FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Identifying Malicious URLs: A Descriptive Pipeline for Analysis and Implementation

João Pedro Valente Fonseca

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Ricardo Santos Morla

Co-Supervisor: Carlos Novo

October 25, 2022

Abstract

People that use the internet are at constant risk. There are billions of malicious Uniform Resource Locators (URLs) available online that can steal credentials and download malware into users' computers. A point can be made that, although there are this many dangerous links, they have a short life. There are a high number of detection tools that report these cases and take them down. However, people are too trustworthy, and black hackers have become very creative in ways to evade detection. Thus, a malicious link can cause much damage in just a few hours online.

Since humans are the weakest link in a system, new ways of protection must be implemented regularly. This protection method must provide the user with an easy and intuitive mode of interaction. The less the complications it provides to the client, the better. For this reason, the goal of this dissertation was the deployment of said tool online, alongside the construction of a pipeline that can be used for other diagnosis tools and even other types of detection.

In this work, we describe the requirements that lead to results to keep the community safe from Phishing and other malicious webpages attacks. The process adopts a classification method, with its own scale, besides categorizing the type of attack to clear any doubts in the user. Other results are presented for the same reason, like the IoCs (Indicators of Compromise). The client can then use these characteristics to report the URL to the responsible authorities if he wants.

In conclusion, this dissertation tackles the problem of malicious website detection while providing a descriptive analysis of the process. The objective is that, in the future, developers can use these guidelines to develop or evolve their tools.

Resumo

Os fieis utilizadores da internet estão em constante risco. Existem milhares de milhões de Uniform Resource Locators (URLs) maliciosos disponíveis online que podem roubar credenciais e fazer download de malware para os computadores dos utilizadores. Pode-ser afirmar que, apesar de existirem um número grande de links perigosos, estes têm um vida curta. Estão disponíveis numerosas ferramentas de deteção para reportar estes casos e eliminá-los. No entanto, as pessoas são demasiado confiáveis e os black hackers tornaram-se muito criativos de forma a evitar a deteção.Assim, uma ligação maliciosa pode causar muitos danos em apenas algumas horas online.

Uma vez que os seres humanos são o elo mais fraco de um sistema, novas formas de proteção devem ser implementadas regularmente. Este método de proteção deve proporcionar ao utilizador um modo de interação fácil e intuitivo. Quanto menos as complicações proporcionar ao cliente, melhor. Por esta razão, o objetivo desta dissertação foi a implementação da referida ferramenta online, a par da construção de um processo que pode ser utilizado para outras ferramentas de diagnóstico e até outros tipos de deteção.

Neste trabalho, descrevemos os requisitos que levam a resultados para manter a comunidade a salvo de ataques de phishing e outras páginas maliciosas. O processo adota um método de classificação, com escala própria, além de categorizar o tipo de ataque para esclarecer quaisquer dúvidas no utilizador. Outros resultados são apresentados pela mesma razão, como os IoCs (Indicadores de Compromisso). O cliente pode então utilizar estas características para reportar o URL às autoridades responsáveis, se assim o quiser.

Em suma, Em conclusão, esta dissertação aborda o problema da deteção maliciosa de sites, ao mesmo tempo que fornece uma análise descritiva do processo. O objetivo é que, no futuro, os desenvolvedores possam usar estas diretrizes para desenvolver ou evoluir as suas ferramentas.

Acknowledgments

Gostaria de começar por agradecer aos meus orientadores, Professor Ricardo Morla e Carlos Novo pelo apoio ao longo deste trabalho. Sei que não foi fácil, mas se este trabalho foi completado foi devido às reuniões que tivemos ao longo do ano e ao feedback que me permitiu progredir.

Estou eternamente grato ao esforço que os meus pais fizeram ao longo destes anos de curso. Reconheço perfeitamente que é um privilégio ser estudante do ensino superior e, principalmente, de uma faculdade como a FEUP. Por isso, não há palavras pelo que eles fizeram por mim. À minha irmã, que já passou exatamente pelo que estou a experienciar, queria deixar um carinho especial. Foi o meu maior apoio durante estes anos.

Aos meus amigos tenho que deixar um agradecimento pelos momentos vividos. Há momentos que passei nesta casa que não seriam capazes de ser vividos noutro sítio. Ao longo destes anos, a minha vida tem sido dividida entre o percurso académico e a minha vida de basquetebolista profissional, por isso, não posso passar a oportunidade de agradecer por último a toda a gente que fez parte destas caminhadas.

João Fonseca

“Ars Longa
Vita Brevis”

Contents

List of Figures

List of Tables

Abbreviations and Symbols

ANOVA	Analysis of Variance
FEUP	Faculty of Engineering at University of Porto
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IoC	Indicator of Compromise
MISP	Malware Information Sharing Platform
ML	Machine Learning
OCR	Optical Character Recognition
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine
TF-IDF	Term Frequency-Inverse Document Frequency
TLD	Top Level Domain
URL	Uniform Resource Locator
VM	Virtual Machine
XGBoost	Extreme Gradient Boosting
YALIH	Yet Another Low Interaction Honeyclient
IDS	Intrusion Detection System

Chapter 1

Introduction

1.1 Motivation

The Internet has been a fast-growing network that connects all. We are connected not only with each other but also with our daily activities. Our preferred businesses are now at the distance of a click, which has facilitated our everyday life, but not without some dangers attached.

Technology keeps evolving to keep people safe and correct systems' vulnerabilities, but the most significant risk to systems is people. Staff members are commonly the weakest link of a system, which is why Blackhat hackers often take advantage of that to acquire unauthorized access to sensitive information.

Hackers often use a method called **Social Engineering** to manipulate people into disclosing confidential information, like Financial credentials, Login credentials, among others. They prey on trust, stress, and greed to reach their objective, knowingly or unknowingly, to the target. In an era where Social Media is used to keep people connected with each other, hackers use it to their advantage, which is why the most pressing cyber attack today is Phishing.

Phishing is a type of Social Engineering attack that can be used for one to a large database of targets. It has been around since the beggining of the Internet and, in 2019, 32% of all data breaches involved phishing [7]. It can be used to affect the most various number of industries. According to the report on Statista [8], the Financial industry was the most targeted in the first quarter of 2022 (Figure 1.1).

There are various types of phishing attacks (Spear Phishing, Smishing, Vishing), but whatever the type or medium used, the one thing that most of them have in common is the use of a URL (Uniform Resource Locator) that usually leads the victim to a malicious webpage imitating a Legitimate Business page, for example, to steal login credentials (a good example is the various PayPal phishing scams [9]).

URLs are not only a potent weapon in Phishing. **Spam e-mails** and **Drive-by-download attacks** are other examples. So, with the growing safety concern, while browsing the Internet, the use of tools, browser plugins, and websites for malicious URL detection has become of the utmost importance.

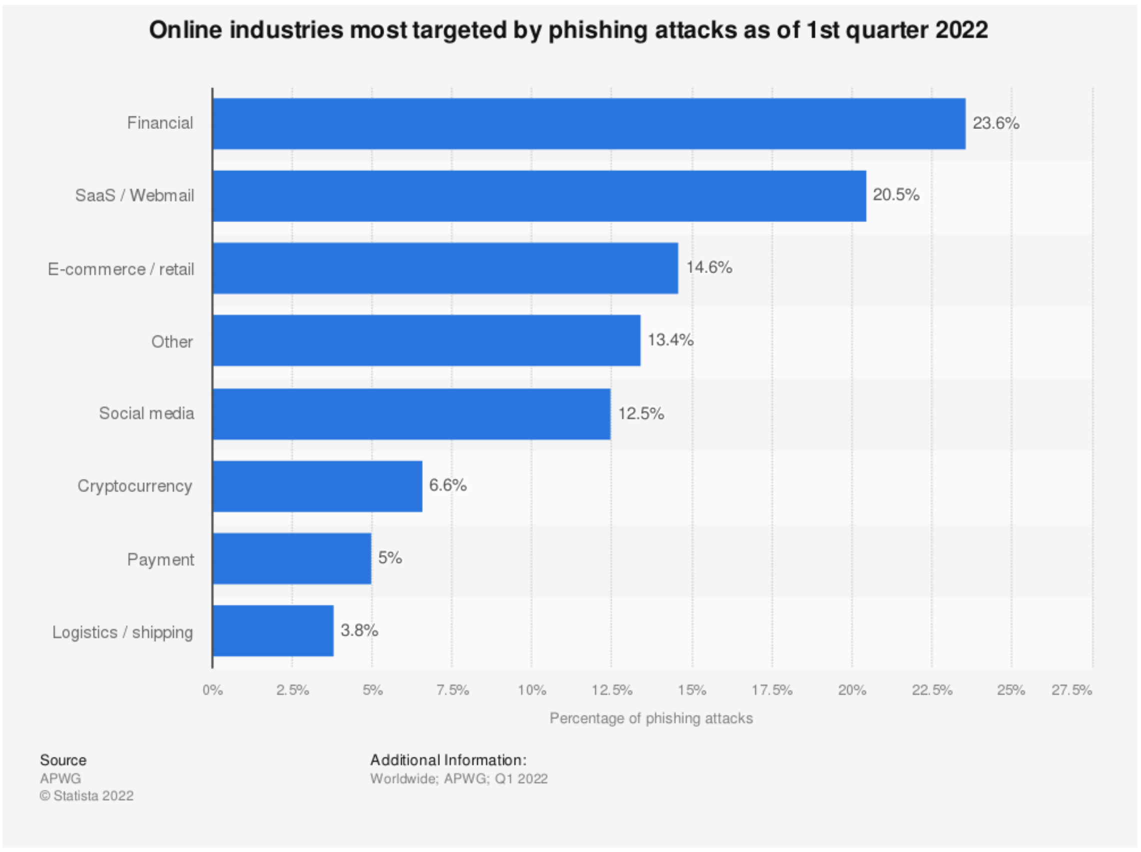


Figure 1.1: Online industries most targeted by phishing attacks as of 1st quarter 2022

1.2 Context

There is a race between cybercriminals and cybersecurity specialists. Only this one does not seem to have an end, and there is a constant change of who is winning. Blackhat hackers' intentions behind attacks can or cannot be clear. However, accessing sensitive information or compromising systems is a common goal. Brute-forcing or finding and exploiting vulnerabilities may achieve the objectives last mentioned. Nevertheless, it can take a very long time to find success, which in a race is not the ideal scenario.

