# PARADISE - Preventive Advertising fRAud Detection Intelligent SystEm

Diogo Frazão
diogo.c.frazao@ist.utl.pt

**Abstract**—The capacity of click fraud to impact business models based on online advertisements emerged as a topic of discussion.Part of Google's financial success through the years was due to the rate per click online advertisement model; however, nowadays, this model has been heavily threatened by fraudulent clicks. Invalid clicks in advertisements, I.e., clicks that do not correspond to a real interest in the advertisement, deviate funds to scammmers. In order to solve this issue some solutions were presented. Unfortunately, the conducted tests revealed some difficulty to target these attacks. In this paper we introduce a new approach to detect fraudulent clicks, by creating a modular system, able to test the user's browser by accessing the website in order to identify if it's a legitimate one or not, and also to test the user's behavior. Additionally, the invalid clicks identification process can be improved by extraction and processing raw data from ad network database.

**Index Terms**—Click Fraud, ads, click fraud detection system, invalid click, data mining.

## 1 Introduction

Online Advertising is a key part of the Internet economy. These ads are the main source of income for millions of websites such as blogs and social networks. Without online advertising, these websites cannot operate.

The Internet has evolved to a point that you can create a business based solely on online advertising, thus important to protect the consolidation of this business model. Given the standardization of this income mechanism in the Internet world, in the third quarter 2015 this industry generated 15 billion dollars only in the US [1], it became the target of fraudulent mechanisms.

There are several media cases of these frauds. In 2006, Google had to pay $ 90 million to victims of **click fraud** [2]. According to the Financial Times, an online advertising campaign by Mercedes-Benz was more seen by bots than by humans, estimating that only 43 % of ad impressions have been seen by humans [3].

Not only news confirm that such attacks are in vogue. According to the 2014 report of the security company Incapsula [4], 56 %of all Internet traffic is generated by bots. Of these 56 % 29 % are malicious source, ever increasing the necessity of analyzing traffic received by websites.

The ecosystem of online advertising is fundamentally divided into four entities participating in the business model (Figure 1).

The ad network is the entity that manages the ads, being responsible for making the bridge between advertisers and publishers.

Publishers display ads provided by ad network on their websites or applications (4).

Advertisers to see your ads available on the Internet has two options: either contact the ad network to see their advertising campaigns displayed a wide network of publishers (5) or directly contact the publisher (6).

When a user surf in a publisher website (1) this website displays multiple ads that make up the page; if it is interested in any product advertised in the ad, the user does

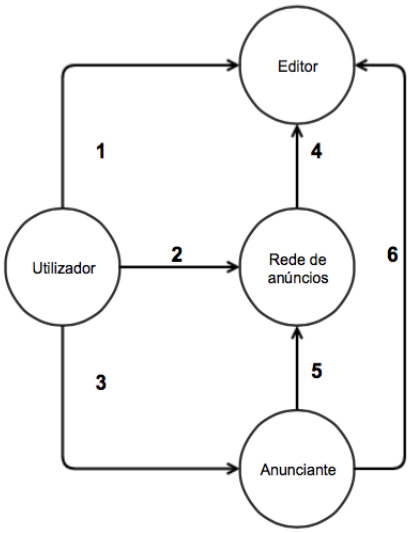


Fig. 1. PPC business model

click on the advertising banner. When performed click, the ad network registers the click (2) and finally the user is redirected to the advertiser's page (3).

The network ads typically holds 30% of the value of a click, as the remaining amount (70%) is paid to the advertiser. For example, when a user browses a website of a publisher and click on advertising ad of brand X, the owner of that brand pays to the ad network the value of this ad click and the ad network pays a fraction this value to the publisher. This type of business model is called Pay Per Click (PPC) [5], It is not the only business model in the online advertising world [6]. There are many other. Cost Per View (CPV) it is charged to the seller an amount whenever your ad'e displayed; Cost Per Lead (CPL), in this model,

the advertiser is only charged if the user creates a lead for him. Example: if the user chooses to sign up in a advertiser website.

The ecosystem of online advertising is not simple. Traffic can either flow directly from publisher to advertiser or pass through a wide variety of intermediates. The **subsyndicators** are website that receive ads by others publishers **syndicators** [7]. For example, the website x has an agreement with an ad network, he sends keywords to the ad network and it returns ads related to those keywords. So website x show ads on your website. Webesite sub-x does not have a contract with any ad network, so, in order to advertise in its page it sends key words to website x. Website x then forwards those key-words to the ad network which gives back an ad list to website x. Website x will then forward said list to website sub-x. In this way, the website sub-x can show ads. For example, Google and Bing provide ads to other search engines like InfoSpace and Yahoo [8] [9]. InfoSpace and Yahoo show ads on your websites but additionally provide those ads for smaller networks.

# 2 PARADISE

This chapter describes the general structure of the click fraud detection system PARADISE.

## 2.1 Modules

Our solution settles into an independent multi-modular system. The goal is to divide a complex project into smaller pieces (modules). The used modules will be the following:

- Browser Checker: validation of the user's browser.
- Data Mining: validation of the user's behavior through machine learning techniques.

These 2 modules will be used to detect two kinds of bots: simple bots and complex bots. We say a bot is simple when it doesn't have the capacity to stage a real browser or to simulate human behavior. Whereas a complex bot can put on some features of a modern browser, such as execute DOM methods. These bots can also replicate human behavior, like surfing the website's pages, do several clicks and interact with several of the site's links.

## 2.2 Browser Checker

This module aims to certify the user's browser. In order to do so, it's necessary to check some of the browser's characteristics. Those will be later examined and the click made will be labeled as real or not.

This module is divided in two components:

- Data collector: this part takes place on the client's side, where is gathered all the information needed to the proper functioning of the module.
- Analysis: all data will be processed and then the click is classified as being made by a bot or by a human.

