processes with the intent of making as much web traffic as possible in the shortest time, obtaining benefits from 'fake clicks'.

This paper aims to identify the Traffic NIT of a set of logs and try to separate it from the legitimate traffic of the human user browser's navigation.

## 2. Background

### 2.1 Digital advertising

The question is : 'What happens when a user visits a website, and some Ad is loaded?'

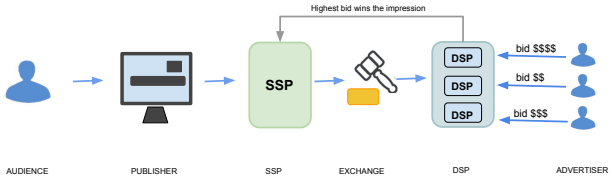


**Fig. 1.** Life cycle of an online Ad.

The process followed from the time the user visits a website until the advertiser pays for the ad, takes place in a few seconds.

When a user visits a website, the browser generates a request to the owner of the website, to view the content of this website (the lawful content and other content that may not be).

At the top or side of the webpages, the publisher of the website may place some ad spaces to get revenue from advertising. In these banner ads, other advertisers can put code or other links to their websites (ad-tags), as a redirection when the ad is clicked on by the user. When the user clicks on the ad, there is a request sent by the browser to the **SSP (Sell Side Platform)** which help publishers to manage their display space on the websites.

The SSP redirects the request to an entity called **Ad-Exchange**, another intermediary, and tells it what kind of inventory profile is available. The SSP can be connected to several different Ad-Exchanges to have the publisher's ad exposed to as many potential buyers as possible. The Ad-Exchange will initiate and auction to the **DSP (Demand Side Platforms)**.

Ad-Exchanges are like 'Auction Houses', where the DSPs will respond with bids based on profiles or pre-established categories of previous advertisers campaigns. They follow profiled users who have previously clicked on similar ads so they can request a URL for the corresponding ad that fits better with any advertiser bid.

The winning bid and the URL associated with the ad, are forwarded again to the SSP, who will forward it back to the user browser to show the ad which the user has clicked on. The transaction is made during the load page, that is no more than 120 - 300 ms (Real-Time Bidding (RTB) ) (8).

When we consider that each intermediary receives a small commission for each ad sent to the user website, and also taking into account the existence of bot groups or automated campaigns that make up to millions of clicks in a short time; we have a **large amount of economic profit in seconds.**

### 2.2 The fraud problem

The use of automated procedures using technology to buy and sell an Ad inventory is what is called **'programmatic advertising'.** Also, this procedure is optimized by using Artificial Intelligence or Machine Learning techniques to increase efficiency.

The relation between publishers and ads agencies that also can automate the budgets is made easy by programmatic advertising. The programmatic ad-networks are an opportunity for small agencies, advertisers, and publishers to set their publicity on websites and make their campaigns more successful. This opportunity is also taken by fraudsters to make a profit. The programmatic ad chain involves so many actors and any of them can be a point of infiltration.

Almost anyone can sell or buy an ad, and this is the idea of an **open ecosystem.** Closed platforms also exist, like Google Adwords or Facebook, that offer their own prices and in some opinions decrease competition and innovation between advertisers.

An additional benefit for the bad actors is that **Ad Fraud is not illegal** technically and is very lucrative. The fraudsters haven't taken high risks and have the advantage that the law is not explicit in these terms (34). There is no fine or sentence like in a credit card fraud.

The monetary reward is based on the volume of transactions, and this is also an opportunity for the advertisers themselves. They gain profit from the clicks on fraud ads, paying for visits with no intention of buying anything from them, or creating an advantage over other advertisers who are competitors.

As everyone can see on many websites, there is more space dedicated to ads that involve fraud than licit content. From the point of view of the publishers, every time a user clicks on an ad, the owner of the website is paid for displaying this ad. This is also used by publishers to trick the internet user in order to get a higher number of clicks (licit or not) to get paid.

From the Cybersecurity point of view, identifying or detecting suspicious websites, fraudulent pages, or botnets that generate non-legitimate traffic emerges as a new important motivation.

We are going to focus on the traffic behavior that can be generated by bot networks or is non-intentional (NIT).

It is well known that these networks can be detected since users who visit these malicious websites are usually the same.

This also happens on other webpages that cannot be malicious ones, but on a smaller scale. This usually leads to the fact that if we find a large amount of traffic from a set of users to a single website; there is a greater probability that this website could be suspicious of fraud.

It should be taken into account that bot networks also generate traffic to lawful websites. This is the traffic that we will have to recognize and discard. As the attackers reuse their botnet networks as a service to different customers; this can be helpful to identify the suspicious traffic that these networks generate.

In other previous publications such as the paper **"Using Co-Visitation Networks For Detecting Large Scale Online Display Advertising Exchange Fraud** (25), some novel algorithms applied to the given traffic, aim to identify the origin of the web traffic and validate if the sellers of online display ads are reliable and legitimate.

That is why, for this paper, we will try to apply the solution proposed in the above-mentioned paper, to reach the same conclusions and validate the novel algorithm.

## 2.3 Co-Visitation Networks

The algorithm used in this paper is a novel algorithm that, extracting knowledge from the data logs, tries to classify websites that produce a large amount of NIT (Non-intentional traffic) obtaining an economic benefit.

Analyzing the co-visitation degree between domains, and setting a threshold number of common visits; we can know if the traffic between them is NIT or not.

To calculate this value, the algorithm is based on the fact that if we have a considerable amount of domains and there are links in the graph between them, it will be considered that the traffic they produce may not be lawful and might not be generated by a real user, being classified as a fraudulent domain. Also this high co-visitation degree could be assigned to 2 domains, if the domains related are very common ones or are sites with common content. But is weird to find 2 domains with no similar content and a big co-visitation degree; so following the algorithm these domains are classified as malicious ones. The authors of the Stitelman's paper (25), consider a domain as suspicious if there is a big co-visitation degree **between almost 5 or more domains related**. So analyzing the content sites of the neighbors' domains in a co-visitation network, we can classify if are fraudulent domains or not.