Phishing, spam and drive-by downloads can fix this time problem. These are typically executed on a large scale and are very low time-consuming. Everything that is required is a list of e-mails, phone numbers, or another communication technology, and a simple fake website. In this case, there is no specific target in mind, but if there is, utilizing the same principle with a more direct approach will suffice.

Nowadays, malign web pages are abundant, and every day more and more appear. On the opposite side, spam filters and malware detection methods are simultaneously evolving. Users trust their web browser's defenser mechanisms when they are surfing the net to keep them protected, but zero-day attacks are imminent. So, having a trustworthy website where to input a suspicious URL when in doubt is a way of complementing an analysis. It classifies the URL while also providing an Incident Report, this is, analyzing and reporting a suspicious URL without having to constantly share all the links the user accesses, unlike in browsers.

1.3 Objectives

This dissertation aims to design a website where users can input a URL and receive a report on details that are of interest to the general population. To achieve this, the main topics explored in this work are as described:

- The design of a website where the users can insert a URL for analysis and get a report that is informative and complete.
- The retrieving of the IoCs, Indicators of Compromise.
- The analysis of the URL using tools that are suitable for the purpose
- The presentation of a pipeline model used for the entire analysis process, since the submission of the URL to the report construction

1.4 Structure

The structure of the document is divided as follows:

- **Chapter 2-** Literature Review: An introduction to some essential concepts related to the scope of the thesis, as well as a summary of recent studies done on the subject of malicious URL detection.
- **Chapter 3-** Requirements and Pipeline Development: Discussion of the requirements and pipeline flow proposed, as well as an overview of the hole process with the chosen tools.
- **Chapter 4-** Pratical Validation: Website design that validates the pipeline design, and discussion on the execution and results obtained and reported.
- **Chapter 5-** Conclusion and Future Work: Conclusions of the work done and how it can be further improved.

Chapter 2

Literature Review

This chapter aims to explain the dangers of Malicious Websites. We start by describing the type of attacks and obfuscation techniques, the origins, and a bit of a walk through the evolution of Phishing in section 2.1. Then, an overview of the type of detection methods used by previous authors and the main limitations of the proposed frameworks. This overview starts with the more simple method in section 2.2, Blacklisting, a rule-based detection in section 2.4 and the presentation of some honeypot techniques in section 2.5. However, the main focus is in section 2.3 on the Machine Learning approaches, since this is a more commonly used and popular method nowadays.

2.1 Background

2.1.1 Social Engineering

The increase of scams all over the internet is concerning. The "Nigerian Prince" e-mail scam is the most known worldwide. It has been a target on many comedy shows over the years since it is one of the longest-running internet fraud examples. However, this is one of a thousand other examples (Paypal Phishing Scams [9], Google Docs Phishing Scam [10], Amazon Scams [11]). These attacks are specific types of Social Engineering attacks.

Social Engineering is the craft of using manipulation to get people to divulge private information. These can lead attackers to get access to the protected systems, money from the user, and other confidential information. These attacks can be classified into two categories [12]: Human or Computer based. But it can also be categorized into the following three categories: Technical, Social and Physical based.

All this categorization falls into two main classes: Indirect and Direct (Bidirectional or Unidirectional) communication. To better understand this, it is needed to break down Social Engineering attacks into components:

- **Social Engineer**: The individual or the group of individuals performing the attack
- **Target**: The person or organization the attacker is trying to take advantage

- **Compliance Principles**: The principle the Social Engineer is going to explore in order to perform the attack (authority, scarcity, friendship,...)
- **Techniques**: It is the main component. It is what classifies the type of attack and determines how the Social Engineer executes it (Phishing, Pretexting, Quid Pro Quo,...)
- **Medium**: It's the method used to interact between the Social Engineer and the Target (e-mail, face to face, telephone, SMS, webpage,...)
- **Goal**: The end goal of this all operation (financial gain, unauthorized access, service disruption)

Using one of this era's most superb hackers, Kevin Mitnick in [1], this work presents the Social Engineering attack cycle used when performing attacks. Figure 2.1 is taken from the said article and presents the different stages of this process.

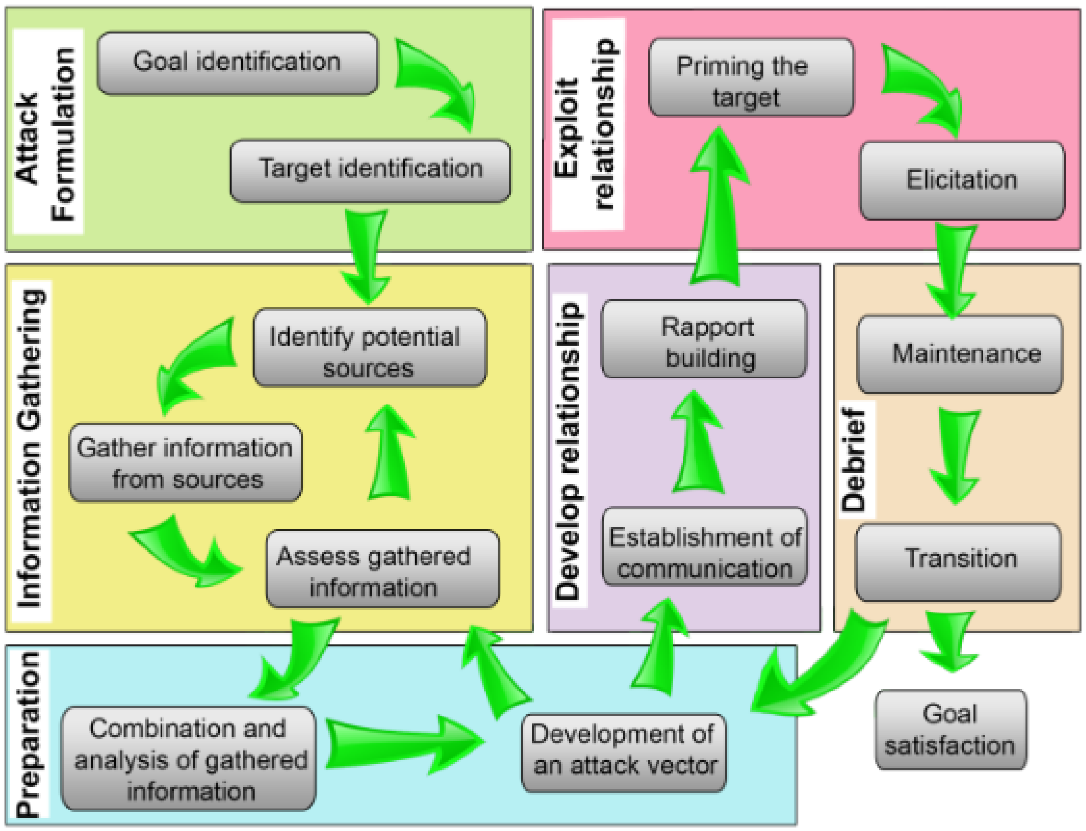


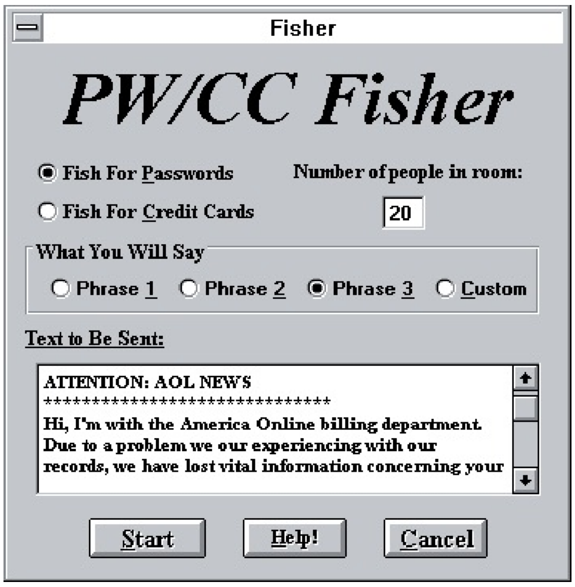
Figure 2.1: Mitnick's Original Social Engineering Attack cycle [1]

In this project, the focus is on Malicious URLs, which, considering the previous components presented, can be used together with e-mail as a medium for Phishing attacks. The following section discusses the background and early attacks performed that affect our daily lives to this day.