### 2.2.1 Data collector

This component is on the server side but is executed in the client. The publisher website needs to import a JavaScript library that's kept in storage at our server. The data is sent in Json format. In addition to data collecting, this component is also responsible for testing specific browser components sent by the server. The data collecting component captures all the clicks made by the user, as for its coordinates.

### 2.2.2 Analysis

The browser is a program whose primary function is to obtain, process and present information resources available on the World Wide Web. The browsers usually communicate with web servers, where are hosted the webpages the users see. The webpages are written in several languages, such as HTML, XHTML, HTML5, haml and JavaScritp. In order to properly envision websites, the browser must support all the languages correctly. The document object module (DOM) is a multi-platform convention and independent of language, serving as an interface between JavaScript language and the objects of Markup (XHTML, HTML5,. . . ). This component is mandatory in a modern web surfer, will be evaluated by PARADISE. The goal is to understand if the browser supports several of the DOM functions, in order to distinguish between a real browser or just a bot surfing the website. One simple way to stage a visit in a browser is to add the User-Agent field with the name and version of a known browser. This module aims to check if the user's browser is real or not. To perform this operation successfully we'll exploit the aforementioned characteristics, in addition to other relevant criteria, in order to ascertain a valid click. This module intends to identify simpler bots, once more advanced ones fully mirror the full behavior of a real modern browser. The data mining module will be used to evaluate these more complex bots. To avoid false positives, all the tested characteristics are present in the 5 most used Internet browsers: Chrome, Firefox, IE, Safari e Opera. Additionally, we'll test specific characteristics associated to the browser in question. To validate the user browser, these properties must be tested:

- Check if the user clicked on the ad banner.
- Check if he moved the cursor to the ad banner.
- Check 3 browser characteristics.

The first test checks if the user really clicked on the advertisement banner. It's quite a basic test, but there's several kinds of bots that just parse the website's html and follow the ad link, not actually clicking the advertisement. If the user fails this checkpoint it's automatically identified as a bot, if not, the next test will take place. This next module gathers all the clicks made by the user. In order to pass, the user must make a click in the advertisement banner. This module aims to identify if the position of the user's cursor shifts towards the ad banner upon arriving to the page that contains the actual ad banner. To pass this test, the click coordinates must intercept the area that contains the banner. Like in the previous test, we want to assure that the click isn't made by a html parser. If the user leaves the publisher's page without clicking the ad, all the data will be discarded, and by entering in a new ad page, data will be collected, and so forth and so on. As previously, the user

TABLE 1
Challenges - HTML DOM

| HTML DOM | Chrome | IE | Firefox | Safari | Opera |
|---|---|---|---|---|---|
| anchors | Yes | Yes | Yes | Yes | Yes |
| images | Yes | Yes | Yes | Yes | Yes |
| URL | Yes | Yes | Yes | Yes | Yes |
| domain | Yes | Yes | Yes | Yes | Yes |
| getElementById() | Yes | Yes | Yes | Yes | Yes |
| applets | Yes | Yes | Yes | Yes | Yes |
| doctype | Yes | Yes | Yes | Yes | Yes |
| forms | Yes | Yes | Yes | Yes | Yes |
| ElementsByName() | Yes | Yes | Yes | Yes | Yes |
| querySelectorAll | 4.0 | 9.0 | 3.5 | 3.2 | 10.0 |
| baseURI | Yes | No | Yes | Yes | Yes |
| documentMode | Yes | 8.0 | Yes | Yes | Yes |
| readyState | Yes | Yes | 3.6 | Yes | Yes |
| renameNode() | No | No | No | No | No |
| ElementsClass() | 4.0 | 9.0 | 3.0 | 3.1 | 9.5 |
| head | 4.0 | 9.0 | 4.0 | 5.0 | 11.0 |
| activeElement | 2.0 | 4.0 | 3.0 | 4.0 | 9.6 |
| scripts | Yes | Yes | 9.0 | Yes | Yes |

can only take the next test round if it passes this one too. The third test sets to target several browser characteristics in order to understand its authenticity. When a click is made on an ad, an HTTP request is sent to the server where that module is located. The server randomly picks 3 challenges that the user must pass, and those tests can be specifically targeted to the used browser or just a general browser check. There's 18 challenges in total, 9 of which browser specific tests. Lastly, the user sends the response to the server, and if one of those tests raises a red flag the user is identified as a bot. A real browser will be able to correctly pass the 3 browser validation challenges, and if such happens, it'll be classified as a real browser. A challenge completed successfully matches a DOM method being correctly executed. However, some click fraud bots won't be able to properly pass all the hoops being immediately classified as a bot. The table 1 show the list of challenges used, as for the used traits used to evaluate the user's browser. All the characteristics featured in the table's first entry are supported by the most common Internet browsers, either for the mobile version, as for the desktop one. The following entries consist in singular characteristics of specific browsers.

The test that checks if the user moved the cursor towards the ad banner is a feature just for non-mobile devices. This is due to the inability to capture all the tablet and smartphone coordinates, once these devices don't have a mouse or a trackpad.

## 2.3 Data Mining Module

This module was designed to collect multiple data regarding the publisher's website users, by using data mining and machine learning techniques to classify the user. Classifiers are an effective way to spot click fraud attacks in advertisements, particularly clicks made by more sophisticated means. These automatic learning systems allow the fraud defense system to perfect and increase its detection performance. All user data gathered is analyzed by our system in order to identified if the user is a bor or not. To do so, we've divided this module into two components:

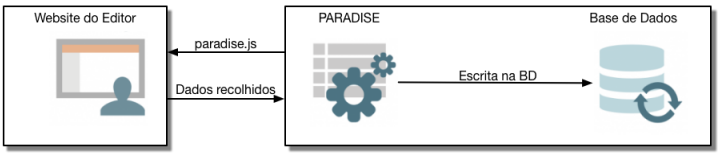


Fig. 2. PARADISE System

- Data collector: Gathers the information regarding the user surfing the publisher's page, and normalizes the data.
- Analysis: Processes the data gathered previously.