**A. Domain - Ip Graph.** The algorithm allows us, with a large amount of data extracted from the logs (datasets), to build a first bipartite graph in which the visited domains will be related to the user IPs that visit them.

Lets consider the graph: Domain_IP_Graph = (dom, IP, edges); where 'dom' will be the domains or websites visited, 'IP' will be the IPs of the users who visit these websites, and 'edges' will be the links that exist between the 'dom' and the 'IP'.

This first graph will help us to calculate the graph of co-visits between domains. From the Domain_IP_Graph we can obtain the total number of visits to a domain per IP, in addition to the total number of different IPs that have visited a domain.

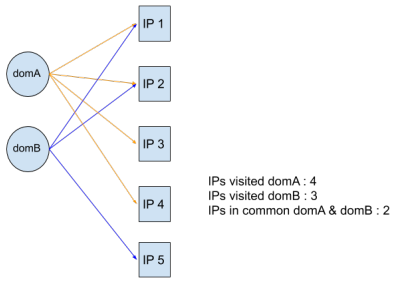


**Fig. 2.** Schema of calculation of a Domain-ip Graph .

**B. Covisitation Graph.** To calculate the **co-visitation graph** as a domains network approach, we will consider domA and domB, two domains or websites, which will belong to our domain space D. There will be a relationship between both domains if there is at least n * 100 percent of users (in our case given by the IPs), who have visited the domA but have also visited the domB.

The link between two nodes or domains represents at least 50% of users who have visited each of those domains, considering this threshold the most useful for that case, to get the connection network.

So the algorithm:

$$ {}^{n}_{d}\mathrm{G} = (V_d \subseteq D, E = \{(domA, domB) : domA, domB \in D, \\ [\Gamma_G(domA) \cap \Gamma_G(domB)] \,/\Gamma_G(domA)$$

**Fig. 3.** Algorithm co-visitation rate .

For example, with a domA domain visited by 400 users, and a domB domain visited by 1500 users. An overlap of 200 is the number of common user IPs between domains (which visit both domains); we would calculate the following:
Is there a link from domA to domB? 200/400 = 0.5
Is there a link from domB to domA? 200/1500 = 0.133
In this case there is a link between these two different domains if the number of common visits divided by the total number of visits to the domain is greater or equal than 0.5. We will consider this value as a weight in the edge. So in the example graph there would be a link from domA to domB, but not from domB to domA. The domains are tightly connected because of the shared traffic overlap.

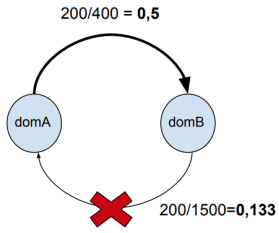


**Fig. 4.** Domain's Co-visitation grade calculation.

## 3. Data Description

The attacks on the Internet are growing in number, and the organizations respond developing security products based on certain intelligence or logs data.

Nowadays, logs can provide more information than we could imagine about activities carried out in the network. Specialists are normally able to know if an attack has occurred by looking for patterns in the logs.

Normally in these logs the data is not clean (blank fields or nonreadable characters). There is a lot of information that we can consider as not useful and that makes these logs heavy in terms of volume data.

The data logs used in this paper belong to a Global Provider inside the Online Advertising Ecosystem of which we cannot reveal the name due to the signed terms in an NDA (Non-Disclosure Agreement). The datasets used include daily samples of the incoming requests that the DSPs exchange with the corresponding AdExchange. Also, are old datasets, and in any case, the user's information has been used with the

intention of identifying the user or to create user profiles. The information only has been exploited to establish links between IPs and domains.

Although we have other records, we only used two of them for this short concept test. Here is a brief description of the fields used from the dataset.

**user_ip** : represents the IP address of the user that creates the ad-request. We consider this field as a string after its normalization filter.

**referrer_domain**: is the publisher ad-request's referrer domain. It refers to the ad-network that has pre-bought the advertisement request. We will consider this record as a string of the publisher URL with the format : "com.somecompany.otherthings".

These logs are given in csv.gzip format, spread over days. In total, we have approximately 300 log files per day of around 200MB to 1GB of data.

For the study and given the limitations in the cluster (standalone mode), we will take a sample of one day data (100G approx.) with files of 200MB consecutive in the time period and then compare the datasets experiments with other day datasets in the same week.

These "raw" logs contain lots of information about the user's behavior and the destinations websites visited, but there is also a huge amount of 'noise' (fields to null / none, non-standardized domains, strange characters, ...).

We filter the data, mainly using the fields' user_ip 'and 'referrer_domain', in a way that if some of them are null, are not a string, or do not have an IP format; we discard the tuple (user_ip - referrer_domain). We won't consider this invalid percentage of records, with the aim of obtaining a smaller amount of data, in advance, easier to represent. We consider each log as a table composed of rows and columns, to cover all data efficiently thanks to the technologies used which are explained in a later section.

## 4. Development framework

The technologies used to carry out the implementation and development of this paper are described below.

**Apache Spark** (3) is an open-source cluster-computing framework; developed by Berkeley AMPLab in California. It is oriented to obtain high-speed cluster computation and data parallelism. Spark is 100 times faster than Bigdata Hadoop and 10 times faster than accessing the data from a disk.

Apache Spark supports several programming languages such as **Scala** (32), Python, Java and R; providing a high-level API.

It is known that the main element of Spark is **RDDs (Resilient Distributed Datasets)** (rdd), a basic data structure where the data is distributed in blocks or records, distributed in the different cluster nodes being grouped by operations. Spark also represents the data in Dataframes and Datasets.

