Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica

Facebook Friend Requests' Analysis: development of tools for data collection and analysis

Tesi di laurea

Relatore

Prof. Mauro Conti

Laureando

Giada Zuccolo

Anno Accademico 2020-2021

Non come chi vince sempre
ma come chi non si arrende mai.

— Frida Khalo

Sommario

This document describes the project developed during the internship period. The internship, which lasted about 320 hours, was carried out in smart working and on behalf of Professor Conti's Cybersecurity research group.
The topic is about studying the phenomenon of friend requests on Facebook and how an attacking profile with certain characteristics is more easily accepted by a victim profile with certain characteristics.
In the first place, the study of the environment was required to outline the requirements and objectives.
Then it was required the implementation of tools that allow the creation of attacking profiles, the search for profiles with certain characteristics, and the sending of the friend request.
The collected data then had to be analyzed in order to create a model of acceptance. This is because the final goal was the development of a tool called `"Zero-Effort Attack"`, which, given a victim profile, defines the characteristics that the attacking profile should assume to be accepted more easily.

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Mauro Conti, relatore della mia tesi, per aver creduto nelle mie idee ed avermi proposto di lavorare a questo progetto, ed insieme a PierPaolo, ringrazio per l'aiuto e il sostegno fornitomi durante il lavoro.

Voglio ringraziare più di tutti la mia famiglia per il costante sostegno e per essermi stati vicini in ogni momento, sia gioioso che difficoltoso, durante questi gli anni di studio.

Ringrazio poi i miei amici e compagni per questi anni passati insieme, da vicino e da lontano. In particolare ringrazio Sofia, per essere sempre stata una spalla e soprattutto un'amica su cui contare.

Ringrazio tutti i miei amici che sono qui a condividere questa gioia con me, soprattutto Clarissa, per l'appoggio che non mi ha fatto mai mancare e per la sua preziosa amicizia che per me è sempre stata un punto fermo.

Last but not least, ringrazio con tutto il cuore Andrea, per avermi aiutata a vedere ed a tirare fuori sempre il meglio di me anche quando io non ci riuscivo, per avermi supportato (e sopportato) fin dal primo istante e per tutto l'amore che mi dimostra ogni giorno.

Padova, Settembre 2021 Giada Zuccolo

Indice

Capitolo 1

Introduction

The exponential technological development and the internet have proliferated new criminally relevant conduct: new and more powerful IT tools have created new ways of committing a crime and have generated criminal phenomena.
For this reason, the area of primordial interest was the cybercriminal reality linked to the growing and hyperbolic development of the web and the technologies connected to it and how the types of crime (and criminals) and the most well-known and ordinary modus operandi have changed their physiognomy and have evolved with it.
Just think of online social networks and how they have an increasing impact on everyday life. Nowadays, social networks occupy a considerable part of daily life: they allow you to share links, photos, videos, thoughts, and to show your interest in some brands or companies. Many companies keep their customers updated through these channels, which can even be very useful for the growth of their customers, with shrewd and studied marketing skills. In a more circumstantial reality, social networks allow you to stay in touch with friends near and far and favor the birth and development of relationships even with unknown users.
It's precisely in this context that the project takes shape.
To start a relationship between two users, they must make contact, in some ways: generally or with a follow action (without some permissions) or with a friend request, where the user that receives the friend request has to accept (or not) the request to make contact with the other user.
In the case of the social network par excellence, namely Facebook, a profile A asks for friendship to a profile B, who can choose whether to accept, leave pending or reject.
In fact, the person behind a virtual profile could be unknown: a person may have created a profile that does not belong to any real person but simulating its existence, to get in touch with some specific users.
This mechanism has led to the birth of online crime aimed at targeting specific people who, initially unaware, find themselves becoming *victims*.
From here, a first project idea was born: put the focus from the attacker's point of view and then create a tool to analyze a victim and suggest to the criminal a series of information be used to create a profile that the victim would more likely accept.

1.1 Organizzazione del testo

1.1.1 Contents

Il secondo capitolo descrive ...

Il terzo capitolo approfondisce ...

Il quarto capitolo approfondisce ...

Il quinto capitolo approfondisce ...

Il sesto capitolo approfondisce ...

Nel settimo capitolo descrive ...

1.1.2 Typographic conventions

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

* gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
* per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*[g];
* i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

Capitolo 2

Description of the internship

In this chapter, there will explain details about the internship: the main idea behind the project, in what chronological order it will be carried out, and the final goal, i.e. the development of the "Zero-Effort Attack" tool.

2.1 A brief introduction to the project

The general idea is studying the phenomenon of friend requests on Facebook, and how a profile with specific characteristics is accepted more easily by another profile with certain characteristics.
These characteristics, chosen after in-depth research in the literature, are defined by parameters (e.g. age, occupation, profile picture), which are divided into classes (e.g. age's class are defined by ranges, such as "age between 18 and 50 years old" or "over 50 years old").
Therefore, there will be developed specific tools capable to create a fake attacker profile with precise parameters and then send a friend request from this profile to more than one victim profile.
The objective is to create a decided number of attacker profiles, based on the various combination of the parameters that we chose to analyze; then, all of these profiles must ask a friend request to a specific number of victim profiles which are categorized into configuration based on the various combination of the parameters that we have decided to analyze. In this way, we are sure that every type of attacker will ask at least one friend request to every type of victim.
Consequently, an amount quantity of data will be collect and analyzed, in such a way as to be able to define a valid *model of acceptance*.

This is necessary to satisfy the concluding goal: define a tool, called `"Zero-Effort Attack"`, which, after entering the URL of the victim, will determine the characteristics that the attacker should include in his profile to be more likely accepted by the victim.
The name `"Zero-Effort Attack"` marks the effort that will be made for the attack, that is 0.
In fact, the only attacker's *effort* is creating the profile on Facebook with the characteristics dictated by the tool, once the victim's profile has been indicated entering his Facebook profile URL.

2.2 Expected products

The student had to keep track of the work done and the progress made against the expected progress day by day.
At the end of the internship period, the student had to produce a written report that keeps track of the work done and describes the results obtained, in particular:

1. what has been learned from the study and research in literature;
2. the study, development, and execution of the tool for automating the search for a profile with certain characteristics;
3. the study, development, and execution of the tool for the automation of the creation of an attacking profile with certain characteristics;
4. the study, development, and execution of the tool for the automation of friend requests from a profile with certain characteristics to another profile with certain characteristics;
5. consecutive data collection;
6. an analysis of the collected data;
7. a model of acceptance.
8. `"Zero-Effort Attack"` tool.

Capitolo 3

Background

This chapter describes the main technologies used in this project: Python, Selenium and Facebook.

3.1 Web Scraping

For this project, the information necessary, that permits the collection of analyzable data, is situated on the Facebook pages: this information must be found and save.
To do it, the Facebook pages need to be scrape. This is called "web scraping", which means the technique and the technologies to extract data from a website.
Scrape manually every single day is a waste of time: a lot of time will be spent to click, scroll and search the desire information, but especially because the content of these pages could be updated very frequently.
For this reason, for this project, automated web scraping will be the bread and butter. The information will be collected every single day in order to have a big quantity of data to analyze in a second moment, but it will happen in an automated way: the code will be written only once, but it will be used for every needed page.
Thus, appropriate tools will be developed: they will scrape the pages with the information to permit to save them in the specific files.
But never forget that the data on the websites are unstructured: a tool that scrapes a web page must understand what kind of information have to scrape and collect, and then it must save them in a structured way, depends on the context.

3.2 Python language and Selenium

Python is an interpreted high-level general-purpose programming language. It's easy to read, thanks to its use of significant indentation.
Python is easy to learn, simple to code, and the clear syntax permit and easier maintenance. It supports modules and packages, which encourages program modularity and code reuse. Furthermore, a small code is enough to do large tasks.
Thus, Python is suitable for web scraping because of all these reasons and also because it has a huge collection of libraries that provides methods and services for various purposes, among which, libraries for manipulation of extracted data.
One of these libraries is called Selenium.
Selenium is an open-source web testing library, used to automate browser activities:

a browser-driver executes a script on a browser instance on a device, and this permits to automate and test web algorithms.
In this project, Selenium will be used to automate procedures that scrape Facebook pages in order to collect data from them.
More specifically, it will be used Selenium WebDriver.
Selenium WebDriver is a web framework used for automating web-based application testing to verify that it performs expectedly: it simulates the behavior of a real user within a browser.