2.1.2 Phishing

So, what exactly is Phishing? Phishing is the combination of Social Engineering and technical methods to convice the user to reveal their personal data [13]. Phishing is a big problem nowadays because, no matter how many plugins browsers have for protection, and tools available for detection, if people are not educated about this subject, there will always be a risk attached to web browsing.

The history of Phishing can be traced back to the mid-1990s. In those years, the online service provider AOL (America Online), one of the early pioneers of the internet, offered dial-up and instant messaging services. This caught the eye of hackers like Koceilah Rekouche, who developed an automated password and credit card stealing software [2]. Performing this attack was simplified to a simple check of a box on the GUI, as presented in Figure 2.2, that it quickly spread, and the word Phishing was coined for this type of attack.



[✓ Fish For Passwords] Fish For Credit Cards

Number of people in room: 20

What You Will Say: Phrase 1 Phrase 2 [✓ Phrase 3] Custom

Text to Be Sent:

ATTENTION: AOL NEWS
****
Hi, I'm with the America Online billing department.
Due to a problem we our experiencing with our
records, we have lost vital information concerning your

Figure 2.2: AOHell's Phihsing Configuration Screen [2]

Although, in this era there are others attacks who are becoming very concerning, like Credential Stuffing [14], Phishing is still the most common. However, there are various types of Phishing, like Spear Phishing, Whaling, Clone Phishing, Smishing, and Vishing [15]. Still, all of the types can be inserted into four categories:

- **Spoofing E-mail**: It's an attack where the hacker sends an e-mail using some other e-mail address, impersonating the legitimate person of organization.

- **Fake Social Network Accounts**: The attacker creates a fake social media account (very common on sites like Facebook, Instagram, LinkedIn) to send malicious URLs or to interact with users in order to have them disclose confidential information.

- **Hacking**: Hackers perform a vulnerability scan on systems to uncover their weaknesses and perform attacks (Brute Force, for example). This is where Phishing can be very useful. An employee can disclose a password to the hacker giving access to the network.
- **Trojan Horse**: It's an executable program that grants control over the device to the hacker.

In conclusion, Phishing constantly threatens Internet users and their businesses. An example is the Ethereum blockchain, which has recently become a target of scams referred on [16]. The popularity has led to an increase in Detection methods. Blacklists, Machine Learning, Virtual Machines execution, Rule-based techniques and Honeypots are just some of the ways authors referred in the next sections use to Detect Malicious Websites.

2.2 Blacklisting

If a malicious URL is detected, it should be reported. That is where Blacklists are useful. This database's infrastructures provide a form to inform users which sites have malicious intent. Blacklists are specially used by web browsers and antivirus due to low expenditure of computing resources, which makes the web browsing delay almost unnoticeable. The effectiveness of these software depends on their size, scope, update speed and frequency, and accuracy [17]. Some examples of known Blacklists are PhishTank [18], Google Safe Browsing [19] and OpenPhish [20].

In [21], the author proposes an automatic Phishing Detection and Incident Response Framework based on Blacklisting. The Incident Response part of the article consists on using Databases to cross check the e-mail and the URL, and warn the victim. In the Phishing Detection framework, a honeypot (method discussed on section 2.5) is added in order to prevent any incidents.

When determining the intent of a website, it is crucial to maintain a false negative (a false negative is classifying a website as benign when it is malicious), which is why the use of Blacklisting can be overlooked. This method has some disadvantages, like being easily evaded by zero-day attacks. Moreover, with the creation rate of these sites, it is difficult for the lists to be kept up to date. So, new techniques need to be implemented to combat this growing concern.

2.3 Machine Learning

The percentage of malicious websites using Secure Sockets Layer protocol has increased up to 83% in 2021 [22]. This is very worrying, considering that in the old days, users considered themselves safe just by looking at the SSL sign. However, nowadays, there is no connection to that, except that the website is most probably malicious if this sign is not there. Looking at the most recent years, in 2020, the Covid-19 pandemic striked and the day-to-day life of people was transformed. Businesses had to change how they operated, leading to a massive transition to online marketing. However, this attracted the attention of cybercriminals, leading to increased cybercrimes. According to the Phishing and Malware report by Vade in the first quarter of 2022 [23], their tool detected 32.9 million malware emails (including malicious URLs and executables).

This represented a 201% increase over February, and with the increase of zero-day attacks, a technique that can learn and constantly be updated was needed. So, the interest in ML models became essential for cybersecurity specialists.

ML is growing in many areas, and cybersecurity is no stranger to this trend. This section covers feature extraction and selection, different types of attributes, dataset construction, data preparation and cleaning, different model training and validation techniques, among other topics.

2.3.1 Web Scrapping and Crawling for Data

Web Scrapping and Web Crawling are two different concepts for data collection. Both can be used for feature extraction, important part of the data analysis process, and are commonly performed using the programming language Python. But, what are exactly the differences?

- **Web Crawling**: A web crawler is a bot used by many search engines, such as Google, due to it's ability to rapidly retrieve big amonts of information on a topic (namely, retrieve webistes). In [24], the author proposes a web crawler that uses only two python libraries: *urllib.request* and *BeautifulSoup4*.

- **Web Scrapping**: It's a program for extracting the data from websites. Often combined with a web crawler in order to retrieve URLs from the web, it mainly focus on structuring information (database or spreadsheet construction). In [25], *scrapy*, a web scrapping technique, is introduced. The framework is as follows:

    1. The web crawler draws the desired links from the web;
    2. Then the data is extracted to get the source links;
    3. Finally, the data is organized and stored into a csv file.

Data collection is the first step in any ML process. Without data, there is no process to train the model and give it the required information. This previously mentioned phase is crucial, but in most cases, data is already provided. There are a large number of datasets available online that can be used. Therefore, some of the articles cited in this work exclude that part, starting with Data Preparation as the first component. The ML life sequence is presented here:

1. Data Preparation
2. Build the Model
3. Deploy the Model

2.3.2 Machine Learning Detection Techniques

We are not discarding the techniques mentioned in section 2.3.1. They can be helpful, like in [26], where the tool *Prophiler* classifies pages collected by a web crawler. This paper aims to reduce the number of web pages that need to be dynamically analyzed to identify malicious web pages, so naturally, the web crawler is very important here. Nevertheless, to simplify, we are focusing on the top 3 components of an ML life cycle in the following presentation of some of the architectures for malicious URL detection.

Classifying a link using ML algorithms can sometimes be associated with a binary outcome, either malicious or not. However, with the growing interest in technology, people want to know what a URL can do and what type of attack they can suffer. An example is [27], where the authors, besides the binary classification, distinguish between Phishing, spam, and malware.

More and more people nowadays use a smartphone. And since mobile webpages tend to be different content-wise from desktop browser pages, a specific ML method for detecting this malicious behavior was built in [28]. This determination was based on the static (passive) as well as dynamic features(active).

There are such a significant number of these types of approaches, that for the rest of this section there will be a separation into two types:

- **Active Detection**: URL classification based on the content of the pages (content and visual based features)
- **Passive Detection**: URL classification based on the attributes such as WHOIS records, IP address, and location, without visiting the page (Lexical and host based features)

2.3.3 Passive Detection Methods

The most common techniques to ensure victims are redirected a Phishing page without them being aware of it are[29] :

- URL hiding (URL shortening)
- Homograph spoofing
- Typosquatting
- Soundsquatting
- Combosquatting

So, to combat this, passive detection techniques were introduced. They have a lot of advantages, like not visiting the web pages. The features are extracted from the URL, domain name, and DNS server, among others. These attributes are retrieved without putting the machine at risk, since there is no need to visit the webpage. Another advantage is not relying on the downloading and analysis of the page content is a lightweight operation, and, with being independent from being tied to a particular application setting, these approaches are resistant to content cloaking, a very used technique by attackers to evade detection.

The authors of [30], take advantage of this features and create a dataset using lexical and host-name features (Table 2.1). In [6] it is the same approach combined with online algorithms to rapidly adapt to the evolving distribution features that characterize malicious URLs over time.

Lexical	Host-based
Hostname	WHOIS info
Primary domain	IP prefix
Path tokens	AS number
Last path token	Greographic
TLD (Top level domain)	Connection speed
Lexical misc.	Host misc.

Table 2.1: Lexical and Host-based features [6]

In [31] the writers make a simple binary classification pipeline. First is the extraction of eight lexical and host-based features from PhishTank. Then, the training and testing of the dataset using Random Forest. When ready, it is the input of a URL and labeling of the same. Finally, a performance analysis using the Receiver Operating Characteristic (ROC) curve, sensitivity, accuracy and a Confusion Matrix.

Classifying URLs can be more than just binary. The MRS (Microsoft Reputation Service) can categorize the URL. For example, the category for *https://sigarra.up.pt/feup/pt/web_page.inicial* is Education. Based on this, Mohammed Nazim Feroz and Susan Mengel in [32] a hybrid approach using clustering and classification to provide a detection method using ranking. The URL is categorized and ranked when inputted, meaning it is attributed a benign and phishing percentage, and a category. These percentages then attribute one of three possible results, Benign (green), Moderate (yellow), or Severe (red). Another categorization model is proposed in [33]. There are three categories: Spam, Phishing and Malware URLs. In the process of data collection, lexical features are extracted for each category and pre-processed. After, there is the training phase and then the prediction. Deep learning models like convolutional neural are the choice for the binary prediction (benign or malign) and for the multi-labeling classifier, an SVM.