**Python** (28). It is a high-level programming language for general-purpose and interpreted (not compiled). Although it can be considered a slow language (slow execution or performance), many developers continue using it assuming that temporary cost due to other advantages and different libraries it offers. Nowadays it is widely used for Machine Learning and real-time Data Analysis.

**Pyspark** is the Python API for Apache Spark. Until now Python was considered less efficient than Scala (programming language for Spark; multiparadigm and scalable); because of the use of **UDF** functions (User Defined Functions) when python code is called in an UDF; but recently added functionality permitted to implement UDFs in python that are called in batches from the Spark JVM reducing the IO overhead. That is why Spark2.3 version offers the ability to define UDFs with full Python performance (5).

For the development of the code of this project, Pyspark and Python have been chosen because of their simplicity at the time of implementation, comprehensive API as well as the great readability of the code.

The information or data in Pyspark is used in structures called **Spark Dataframes**. A Dataframe can be considered to be a table with rows and columns similar to a relational database. The Dataframe is distributed in records among all the nodes of the cluster; where computing the operations are distributed to make tasks more agile offering a high-level of abstraction. There is another type besides Spark Dataframe, called **Pandas Dataframe** with which you can also work in the Spark environment.

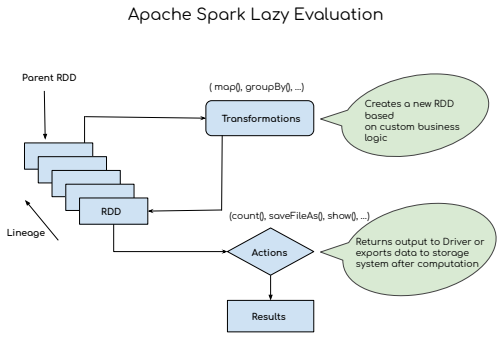


**Fig. 5.** Spark Operations.

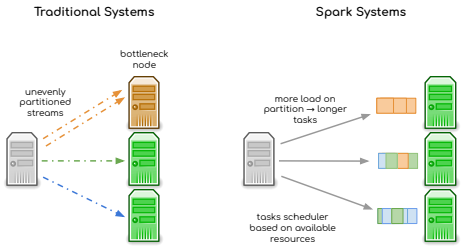


**Fig. 6.** Spark Dynamic load balancing.

But the main advantage we have using Spark Dataframes is that **Spark is lazy**(6) and works in a **distributed environment**. That means, it divides the operations into transformations and evaluations that accumulate until one of them needs the calculated data of another previous operation. Thus, Spark, having the complete operations string to calculate, can divide them into blocks for optimal execution and distribute

them by the cluster nodes, with faster and more efficient computing; storing only the intermediate data for each operation in its memory; allowing to use a large volume of data.

On the contrary, this does not happen with **Pandas Dataframe**, which stores all the data in its memory and then immediately executes each operation in **local**. It is used with smaller amounts of data.

**GraphFrames** (4) is a library to represent graphs: vertices (domains or ips in our case) and edges (relations or links between them).

GraphFrames is designed to use Dataframes, this is the main difference with **GraphX** another library designed to be used with Spark and RDDs. Like in mathematics Graph Theory, GraphFrames also allows us to execute queries and algorithms designed for graphs, easily search for patterns, vertices, subgraphs, or the shortest path between two nodes.

Because of the huge volume datasets, the interest in the use of Graphs is growing in several techniques of Machine Learning, and could also be applied to Cybersecurity algorithms.

In this part we want to mention the use of **Graph Motifs finding** from Graphframes in the code generated for this paper. (11) GraphFrames library gives the possibility of using the motifs finding algorithm through the method *'find(pattern: String)'*. Motifs finding is a graph pattern matching that checks a value against some pattern. In a graph, the pattern to look for is an expression used to define some connected vertices. Motifs finding is an efficient manner of executing multiple joins looking into a graph structure. An example of a search for structural pattern in a graph, used in the code developed for this paper :

```
df_motifs =
= g_domip.find("(a)-[e]->(b); (c)-[e2]->(b)").filter("a != c")
```

**Fig. 7.** Motifs finding example of use for the algorithm.

GraphFrames at the moment is not part of the Apache Spark API, since it uses GraphX and it does not use Dataframes. Finally, since GraphFrames does not have a specific module for graphical representation of graphs, we used IGraph, in addition to Python matplotlib for some graphs and networks plots.

**IGraph** (16) is a collection of open source graph analysis tools.

To make the environment more dynamic we use **Jupiter Notebook** (27), which is a client-server interface that allows us through a web client application used from any web browser to write code, document it, execute it in parts, visualize data in graphs, perform intermediate calculations, etc.

## 5. Algorithm Design

In this section, we described the design of our implementation of the Co-visitation Networks algorithm (4).

The algorithm developed, follows the flow chart on Figure 8, to get a classification of suspicious domains for the input data.

In the image, we can see some modules, like 'loader', 'preprocess', 'patterns_extractor', 'plot' or 'saver'. Every module is composed of Python functions that could be used separately.

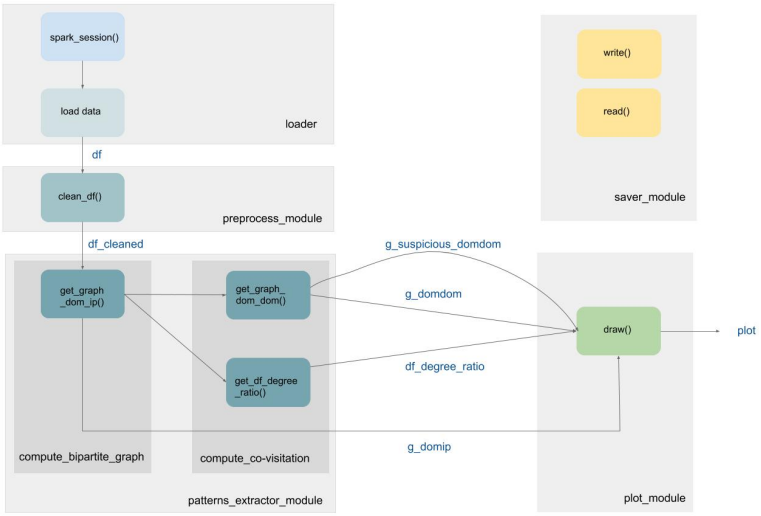


**Fig. 8.** Flow chart algorithm code. Determining the co-visitation network.

The Python code is distributed in a few classes, making a **'utils' package**. Each of the classes, as its own name describes, are composed by functions that can be applied to the component cited in the name of the class.