The reason why we've made this separation lies on the two completely distinct features the present in this module. By separating them, we've assigned different responsibilities each.

### 2.3.1 *Data Collector*

This component targets multiple user features, so that they can be later examined.

Every time a user gets to the publisher's website several kinds of information are collected but its storage only takes place if the user clicks on the ad banner, otherwise, all data is discarded. Afterwards, the stored data will be analyzed by our next component.

All the data collecting code was developed in JavaScript and it's based on the server where our anti-fraud clicking system (PARADISE) is stored. This will be thoroughly explained in section 3. The publishers must import this remote JavaScript file to their website, so that our server can access the information as it's shown in figure 2.

We've decided to only collect data whose modification is complex or impossible.

In the beginning, we selected the highest amount of data possible, but unfortunately we realized that this information could be easily manipulated by more complex bots.

All this data belongs to the application layer and is possible to forge [10]. For this reason, we decided to discard them, since we don't want to have our machine learning system with compromised data. An example of this is the user agent, a highly used characteristic in click fraud detection systems. However, it can be manipulated because there are several browser plugins that allow for it to be changed. This leads us to the conclusion that a bot can easily change this field too.

Once the data is in different scales, we need to standardize it. The pre-processing stage has an important part in the identification of click fraud attacks. To normalize continuous data, we use the following formula. It ranges the standardization between a number from 0 to 9.

$$X' = a + \frac{(X - X_{min})(b - a)}{X_{max} - X_{min}}, a = 0 \wedge b = 9$$

The analysis component will only be able to use the data in the machine learning system once all the information is standardized.

#### 2.3.2 Analysis

This component classifies normalized data from the previous component, through consistent classification models. We'll use an unsupervised method to cluster our data. This choice was due to the fact that our evidence was from different groups and they're not completely identified. Our training dataset is composed by the following data:

- Simples Bots (no human behavior)
- Complex Bots (with human behavior)
- Real Users

In the training phase all the information is heterogeneous and unidentified. The ultimate goal is for each cluster to describe a kind of user or bot, a classification explained after the training phase. The algorithm chosen to solve this problem is k-means. It was chosen once it's an unsupervised algorithm.

We chose an unsupervised one once we don't know the different kinds of behavior we will encounter, and such algorithms don't require that info: they alone separate the samples in classes. This is because users have different behaviors in different websites. An Youtube user has a completely different behavior than a Wikipedia one. The k-means is also a quite fast algorithm and not intensive, memory wise. Another reason to use this algorithm is that we know in advance the number of clusters the output will have. The k-means is a clustering algorithm that forms clusters based on the entities' characteristics. By applying the algorithm, we'll be splitting the sample in k homogeneous groups, so that each cluster's elements are the most similar possible. As for elements in different clusters, they're expected to be less similar compared to elements in the same cluster. The k-means uses the euclidean distance to make this division into groups by the centroid criteria. The algorithm will be able to add the users and the bots in the appropriate clusters. Users or bots from the same cluster are more similar amongst themselves than the ones from other clusters.

The k-means works in the following manner:

1) Estimate the k by using the silhouette analysis.
2) Produce centroids according to the number of clusters (K). All the centroids are different and created differently.
3) Calculate the distance between centroids and each point.
4) Assign each point to the nearest cluster (closest centroid).
5) Recalculate the new centroids in each cluster.
6) Stop if the centroids don't change, otherwise, repeat step 3.

The k-means simply makes clusters from users/bots, it doesn't give any indication on what kind of cluster it is. This decision must be made by heuristics or by us. The number of clusters is defined by us. To select the number of clusters (k), we use the silhouette analysis. This method interprets and validates the cluster consistence by assigning a score. The silhouette method shows a good performance in several occasions

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

After the data is splitted amongst k clusters by the k-means algorithms, for every given i, a(i) is the dissimilarity average of i and of all the entities in the same cluster. As for b(i), it stands for the minimum dissimilarity average of i and of other entities in each cluster that i is not a member. When the silhouette coefficient shifts towards +1, it means the clusters in that dataset are well classified. The 0 value means that the entities might be in that cluster as well as in any other. If the silhouette coefficient is negative, it tells us that the entity is wrongly classified. In our answer to find the appropriate number of clusters, we've tested s(i) to different values of k. This analysis is thoroughly explained in the section 4.4.3.

### 2.4 Module execution

Our system tries to save up the most resources possible without affecting the performance of the fraud click detection. The first module to be executed is the Browser Checker. Through several browser authentication techniques this module validates when a browser is real or not. This module is the first to be executed, once it doesn't require such high resources as the data mining module. The goal is to filter the highest possible number of fraudulent click without using much more complex techniques such as machine learning and data mining. Thus, through a set of heuristics defined, this module filters the most fraudulent clicks possible. Therefore, the data mining module just needs to use its resources to track the most advanced click fraud attacks. The browser checker and the data mining modules don't communicate with each other, with just the database acting as an intermediary. Both these modules are executed remotely, and totally independently. In case one of the modules fails, the other can assume the total responsibility of analyzing all the click fraud attacks. Although, there's a clear trade-off in this event, once the browser checker module doesn't have the capacity to identify more complex attacks, thus, all those attacks will remain unnoticeable. In the other scenario where it's the data mining module that's fully responsible to detect fraudulent clicks, the system would consume an high number of resources once the amount of attacks to analyze would be much bigger. In this last option, that would be wasteful. We guarantee that the system still runs if a module fails, thanks to this architectural design.