3.3 Facebook

Facebook is an OSN, born in February 2004. It permits people to share content: Facebook users could create a personal profile, add other users as friends, and exchange messages. Additionally, users could share their status, news stories, notes, photos, videos, and allow their friends (or friends of friends) to comment on them. Furthermore, users may join common-interest groups, organize events, and create fans pages for a workplace/business, a school/college, or even a brand/product.
Right now Facebook is not as popular as it used to be, especially among the younger ones. Younger generations prefer to use others OSNs, like Instagram or Tik Tok: in fact, in recent times Facebook has seen the average age of its users increase.
But, despite this, it is currently still the most used social network.
Furthermore, Facebook is not a new area in literature, but compared to other OSNs like Instagram or Twitter, has many more analyzable factors, so there would be a lot more material on which to study the knowledge already known and critically examine the resulting data of this project.
In short Facebmake the world and the people more open and connected.
However, it is unavoidable that this platform may also provide incentives for criminals to carry out illegal activities.

3.3.1 Spotlight on safety in social networks

This project was born after reflecting on safety in social networks.
In an increasingly interconnected world, everything becomes virtual more and more rapidly, including crimes involving people themselves in the flesh, not just the data about them. The digital identity of a person today is the data that the person makes public.

This project wants to shed light not on the technological weaknesses that may exist, but on the people who are inside the Net, behind a virtual profile.
Facebook uses a policy system to try to prevent problems related to the safety of people on the platform: starting from the constraint of minimum age to register, ending to checks for criminal or inauthentic behavior.
Fake profiles, in fact, have always been a problem for all platforms that allow users to register. Nowadays, creating and then using a fake profile on Facebook integrates the crime of person substitution envisaged and governed by art. 494 of the criminal code even if you use a caricature image.
A person can create a profile and be real to themselves, but the same person can create a fake profile that identifies another person.
This study wants to highlight that behind a virtual profile you can never be sure of who really is and show the weaknesses of the average user who uses these platforms.

Capitolo 4

Literature search

The research in the literature and the study of founded projects was useful for the knowledge of the possible areas and above all, it was necessary to verify that no other project had as a focal point the attacker who wants to hit a specific victim and must make sure to be accepted as a possible friend.
Furthermore, these projects were useful for delineating the parameters from which to structure the attackers' and victims' profiles.

4.1 List of papers

Many papers are dealing with this social network, each of which opened to the perspective of many other projects. The most important and useful papers found in the great research phase are now shown and discussed:

1. INVESTIGATIVE TECHNIQUES IN THE DIGITAL AGE - CYBERCRIME AND CRIMINAL PROFILING
   In this paper, it is made clear how cyberspace offers the possibility of carrying out illicit acts with the perception of going unpunished, showing the new various types of crimes such as cyberstalking, cyberbullying, online sexual offenses, etc. Cyberspace has allowed the criminal to evolve the techniques and approaches to the victim, so much so that it is not always immediately obvious that the intentions of one account towards another can be malicious.
   This new world, moreover, places investigators to consider the crime scene differently: a crime born of a victim's approach on the web places the computer itself as a victim or as a witness, but above all as a crime scene and therefore as a space to be analyzed. that could help (or penalize?) the investigators.
   LINK: https://oapub.org/soc/index.php/EJSSS/article/view/821/1403

2. OSNs AS SUPPORTING EVIDENCE: A DIGITAL FORENSIC INVESTIGATION MODEL AND ITS APPLICATION DESIGN
   This paper states that digital crime investigations are performed without adequate guidelines, as there isn't a consistent standard and model, only a set of procedures and tools, and above all, there is no model built specifically for social networks. This paper, therefore, proposes a standard survey model to be used for social networks, incorporating existing traditional frameworks and strategies..
   LINK: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6125728

3. INTEGRO: LEVERAGING VICTIM PREDICTION FOR ROBUST FAKE ACCOUNT DETECTION IN OSNs
   The *Integro* project is a scalable defense system that helps OSNs detect fake accounts using a meaningful user classification scheme. This paper, in addition to providing an explanation of how *Integro* works, provides interesting explanations on the behaviors that differentiate the attacker and the victim.
   LINK: https://www.sciencedirect.com/science/article/pii/S0167404816300633

4. SOCIALSPY: BROWSING (SUPPOSEDLY) HIDDEN INFORMATION IN OSNs
   This paper highlights how current privacy settings in social networks are not as effective as users might think, focusing on Facebook. It shows how easy it is to retrieve information that a user thinks they have set as private.
   LINK: https://arxiv.org/pdf/1406.3216.pdf

5. EVALUATION OF THE LIKELIHOOD OF FRIEND REQUEST ACCEPTANCE IN OSNs
   This paper explains how OSN users often run into breaches or security issues due to rash acceptance of the friend request, which can lead to the disclosure of personal information and vulnerability to an attack. The document proposes a method to evaluate the probability of becoming a friend having defined a model data: a future friend and incoming friend requests are evaluated with reference to this model which takes into account the attributes (such as common interests) and behavioral properties (such as seat frequency).
   LINK: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8731968

6. CAN FRIENDS BE TRUSTED? EXPLORING PRIVACY IN OSNs
   This article presents a case study describing the privacy and trust inside a small population of online social network users. Taking Facebook as a reference, the frequency with which people are willing to disclose personal data to an unknown online user was determined. While most of the users sampled did not share sensitive information when requested by the stranger, it turned out that several users were willing to divulge personal details to a stranger if there is a mutual friend.
   LINK: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5231850

7. TO BEFRIEND OR NOT: A MODEL OF FRIEND REQUEST ACCEPTANCE
   This paper was the most useful, as it provided a valid model for the choice of parameters: it explains how a friend request acceptance model was developed that explains how various factors influence user acceptance behavior. This paper highlighted how there are 4 decision-making factors to which a user appeals when he has to accept a friend request, in particular, **"friendship factors"**, i.e. all the visible aspects of the profile (name, gender, profile picture, city of origin, residence, school, mutual friends, interests, etc.); **"privacy factors"**, dictated by a personal concern of a user and/or awareness of her, perhaps also due to personal experiences or friends; **"environmental factors"**, when friend requests are accepted without actually considering those who arrive due to lack of concentration and/or time to check; in the end **"capacity of interface"**, that means that some types of users spend a lot of time to understand what kind of person has asked them for friendship, with careful evaluation of all the

information and photos of that profile, until sending a private message. Some users keep the friend request pending and reserve the right to check the profile occasionally, to see changes within it.
LINK: https://www.usenix.org/conference/soups2014/proceedings/presentation/

4.2 Summary and starting points

Continuing with the research, the focus has increasingly focused on the type of projects that could have turned out to be similar to the idea of putting oneself on the side of the attacker who wants to hit a specific victim and must ensure that he is accepted as a possible friend.
All the papers found were useful to outline the parameters from which to start to structure the attacking and victim profiles.
In the next chapters, the parameters and the development choices will be motivated based on the aspects that the study in the literature has allowed evaluating.

Capitolo 5

Structure design

The attacker knows that it is not easy to be accepted by his victim and, for this reason, he wants to understand what choices he must take to create the best profile possible to be accepted without too much delay by the victim. Therefore, the parameters must be defined base on the victim, to be accepted with greater possibility.
Thanks to the study of some specific papers mentioned above and thinking about how much to deepen the search for the best profile, in this chapter will be define and explain the chosen parameters.
Then, from the parameters, the consequent profiles configurations will be shown.

5.1 Definition of the project's line guide and the chosen parameters

The definition of the parameters was conditioned by the degree of depth that the project wants to achieve: clearly more parameters lead to a greater number of profile configurations, but the data collection must be more in-depth and on a large scale.
Furthermore, Facebook does not check that an entered date is true, it only checks if it is valid by its standards. This means that a person could enter data that is valid for Facebook but not true in reality. Assuming that there is always a real person behind a profile, untrue data would lead to an invalid analysis.
Therefore, the parameters chosen are parameters that when registering on Facebook, a person would be required to be honest.
It should be underlined how the resulting configuration of the profiles can be considered as the starting point: in the future the study could be deepened, adding other parameters, but this can be done by taking this as the starting configuration.

5.1.1 1st parameter: gender

Gender is almost the most relevant factor, so much that most of the time is the decisive one.
If looking at the profile's name, the gender is not clear, an user can check the bio of this profile. In fact, in the bio, the correspondent label of the gender will specify because a person who wants to register on Facebook must declare it and could be "woman", "man" or "custom option" where a user can set its pronouns.
By the way, many papers of those cited report that a friend request from a female user

profile is accepted easier than a friend request from a man user profile, regardless of the gender of the profile that receives it.

Classes

The parameter `gender` ∈ {`female, male`}, where:

* `female/F`, if the victim's profile user declares to be a *woman*;
* `male/M`, if the victim's profile user declares to be a *man*.