**Fig. 9.** Utils Python package components.

A Python class is classified according to the functions that it contains, and the different input components for the function.

In Figure 9, is a description for each Python class function components. In the following lines there is a brief explanation to apply the functions developed.

**df_utils.py**: functions that could be applied to a Spark dataframe.

**gf_utils.py**: functions to apply to a GrapFrame graphs.

**row_cleaners_utils.py**: functions to clean and normalize the data from the columns of an Spark dataframe (domain and IP cleaners ).

**draw_utils.py**: functions to draw the graphs (networkx,

igraph), represent histograms or overlap matrices.
**read_write_utils.py**: functions to save or read the calculated data onto disk (graph or Spark dataframe).
**spark_utils.py**: function to create a Spark session.

Each package or function could be easily modified.

Our contribution code still being developed, to make the package simple to use and install.

The code is open-source and at the moment is published on a public GitHub repository: **"https://github.com/ayalo/TFM_fraud_36"**

## 6. Results and Experiments

### 6.1 Overall results

First, we take a sample data, limited to 4 log files of 200MB of data each, generated in one day consecutively.

From the logs we create a bipartite graph of Domains and IPs. The edges of the graph represent that the IP of the source node has visited the domain (or app) of the destination node. Figure 10 shows an example of this bipartite graph.

On the left side of the bipartite graph, we see the domains visited by their corresponding users on the right side, represented by their connection IP.

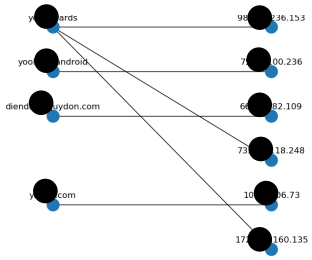

**Fig. 10.** Example of Domain-ip Graph .

With this structure, it is easy for us to check the number of visits that a domain receives, as well as the number of common users that receive two domains between each other.

Once this graph is generated, we obtain the Co-visitation graph.

We can see examples in Figures 11, 12 and 13.

On the right side of the Figure 11 we see the complete graph in which all the domains consulted on the day are represented; meanwhile on the left side, we see a sub-graph with the domains classified as suspicious ones.

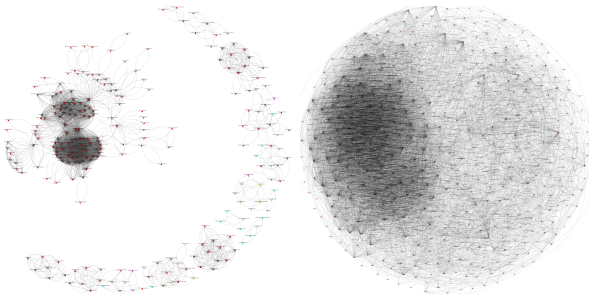

**Fig. 11.** Example Co-visitation Graphs from the dataset.

With the classification seen on this last Figure, we can appreciate a large central cluster composed of a considerable number of nodes and connections between them. This cluster will be associated with NIT traffic. Other smaller clusters around are corresponding to groups of licit traffic and visits of real users to these domains that could be malicious or not. These ones, never have a large number of nodes.

The weights in every calculated graph are the corresponding co-visitation degree (edge_ratio in our code) between the connected domains. We want to obtain the complete graph of the data sample, with the idea of being able to compare sub-graphs between legitimate domains and those considered malicious. That could be seen in 12.

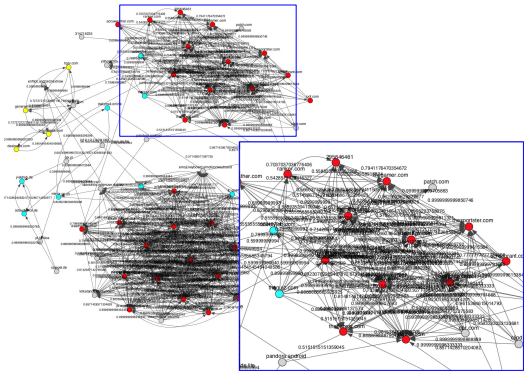

**Fig. 12.** Example Co-visitation Graphs weighted.

In the co-visitation graph Figures, we have several groupings of websites with different content where everyone shares visitors traffic between them.

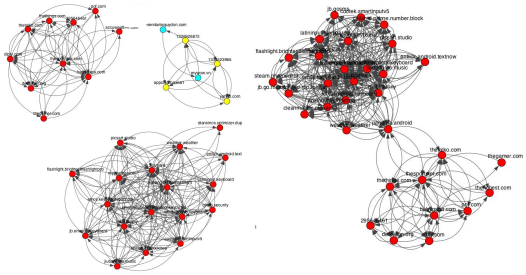

**Fig. 13.** Example Co-visitation Sub-Graphs for malicious domains.

If a website has many visits and many visitors, and we assume it is legitimate, it should not have many relationships with domains that are not similar in content.

With similar content of domains and therefore of users who visit them, the coincidence in the graph is represented with an order of not up to 50%, and also popular websites should have more incoming edges than outgoing ones. All the grouped domains that exceed this percentage are those that we will consider malicious domains and will be related to each other. Also these sites don't have similarities on content. Most websites are made up of a bunch of links with related pictures as can be seen on the Table 1. The main purpose is to be attractive to the user and raise interest to the point of

clicking on those links.