## 3 Implementation

This chapter addresses the key choices and decisions on the implementation of the Paradise system.

### 3.1 Database

The Paradise system requires a database to enter information concerning the various users who use the publisher's websites. We will use a NoSQL database.

The NoSQL databases are designed to support large data sets, including hundreds of millions of entries. Most NoSQL database architectures can scale horizontally, this means that when we have a scalability problem we don't need to change to a larger and more robust server. We can simply add another node to our architecture by distributing

the load among them. Unlike relational databases, NoSQL does not use ACID properties.

That being said, our next challenge is to identify which database is more suited to our system within that set of non-relational databases.

MongoDB is a document oriented database system. This system breaks with the traditional structure based on relational tables in favor of JSON format documents. MongodDB includes the operations create, read, update and delete (CRUD) API so it has the ability to avoid the delay of advanced queries. This database is also characterized by its strong consistency, which assures that if a write was successful we will always get the correct value. This is a key requirement since Paradise is a security system and has to be as accurate as possible in identifying bots.

This happens because MongoDB uses a single-master system and will always read data to the primary node. This database also provides partition tolerance which means that the system continues to operate even if there are communication problems. MongoDB limits the documents to have a maximum size of 16MB.

Due to the reasons mentioned in this section we decided to choose the MongoDB database to implement the PARADISE system.

### 3.2 Web Application

To implement the server we decided to use Node.js. Node.js is a platform to develop server-side web applications. Its goal is to be a web server with high capacity scalability.

Node.js runtime environment is Google's V8 and JavaScript engine . Google V8 is a JavaScript interpreter developed by Google and also used in the Google Chrome browser. Node.js uses an asynchronous model, I / 0 non-blocking and event-driven. This architecture is designed to optimize the throughput and scalability of Web applications. The tool was already tested in various scenarios and used in companies like netflix, paypal and linkedin. Node.js is a platform known to support tens of thousands of concurrent connections on a single machine, ideal for web applications in real time. This fits perfectly in the architecture of our system, since Paradise is a security system that receives information from multiple users in real time.

As we are using Node.js, we'll use the same language for client, server and database. And because MongoDB is also implemented in Javascript, we can keep the data in native JSON since we send it from the browser until it writes in the disk (database). Node.js has a very active community, with over 70.000 libraries registered for Node.js.

In the Node.js server are all the JavaScript files that many publisher's websites have to import, so the server has access to user information. This server also has the responsibility of doing all the insertions in the database.

### 3.3 Data Transfer

In this section we will explain how to transfer data between the user's computers and our server. This is a key point, since the data collection component of both modules needs to perform this transfer. All collected data is transferred to the server and then inserted into the database through socket.IO. Socket.IO is widely used for web applications in real-time, such as instant messaging software, collaborative documents and data analysis applications. From the technical point of view, our detection system can be considered a real time application, since the user data collection is done in real time.

The IO socket allows bidirectional communication between clients and servers. It consists of two parts: a library on the client's side that runs on the browser and a library that runs in node.js. Both components have an identical API. One of this library's features is the fact that it's compatible with most browsers, both on mobile and non-mobile devices. This was one of the main reasons why we used it, since it covers a wide range and versions of browsers.

### 3.4 Data Mining Module

The data mining module monitors the database so that whenever a new user is entered our system is notified. In our system a new user corresponds to a session. This is done through a thread that is alert to the collection where the data is stored. When a new user is inserted it is detected by the module and the data is normalized by this component as described in the previous section.

After the data is normalized, the system uses the k-means algorithm to classify the user and it will predict the cluster where the user is. If the user falls into the set of clusters defined as humans, the user is classified as real. Otherwise, it is classified as a bot. The library we use for data mining and machine learning is scikit-learn. Scikit-learn is a learning library for the Python programming language. It is an open source library. After the user is finally classified, a change in the document is made indicating whether the user is a bot or not. After that, the document is saved in the database.

The transfer of data between the client and the data mining module is done using the socket.IO library, explained previously.

When a user enters a publisher's website, an IO socket is automatically opened and a unique ID is created on the server side to identify the client. We don't use cookies to identify users because there are multiple attacks using cookies. Due to this, our user identification is made by opening a socket, and when the user closes the website window the socket is also closed. If the customer re-enters the website, a new ID is generated on the server side and the new data is captured. All data captured from the clients is sent in JSON format to the server, which eases the insertion into the database. Data transfer to the server is done in real-time whenever one of the events defined in the architecture takes place. For example, when a user navigates to a new page the server is immediately informed about the page where the user is. If the user makes a click on the publisher's page the server also receives this information immediately.

### 3.5 Browser Checker Module

When there is a click on the ad, a request is sent from the client to the server asking for a set of tests. The request is sent through the socket IO library. When the server receives this request it searches for three tests in the database and sends it to the client. The server takes into account the browser version, looking for specific tests for each version.

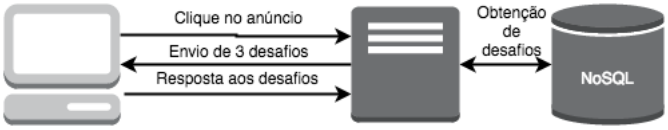


Fig. 3. Browser checker

For example, the server can send tests that only work in that browser version or it can send tests that the browser can't solve. The client sends the test responses to the server. Finally, the server checks whether the tests are correct and if one fails the customer is identified as a bot. All messages exchanged use the bidirectional channel on the TCP protocol. This protocol guarantees the orderly delivery of the messages as we can see in figure 3.

### 3.6 Execution Order

The entire system shares the same database but each module has a collection associated with it. A collection in MongoDB is where data is stored. It's the correspondent to a table in the relational model. Whenever a user logs on a website an entry is added to each of these collections and these two entries share the same unique identifier. Every time the Browser Checker module identifies a user as a bot, it deletes the user entries in both collections and the user is classified as a bot.