5.1.2 2nd parameter: image profile

The profile picture is the first thing, along with the name, a user sees when they have a friend request. As reported in some above-mentioned papers, the profile image already serves to give an idea of the person who is asking for friendship, because it shows some details that do not need to be checked (such as gender or an indicative age range). A profile with a hidden or fake image (eg. a landscape) is more mysterious in the eyes of a user who receives the friend request, beacuse the real person behind this profile is not clearly identifiable.
Therefore, the main idea is to verify the presence of at least one person in the profile picture and classify the user profiles based on the outcome of this verification.
Facebook uses a technology that allows recognizing the profile image's content (faces, objects, ...), and automatically creating a description that will be enclosed in the `alt` tag. The presence of one or more people (and occasionally other details about the panorama and/or some objects present) is always specified within the `alt` tag. On the other hand, if the image is fictional (cartoons, drawings, etc.) or is an image without a person (eg. landscapes), the `alt` tag contains `"No description of the photo available"`.
This technology has been exploited to classify each victim's profile picture.

Classes

The parameter `real_img` ∈ {`true,false`}, where:

* `true`, if in `alt` tag reported the presence of one or more people;
* `false`, if in `alt` tag not reported the presence of one or more people;

5.1.3 3rd parameter: age

Age is a very relevant factor. From the literature, it appears to be one of the first factors that a user who receives a friendship goes to check the profile from which the friend request started, if the age is not deducible from the image profile. The more similar age, the more likely it is that the friend request will be accepted.
A person who wants to register on Facebook, must declare a real birth date (and must be at least 13 years old) and, once the profile has been created, can choose whether to make the date public or not. Some profiles publish only the month and day, other profiles only the year, others the complete date, others hide everything.
To classify profile user's age, the date of birth that it reports will be taken, and the age will be calculated based on the current year, so as to be able to assign to this profile its membership class.

Classes

Age is classified into 3 ranges, so the parameter `age_range` $\in \{2, 3, 4\}$, where:

* `2`, if the age victim's profile user is between 18 and 50 years;
* `3`, if the age victim's profile user is greater than 50 years;
* `4`, if the age victim's profile user is hidden;

5.2 Parameters' configuration to creating profiles

To delineate the types of profiles (the same for both victims and attackers) it was necessary to define profile categories, which are nothing more than the combinations of all the various parameters.
In this case study, the parameters that will be configured are:

1. `gender` $\in \{$`female, male`$\}$;
2. `real_img` $\in \{$`true,false`$\}$;
3. `age_range` $\in \{2, 3, 4\}$.

Combining all various parameters with possible values, there are 12 profiles and $12 \cdot 12 = 144$ possible attack combinations.

All the various resulting configurations are now shown in tabular form:

gender	real_img	age_range	LABEL
female	true	2	FT2
female	true	3	FT3
female	true	4	FT4
female	false	2	FF2
female	false	3	FF3
female	false	4	FF4
male	true	2	MT2
male	true	3	MT3
male	true	4	MT4
male	false	2	MF2
male	false	3	MF3
male	false	4	MF4

The label

As can be seen from the table above, each type of profile is classified with a label.
The label is the union of the actual value assumed by the parameters in that particular circumstance: the first character identifies the gender (`"F"` in the case of `"female"`, `"M"` in the case of `"male"`), the second character indicates the value of the `"real_img"` parameter (`"T"` in case it is `"true"`, `"F"` in case it is `"false"`) and the last character is the numeric value that indicates the age range (it can be `"2"`, `"3"` or `"4"`).

All victim profiles are classified with these labels. in fact, each collected *victim* profile is pigeonholed into one of these 12 categories.
However, as regards the *attackers* profiles, they will be created starting from these configurations and in an automated way with the help of the tool called `create-profile.py` (details in the chapter 6.1.1).
Furthermore, if attackers and victims have the same label it could be brought to create confusion in the second phase of the organization of the attack combinations. For this reason, the attackers' profiles are identified with an identifying code "`PF-`n" where n is an increasing identification number assigned when the profile is created.

5.3 Development planning

Development takes place in 4 consecutive phases:

1. create the 12 attacking profiles;
2. collect victim profiles and classify each of them in its category;
3. each attacking profile will require the friendship of 3 victim profiles for each category (details in chapter x);
4. analysis of the collected data (how many profiles have accepted the friendship, which category they belong to and which category the respective attacker belongs to, ...).

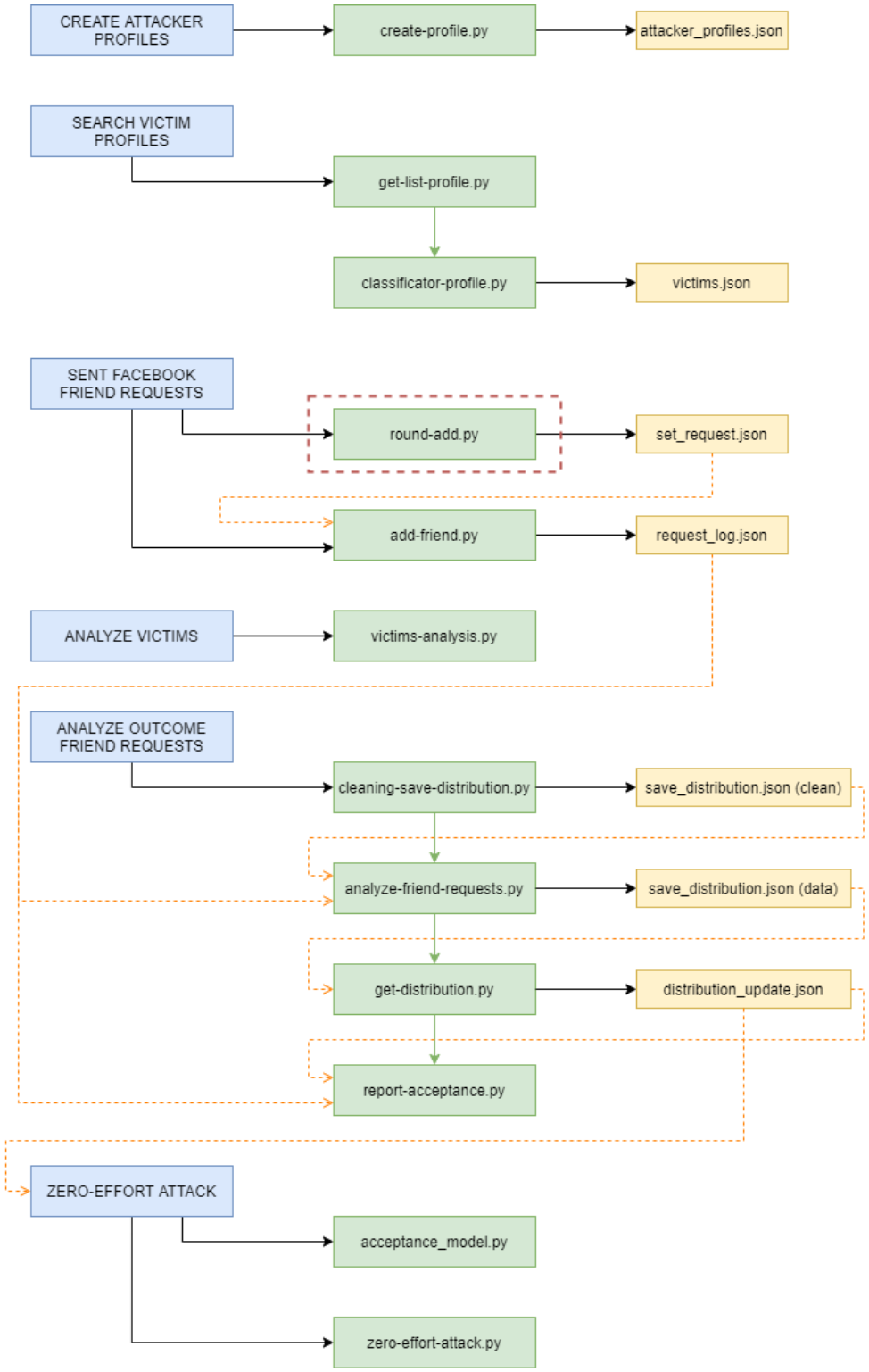
5.4 Architecture system

The following page graphically shows the general architecture of the system:

blue rectangles indicate the possible actions that can be performed.
They are listed vertically to show the chronological sense of their execution.

green rectangles indicate the developed tools.
Note that the `round-add.py` tool is surrounded by a *red rectangle with a dashed border* to indicate that it must be run only once and before the first run of the `add-friend.py` tool.
Furthermore, the *green arrows* that connect one tool to another, indicate a chronological order in which the tools must be executed each time.

yellow rectangles indicate the resulting datasets after running that tool.
These datasets can be used for subsequent actions.In fact, note the *orange arrows*: they indicate which specific dataset is required in order to run the tool to which it is connected.