PhishStorm [34] is an ML mechanism that uses distributed real-time computation to infer intra-URL relatedness. For this purpose, it focuses on the use of lexical features. PhishScore [35] is the continuation of the work of PsishStorm, by adding new features.

CatchPhish [36] proposes a technique that uses hostname, full URL, and phishing words from the suspicious URL for the classification. A Term Frequency-Inverse Document Frequency (TF-IDF) algorithm was used to calculate the score of each term.

More recent events, like the covid-19 pandemic, have brought to light new problems. The appearance of a considerable number of malign websites related to this subject is increasing on a week-to-week basis. The simple addition or repetition of a letter or bitsquatting in the domain threatens users. A simple typosquatting, and people could be entering sensitive data into an attacker's hands. To avoid that, in [37] domain name detection is performed using a small number of lexical features retrieved from the URLs and applied to online and batch learning models. Six ML algorithms were involved: Decision Tree, Random Forest, Gradient Boosting, Extreme Gradient Boosting (XGBoost), a Support Vector Machine (SVM), and a Multilayer Perceptron. The conclusion taken is that the XGBoost algorithm outperformed the other models.

With the introduction of the text limitation on social media platforms to prevent malicious websites from being shared, the use of short URLs for this purpose has increased. There are various shortening services available. However, Bitly and TinyURL are the most famous. Bitly [38] does not provide a CAPTCHA to test human identity at the time of shortening. For protection against spam, it claims to use a blacklisting method. In [39] the writers use a combination of content (the content of the tweet), context, and social features to detect a tweet containing a malicious URL.

2.3.4 Active Detection Methods

With the growing popularity of URL shortening services, passive methods have been stuggling to keep up. To counter this, active detection methods are explored in the articles we will present in this section. Active techniques take advantage of Content-based features, like characteristics of the JavaScript and HTML code, obfuscation IoCs, and visual elements of the page. For example, in [40], a collection of 41 attributes are extracted. These are of four categories: web-based network traffic, URL keywords, web host information, and web content. An ANOVA (Analysis of Variance) test and XGBoost algorithm reduce the number of features to the most essential. 17 is the new number of features. Finally, the dataset is used to learn the XGBoost classifier. This example is for drive-by download links only. In [41], the author is not only capable of detecting the drive-by download attack but also the attack class: Plugin memory violation, Plugin unsafe API, and Browser memory violation.

In 2007, Yue Zhang, Jason Hong, and Lorrie Cranor developed *CANTINA*, a novel content-based approach for detecting phishing websites [42]. The software work flow is as follows:

- Given a web page, calculates the TF-IDF score of each of the terms on the page
- Generates a lexical signature by taking the five terms with the highest TF-IDF weights
- Feed this to a search engine, in this case Google
- If the domain name of the current web page matches the domain name of the N top search results, it is considered a legitimate web site. If not, it is considered a phishing site.

Using images for content-based phishing analysis can be very effective. A tool called *GoldPhish* [43] is an example. To better explain the design approach can be broken down into three major steps: First, capture the image of the current website in the user's browser. Second, using Optical Character Recognition (OCR) techniques, the tool converts the image into computer-readable text. Third, similiar to *CANTINA*, input the converted text into a search engine to retrieve results. The same process is used in [44], the only difference being, instead of capturing the entire page, they focus on the website logo only.

The last visual based tool we are going to talk about here is Phish-IRIS [3]. This technique uses compact visual descriptors, such as CSD (Color Space Descriptor), DCD (Dominant Colour Descriptor), and HTD (Homogeneous Texture Histogram). These models are then applied on a single screenshot of the whole page, just as the two visual techniques mentioned in [43] and [44], and also on a more multi-level representation that divides the input screenshot into equal sized 2x2, 3x3 and 4x4 grid cells, for feature extraction. The attributes are used for training the SVM and Random Forest prediction models. The whole system Flowchart is shown in Figure 2.3.

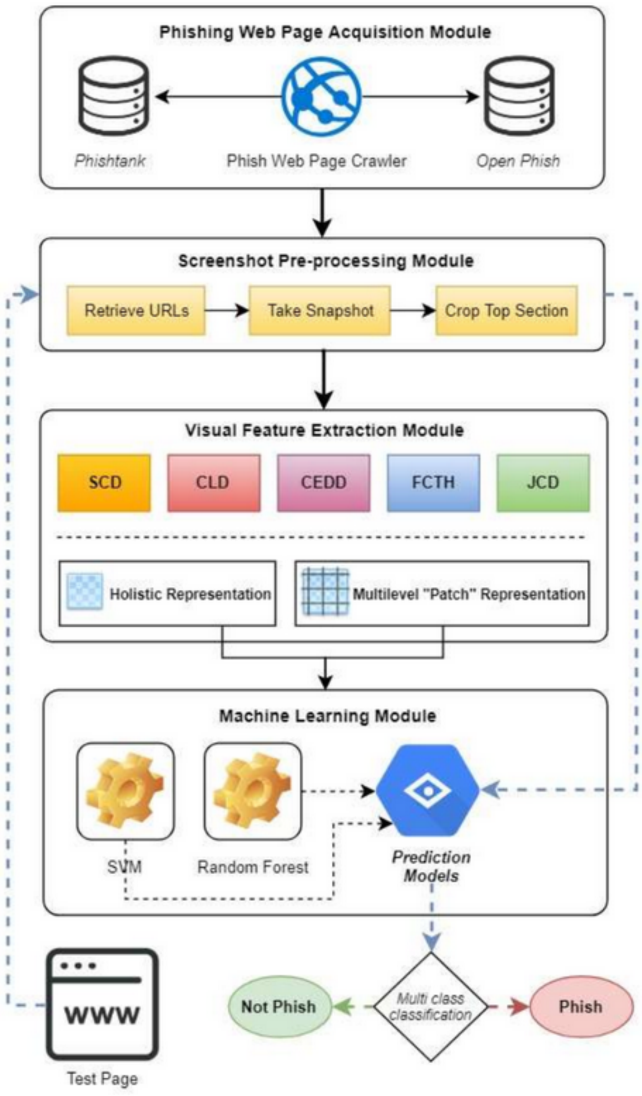


Figure 2.3: Phish-IRIS System Flowchart [3]

Finally, in [45] the authors purpose an intelligent phishing detection architecture using a new approach that manipulates features of images, frames and text of phishing websites.

2.4 Rule-based Detection

This section discusses the possibility of rules, mostly YARA rules, used as a way to detect malicious webpages. The work developed in [46] is on an automated, low-interaction malicious webpage detector *WebMon*. Although this tool is ML and YARA-based, we focus on the last part.

Malware can be encrypted with EK (exploit toolkits) in the code for the website, composing a drive-by-download attack. That is the area where YARA rules are more effective. WebMon consists of a queue server, Docker, with multiple containers and a database. However this rule-based approach only detects drive-by download pages.

In [47], the approach detects any type of malign links. Rules have two outcomes, either the rule is satisfied or not, 0 or 1. Consequently, the compliance with the rules is transformed into features to be later applied to ML models. A few examples are the following features:

- **Feature 1**: if the URL contains the IP address is 1, if not 0.
- **Feature 2**: if website uses https protocol for data transfer is 1, if not 0.
- **Feature 3, 4 and 5**: if a selected blacklists keywords appear on the domain, path or query part of the link is 1, if not 0.

2.5 Honeypots

The last studied approach is the use of Honeypots. Honeypots function like a decoy alongside traditional detection systems, like firewalls and IDSs. This way, attacks can be deflected into the honeypot, where the system can be exploited. Honeypots can be categorized in two types [48]:

- **Active**: Also called Honeyclients, these interact with the webpages to detect malicious intent.
- **Passive**: Passively waits for the attack to detect them.

For this thesis, we are going to be focusing our attention on the Active honeypots, the Honeyclients.The reason behind this choice is that, since this is a detection tool, there is no use for passive Honeypots. We are trying to actively detect, so there is a need for software that will go online in search of malware. Which can classified as two types themselves:

- **High-interaction Honeypots**: These use real systems with real applications that can be infected and grant full access to the hacker. The indicated try to take advantage of this fact to make the hacker launch further network attacks.
- **Low-interaction Honeypots**: These use emulated parts of systems, such as network stacks or browsers. As expected, they can't be exploited to give full access to the hacker.

One of the first of this type was PhoneyC [49]. This tool simulated a browser to obtain the content of a webpage. It then ran it through an antivirus to check for alerts. Besides this, the code was parsed by language to be analyzed by the respective script engine for alerts. The final step was obtaining the links from the HTML page to be again inputted to PhoneyC for analysis.

In [50], a combination of a web crawler with a Low-interaction Honeyclient was used to identify malicious web programs from a list of websites. PythonHoneyMonkey [4], another example, uses a JavaScript tool called Spider Monkey to detect obfuscated code sent to the browsers via web servers. This application also presents different Operating Systems to simulate for detection (Figure 2.4).