If the user is classified as a non bot by this module, it will go to the data collection of the data mining module and will change a field so that the Data Mining module is able to classify it. This is done through a Boolean and when the boolean is true, the Data Mining module can classify the data.

The Data Mining module monitors the collection associated to itself but only treats data when the Boolean mentioned earlier is true. When this module classifies a user the data is deleted from the two collections. If the Data Mining module runs out of data to sort it, it will send a message to the Web Browser Checker module. If the module answers it will wait for new data to use, but if it doesn't answer it treats the data that is in the collection even if the Boolean is not true. Both modules handle the data in order of arrival as a FIFO system.

## 4 Evaluation

In this section we will evaluate Paradise using some metrics. The success of the click fraud detection system is determined by its ability to detect users that are associated with click fraud in an effective and correct way.

### 4.1 Setup

Throughout the evaluation phase all modules from the Paradise system, database and publisher website were hosted in the cloud. The chosen cloud was Digital Ocean because it offers server credits for college students. Unfortunately, the credits provided were only enough to use in one machine. The machine we used had the Ubuntu operating system version 4.14 installed and the version of MongoDB was the 3.2.4. This machine had 1GB of memory, 30GB disk and a processor with one core. The machine was located in London.

TABLE 2
Datasets

| Dataset | Number of clicks |
|---|---|
| Bots simples | 100 |
| Bots complexos | 900 |
| Utilizadores reais | 74 |
| Total | 1074 |

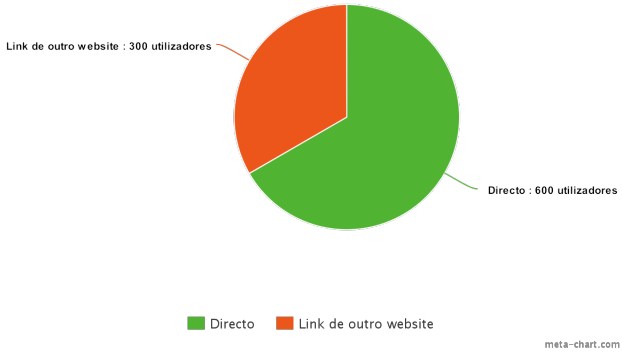


Fig. 4. Complex bots referral

### 4.2 Publisher Website

To make an evaluation as close to reality as possible, we created an publisher's website. The aim was to create an advertising campaign on this website and use the Paradise system in order to monitor all clicks made. The website address was www.bananas.live. This website contained many articles about bananas, as well as videos and images. The content was written in english and the website had about five pages, but only one of them had ads.

Bananas.live was created using only HTML, JavaScript and CSS and it had our security library imported. The banner advertising is located on the home page. This page includes an article about the nutritional facts and benefits of bananas. The advertising banner has the dimensions 300 by 250 pixels. When the user clicks on the ad, he is redirected to a landing page.

### 4.3 Datasets

To evaluate the modules of the PARADISE system it was necessary to create multiple datasets. Each dataset specifies a certain user and a different simulation. There are three datasets collected separately:

- Simple bots Dataset: a set of simple bots in order to test the Browser Checker module;
- Complex bots Dataset: this dataset consists of a set of complex bots capable of simulating human behavior.
- Real users Dataset: real visitors interested in the ad.

Table 2 contains a summary of the number of clicks made for each scenario. A total of 1 074 fraudulent clicks was generated. The figure 4 shows complex bots referral. The figure 5 and 6 shows referral and the platforms used by real users.

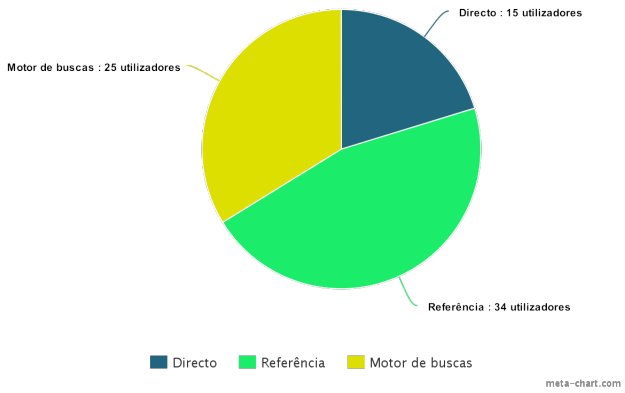


Fig. 5. Real users referral

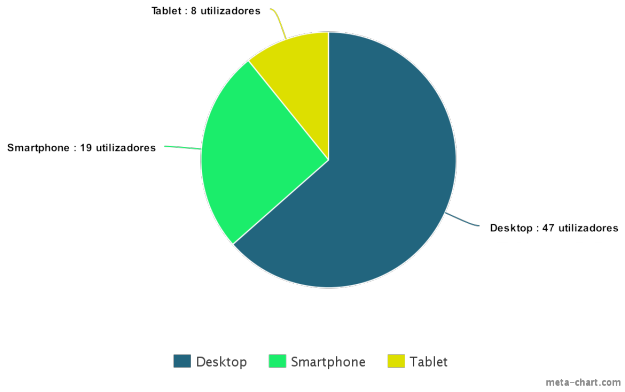


Fig. 6. Real users platform

### 4.4 Experimental evaluation

In this section, we present the metrics used to evaluate the proposed architecture. We will also explain how each module was tested and what were the metrics used to evaluate each module.

#### 4.4.1 Metrics

The evaluation of the proposed architecture was made experimentally, building a prototype. The benefits of each one of the two modules will also be evaluated.