CREATE ATTACKER PROFILES
create-profile.py
attacker_profiles.json
SEARCH VICTIM PROFILES
get-list-profile.py
classificator-profile.py
victims.json
SENT FACEBOOK FRIEND REQUESTS
round-add.py
set_request.json
add-friend.py
request_log.json
ANALYZE VICTIMS
victims-analysis.py
ANALYZE OUTCOME FRIEND REQUESTS
cleaning-save-distribution.py
save_distribution.json (clean)
analyze-friend-requests.py
save_distribution.json (data)
get-distribution.py
distribution_update.json
report-acceptance.py
ZERO-EFFORT ATTACK
acceptance_model.py
zero-effort-attack.py

Capitolo 6

Development of the tools

In this chapter the study behind the development of every single created tool will be explain, and technical details about implementation.

6.1 Tool for automated create of attacker's profile

This section reports the study and development process of the tool for the automated creation of an attacker profile.

6.1.1 The tool: `create-profile.py`

The attacker profiles were created from the various profile category configurations identified in chapter 5.2. The process is automated, that is, using a tool, called "`create-profile.py`", which after requesting specific values as input, responds in output with a complete card of the attacking profile's personal data.

The tool, run from the terminal, requires as input:

* `cod_id`,
  an identification code to identify the profile in a simpler way (with this parameter, `PF-`n will be created, where n is this `cod_id`).
  Accepted values: numeric parameter (`int`);

* `gender`,
  i.e. the gender of the attacker
  Accepted values: string parameter s, with $s \in \{$`female, male`$\}$;

* `real_img`,
  boolean value for the type of image profile the attacker must have.
  Accepted values: boolean parameter (`true` or `false`);

* `age`,
  i.e. the age of the attacker.
  Accepted values: numeric parameter (`int`);

* `agehide`,
  boolean value for whether the age will be visible in the profile or not.
  Accepted values: boolean parameter (`true` or `false`).

Moreover, the tool use datasets (created ad hoc) to generate a most complete and realistic profile possible:

* `dataset-name.json`, to randomly assign the name to the profile, based on the input `gender` parameter; this dataset is formed with the most common Italian names, both male and female, more than 100 for each.
* `dataset-surname.json`, to randomly assign the surname to the profile; this dataset is formed with the 100 most common Italian surnames.
* `dataset-city.json`, containing the list of Italian provinces to randomly assign the city of residence to the profile;
* `dataset-occupation.json`, to randomly assign the job occupation to the profile;

6.1.2 Develop of the tools

First of all, using `dataset-name.json`, a random name is extract and assign to the profile, based on the input `gender` parameter.
Same thing for the surname where, using `dataset-surname.json`, a random surname is extract and assign to the profile.

After that, `real_img` parameter and `gender` parameter will be check, and, the label to identify the attacker the label begins to be created.

With the `age` parameter, the tool will then create an appropriate date of birth to be entered in the Facebook profile, both in the case `agehide` parameter is `"true"` or `"false"`, because to register on Facebook it is necessary to enter a date of valid birth. With the correct age range, the last character of the label could be set now, so the *label* is created.

Despite `residence` and `occupation` are superfluous parameters fot this case of study, to make each attacker profile more complete and realistic, actual values were nevertheless entered, which obviously were not valued in the analysis for research purposes.

About `occupation`, a minimum of control has been included, to make it consistent with age and leaving aside some special cases: each profile under the age of 18 will have a `"Studentessa"` or `"Studente"`, while each profile over the age of 65 will have `"Pensionata"` or `"Pensionato"` as occupation.

About `residence`, an attempt was made to scatter the various attacker profiles throughout the Italian peninsula. This was done in order not to risk having more attacker profiles in too close areas.

Afterwards, the tool uses the temporary fake email generator service (at `https://emailfake.com/` site) to create the fake email with which this profile will then log into Facebook. The username of the email is made up of `nome.cognome.age`, for each profile. The tool will go to the temporary false email generator site, enter the username created in the appropriate input section and then save the complete email with the random domain that the site will propose.

Consequently, the tool generates a password of 6 characters, formed by an uppercase letter, 4 lowercase letters, 2 numbers, and a special character, to use to log into Facebook.

In the end, the tool generate the identification *code* with the `cod_id` (for example `PF-3`), and save the data in the dataset `attacker_profiles.json`.

6.1.3 Create Facebook accounts

Originally, one idea was to create the Facebook account in an automated way, using a tool developed ad hoc.
Unfortunately, however, in the study phase, it was seen how the social network applies very strict policies that often constrained the creation of these profiles.
First of all, it required to enter a code that he would send by email and occasionally forced the resolution of a ReCAPTCHA to continue.
Sometimes the email with the code not arrived, so the procedure broke up and the profile was blocked.
In other cases, Facebook continued to ask to resolve ReCAPTCHAs, even though they were always resolved correctly, and, after a while, the profile was blocked.

Therefore, the creation is done manually, but, despite this, there were still some significant problems: in fact, many profiles, after entering the initial data were immediately blocked and could only be unlocked by entering a mobile number.

So it was necessary frequently changing browser, connection, device, operating system, and even site for the generation of temporary emails.
To avoid running into further blocks, the profiles were created gradually, that is, days apart from each other.

Once created, each profile was included in numerous Facebook groups of different genres (reading, music, sports, etc.) and occasionally some content was shared in their personal profile.

Furthermore it's important to underline that some profiles with `real_img` parameter set as `false` are minors, although in chapter 5.1.3 it can be seen how the 3 age ranges only consider adult people.
This was done because the age of these profiles, as evidenced by the corresponding *label*, is not visible to the victim profiles who find the friend request, as the age is hidden.

The last step concerns the profile picture. It is necessary to specify that the profile pictures where the `real_img` parameter is set to `true` are not of real people, as they were fished from this site: https://thispersondoesnotexist.com/. It generates a realistic picture of a person every time the page is refreshed, using artificial intelligence.
So, the profile pictures were obviously chosen after creating every single attacker profile with the tool, because it was necessary to be consistent with the age of the profile user.
In the end, the profile pictures where the `real_img` parameter is set to `false` were chosen looking random images in Google Images.

The 12 attacking profiles that have been generated are now reported.

COD	LABEL	IMG	NAME	SURNAME	AGE	RESIDENCE	OCCUPATION
PF-11	FT2		Federica	Grasso	38	Milano	Agente Assicurativo
PF-9	FT3		Giulia	Moro	72	Vercelli	Pensionata
PF-6	FT4		Giulia	Ferrari	32	Palermo	Istruttrice di Yoga
PF-8	FF2		Ilaria	Riva	26	Verona	Stilista
PF-10	FF3		Sara	Rizzo	59	Bologna	Fiorista
PF-7	FF4		Michela	D'Angelo	16	Taranto	Studentessa
PF-5	MT2		Leonardo	Neri	45	Roma	Tatuatore
PF-3	MT3		Carlo	De Angelis	57	Siena	Chimico
PF-0	MT4		Giuseppe	Martini	39	Ancona	Farmacista
PF-2	MF2		Samuel	Colombo	44	Caserta	Tappezziere
PF-4	MF3		Valerio	Mazza	57	Catanzaro	Dentista
PF-1	MF4		Alessio	Fontana	17	Pescara	Studente

6.2 Tool for automated search of a victim's profile

This section reports the study and development process of the tool for searching for a victim profile in an automated way.

The initial idea was to develop a tool that would scrape and categorize the profiles at the same time they are searched for.
For example, the tool logs into Facebook with the credentials of an attacker profile, somehow it gets a list of possible victim profiles and, one by one, it enters on it to scrape and categorize it, saving the result in the appropriate file. This means that the tool would move to the next profile until the list is over.
Computationally, this process is long and cumbersome, scraping would have increased significantly and, above all, this procedure is all done with a single attacker profile: this could be bring Facebook to block it, as it would always perform the same action in a loop for a long time.

Furthermore, the Facebook search interface is not helpful for this case study, as it gives the possibility to set the parameters of "city", "education", "work" and "friends of friends", and these parameters have nothing to do with this case study.

For all these reasons, it was necessary to proceed differently: first using a tool called `get-list-profile.py` to save a list of eligible victim profiles and then a subsequent tool called `classificator-profile.py` which determines for each profile if and what kind of victim it is.
The idea and operation of these two tools are now explained in detail.

6.2.1 The tool: `get-list-profile.py`

This first tool saves a list of profiles which will then be analyzed and classified by the next tool.
In detail, the tool, run from the terminal, requires as input:

* `cod_id`, that's the attacker's profile code;
* `type_victim`, i.e. gender of the victim profiles that will be collect.
  Accepted values: string char s, with $s \in \{$`f, m`$\}$;.