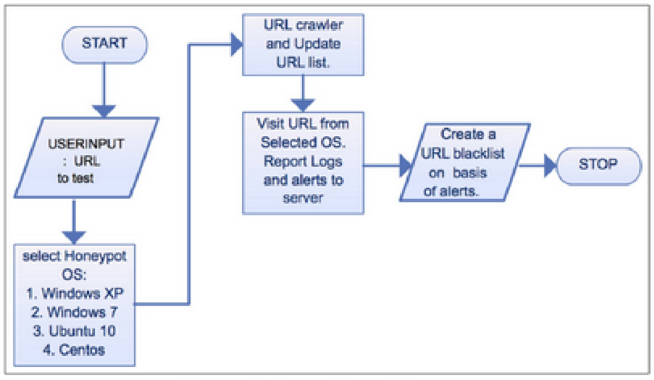

START
USERINPUT : URL to test
select Honeypot OS: 1. Windows XP 2. Windows 7 3. Ubuntu 10 4. Centos
URL crawler and Update URL list.
Visit URL from Selected OS. Report Logs and alerts to server
Create a URL blacklist on basis of alerts.
STOP

Figure 2.4: PythonHoneyMonkey Work Flow [4]

Thug [51] and YALIH [5] are two tools we will take advantage of further in our work. YALIH, uses a more signature based approach, employing ClamAv antivirus and YARA-rules for detection (Figure 2.5). As for Thug, the focus is to mimick the behavior of a web browser in order to detect malicious content. Further in this document, a deeper explanation of this tool will be given.

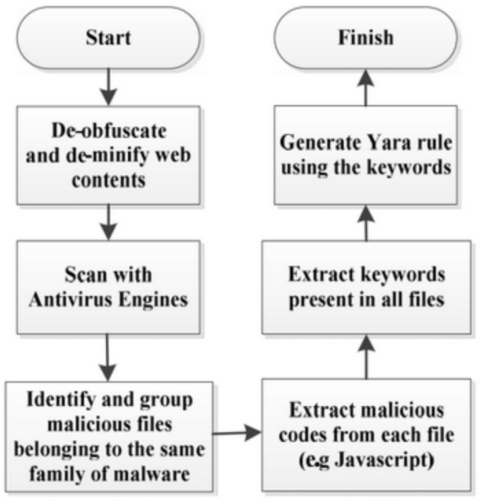

Start
De-obfuscate and de-minify web contents
Scan with Antivirus Engines
Identify and group malicious files belonging to the same family of malware
Extract malicious codes from each file (e.g Javascript)
Extract keywords present in all files
Generate Yara rule using the keywords
Finish

Figure 2.5: Steps in automatic YARA rules generation [5]

For this thesis, we are focusing on Low-interaction Honeyclients. The reason for this choice is because they present a more effective process of analysis when integrated with the rest of the methods.

2.6 Discussion

This chapter aims to paint a picture of how advanced this area is. The last article we are going to discuss here is the proof. In [52] the authors developed a tool that automates human behavior for detecting phishing websites. The tool actually imitates human behavior, like feeding a website with a login sector with fake credentials to learn if it is legitimate or not.

Thus, with many options, what is the best approach to take? Further in the document, a list of requirements is available, determining the goals for our choice. However, since the goal is to build a website where students can enter a domain and trust the results, ML did not seem like an excellent strategy. Has referred to before, this is a tool for analysis on a big scale when the requirements demand a faster process and not much focus on just one input at a time. Therefore, combining other software mentioned in this chapter at the cost of a slower executing time seemed to be the preferred option.

Chapter 3

Requirements and Pipeline Development

This chapter serves as the presentation of the initial steps in order to design the analysis pipeline. In section 3.1, we gather the requirements that need to be satisfied by the process, this is, what the target population wants to acquire from the use of this tool. Afterward, we dive into the pipeline description, where we go into detail on the goal for each phase and the tools to achieve it. The phases are as follows:

- In section 3.2 an overview of the pipeline is exhibited;
- In section 3.3 the detection of private information;
- In section 3.4 a early analysis of the reputation score of the URL is done;
- In section 3.5 some IoCs are presented and the process of their retrieval is explained;
- Finnaly, in section 3.6 a dynamic analysis is performed on the URL with different stages.

3.1 Requirements for URL analysis

Every project has objectives that we aim to achieve. These have already been disclosed in section 1.3 and are very discrete and straightforward. Objectives act more as a guideline, as a project can be successful with not all of these accomplished. However, another parameter of a project is the requirements. They differ from the previously mentioned as these are a necessary conditions, something that must be accomplished in order for the success of the dissertation.

In previous chapters, we have mentioned that this dissertation contains two main focuses: the development of a tool that can be publicly available for URL analysis and a pipeline description of this analysis. Therefore, we can divide the requirements into two main groups: Tool development requirements and analysis pipeline requirements.

First, we will address the tool development requirements. These are characteristics that are necessary when we are developing the website. The first two can be considered the most critical:

Tool Development Requirements	Analysis Pipeline Requirements
Report the URL classification	Determine the end results
Report the URL categorization	Present the steps of the process
Check for private information on the URL	Present type of tools to use in each phase
If private information is detected, warn the user of this presence	Filter the results to the necessary
Report the effective URL	Construct a database with the results
Connect the results database with the website	
Construct a pleasant graphical interface	

Table 3.1: The two main objectives of this thesis and their respective requirements

report to the user if the URL is benign or malicious, and if the latter is true, what type of attack the user can be the target. A FEUP student does this thesis. Therefore, it is natural that most people using this website will also be engineering students. So, it is required that the tool report the nature of the link and the categorization for the population of users to make a decision with the most information possible. This raises another necessity, the need to report the presence of private information. As referred, the target population is mainly constructed of engineers or future engineers. And, as such, these people are more aware of the effects of having their private information available online. So, before advancing with the analysis, we need to have the tool flag the presence of this type of information to the user. Proceeding with this train of thought, the effective URL is another necessity of this report. All this information needs a storage place, a database. A database with the information gathered from the pipeline execution needs to be connected to the website. This way, any time an inputted URL has already an entrance on the dataset, there is no need for the user to wait long for the results. The website can retrieve them and present the report.

In short, a pleasing graphical interface for the website is the final requirement for this first part. Because poorly constructed webpages do not inspire a sense of confidence in the user.

The second part is the analysis pipeline requirements. Here, the requirements aim to outline the principles on which any analysis process can base itself to obtain this type of results. First, we need to determine the end results we aim to obtain at the finish of the analysis. These are the end goals that every step of the process will be devoted to achieving, which remarks the second requirement of this part, the presentation of each step. These are the levels that every tool can base themselves on to analyze a link and gather essential information each step of the way. Therefore, we need to introduce the type of tools to use in every step and some options from which people can choose. At the end of the analysis, we need to filter the results we gather to only the essential ones. Every tool we choose will present more results than necessary. Therefore, we must interpret and filter them to the strictly necessary for this analysis. This can increase the pipeline execution speed and simplify the process for other people with less knowledge in this area. To finish, a database that stores these results is required. This will link the pipeline of analysis we follow and the tool we use to interact with the user.

The summary of the requirements of this work is presented on Table 3.1.

The following section describes the overview of the pipeline described by the requirements above.

3.2 Pipeline Overview

In section 3.1, we present the basis for this work's construction: the user's requirements, and, from these remarks, we can construct the pipeline for the malicious URL analysis that the website is going to present.

Following the this Pipeline Overview, we will make a presentation on each phase and what are their goals. In this thesis, we aim to present the approach we took and the overall view of the process for other developers that may need to design a new detection method. For that reason, different tools are mentioned in the sections that can accomplish each of these goals.

The requirements already obtained and the description of each of the steps of the process will be presented on the following sections individually. Therefore, a pipeline overview is required to understand how they interact. A workflow diagram is shown in Figure 3.1 to understand better.

The flow starts with a URL submission. The first action is to check if other users have already submitted the link. If true, the report is constructed from the data encountered. If not, the process advances to check the URL for private information. In the case there is this type of data, the user is notified and decides to continue with the process or to end it. Next, we import the reputation score and store it. Before gathering the IoCs, if needed, we retrieve the effective URL. The results of the previous action are stored. Finally, we perform a Dynamic Analysis. The classification of the URL (a malicious rate of 0 to 10) and, if malicious, the categorization (Phishing or a type of drive-by download attack is specified) are stored.

Figure 3.2 is a simple Swimlane explaining the Dynamic Analysis process of the pipeline. We go into a more detailed explanation on Section 4.2.

3.3 Detection of Private Information

When dealing with users, exceptional attention to their personal information needs to be a focal point. Therefore, when a user inputs a URL, if it contains any personal information, like an e-mail address, phone number, name of the person, IP address, among others, we must abort the process until permission by the user is granted. Thus, it is only natural that the first phase of the process is the checking of any type of the previous mentioned data.

Firstly, we take advantage of an open-source software called *Hakrawler*, designed by a Pentester named Luke Stephens (@hakluke) [53], for subdomain enumeration which uncovers all the hidden URLs asociated with the main one. This procedure is a way of checking all the related links to the original one. If any contains any type of private information or indications of it, the user is warned.