Our proposed classifier faces four possible outcomes. If the click is fraud and is classified as fraud, it counts as a true positive (TP) If the click is legitimate and is classified as legitimate, it counts as a true negative (TN). If the click is fraud and is classified as legitimate, it counts as a false negative (FN); If the click is legitimate and is classified as fraud, it's a false positive (FP). We will evaluate the system with the following formulas [11]:

$$\text{True Positive Rate} = \frac{TP}{P}$$

$$\text{False Positive Rate} = \frac{FP}{N}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

TABLE 3
False positive users

| ID | ISP | Browser |
|---|---|---|
| 45 | ovh hosting inc. | Chrome 29.0.1547.41 |
| 47 | ovh hosting inc. | Chrome 29.0.1547.41 |
| 20 | ovh hosting inc. | Chrome 29.0.1547.41 |
| 5 | Sprint | Maxthon 3.4.5.2000 |

$$\text{Accuracy} = \frac{TP+TN}{P+N}$$

Precision is the proportion of true positives among all positive results, both true positives and false positives. Accuracy, on the other hand, is defined as the proportion of true outcomes, both true positive and true negative, of the total number of cases analyzed. An accuracy of 100% indicates that the measured values are exactly the same as the actual values.

#### 4.4.2 Browser Checker Module

The datasets used to evaluate this module were the real users dataset and the simple bots dataset. We did not use the complex bots dataset because they are not identified by browser checker module. The module is used for the collection of data about actual users and simple bots On the simple bots dataset, the system was able to identify all users as bots. Of the 74 real users, 4 were identified as bots, which caused some surprise.

Through these data we calculated the precision and accuracy values and evaluated our system. The equations below show the results of these calculations.

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{100}{100+4} = 0{,}97$$

$$\text{Accuracy} = \frac{TP+TN}{P+N} = \frac{100+70}{100+74} = 0{,}98$$

The number of the false positives is small. We decided to make an individual analysis of the traffic of these 4 users. Table 3 shows the false positive users.

Looking at the first three elements we can conclude that the user's ISP is OVH hosting inc. This ISP is a bit strange since OVH is a service cloud provider, providing dedicated servers and cloud hosting. It's possible that this user is not a real user but a bot instead, since the traffic is coming from a cloud. We can't be absolutely sure because it can also be a real user using a VPN or a OVH server.

Analyzing the last element, we find out that the user used the Maxthon browser, which is a freeware browser developed by a Chinese company. We believe this browser is not compatible with socket.IO library and that's why it was not possible to classify the user correctly.

The results with the simple bots dataset are very positive because this module only works as a filter. This module's main purpose is ensuring that the data mining module doesn't have to waste resources analyzing this type of bot.

#### 4.4.3 Data Mining Module

In this subsection we'll describe the two stages that make up the classification of users by our classifier. The first phase is called the training phase and consists of teaching our classifier to build a robust model capable of identifying click

TABLE 4
Training Dataset

| Dataset | Sessões |
|---|---|
| Utilizadores reais | 49 |
| Bots complexos | 600 |
| Total | 649 |

TABLE 5
Silhouette Score

| K | Silhouette score |
|---|---|
| 2 | 0.65 |
| 3 | 0.68 |
| 4 | 0.71 |
| 5 | 0.71 |
| 6 | 0.85 |
| 7 | 0.87 |
| 8 | 0.84 |
| 9 | 0.85 |

TABLE 6
Silhouette score rule

| Silhouette range score | Interpretation |
|---|---|
| <0.26 | No substantial structure has been found |
| 0.26 - 0.50 | The structure is weak and could be artificial. |
| 0.51- 0.70 | A reasonable structure has been found |
| 0.71-1.0 | A strong structure has been found |

TABLE 7
Training phase

| | Desktop | Smartphone | Tablet | Bots | Total |
|---|---|---|---|---|---|
| Cluster 0 | 3 | 0 | 0 | 305 | 308 |
| Cluster 1 | 3 | 0 | 0 | 292 | 295 |
| Cluster 2 | 0 | 7 | 3 | 0 | 10 |
| Cluster 3 | 12 | 0 | 0 | 0 | 12 |
| Cluster 4 | 17 | 0 | 0 | 0 | 17 |
| Cluster 5 | 0 | 1 | 3 | 3 | 7 |
| Total | 35 | 8 | 6 | 600 | **649** |

fraud attacks. In the classification phase, the second one, we will predict the results using the model we built previously.

#### 4.4.4 *Training stage*

At this stage, that has the classification stage name, the aim is to train our classifier to be able to identify clusters related to click fraud. Subsequently, the classifier can be used to predict future clusters objects. For this, we will use a set of data known as the training set. In our case, our training set consists of two datasets: the complex bots dataset and the actual visitors dataset. Our training dataset consists of two thirds of each of these datasets. As we are using an algorithm of unsupervised learning, the data is not identified, it's just for us. Table 4 shows the number of data taken from each dataset.

After the training data set is complete, it's necessary to indicate the cluster number (k) to the k-means algorithm. To choose the optimal number of k, we made a silhouette analysis for k values between 2 and 9. We chose these two values because we know that there can be at least two clusters (bot and real visitor) and chose the value 9 because there are 9 criteria. Table 5 shows the silhouette score for different K.

We will not choose the value of K that corresponds to the higher silhouette score, because a greater number of silhouette corresponds to a larger number of clusters and we know that we just need 2 to make the system work. What we want to find is the smallest K within a good silhouette score. Table 6 shows the rule used to choose the best silhouette value. To choose a K, the silhouette score value must be greater than 0.70. The k values that meet this requirements are 5,6,7,8 and 9. We won't choose k = 5 but instead k = 6 because we think that the difference between the two is substantial, and with k = 6 we have a stronger structure. After this process is completed we run the k-means algorithm for k = 6. We can see that the clusters 0 and 1 are constituted essentially by bots. There are 6 humans inserted in these two clusters. Analyzing these 6 users we conclude that all of them use desktop and that the traffic to the publisher's website is always direct.