With this classification of suspicious domains, the advertisers could avoid buying impressions from these sources.

We focused on the domains related to the domain "[redacted]emyjob.com" 1; that is classified as fraudulent, and its relations ordered by co-visitation degree higher than 50 % with other domains. Considering a legal domain, "[redacted]hoo.com", the domains related with this one, are been visited by real users which has a Co-visitation degree smaller than 50 % assigned in the classification. We must note that there may be relations between legitimate domains and those considered suspicious, but the co-visitation degree of the legitimate domain concerning these will remain <0.5.

**Table 1. Co-visitation rates of a Suspicious domain "[redacted]emyjob.com" (highest overlap) on the left, and normal domain "[redacted]hoo.com" co-visitation on the right**

| "[redacted]emyjob.com" | |
|---|---|
| [redacted]ly.com | 0.743832 |
| [redacted]finition.org | 0.673838 |
| [redacted]nq.com | 0.668961 |
| [redacted]ather.weather | 0.640849 |
| [redacted]hoo.com | 0.595524 |
| [redacted]bygaga.com | 0.583189 |
| [redacted]esportster.com | 0.578026 |
| [redacted]tivebeat.co | 0.567986 |
| [redacted]talko.com | 0.567699 |
| [redacted]things.com | 0.563683 |
| [redacted]gamer.com | 0.549053 |
| [redacted]nker.com | 0.543889 |
| [redacted]pple.com | 0.540447 |
| [redacted]cuweather.com | 0.531554 |
| [redacted]erichest.com | 0.525530 |

| "[redacted]hoo.com" | |
|---|---|
| [redacted].com | 0.045187 |
| [redacted]ather.com | 0.029487 |
| [redacted]pple.com | 0.028301 |
| [redacted]esportster.com | 0.027971 |
| [redacted]cuweather.com | 0.027190 |
| [redacted]bygaga.com | 0.025908 |
| [redacted]inition.org | 0.022630 |
| [redacted]dgereport.com | 0.022599 |
| [redacted]talko.com | 0.017032 |
| [redacted]things.com | 0.016881 |
| [redacted]716233 | 0.016794 |
| [redacted]today.com | 0.016291 |
| [redacted]y.com | 0.016112 |
| [redacted]lmath-games.com | 0.016006 |
| [redacted]lick.android.textnow | 0.015414 |

--censored domain names.

With the aim of compare the behaviour of these two domains selected before, we are going to study the data. We try to represent the dataset dividing the space data in bins, in order to return the number of samples that are between these slots on an histogram. This could facilitate the calculation of the mean for the data in each bin.

The outDegree and the co-visitation rate, are calculated considering the domain as unique in our domain space D 3. Obtaining the mean of these two metrics of one specific suspicious domain, it could be seen that for a bigger co-visitation degree (or edge_ratio), the smaller the outDegree for the studied domain 14.

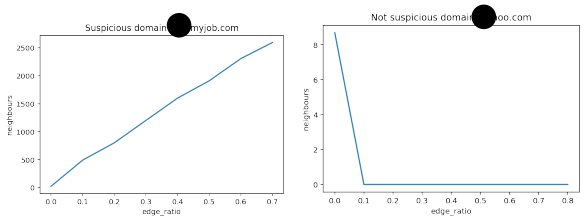


**Fig. 14.** Mean Co-visitation degree - outDegree for the domains : "[redacted]emyjob.com" on the left and "[redacted]hoo.com" on the right.

The relation between the co-visitation rate (edge_ratio) and the neighbors (outDegree); generates an increasing plot in ratio for a considered suspicious domain ("[redacted]emyjob.com"); while for a considered legal domain ("[redacted]hoo.com") the evolution is entirely different 14.

As we said, although "yahoo.com" is related to other highly visited domains, its coefficient of co-visitation remains low.

With the intention of visualize the calculated results, in Figure 15 we can see a heat map or overlap matrix (matrix on the left). In this matrix, the domains have been represented in the 'x' and 'y' coordinates; and the co-visitation degrees between them can be seen by different colors.

When higher and to the left of the matrix in this case, we see a bigger degree of overlap between domains; which means that the probability that the domain could be fraudulent is greater.

It has been verified that many of the domains have an overlap degree of at least 60 % and in some cases, it reaches up to 90%.

In order to compare, we have made another similar matrix, taking a random sample of domains from the data set (matrix on the right of Figure 15).

We can see in this matrix that there is practically no overlap between domains, and the co-visitation degree between them is usually low.

## 6.2 Suspicious domains patterns

During the application of the algorithm, we have taken data of several consecutive days to be able to contrast domains classified as suspicious with the legal ones.

The percentage of suspicious domain classification for 3 days of data, taking a total of 1GB logs per day, is over the 20-30 % 16. Our intention is to demonstrate that with a larger volume of data we manage to classify domains with equal precision.

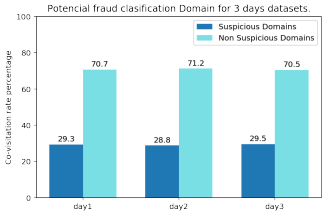


**Fig. 16.** Percentage of cataloged suspicious domain from our dataset.

In our directed graph, outDegree is the number of edges or links where the domain is the source. We refer the outDegree of a domain with the number of unique neighbors. The neighbors are other unique domains.