Afterwards, usually, if a URL contains personal information about the user, it is visible. Nevertheless, there are cases where this data comes hiding behind some type of coding. Several

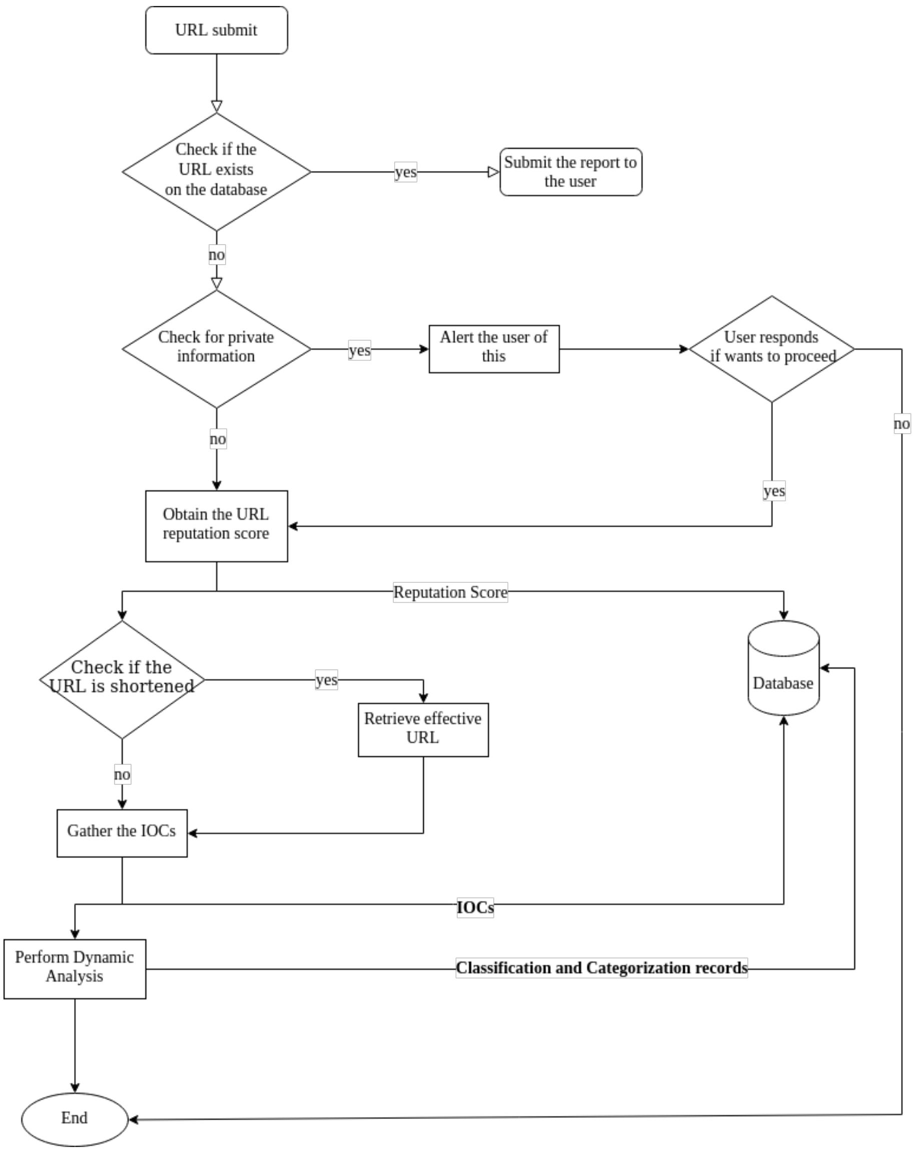

URL submit
Check if the URL exists on the database
yes
Submit the report to the user
no
Check for private information
yes
Alert the user of this
User responds if wants to proceed
no
no
yes
Obtain the URL reputation score
Reputation Score
Database
Check if the URL is shortened
yes
Retrieve effective URL
no
Gather the IOCs
IOCs
Perform Dynamic Analysis
Classification and Categorization records
End

Figure 3.1: Flow chart of the analysis pipeline

approaches can be taken here, like the use of CyberChef [54] for decoding. Therefore, to simplify things, the straightforward JavaScript code for decoding listed bellow was used.

```
let decoded = Response.Write(Server.URLDecode(urlString));

console.log(decoded);
```

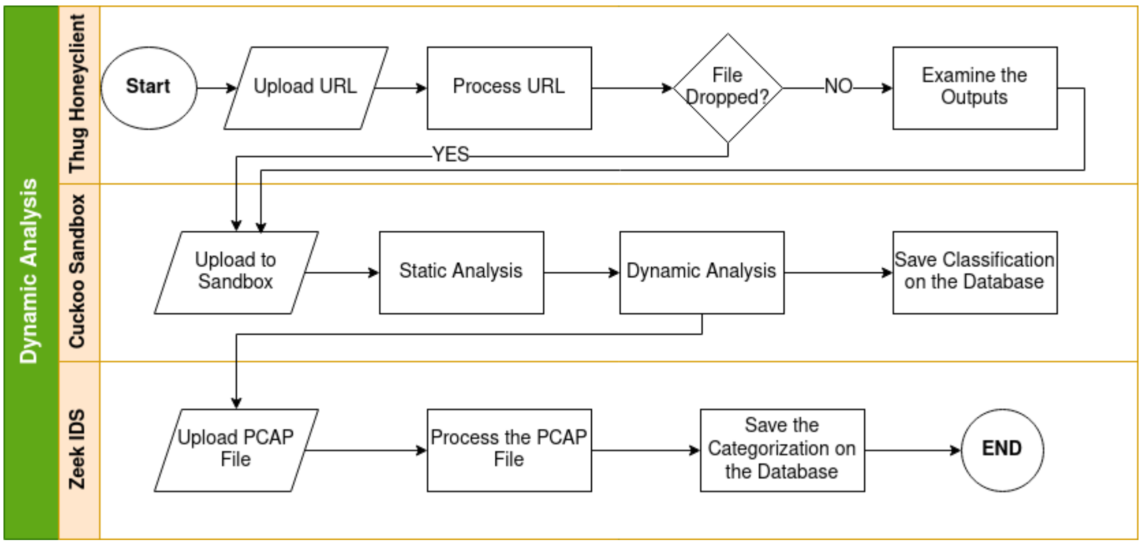


Figure 3.2: Dynamic Analysis Swimlane

For parsing the information presented on the URL the same language option as before was selected. The code is listed as follows:

```
//function getVal(){
//    const inputUrl = document.querySelector('input').value;
//    console.log(inputUrl)
//}

var readline = require('readline');
var resp = "";

var leitor = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

function parseURLParams(url) {
    var queryStart = url.indexOf("?") + 1,
        queryEnd   = url.indexOf("#") + 1 || url.length + 1,
        query = url.slice(queryStart, queryEnd - 1),
        pairs = query.replace(/\+/g, " ").split("&"),
        parms = {}, i, n, v, nv;

    if (query === url || query === "") return;

    for (i = 0; i < pairs.length; i++) {
        nv = pairs[i].split("=", 2);
        n = decodeURIComponent(nv[0]);
        v = decodeURIComponent(nv[1]);

        if (!parms.hasOwnProperty(n)) parms[n] = [];
        parms[n].push(nv.length === 2 ? v : null);
    }
    return parms;
}

leitor.question("Which is the URL to analyze for private information?\n",
function(answer){
    var resp = answer;
    urlParams = parseURLParams(resp);
    console.log(urlParams);
    leitor.close()
});

//var urlString = "www.mints.com?name=something";
//   urlParams = parseURLParams(urlString);
//
//   console.log(urlParams)
```

A common practice by hackers is to cipher the user's private information. AES 256, triple DES, RSA and other encryption techniques can be deployed, making the information inaccessible to us analyzing the link. Therefore, the user is warned of this possibility when the user clicks on the button to analyze the URL.

Another malicious practice that has become more and more common in present days is the use of a specific URL for a specific user that does not contain any private information. A modern example is the messages people receive on social media with a shortened URL from an infected user (sometimes, these are people we know). These vary from target to target, and if the hacker receives a request in a particular domain, it transmits information about a specific user (like that the user frequently checks the incoming e-mails, that it opens these links, making him vulnerable, the browser, the time it was opened, among others). In conclusion, although it does not transmit any private information to us, the attacker can start sending more e-mails to this address in particular (Spear Phishing). Therefore the user is warned of this possibility and cautioned to proceed at his own risk.

3.4 URL Reputation Score

This section is the first actually to classify the URL. Here, we take advantage of how fast and lightweighted that **Blacklisting** approach is. Several databases and tools are available online to obtain this score. To name a few: PhishTank [18], URLVOID [55], OpenPhish [20], SURBL [56], Malware Domain List [57], Should I Click [58], VirusTotal [59], among others.

As mentioned in section 2.2, blacklisting tools have a significant disadvantage in detecting malicious links. However, this step's primary focus is not on obtaining an undeniably correct result, but on presenting the users with a preliminary result, something they can base on, but not with too much certainty. One of the problems with the mechanisms described in future sections to obtain a more reliable result is the time it takes to execute them. Therefore, with this intermediary step, we provide the user with an outcome that has a false positive rate of 0% (meaning if it determines the URL malicious, it is 100% correct) while adding no delay to the execution time.

The type of result can variate on the tool used. For example, VirusTotal provides a numeric outcome. If the designer decides to use a blacklist like OpenPhish, the outcome is binary (OpenPhish is a database of phishing links, the input is either matched on this tool or not). Therefore, it falls to the person developing the analysis process to decide which one to use.