First of all, the tool examine the attacker's `cod_id` to verify that it exists and, if so, save its email and password to log in to Facebook. This is because it is not possible to search for profiles on Facebook without logging in.

At this point, based on the gender entered in the input, the tool randomly chooses a name from `dataset-names.json` and searches that name.
Facebook responds by showing a list of profiles with that name. But, more precisely, Facebook would show a list of profiles residing in a circumstantial area to that of the profile with which you are searching. To broaden the search range, we have chosen to use provinces instead of cities (and therefore to insert only the Italian provinces within the `dataset-city.json` dataset) and to disperse the attacker profiles throughout the Italian territory.

Furthermore, each attacker profile searched for a minimum of 4 different named profiles with `gender = "f"` (i.e *female*) and a minimum of 4 different named profiles

with `gender = "m"` (i.e *male*). A minimum or maximum limit was not given to the profiles to be collected, also because they were not yet assigned to the corresponding label.

In any case, from the list of resulting profiles, the tool clicks on the "Show all" button to load an even greater number of profiles and, scrolling the page a couple of times, saves in a `file` all user-profiles' url with the searched name.

Several copies of this tool have been created, which differ from each other for the file where they save the list of profiles. This was done to be able to work in parallel, in order to collect as many profiles as possible in the shortest possible time.

6.2.2 The tool: `classificator-profile.py`

This second tool analyzes each profile of a given list (output of the previous tool) to be able to attach to each of them its corresponding label, and then save them in the appropriate dataset, called `victims.json`.
In detail, the tool, run from the terminal, requires as input:

* `cod_id`, that's the attacker's profile code;
* `file`, that's the name of the file containing the list of victim profiles that must be analyzed and categorized.

First of all, the tool examine the attacker's `cod_id` to verify that it exists and, if so, save its email and password to log in to Facebook. This is because it is not possible to search for profiles on Facebook without logging in.

At this point, the tool opens the correct file based on the `file` parameter passed in input (the file is in the same folder as the tool). It reads line by line, taking each URL that will lead to a saved victim profile.
Once the URL has been selected, the tool moves to the dashboard of this profile, in order to analyze its details.
Here, the tool first checks for the presence of the button to add the profile to friends: some profiles do not allow you to be added to friends, but you can only follow them or send a private message. If the button is present, it will continue the analysis, otherwise, it will go to the next URL.

As a first point, the tool checks the profile picture. The algorithm uses the `alt` of the profile image to identify the presence (or not) of people within the image, a sufficient condition to make the `real_img` parameter as `true` (details in chapter 5.1.2). Once it have checked the `alt` of the profile image, the tool will declare the image as `"T"` when the value of the "cod_id" parameter is "true", otherwise it will declare `"F"`.

After that, the tool will have to analyze what kind of URL the victim profile has. In fact, there are two types of profile URLs:

* `https://www.facebook.com/profile.php?id=`*`numerical_sequence`*
  e.g. https://www.facebook.com/profile.php?id=3322661122
* `https://www.facebook.com/`*`character_string`*
  (sometimes followed by *`.numerical_sequence`*)
  e.g. https://www.facebook.com/mario.rossi.32

Due to this difference, a check has been implemented to understand what type of URL you are dealing with and, consequently, the correct link is assigned to move within the profile to find the information.

At this point, the tool knows what type of URL it has to create for moving to the "about" page, where there is the basic information of the profile.

Once here, thanks to a regular expression, the tool checks for the presence of a 4-character string, indicating the year of birth. If it is present, the age is calculated and the corresponding range between "2" or "3" is assigned, otherwise, the assigned range is "4" (details in chapter 5.1.3).

As a last step, the tool analyzes the gender of the profile, first looking for the presence of the string "Man" or "Woman" and, if it is not found, it moves to `dataset-name.json` to search in which category that name is found. It will assign `F` if the profile is of a woman, `M` if it belongs to a man.

Once at this point, the tool has created the corresponding label and this is where the *check system* comes into play: before categorizing the profile, it asks for confirmation via the terminal, showing the label created. If it is correct, it's necessary to answer affirmative (with an "ok"), otherwise, the correct label must be enter.
Now, the tool has the exact label, so it can save the profile within the `victims.json` dataset, obviously after checking if that profile does not already exist.

Once the process is finished, move on to the next URL, until the list is over. Then, it destroys the `file` with the profile list and ends its execution.

A maximum limit for the profiles to have for each category has not been decided.
On the contrary, a minimum number has been defined: each attacking profile will have to perform 3 friend requests for each category of victim profile (including the category in which it belongs).
Considering that there are 12 categories of attacker profiles, the minimum number of victim profiles for each category is $12 \cdot 3 = 36$ (so collect $12 \cdot 36 = 432$ victim profiles).
An analysis tool aided this collection process, showing how many profiles were needed to reach the minimum value based on those already collected (details about the tool in chapter 7.1).

6.3 Tool for automated friend request

This section reports the study and development process of the tool for the automated friend request from an attacker profile to one or more victim profiles.
This tool is the simplest in terms of implementation, as it was only necessary for it to log in and ask for friendship from a profile given in input.
But, considering that:

1. each *attacker* profile must send the friend request to 3 *victim* profiles for each category;

2. a *victim* profile must not receive the friend request from more than one *attacker* profile;

3. each *attacker* profile must have the same number of total friend requests to make;

4. an *attacker* profile mustn't send too many friend requests in close moments, because Facebook would block the profile.

For this reason, first a tool called `round-add.py` was created and then the definitive friend request tool, called `add-friend.py`.

6.3.1 The tool: `round-add.py`

This first tool must set the friendship requests that each *attacker* profile has to send: it assigns to each attacker profile 36 victim profiles to which to send the friend request. Each URL within the `victims.json` dataset is seen as an element of an array, and the name of the array is given by its category name.
This tool must know:

* the exact number of *victim* profiles to witch each *attacker* profile must send the friend request (36 in this case of study);

* the label of each *victim* to witch one *attacker* profile must send the friend request;

* the exact number of *attacker* profile (12 in this case of study);

* the `cod` of each *attacker* profile;

* the number of friend requests that each profile will have to send for each category (3 in this case of study).

The tool process this information and within the `set_request.json` dataset, it will assign to each attacker profile the list of URLs of victim profiles to be attacked. It is nothing more than a list of positions of elements within each array containing the URL list of victim profiles. To calculate the index of each element, the numerical value of the *attacker* profile is considered, to simulate a "jump" to the correct element, as many as there are attacker profiles available.
At this point, each attacking profile has its personal list of victim profiles to attack and the next tool comes into play.

The tool, run from the terminal, doesn't need any input parameters:

6.3.2 The tool: `add-friend.py`

Given an attacker profile and its respective list of victim profiles that he will have to attack, this second tool takes care of requesting the friendship from the *attacker* profile to the *victim* profile.
Hovwever to avoid being blocked by Facebook, it was decided to structure this attack in rounds: at each round, identified with an index, an attacker profile will have to send the friend request from `n` victim profiles.
The tool point to the first element of the victim profiles list of one specific attacker profile, skipping the elements that are part of the previous rounds.
There is a list that specifies how many friend request must be sent in every round (values ranging from 1 friend request to 4 friend requests). The index of each elements of this list is the identification value of the round.

For example, at the 3rd round, knowing the number of profiles to which must request friendship in the previous rounds, the sum of them point to the first element of the victim profiles list to which the current attacker profile must send the friend request in this round.
In detail, the tool, run from the terminal, requires as input:

* `cod_id`, that's the attacker's profile code;
* `round`, the numerical value that indicates the attack round to be satisfied.

At this point, each *attacker* profile will log into Facebook and send the friend requests to the victim profiles taken from the correct portion of the list in that particular round.
To manage this delicate phase more accurately, an excel file has been built to keep track of the rounds already executed.
Once the friend request has been sent to a *victim* profile, before moving on to the next one, the tool saves the information in a dataset called `request_log.json`: for each attacker profile, the list of attacked victim profiles are indicated, with its specific category.

Capitolo 7

Data collection

In this chapter all the developed analysis tools will be explain.

For data collection, it wouldn't have been required to develop special tools. Nevertheless, they have been implemented because they are useful and, above all, usable even in contexts where the amount of data collected is considerably greater.

Moreover, it was preferred to develop separate tools that perform individual analysis functions. This was done thinking about the fact that these tools could also be used individually in the future in contexts where it will not be necessary to follow a chronological order of execution every time.

In addition, a *time control system* has been developed: the analysis tools also keep track of the date on which they are performed to make visible the trend over time because some victim profiles may accept the friend request later because they previously kept it suspended.
Each result of each analysis session is enclosed in a folder named with the date on which the analysis session was carried out.