In the classification stage, all elements included in the cluster 0 and 1 are classified as bots. Clusters 2, 3 and 4 are composed solely of real users. All cluster 2 users are using mobile platforms such as smartphones or tablets. All cluster 3 users use desktop and the source of their IP addresses is the United States of America. The traffic is referenced by search engines or another websites. All cluster 4 users are desktop and none of them has an IP address of American origin. Users in this cluster have little interaction with the website compared with the previous cluster. We were able to conclude this by checking the number of page views, time spent on each page and mouse cursor moves. For the classification phase, all elements included in cluster 2, 3 and 4 will be classified as real users. The cluster 5 is composed of four human and three bots. All humans in this cluster use mobile devices (tablets and smartphones) and viewed only a website page: the page where the banner advertising was located.

Since all humans in this cluster use mobile platforms (smarphone and tablet) their movements are zero. The 3 bots present in this cluster resemble a bit the previous users, since they make very few moves with the mouse and visited only one website page. The traffic of all elements in this group is direct.

Due to the fact that this cluster has a similar percentage of bots and humans, all the users that are classified in this cluster will have to be analyzed manually. Because of this, the cluster doesn't identify whether the user is human or a bot. Since this is the cluster with fewer elements, the analysis will not be too costly. Table 7 shows the users of each cluster.

#### 4.4.5 *Classification Stage*

In this stage we'll evaluate the ability of our model in targeting click fraud attacks. To do so, we'll use the 2 previously described datasets, but we'll also use the rest of data. The table 8 shows some characteristics of the test set. For every input in this dataset, the k-means algorithm will predict on which cluster the user will be classified. As we've seen previously, if the user is assigned to the cluster 0 or 1 is considered a bot, if he goes into the cluster 2, 3, or 4 is considered to be human. Lastly, if the user is classified as a cluster 5 user, a manual verification is needed. To make a more thorough analysis of our system, we'll make two tests,

TABLE 8
Test dataset

| Dataset | Sessions |
|---|---|
| Utilizadores reais | 25 |
| Bots complexos | 300 |
| Total | 325 |

TABLE 9
Users of each cluster

| | Desktop | Smartphone | Tablet | Bots | Total |
|---|---|---|---|---|---|
| Cluster 0 | 0 | 0 | 0 | 146 | 146 |
| Cluster 1 | 2 | 0 | 0 | 151 | 153 |
| Cluster 2 | 0 | 7 | 2 | 0 | 9 |
| Cluster 3 | 7 | 1 | 0 | 0 | 8 |
| Cluster 4 | 3 | 0 | 0 | 0 | 3 |
| Cluster 5 | 0 | 3 | 0 | 3 | 6 |
| Total | 12 | 11 | 2 | 300 | **325** |

one for when the assignment to cluster 5 is related to the click fraud, and another one when it's not.

Our datasets are unbalanced, once the complex bot is 12 times superior to the ones of real users. To get a robust processing of our classifying metrics, it's important that both classes are represented evenly. Once we're using 2 datasets, we've divided the larger dataset (complex bots) for the smaller (real users). The goal is to find the common multiplier. Thus, in the metric processing we multiply this value to the sample value to balance the datasets.

That being said, we've used the k-means algorithm to predict the output of our dataset. The table 9 shows how each cluster is structured after the algorithm is completed.

Looking at Table 9 we can conclude that cluster 0 is exclusively composed by bots, and cluster 1 mostly too, except the 2 users classified as bots incorrectly. The cluster 2 is characterized by users that use mobile platforms like tablets or smartphones. In its majority, the cluster 3 is made by users that use a desktop, except for the one that uses a smartphone. The users in cluster 5 are splitted amongst bots and humans, with a 1 to 1 ratio.

In this test we'll consider that the clusters 0 and 1 are made by bots, and the clusters 2, 3, 4, and 5 are made by real users. The table 10 is a confusion matrix. In this matrix we can find the real and false positives, and also the real and false negatives. The TN and FP values already take into account the multiplying factor. This table allows to visualize the performance of our algorithm.

With the data from this table we've calculated the precision, accuracy, rate of false positives and rate of real positives. These values will help us evaluate our system. The equations below show the results from these

TABLE 10
Confusion Matrix (Cluster 5 = humans)

| n=325 | Predicted: No | Predicted: Yes | |
|---|---|---|---|
| Real: No | TP 297 | FN 3 | 300 |
| Real: Yes | FP 24 | TN 276 | 300 |
| | 321 | 312 | |

TABLE 11
Confusion Matrix (Cluster 5 = bots)

| n=325 | Predicted: No | Predicted: Yes | |
|---|---|---|---|
| Real: No | TP 300 | FN 0 | 300 |
| Real: Yes | FP 60 | TN 240 | 300 |
| | 260 | 240 | |

calculations.

True Positive Rate = $\frac{TP}{P} = \frac{297}{300}$ = 0,99

False Positive Rate = $\frac{FP}{N} = \frac{24}{300}$ = 0,08

Precision = $\frac{TP}{TP+FP} = \frac{297}{297+24}$ = 0,925

Accuracy = $\frac{TP+TN}{P+N} = \frac{276+297}{300+300}$ = 0,955

After these calculations are complete, we've repeated the process, except this time we classified the cluster 5 as a click fraud cluster. The goal is to compare both and draw conclusions from the classifier's performance. The tabela 11 shows the new confusion matrix.

That being said, we've repeated all the metrics calculations. The equations below show the results obtained.