Also, the co-visitation rate calculated with the algorithm proposed and the number of common_ips, that refers to the absolute number of different IPs that the domain is linked with in the co-visitation graph, is presented in Figure 17.

| | | day 1 | | | day 2 | | | day 3 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | outDegree | co-visitation | common_ips | outDegree | co-visitation | common_ips | outDegree | co-visitation | common_ips |
| Suspicious Domains | [redacted]ivernon.com | 279 | 0.652 | 3.907 | 419 | 0.436 | 13.9476 | 320 | 0.610 | 7.0 |
| | [redacted]scopefriends.co.uk | 147 | 0.605 | 6.054 | 152 | 0.5706 | 7.4146 | 155 | 0.566 | 6.794 |
| | [redacted]ploopline.com.br | 90 | 0.529 | 3.7 | 87 | 0.513 | 4.103 | 66 | 0.677 | 9.485 |
| Worthy Domains | [redacted].com | 2815 | 1.873E-4 | 9.390 | 2945 | 2.008E-4 | 7.962 | 2626 | 2.2996E-4 | 8.006 |
| | [redacted] | 4999 | 7.533E-5 | 13.146 | 5002 | 8.194E-5 | 14.357 | 4900 | 7.559E-5 | 12.391 |
| | [redacted]efire.com | 3775 | 2.109E-4 | 13.957 | 3737 | 1.987E-4 | 14.066 | 3884 | 1.941E-4 | 13.525 |

**Fig. 17.** OutDegree, co-visitation rate and common_ips of 3 cataloged domains.

It should be noted that although domains considered lawful have a greater outDegree, and greater number of common_ips,

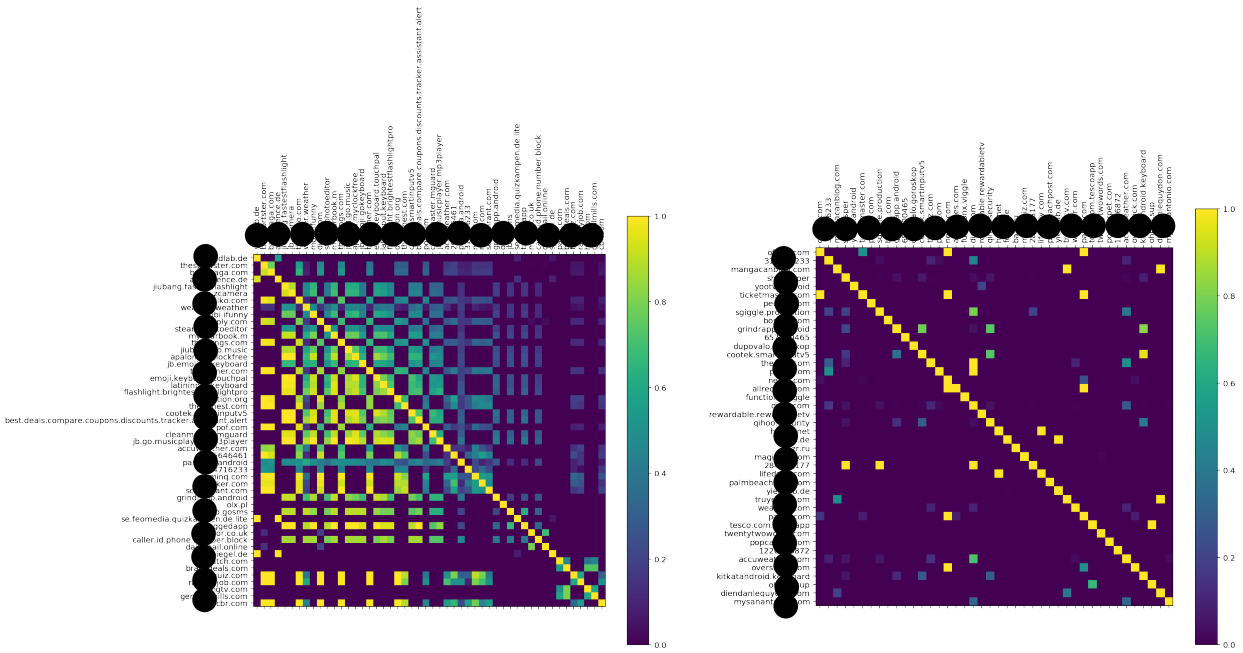

**Fig. 15.** Overlap matrix of suspicious webs con the left and a random sample on the right.

the assigned co-visitation rate is still very low (Figure 17, Table 1).

However, for domains classified as suspicious, the degree of co-visitation exceeds 0.5.
In order to get knowledge about the log data used :

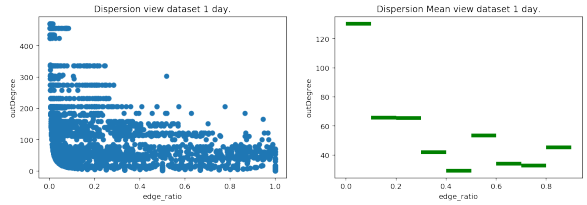


**Fig. 18.** Example mean dispersion edge_ratio / outDegree from the dataset.

To get an idea about the data we are managing, we want to take a view of the dispersion for the co-visitation degree (edge_ratio) and the outDegree values in our dataset, Figure 18 left side. The mean for this dataset is presented on the right side of the Figure 18.

With the previous Figure 16 we can have an idea about the volume data and where the suspicious domains are going to be related in different clusters.

Comparing the co-visitation degree over 3 suspicious domains with 3 legal ones, over 3 days data.

Note that the mean and median distributions are above 0.5.

Looking the distribution data day1 for the domain 'lillian-vernon.com' 19, the median is on 0.8 and the mean with a variance maintained above 0.5. Almost the entire distribution is over 0.5 co-visistation degree (edge_ratio) assigned. There is some data that is out of this variance like the minimum which is 0.3.

Though not always reaching conclusive results for the suspicious domains, there is a quite big difference when comparing with the domains considered legitimate.

For the values of two licit domains 20; the image shows all the points of the distribution lower than 0.5, even less to 0.01 co-visitation degree. These results indicate the same conclusion as in 14.