7.1 Analyze victims: `victims-analysis.py` tool

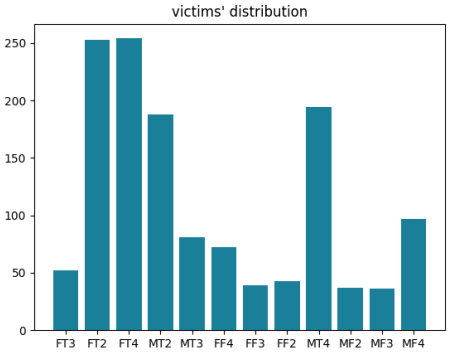
To determine the evolution of the `victims.json` dataset growth during the execution of the `classificator-profile.py` tool (details in chapter 6.2.2, it was created a simple tool, called `victims-analysis.py`.

This tool show on the terminal how many profiles for each category of victim profiles have been saved up to then in the `victims.json` dataset.

Furthermore, to make distribution more intuitive, it shows (but not save) a bar graph that highlights the difference in the victim profiles collected for each category.

This tool was very useful during the execution of the `classificator-profile.py` tool because it allowed to keep under constant control the trend of the dataset, and, above all, to be able to check that every category will reach the minimum value.

```
$ python victims-analysis.py
Tot = 1346
('FT4', 254)
('FT2', 253)
('MT4', 194)
('MT2', 188)
('MF4', 97)
('MT3', 81)
('FF4', 72)
('FT3', 52)
('FF2', 43)
('FF3', 39)
('MF2', 37)
('MF3', 36)
```



Looking at this image it is immediately apparent that the dataset is heavily unbalanced. This situation is discuss in detailed in chapter 8.1.2.

7.2 Analyze outcomes of friend requests

After sending all the friend requests with each attacking profile, we started to launch these various analysis tools. To do this as automatically as possible, a series of tools have been created, each based on its specific functionality.
The developed tools are:

1. `cleaning-save-distribution.py`;
2. `analyze-friend-requests.py`;
3. `get-distribution.py`;
4. `report-acceptance.py`.

These tools must be run in exactly this order each time. Now they will be explain.

7.2.1 The tool: `clean-save-distribution-contents.py`

This first simple tool cleans the contents of the `save_distribution.json`, i.e. it will set its each value to 0.
Considering that these tools must be run one after the other on each analysis session, this particular tool must be run first because it modifies the starting file for the analysis. If it contains data from previous analysis sessions, it is invalid because the data would overlap. So, with this tool, the content of this dataset is cleaned up as if it had just been created and therefore ready to accept the new data for a new analysis session.

Although the function is small, this mini tool was created separately, as it could be used at any time (for example if some error occurs during the analysis).

Potentially, this tool could also be run only at the end of each analysis session. The important thing is that every time a new analysis session begins, `save_distribution.json` dataset must be cleaned.

7.2.2 The tool: `analyze-friend-requests.py`

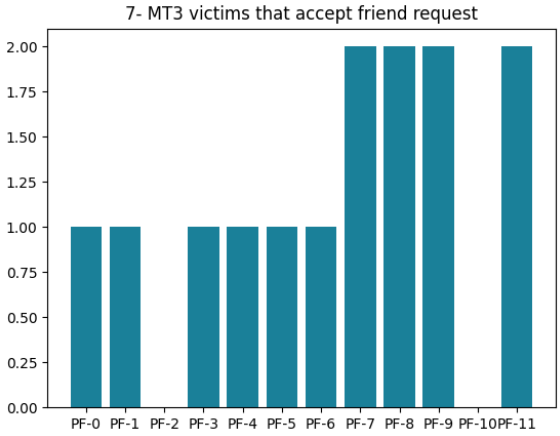
This tool verifies for each attacking profile which of the victim profiles has accepted the friend request.
Using a list with all the identification codes of the attacking profiles, it fetch the information necessary to log in on Facebook and then it save the list of friends' URLs in a temporary file.
A function will read the temporary file line by line and find a match with a URL in the `request-log.json` file in the section dedicated to the attacking profile under consideration. For each profile found, will increase the count inside the `save_distribution.json` file, so that at the end of the analysis of each attacking profile, it is possible to see, for each victim, how many and what type of profiles attackers accepted the friend request.

In the end, It also will show in the terminal how many friend requests have been accepted out of the total of those sent.

7.2.3 The tool: `get-distribution.py`

This tool shows the distribution of the friend requests accepted: for each victim, it creates a graph that displays how many friend requests the victim accepted and from which attacker profiles they came from.
The tool will first create a folder called "report" with the date the tool is started (for example "report-2021-08-02").
Then, inside it will save bar graphs that show this information.
Each image contains a graphic about a specific victim and is saved with today's date and victim type (for example "2021-08-02 - 7-MT3").
Furthermore, it creates `distribution_`*current_ date*`.json` dataset and update the content of `distribution_update.json` dataset, with the report of the data in `JSON` struct. They are the same file, but the first one is useful to see the report during the time, the last one will be use to have the last updated data to create the *model of acceptance* (details in 8.2)

The graph shown below is an example.
It represents how many "MT3" victims, ie males over the age of 50 and with a real profile picture, accepted the friend request and which profiles they came from.

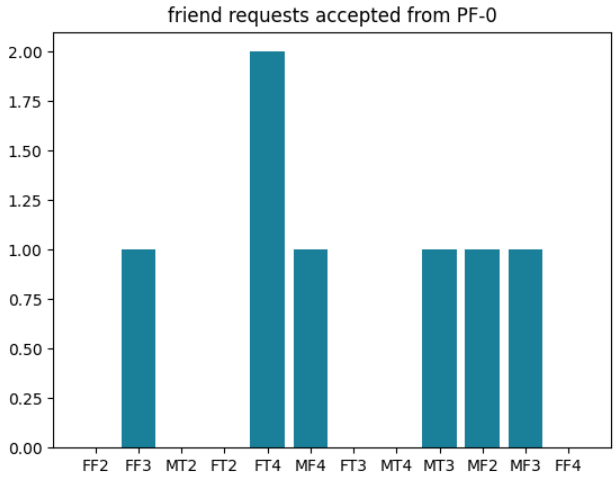


7.2.4 The tool: `report-acceptance.py`

The last tool to run, called "`report-acceptance.py`", focuses on the attacker who made the friend request.

First of all, it creates a folder called "`report-request`" where it saves a bar chart for each attacking profile showing, for each category of victim profile, how many requests have been made by that attacking profile.
In this case study, it would not be necessary considering that there are 3 friend requests for each victim profile, but if you have to work with more data this function will be needed again.
Then, it will create a second folder, called "`report-acceptance`", where it saves a bar chart for each attacking profile, showing, for each category of victim profile, how many requests have been accepted to that attacking profile.
It will also save this information in the`distribution_data` file where *data* means the date the tool was executed (for example "`distribution_2021-08-02`").

An example is a graph below, inserted by the tool in the "`report-acceptance`" folder, which represents how many victims (and which category they belong to) have accepted the friendship to `PF-0`, that is the attacking profile with the label "`MT4`".



Capitolo 8

Data analysis and the outcomes

This chapter illustrates the analysis of the collected data, enriched by discussions and experiments.

8.1 Discussions, experiments and premises about collected data analysis

Before discussing the obtained acceptability model and describing the "Zero-Effort Attack" tool, it is necessary to pin some premises and discuss some hot spots. Thus, some small experiments carried out are cited to show how the research field is very large and full of possible paths to take.

8.1.1 Number of friend request

First of all, a clarification about the number of friend requests.
According to the law of large numbers, with a greater amount of data collected, the result of the analyzed data is more certain and therefore more valid.

Thus, is correct to affirm that `3` is a small number, but it was decided thinking about the smallest exhaustive number of friend requests to do considering the available time.
A greater number would have led to a greater number of logins and friend requests, actions that could have led Facebook to block profiles.
With more time, logins and friend requests would be spread over time, so as not to be blocked.

So, `3` is the perfect number for this case of study (i.e. for the internship), because only one request not could be valued, 2 requests bring the analysis in a *fifty-fifty* situation, with 3 requests one of the two possible results (friend request accepted or not) triumphs over the other.

It's also important to underline that the result of **the analysis is preliminary**: with more requests (but anyway the number must be odd) the model of acceptance could undergo substantial changes.

Also for this reason, during the development of all tools, it has always been born in mind that they must be implemented to work with a variable amount of data.

8.1.2 `victims.json` dataset: why is it so unbalanced?