True Positive Rate = $\frac{TP}{P} = \frac{300}{300}$ = 1

False Positive Rate = $\frac{FP}{N} = \frac{60}{300}$ = 0,2

Precision = $\frac{TP}{TP+FP} = \frac{300}{300+60}$ = 0,833

Accuracy = $\frac{TP+TN}{P+N} = \frac{300+240}{300+300}$ = 0,900

The accuracy is calculated using right classifications (real positives and real negatives). We've found that we can get better accuracy if we state that the cluster 5 belongs to human users. This happens once the TN value raises from 240 to 276 while the TP value just rises from 297 to 300. By stating that the cluster 5 belongs to humans, we're increasing accuracy. This happens because some bot clicks end up on this cluster due to having similar traits with humans.This only applies if the number of human clicks are assigned to the right clusters.

The metric rate of false positives is used to evaluate the performance from the portion of human clicks that were classified as bots. Considering the cluster 5 as human, we have a value of 0,08, and this value increases to 0,2 when the cluster is considered a bot. This is due to the fact that after the multiplication effect 12 is applied, the cluster 5 becomes majorly composed by humans, which increases a spike in false positives.

The rate of real positives is used to evaluate the number of invalid clicks classified as click fraud. The first observation we've made is that this rate is high in both tests. The maximum value is 1 and the minimum is 0,99. This happens once only 1% of the click fraud attacks go to the cluster 5.

The accuracy compares the false positives with the real positives, trying to capture the effect of a large number

TABLE 12
Paradise evaluation dataset

| Dataset | Sessões |
|---|---|
| Utilizadores reais | 25 |
| Bots simples | 100 |
| Bots complexos | 300 |
| Total | 425 |

TABLE 13
Matrix confusion

| n=425 | Predicted: No | Predicted: Yes | |
|---|---|---|---|
| Real: No | TP 392 | FN 8 | 400 |
| Real: Yes | FP 112 | TN 288 | 400 |
| | 304 | 296 | |

of negative examples in the performance of the algorithm, therefore, the higher the accuracy, the greater is the quality of the experiment. The precision when cluster 5 is related with click fraud is 0,925, and when it's not this value is 0,833, a 0,092 difference. The precision of both scores is rather good, showing that the system has a high consistency in detecting click frauds.

### 4.5 Paradise Evaluation

Throughout this section we present an evaluation of PARADISE. The metrics defined previously will be used to evaluate the whole system. In order to train the classifier we used the dataset described in Table 4; this dataset is comprised by complex and simple bots, as well as, real users. Table 12 show this dataset characteristics. As in last Section we will also use a multiplication factor of 16.

As shown in last sub section our system presents a better precision and accuracy when the cluster 5 is referenced to real users. Thus, the cluster 0 and 1 is related with click fraud and the clusters 2, 3, 4 and 5 are associated with real users. Table 13 is a confusion matrix. Here is presented true positives, false positives, false negatives and true negatives. Values VN and FP already include the multiplication factor. This table depicts the performance of PARADISE.

True Positive Rate = $\frac{VP}{P} = \frac{392}{400} = 0{,}98$

False Positive Rate = $\frac{FP}{N} = \frac{112}{400} = 0{,}28$

Precision = $\frac{VP}{VP+FP} = \frac{392}{392+112} = 0{,}778$

Accuracy = $\frac{VP+VN}{P+N} = \frac{392+288}{400+400} = 0{,}85$

We conclude that the system's precision decreased due to the high number of false positive values. The main reason on why this number is so elevated is a possible dataset contamination with bots. The use of a multiplication factor to balance dataset may not be the best way to resolve this issue. In order to achieve more trustworthy results the ideal case scenario was both the real users and bots dataset to have the same size.

Even with this problems our system shows good precision and accuracy levels.

## 5 Conclusion

This thesis describes the PARADISE system, a mechanism that detects click fraud attacks. Our goal was to create an architecture capable of identifying click fraud attacks in a smart way that would work on the ad network side. In our system, the user's browser is analyzed, as well as the user's behavior. Through experimental evaluation we can conclude that the two modules present good results and that we can apply our system to a real ad network. The evolution of click fraud attacks is increasingly fast and the attacks are becoming more advanced and complex to identify. Our system has two independent modules but additional modules can be added in the future to strengthen the system.

## References


[1] Martine Paris q3 2015 digital ad revenues climb. http://www.iab.com/news/q3adrevenue/. Date: 12-10-2015.

[2] Martine Paris google agrees to pay 90m for 'click fraud'. http://www.foxnews.com/story/2006/03/09/google-agrees-to-pay-0m-for-click-fraud.html. Date: 09-03-2006.

[3] Financial Times. mercedes online ads viewed more by fraudster robots than humans. http://www.ft.com/cms/s/0/788d6d42-da6c-11e3-8273-00144feabdc0.html . date: 26-05-2014.

[4] Incapsula. 2014 bot traffic report. https://www.incapsula.com/blog/bot-traffic-report-2014.html . Date: 18-12-2014.

[5] Boris Mordkovich and Eugene Mordkovich. *Pay-per-click Search Engine Marketing*. LULU, 2007.

[6] Bart Weller and Lori Calcott. *The Definitive Guide to Google AdWords*. Apress, 2012.

[7] N. Daswani and M. Stoppelman. The anatomy of Clickbot. A. *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, page 11, 2007.

[8] Microsoft, Yahoo! change search landscape. http://news.microsoft.com/2009/07/29/microsoft-yahoo-change-search-landscape/ . Date: 29-09-2009.

[9] Google Inc.google services agreement for infospace llc. http://www.sec.gov/Archives/edgar/data/1068875/000119312509042774/dex . Date: 12-03-2014.

[10] Deepti Rajwal, Deepali Band, and Prof Atul Yadav. Study Of Different Attacks On Network & Transport Layer specifically on transport layer . Transport layer use & UDP ( User Datagram Protocol ) and attacker takes help this two protocol working function for attack . TCP / IP is a most widely used protocol suit today . OSI model , which is responsible for the process to the sender . This is connection oriented protocol . A a transmission before sending of data . 2(3):692–695, 2013.

[11] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.