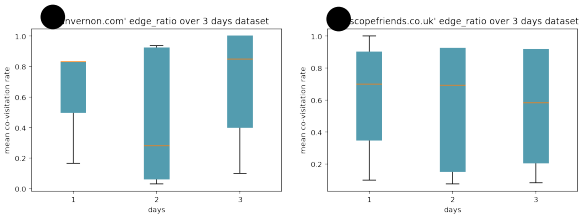


**Fig. 19.** Boxplot Co-visitation rate distribution of 2 suspicious domains over 3 days dataset.

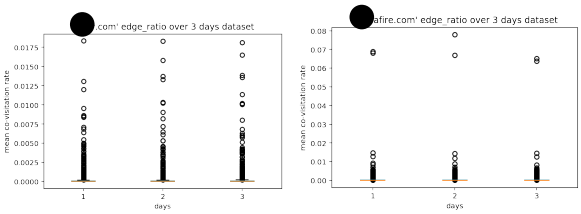


**Fig. 20.** Boxplot Co-visitation rate distribution of 2 legitimate domains over 3 days dataset.

## 7. Related work

Nowadays, innovations on the application of Machine Learning, Artificial Intelligence or Data Mining Techniques applied to Cybersecurity are becoming increasingly common and bring about significant advances.

Although these techniques are often applied not especially into looking for patterns or malicious URL detection, when the methods used are rather inefficient or outdated.

The solution applied in this report intends to detect invalid traffic from the part corresponding to the seller into the life-cycle of an ad. Meanwhile, in other studies, the detection is based on identifying known attacks, based on the users part, the interaction of the user with the webpages, whether it be the users vulnerability or the easy access to this type of information.

The useful data to detect malicious traffic from the part of the DSPs or Exchanges is usually private and opaque. Many of them offer only the information part that doesn't affect their interests for the propose of not damaging their business or obtaining the maximum monetary profit.

In addition to the Stitelmant's (25), on which the paper is based; another approach is proposed in the paper **Nameles** (2). **Nameles** identifies malicious ad request patterns using an algorithm based on Shannon entropy. A link between these techniques is necessary for it to be applied to Cybersecurity URLs fraud detection with the intention to be able to approach the objective of this report more efficiently.

Spam Detection, Web Page Classifications, and Malicious URL Detection techniques are based on similar known algorithms (9).

On the following lines, there is a classification of the methods and techniques applied til these days; cited on the paper " Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs" (19).

Some **methods applied for URLs Classification** like :

- "logistic regression over some hand-selected features to classify phishing" (30); - "comparative analysis of phishing and non-phishing" (22);

- "drive-by exploit URLs and use a patented machine learning algorithm" (23);

- "classify phishing by thresholding a weighted sum of some features (content-related, lexical, and WHOIS- related) (39)".

Other methods for Machine Learning in Related Contexts :

- "The use of statistical methods in machine learning to classify phishing emails" (15);

- "Introducing models of text classification to analyze email content" (1).

Or **methods for Non-Machine Learning Approaches**:

- "Predictive Blacklists use a Page Rank-style algorithm" (20);

- "Exploits by using behavioral detection (monitoring anomalous state changes)" (37);

- many commercial efforts to detect malicious URLs, as McAfee SiteAdvisor (21).

As we can see in the Section "Future Work", there is an intention to mix the data prediction techniques applied these days, to the Machine Learning, graphs and co-visitation network techniques, extracting the knowledge from ad-network logs.

## 8. Conclusions

The problem about the Unintentional Traffic growth generated from web browsers, cause the investment of millions of dollars using a large amount of information from logs.

These logs contain useful information that in expert hands could be translated to know if an attack has happened. But these logs also contain a vast quantity of irrelevant information for advertisers, which need to model appropriate profiles to their users, filtering out the valuable information.

We present an implementation of the algorithm for the automatic detection of malicious domains with large data sets for PySpark. Our library has been designed with special focus in scalability, easiness of use and adaptability to different types of data sources. That is why we choose PySpark and Graphframes as base frameworks of our implementation.

In addition, we have applied the library to a more recent dataset of advertising logs replicating some of the findings of the original paper by Stitelman et al.(25). We have proved that the algorithm works and identify suspicious domains even on relatively small samples of data.

Techniques for noise logs elimination have been developed, as well as the analysis of graphs to identify the number of domains visited or the co-visitation degree between domains.

## 9. Future Work

It is known that mathematical and statistical operations on graphs are limited, and the application of machine learning methods is a tendency and a new challenge.

For the application of Machine Learning methods, graph embeddings could be a solution.

**Graph embedding** transforms a property of the graph into a vector or a set of vectors in order to have simpler and faster operations than on graphs.

But Graph embedding presents other challenges. Capturing the graph topology could be a problem because more embedder encoded will produce better results, but the **dimensionality creates complexity** in order of execution time. Embeddings are compressed representations of the properties of the graph nodes.

The embeddings would be compared with an adjacency matrix of a graph, which describes the connections between the nodes in the graph. The memory required to calculate this matrix on large graphs could be an impossible task in some cases.

With graph embedding, the relations between the nodes or the node's properties, are packed in a vector which has a smaller dimension and is more practical to be used in statistical calculations. An open issue is how to scale a large graph embedding like in a word embedding.

In a word embedding problem, similar words have similar embeddings. Predict the relations between malicious domains, like a word embedding algorithm of machine learning is trained to predict a neighbor word in a sentence (12; 38; 31; 14).

A different approach is that note we set for the novel algorithm used in the paper (see Figure 4 and 4), we have fixed the value n=0.5. An idea could be to evaluate this threshold evolution over time, in order to train an IA model for later, apply other techniques like Graph Embedding described after.

**ACKNOWLEDGMENTS.** This paper would never have been completed without the help and support of my friend **Luis Peinado Fuentes** <q3luis@gmail.com>, who advised me while on holiday. I also would like to thank **Antonio Pastor** and **Patricia Callejo** whose help was invaluable many times.