After the first phase of data collection, it was possible to notice that most of the users on Facebook do not enter a lot of data in their profile, making the dataset of data collected immediately very unbalanced.
More precisely, few users enter their age, city of residence, and occupation (many times it is present but blatantly bogus).
Added to this is the fact that very few underage users who make it clear their age are very few (about 2 out of more than 1000 total profiles collected).
In many cases, age can only be guessed by looking at the profile image, but this is not an evaluation criterion on which to reason.
Thus, some categories contained a greater number of profiles and this caused the dataset to be very unbalanced.

In the first place, it was decided that these profiles were categorized anyway, but for every category, the number of profiles examined is the same.
The extra profiles were useful when it was necessary to replace some victim profiles that, for some reason, could no longer be considered.

Then, it was decided to remove the classes of the parameters that consider the minors, and the `"current_town_value"` and `"occupation_value"` parameters.
All three are interesting factors to analyze in future works (9).

8.1.3 Profile picture: is it really so important?

For this project, 12 attacking profiles were created, 6 of which with a real profile image, depicting a (non-existent) person.
The profile pictures were chosen consistently with the age that the corresponding attacker profile claimed to have.

For example, the attacker profile `"PF-9"` is a 70-year-old woman and her profile picture of her represents a woman who could be said to be around that age. The value of the `"age_range"` parameter of this profile is `"3"`, as the age is greater than 50 years. In this range, however, there are also people of just 52 years and 20 years of difference are not exactly few if you wanted to make a comparison.
The "PF-9" profile was not widely accepted, but how can one be sure that the same fate would have had a profile 20 years younger but still belonging to `age_value` = 3?

For this reason, they took some profiles with the real profile image and changed the profile image with one that, at first glance, showed a different age from the one declared (not too different of course).

A note before showing the results: to make the change more consistent, the year of birth that the profile declares should also have been changed.
With an extra profile used for testing of various kinds, this experiment was done, but Facebook took steps by blocking the profile.
So, it was decided to change only the profile image, taking as good as stated in the

chapter 5.1.2, or that when a user receives the friend request he is easily influenced by the profile image, so much so that many times he does not even check the true age on the appropriate page.

The results are now illustrated:

COD	LABEL	OLD IMG	NEW IMG	FRIEND REQUEST ACCEPTED
PF-9	FT3			+3 [MT2, MT2, MT2]
PF-5	MT2			+2 [FT2, FT3]
PF-3	MT3			+0
PF-0	MT4			+2 = [FF3, FT3]

(A note: after changing his picture profile, PF-5 received a friend request from a woman, the first one request for him)
The importance of the profile image is therefore a valid starting point on which to deepen with a dedicated case study.

8.1.4 Unexpected situations

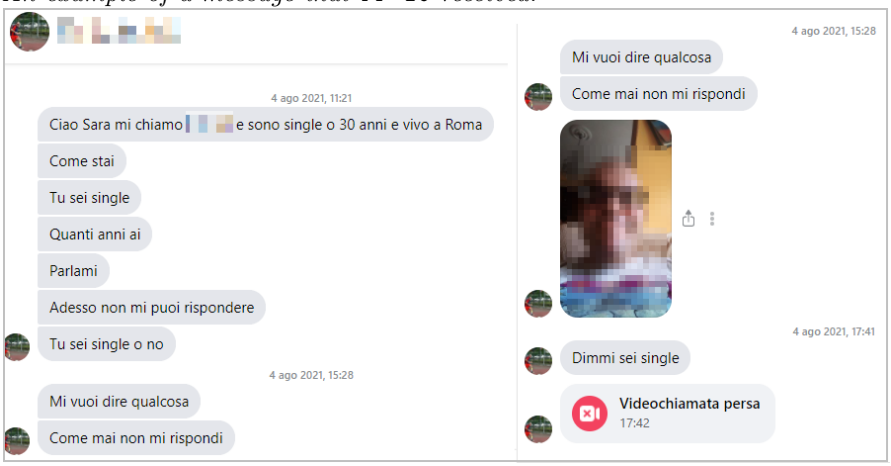
Friend requests from an external profile to an attacker profile

This is the most unexpected situation: the attacker profiles must only to send the friend request. In a sense, one could say that they would have to work in the shadows. However, some attacking profiles found themselves struggling with friend requests from other real profiles!
It was decided to accept all friend requests to see the behavior that these profiles would then assume, but there is nothing relevant to report, except that some profiles have started sending many messages to these attacking profiles (details in the next chapter). Attackers who have received friend requests are reported to be: PF-0, PF-6, PF-7, PF-8, PF-9, and PF-11. In particular, the PF-6 and PF-11 profiles have really received many requests for friendship: analyzing them both are profiles of women, both have the real profile image and are young. The papers said that friend requests from women are more easily acceptable, but even that these profiles received so many requests, well, it was not imaginable!

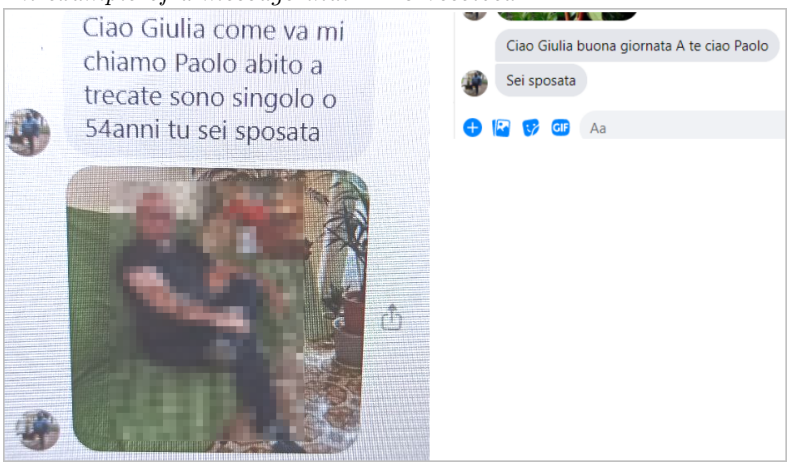
Victim profiles wrote messages to (female) attacker profiles, but not with so good intentions

Facebook is a platform where people can meet and start a conversation.
Well, it was assumed that some victim profiles could accept the friendship and then write messages, to understand if they really know each other in reality or even just to make new acquaintances. The thing that was not at all expected was the number of messages received.
The profiles of the attackers who received messages are all women involved in this project: `PF-6, PF-7, PF-8, PF-9, PF-10, PF-11`. Those who have received the most crops are `PF-6, PF-9, PF-11`. All these profiles received many messages only and exclusively from men.Many of these continued to write insistently even though they received no response.

An example of a message that `PF-10` *received:*



An example of a message that `PF-9` *received:*



(A few days later, this first message was then deleted (leaving only the photo) and replaced with the messages shown on the right).

An example of a message that `PF-11` *received:*

(This is an example of a tireless profile: despite having received no response, he still continues to text almost daily).

A profile removed an attacker profile from friends

In short, a `"MT3"` victim profile removed the `"PF-8"` attacker profile from his friends list a few days after accepting his friend request.
This situation was unexpected in the first place, but, thinking about it, it is not so strange.
As some papers said, there are some people that check very carefully the person who sent them the friend request: sometimes, first, they check the profile to decide if accept the friend request; other times, first they accept the friend request and then check the profile, in order to remove it from the friends if there are some characteristics that the person doesn't like.
Probably the profile more empty than usual (it is a fake profile where not a lot of content has been shared) and with a not real profile picture, made the victim suspicious, so much so as to remove this profile from the friend's list.

8.2 Model of acceptance

With a tool, called `acceptance_model.py`, which analize the content of `distribution_update.json` dataset (details in 7.2.3), an acceptance model will be created.
The tool, run from the terminal, requires the `distribution_update.json` dataset and for each type of victim, it describes the acceptance rate of every type of attacker profile. Considering that the number of friend requests from an attacker profile to each type of victim profile is "3", an attacker profile could have 4 types of percentual, based on the number of how many friends request that type of victim have accepted:

* `1%` if the acceptance rate is `0/3`;
  (It was decided to put 1% instead of 0% because there can always be an exception);
* `33%` if the acceptance rate is `1/3`;
* `66%` if the acceptance rate is `2/3`;
* `99%` if the acceptance rate is `3/3`;
  (It was decided to put 99% instead of 100% because there can always be an exception).