Here, the question of how different the tool described is from VirusTotal emerges. Well. VirusTotal has difficulty detecting zero-day attacks due to its reliance on blacklists and anti-virus engines, has for our tool, since we are classifying based on methods of blacklists combined with characteristics of the behavior, among other execution parameters, it does not share this downside. Another aspect of this tool is that it is developed for day-to-day use. Because of that, we provide a complete report that also includes the categorization of the URL, letting the users know what would happen if the link was clicked.These are two of the main differences between these softwares.

3.5 Retrieve IoCs information

In this phase, the goal is to retrieve the IoCs that characterize the URL. Indicators of Compromise are data that can identify potentially malicious activity on a system. This data characterizes the malware and its behavior. In the case of URLs, the IP, geographical, and domain name data identify each one. Therefore, in order to build a report on a malicious website, this information is required. These IoCs can be critical to detection tools. Taking down a malicious website does not remove the hypothesis that another website with the same characteristics can be online in the future or even right now. An example of this is browser plugins. Typically, if an IP address is flagged as malicious, if the user tries to access a website with this data, the browser will warn him.

Now that we established that retrieving this data can be important in every tool of this genre, we face another problem: how to retrieve this data about the URL? To complete this job, there are some open-spource tools available, such as socialinvestigator [60] and some online websites [61]. Another example, is Maltego [62]. Maltego can be a very powerful tool in gathering information.

3.5.1 URL shorteners

In the previous section, the majority of the examples require the domain name or the IP address to retrieve the IoCs, which can be an issue if the URL is shortened. As already established, these shorteners are a common strategy attackers use to mislead users into following a link they otherwise would not. The most known URL shortening services are Bitly and TinyURL. Thus, before gathering data about the link, we need to expand it, if the case, of course.

A simple website that recovers the original, effective URL should suffice in this intermediary step.

In conclusion, this step does not analyze the link's nature but the characteristics. It is an information gathering phase, just like Section 3.3, differentiating on the type of results. Here IoCs are the result. These IoCs can be used to build a report (or, in this case, use the one provided by this tool) and report it to the competent authorities. Sites like Google Safe Browsing or even the Malware Information Sharing Platform (MISP, an open source software solution for collecting, storing, distributing and sharing cyber security indicators and threats about cyber security) can be utilized to perform this task.

3.6 Dynamic Analysis

As explained earlier, the main objective of this thesis is to allow the users to have a way of submitting a URL and get an analysis report that is accurate and fulfills their requirements. We have already described the early stages of the process. Accordingly, following the flow, this section describes the last phase, the Multi-labeling classification pipeline using dynamic analysis.

Although it may seem confusing to have a pipeline inside another pipeline, the requirements demand this type of analysis in the last step. One of the essential requirements is not only the input link being classified as malign or not but also categorized. The categorization process demands more investigation because this labeling distinguishes phishing attacks from drive-by download attacks, but inside this last category, there are other categories of payloads (Trojan Horses, ransomware, rootkits...). Therefore, to obtain such categorization, we did not find one in our investigation but a collection of tools that, if put together, can accomplish it.

3.6.1 Low-interaction Honeyclients

Section 2.5 describes low-interaction honeyclients as one of the most robust tools for URL analysis. These can simulate a web browser, permitting the link to perform as intended while logging all possible information. A few reliable options are available from our investigation on the subject. Therefore, in the selection process we consider three major objectives that the Honeyclient needed to focus on:

1. Luring the attackers with a bait. For example, adding fake listen ports.

2. Identifying the attackers from their actions. For example, if the web client is trying to acces the fake ports, it will be tagged as malign behavior.

3. Gathering information about the attacker from the logs of their action.

The studied tool that best fulfilled these requirements was Thug. Other tools were considered for this part, like YALIH or HoneyMonkey. Nevertheless, the decision to use Thug, based on our main goal (classifying and categorizing a URL), was made after an analysis of the options available. The previously mentioned software provided more goal-oriented results based on the URL's behavior. Besides, there was no need to allocate hardware to run this tool. For that reason, this Honeyclient was the starting point for this multi-labeling analysis.

3.6.2 Sandbox Environment

In the previous section, the topic was the first tool of the classification pipeline, which, in our case, is Thug. The honeyclient, as mentioned, can collect essential data, which can now be used in the categorization. Therefore, in this step, the sandbox environment uses the payload files to classify the type of executables, if existing, of course.

Sandboxing uses an isolated environment to test malicious content without affecting any application or system on our device. Multiple times, software developers use these environments to test new packages and applications because it is a safe and effective technique to validate their implementation. For these reasons, for classifying an executable file is the prime approach.

The first approach we tried was FlareVM [63]. FlareVM is an open-source windows-based security distribution VM that contains multiple tools pre-installed for malware analysis. We could perform static and dynamic analyses on the payloads with this VM. However, there were two main reasons for not using this mechanism on our pipeline:

1. There were too many tools, most of them were not useful. Therefore, unnecessary installation occurred.

2. The analysis process was not automated. The use of this tool meant another pipeline embedded into this process. This would significantly increase the execution time and complicate the process unnecessarily with other open-source software that can perform the same analysis.

For that reason, from the investigation, Cuckoo Sandbox was the choice. Cuckoo can analyze any file in a matter of minutes in a safe manner. Its output can provide the classification of the type of payload, with the extra of being able to analyze URLs as well (which can be used for double-checking the URL). The other reason for this choice was that it outputs a PCAP file. This is the entry for the last tool of the process.

3.6.3 Network traffic Analysis

Network traffic can be very incriminating for a URL, especially a Phishing one, because of remote manipulation. Consequently, an analysis of the PCAP file can provide further or at least confirm information.

When talking about network traffic analysis, one name comes to mind, Wireshark. Wireshark is probably the most famous tool for this type of forensics. However, we decided to go a different way and use Zeek, an open-source software framework for this exact purpose. The reasons for this choice are simple, it was just more accessible and a cleaner way of performing the task because, important logs are already separated into different files.

3.7 Discussion

In any Project's workflow, the first step is gathering the requirements to accomplish this. Then the search for options to fulfill these is executed and the pros and cons of each are evaluated. Only after this is the pipeline developed. These were the bullet points covered in this chapter.

In this last section, we look at the overview of the entire analysis process to explain how all the phases interact with each other to get to the objective presented in the paragraph above.

Afterward, the execution of all of these tools is represented in the next chapter, along with its validation and the results.

Chapter 4

Pratical Validation

The previous chapter outlined the proposed architecture for fulfilling the requirements for the URL analysis. Here, we discuss the validation of this architecture, starting with the website's design in section 4.1. In section 4.2 we detail the execution of the process described before, step by step. Section 4.3 interprets the results produced and stored in a database. Finally, we use the data to build a report for the website user, discussed in section 4.4.

4.1 Website Design and Implementation

Chapter 3 focused on the work behind the scenes. When a consumer uses a product, most do not care how the system operates, only that it gets results. Thus, the same principle is applied here. A method for the users to submit a URL to us for analysis was essential. Consequently, a website was the decision from the beggining to provide this interaction between the two parts.

The website implementation works as a validation of the pipeline. After it was constructed, this was the technique required to apply it, validating the legitimacy of the results. For that reason, the website needed to check some requirements as well. It needed to be a simplistic site, and most important, effective.

Considering every demand stated above, the website presented in Figure 4.1 was the choice. It is a simple page that allows users to register and log in to perform the analysis. After, a text box where URLs can be submitted for analysis, and in the footer, some information about the developers.

A report is built for the user when the analysis process is complete. The process of gathering this information is the focus of the following sections. Although, after the results are compiled, the website is again the technique we use to present the report to the user.

4.2 Pipeline Execution

Chapter 3 presents an overview of the architecture for URL analysis we propose. In this section, we detail the execution, presenting the results of every step of the process.