## References


[1] A.Bergholz, J.-H.Chang, G. F. a. (2008). Improved phishing detection using model-based features. in proceedings of the conference on email and anti-spam (ceas). *Paper*.

[2] Antonio Pastor, Ruben Cuevas, A. C. M. P. P. C. P. V. M. K. A. A. (2019). Nameles: An intelligent system for real-time filtering of invalid ad traffic. *Paper*.

[3] ApacheSpark (2019). Lightning-fast unified analytics engine. https://spark.apache.org/.

[4] Databricks (2017a). Graphframes user guide for python. https://docs.databricks.com/spark/latest/graph-analysis/graphframes/user-guide-python.html.

[5] Databricks (2017b). Introducing pandas udf for pyspark how to run your native python code with pyspark, fast. https://databricks.com/blog/2017/10/30/introducing-vectorized-udfs-for-pyspark.html.

[6] DataFlair (2018). Lazy evaluation in apache spark – a quick guide. https://data-flair.training/blogs/apache-spark-lazy-evaluation.

[7] Deebanchakkarawarthi G, Parthan AS, S. L. S. A. (2019). Classification of url into malicious or benign using machine learning approach. *Paper*.

[8] Developers, G. (2019). Latency restrictions and peering. https://developers.google.com/authorized-buyers/rtb/peer-guide.

[9] Doyen Sahoo, C. L. and Hoi, S. C. (2017). Malicious url detection using machine learning: A survey. *Paper*.

[10] Enberg, E. J. (2019). What's shaping the digital ad market. https://www.emarketer.com/content/global-digital-ad-spending-2019.

[11] github graphframes (2018). Motif finding. https://github.com/graphframes/graphframes/blob/master/docs/user-guide.md.

[12] Godec, P. (2018). Graph embeddings — the summary. https://towardsdatascience.com/graph-embeddings-the-summary-cc6075aba007.

[13] Guolei Yang, Neil Zhenqiang Gong, Y. C. (2017). Fake co-visitation injection attacks to recommender systems. *Paper*.

[14] Hongyun Cai, V. W. Z. and Chang, K. C.-C. (2018). A comprehensive survey of graph embedding: Problems, techniques and applications. *Paper*.

[15] I.Fette, N.Sadeh, a. (2007). Learning to detect phishing emails. in proceedings of the international world wide web conference (www). *Paper*.

[16] igraph (2018). igraph – the network analysis package. https://igraph.org/.

[17] Jobs, A. C. (2019). Ad clicking jobs • part time online work from home • get paid to click ads. http://www.adclickingjobs.com/.

[18] JuniperResearch (2019). Advertising fraud losses to reach 42 billion in 2019, driven by evolving tactics by fraudsters. https://www.juniperresearch.com/press/press-releases/advertising-fraud-losses-to-reach-42-billion.

[19] Justin Ma, Lawrence K. Saul, S. S. G. M. V. (2009). Beyond blacklists: Learning to detect malicious web sites from suspicious urls. *Paper*.

[20] J.Zhang, P.Porras, a. (2008). Highly predictive blacklisting. in proc. of the usenix security symposium. *Paper*.

[21] McAfee (2018). Mcafee siteadvisor. https://www.mcafee.com.

[22] McGrath, D. K. and Gupta., M. (2008). Behind phishing: An examination of phisher modi operandi. in proc. of the usenix workshop on large-scale exploits and emergent threats (leet). *Paper*.

[23] N. Provos, P. Mavrommatis, M. A. R. and Monrose, F. (2008). All your iframes point to us. in proc. of the usenix security symposium. *Paper*.

[24] Ops, W. (2019). The methbot operation. https://www.whiteops.com/methbot.

[25] Ori Sitelman, Claudia Perlich, B. D. R. H. T. R. F. P. (2013). Using co-visitation networks for detecting large scale online display advertising exchange fraud. *Paper*.

[26] PPCProtect (2019). The ultimate list of ad click fraud statistics 2019. https://ppcprotect.com/ad-fraud-statistics/.

[27] ProyectoJupiter (2015). User documentation. https://jupyter-notebook.readthedocs.io/en/stable/notebook.html.

[28] Python (2019). Python get started. https://www.python.org/.

[rdd] rdd (-). Apache spark documentation class rdd. https://spark.apache.org/docs/1.6.2/api/java/org/apache/spark/rdd/RDD.html.

[30] S. Garera, N. Provos, M. C. and A, A. D. R. (2007). Framework for detection and measurement of phishing attacks. in proceedings of the acm workshop on rapid malcode (worm). *Paper*.

[31] Sasak, B. M. (2018). Graphs are a game-changer for cybersecurity. https://dzone.com/articles/graphs-are-game-changing-for-cybersecurity-5-minut.

[32] Scala (2019). The scala programming language. https://www.scala-lang.org/.

[33] Ting-Fang Yen, Alina Oprea, K. O. T. L. A. J. W. R. E. K. (2013). Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. *Paper*.

[34] Vidakovic, M. L. R. (2015). Graph embeddings — the summary. https://marketingland.com/programmatic-advertising-fraud-exists-149927.

[35] Wikipedia (2019a). Google adsense. https://es.wikipedia.org/wiki/AdWords.

[36] Wikipedia (2019b). Pay-per-click. https://en.wikipedia.org/wiki/Pay-per-click.

[37] Y.-M. Wang, D. Beck, X. J. R. R. C. V. S. C. and King, S. (2006). Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities. in proceedings of the symposium on network and distributed system security (ndss). *Paper*.

[38] Yan Jia, Yulu Qi, . (2018). A practical approach to constructing a knowledge graph for cybersecurity. https://www.sciencedirect.com/science/article/pii/S2095809918301097.

[39] Y.Zhang, J.Hong, a. (2007). Cantina:acontent-based approach to detecting phishing web sites. in proceedings of the international world wide web conference (www). *Paper*.