The tool does a sum of percentages for each type of victim for each type of attacking profile, creating the model of acceptance that is now shown:

VICTIM	ACCEPTANCE RATE
FT2	with ≈ 33% → MT4, MT2, FT4, FF3, FT2
	with ≈ 1% → MF4, MF2, MT3, MF3, FF4, FF2, FT3
FT3	with ≈ 66% → MT3, FT4
	with ≈ 33% → MT4, MF4, MT2, FF4, FT2
	with ≈ 1% → MF2, MF3, FF2, FT3, FF3
FT4	with ≈ 66% → MT4, FT4
	with ≈ 33% → MF4, FT3, FF3, FT2
	with ≈ 1% → MF2, MT3, MF3, MT2, FF4, FF2
FF2	with ≈ 66% → FT4
	with ≈ 33% → MF2, FF2, FT3
	with ≈ 1% → MT4, MF4, MT3, MF3, MT2, FF4, FF3, FT2
FF3	with ≈ 66% → MT4, FT2
	with ≈ 33% → MT3, FF3
	with ≈ 1% → MF4, MF2, MF3, MT2, FT4, FF4, FF2, FT3
FF4	with ≈ 33% → FT2
	with ≈ 1% → MT4, MF4, MF2, MT3, MF3, MT2, FT4, FF4, FF2, FT3, FF3
MT2	with ≈ 99% → MT2, FT4
	with ≈ 33% → FT3, FT2
	with ≈ 33% → MF4, MF2, MT3, FF2
	with ≈ 1% → MT4, MF3, FF4, FF3
MT3	with ≈ 66% → MT4, MT2, FT4, FF4, FF2, FT3
	with ≈ 33% → MF4, MF2, MT3, MF3, FF3, FT2
MT4	with ≈ 99% → FF2
	with ≈ 33% → MT4, MT2, FT4, FF3
	with ≈ 33% → MF4, MF2, MT3, FF4, FT3, FT2
	with ≈ 1% → MF3
MF2	with ≈ 66% → MT4
	with ≈ 33% → MF4, MT2, FT4, FF2, FT2
	with ≈ 1% → MF2, MT3, MF3, FF4, FT3, FF3
MF3	with ≈ 66% → MF2, FT3, FT2
	with ≈ 33% → MT4, MT3, MT2
	with ≈ 1% → MF4, MF3, FT4, FF4, FF2, FF3
MF4	with ≈ 99% → FT4
	with ≈ 33% → FT2
	with ≈ 33% → MT4, MF4, MF3, MT2, FF4, FF2, FF3
	with ≈ 1% → MF2, MT3, FT3

8.3 Zero-Effort Attack

`Zero-Effort Attack` is the name of the final tool. This was the goal of the internship. Its name is derived from the effective effort that the attacker gonna do, i.e. `zero`. That's because the attacker must put the URL of the victim profile on the terminal and the tool will show in the output the model of acceptance (chapter 8.2) for that type of victim.
In detail, the tool, run from the terminal, requires as input:

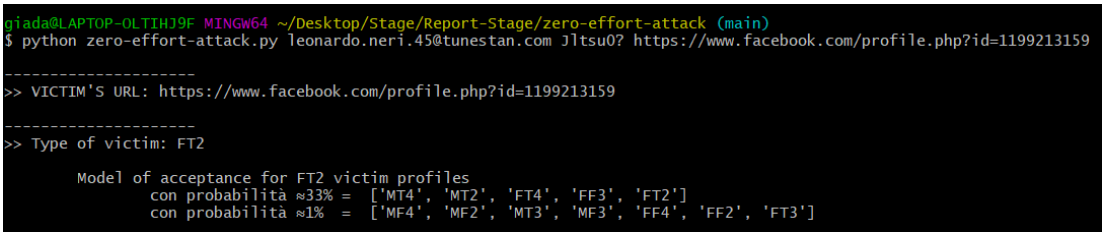
* `email`,
  the email that the attacker use to log into Facebook.
  Accepted values: string parameter (`string`);

* `pwd`,
  the password that the attacker use to log into Facebook.
  Accepted values: string parameter (`string`);

* `url`,
  the URL of victim profile to attack.
  Accepted values: string parameter (`string`);

This tool get the `email` and `password` parameters to log into Facebook.
After that, it goes to the page of the victim profile thanks to the `URL` parameter. Now the tool can be analize the victim, and the procedure is the same of the `classificator-profile` tool (chapter 6.2.2).
Once the victim's profile has been analyzed, it categorizes it by assigning it the corresponding label.
With that label, the tool will examine the acceptance model and show in the terminal which type of attacker profiles are accepted and with what probability, to allow the attacker to create the best profile.

The image below display what the terminal show in the output to the attacker in the end of the procedure.



Capitolo 9

Future works

As also specified at the beginning of chapter 8, the analysis and assessments reported are based on the results that have emerged so far with the data collected.
For example, it could be considered that many profiles who have not accepted friendship could be simply inactive, since for many now Facebook is no longer the reference social network) or not very present in social networks. A list of possible future works is therefore now made, starting from these considerations.

9.1 More than 3 friend requests

Going back to the discussion made in chapter 8, the number of friend requests that each attacking profile sends could be greater and this would lead to greater accuracy of the results found.
The tools that manage the organization of friend requests (chapter 6.3) have been built to manage a very large amount of data, so already with this code created, you could deepen the request by increasing the number of requests, to see if and how the final result changes.

9.2 "Current town", "occupation" and younger users

It should be noted that other parameters could be interesting to study, which are "working occupation" and "current town".
In particular, often the friend request is dictated by work/study interests, so checking the job occupation that the profile declares to employ.
Same for the current town: maybe a profile is more secure to accept friend requests only for those profiles that indicate a current town like its o near its, or is this indifferent?
With a case study that also includes these parameters, the accuracy would certainly be greater and perhaps it would open the way to probably more interesting scenarios, although perhaps more linked to a psychological field.
In the same way, it could be considered among the various age ranges, even a range for minors, who are very often the weakest users and the victims preferred by criminals.
The problem is that there are not many profiles that claim to be minors because Facebook is no longer the social network used by these new generations; moreover, those few profiles that exist almost never make their age visible even to non-friends.
Considering that Facebook does not allow searches for people based on age, it is very

difficult to be able to include minors in some case studies, even if it would be very interesting and useful.

9.3 Profile pictures

As already reported in chapter 5.1.2 and demonstrated in chapter 8.1.3, profile pictures greatly influence the outcome of a friend request. The results show that if the attacking profile has a real image it is more easily accepted by the victims, and at the same time, a victim profile with the real profile image is more likely to accept friend requests than a profile with a hidden image.
An example of future work could be to create multiple profiles with different ages belonging to the same range (or by dividing the age into a larger number range) in order to effectively verify how much the profile image of a real person can influence the analysis, and to cross-reference the data in cases where the profile picture is false.

9.4 More accurate detection of profile pictures

A future job could certainly be to develop a more accurate detection system of profile images. As an example, **YOLO** (**Real-Time Object Detection**, detailed here: https://pjreddie.com/darknet/yolo/) is reported.
YOLO is a state-of-the-art, terminal-based, real-time object detection system that analyzes an image and declares which elements are part of it and with what percentage. An image of a person turned from behind was not always classified as a true image, as Facebook's technology could not recognize the person present and therefore did not insert in the `alt` tag.
Among other things, some recent articles report how the Facebook technology that determines the automatic content of image `alt` tags, is continuously improving for the detection of more details.
In this case study, due to timing issues, it was not possible to completely fine-tune the use of YOLO, but we want to underline however how the percentage of error that the current system produces is very low (about 2 profiles every 50 examined), and that to remedy any possible inaccuracies, a check system has been implemented, discussed in chapter 6.2.2.

9.5 Check data

A big problem in this case study is checking that the data a user has entered is valid, correct, and consistent.
For example, a 30-year-old man can safely write that as a job he is a Milan footballer when it is not blatantly the true.
Similarly, a user could write unclear things, for example, more than one profile was found that reported "I work for myself" as a job. And it might well be right, but categorizable in no way.
In the end, many people write that I work in a particular place, but they do not specify their role: for example, a mechanic might write the name of the shop where he works and not the fact that his job is to be a mechanic.
So a future work could be to take into consideration all the possible variant and to know how to evaluate and analyze them.

Capitolo 10

Conclusions

10.1 Summary

The "`Zero-Effort Attack`" tool, which was the objective of this internship, has been developed and is fully functional. So the goal was achieved brilliantly and much food for thought was reported for any future in-depth work.

10.2 Acquired knowledge

During this internship the student will have the opportunity to deepen and put his knowledge into practice, in particular:

* in the field of the automated management of browsers through the use of the Selenium framework;
* Python programming;
* tools and methodologies for data collection.

10.3 Personal evaluation

This project allowed me to deepen and put into practice knowledge related to programming with Python and Selenium and, above all, it allowed me to get my hands in one of the fields of Cybersecurity, which is the safety of people in a now ordinary place like social networks.
It would have been much more interesting to have been able to develop even just one of the points presented in chapter 9, but the short time and optimal planning that was nevertheless subject to changes to solve problems of an unpredictable nature didn't make possible the more in-depth development of the algorithm.

Appendice A

Appendice A

Citazione

Autore della citazione

Bibliografia