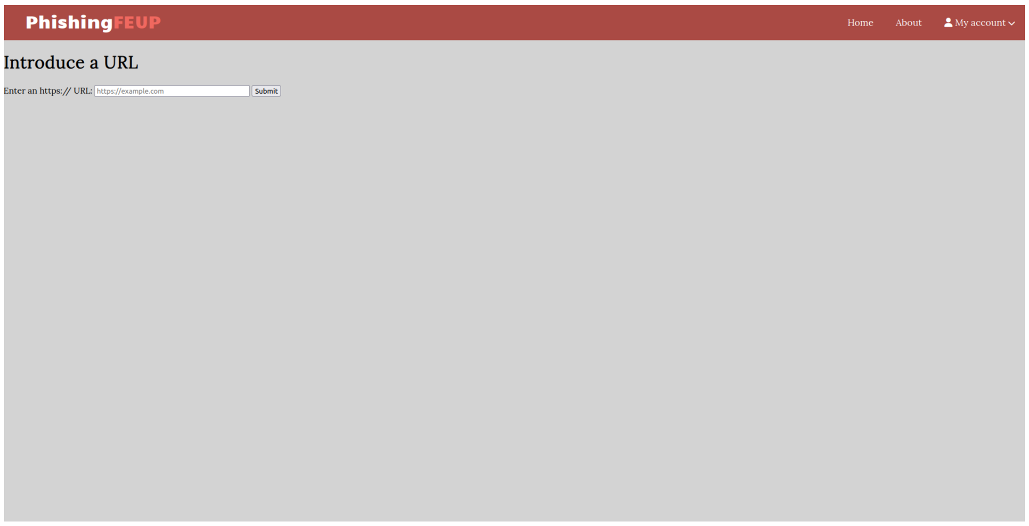
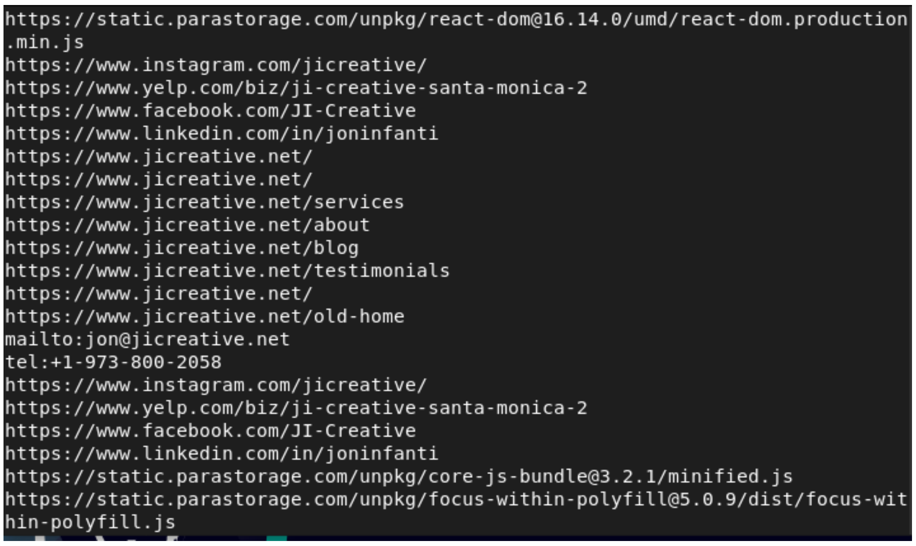


Figure 4.1: A screenshot of the analysis webpage

4.2.1 Detection of Private Information

As explained in Section 3.3, uncovering the user's personal data aborts the process. Therefore, although simple, the execution of this step can halt everything. So, the first tool is Hakrawler, and its execution is very straightforward. We just need to run the command *"echo <input URL> | docker run –rm -i hakluke/hakrawler"* and it uncovers all the hidden URLs. This is done in an attempt to discover some result linked to the initially submitted website that contains some type of personal information. For example, when we run with the `https://www.jicreative.net/`, Figure 4.2 shows a small amount of the output. As we can see, the results show some personal information.



```
https://static.parastorage.com/unpkg/react-dom@16.14.0/umd/react-dom.production
.min.js
https://www.instagram.com/jicreative/
https://www.yelp.com/biz/ji-creative-santa-monica-2
https://www.facebook.com/JI-Creative
https://www.linkedin.com/in/joninfanti
https://www.jicreative.net/
https://www.jicreative.net/
https://www.jicreative.net/services
https://www.jicreative.net/about
https://www.jicreative.net/blog
https://www.jicreative.net/testimonials
https://www.jicreative.net/
https://www.jicreative.net/old-home
mailto:jon@jicreative.net
tel:+1-973-800-2058
https://www.instagram.com/jicreative/
https://www.yelp.com/biz/ji-creative-santa-monica-2
https://www.facebook.com/JI-Creative
https://www.linkedin.com/in/joninfanti
https://static.parastorage.com/unpkg/core-js-bundle@3.2.1/minified.js
https://static.parastorage.com/unpkg/focus-within-polyfill@5.0.9/dist/focus-wit
hin-polyfill.js
```

Figure 4.2: Output of Hakrawler with the input URL of `https://www.jicreative.net/`

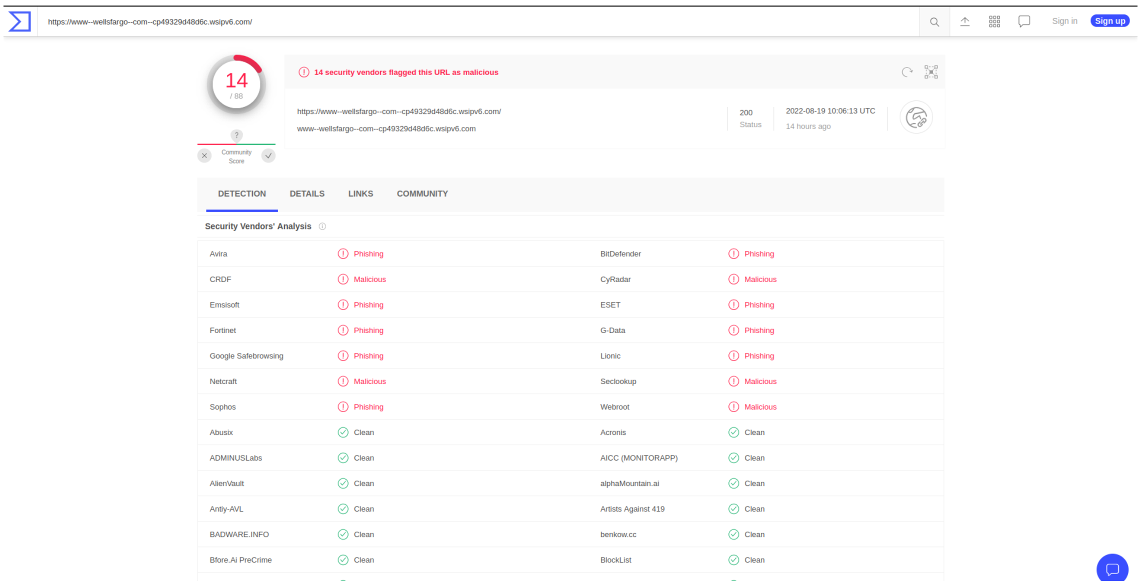
The execution of the decoding block is also very straightforward. If the URL we input is encoded, it returns the decoded correspondent, if not, it simply returns the same one. Following this, we submit the URL to the parsing script to check for personal information. The execution is simple. For example, if the input equals something like `www.mints.com/?name=something`, the code parses the domain name from the path and the query and outputs the information found. In this case, the output would be: **name: [’something’]** .

In conclusion, if no private information is found (which, considering our experience is the most common case), the execution carries on to the next section: Checking URL reputation.

4.2.2 URL Reputation Score

In section 3.4, there is a list of options from which to choose. We select two, VirusTotal and YALIH, each one with a different purpose.

First, we insert the URL in the VirusTotal website. VirusTotal inspects the item with over 70 antivirus and Blacklisting services. Using `https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com/` has an example retrieved from the OpenPhish database, we submit it to VirusTotal, and obtain the report shown in Figure 4.3.



https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com/

14 / 88

14 security vendors flagged this URL as malicious

https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com/
www--wellsfargo--com--cp49329d48d6c.wsipv6.com

200 Status

2022-08-19 10:06:13 UTC
14 hours ago

Security Vendors' Analysis

Avira	Phishing	BitDefender	Phishing
CRDF	Malicious	CyRadar	Malicious
Emsisoft	Phishing	ESET	Phishing
Fortinet	Phishing	G-Data	Phishing
Google Safebrowsing	Phishing	Lionic	Phishing
Netcraft	Malicious	Seclookup	Malicious
Sophos	Phishing	Webroot	Malicious
Abusix	Clean	Acronis	Clean
ADMINUSLabs	Clean	AICC (MONITORAPP)	Clean
AlienVault	Clean	alphaMountain.ai	Clean
Antiy-AVL	Clean	Artists Against 419	Clean
BADWARE.INFO	Clean	benkow.cc	Clean
Bfore.Ai PreCrime	Clean	BlockList	Clean

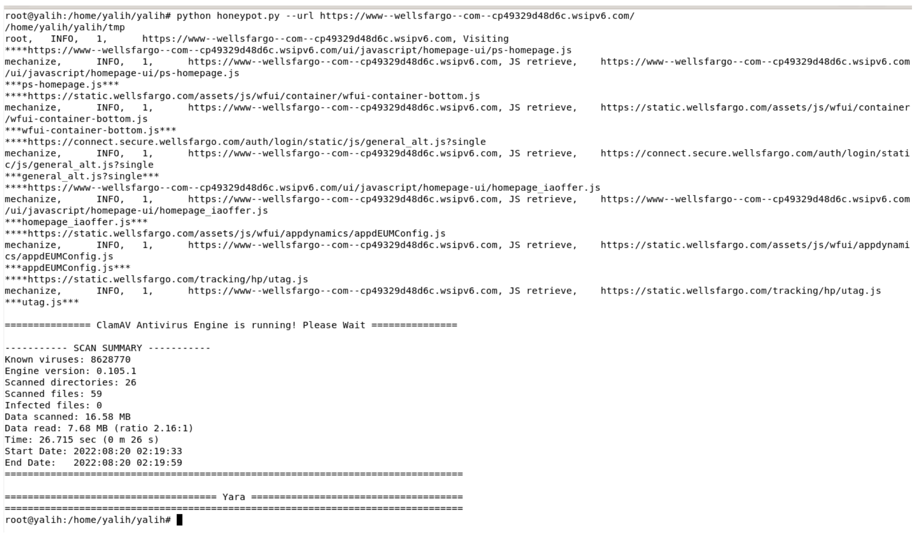
Figure 4.3: Analysis report from VirusTotal

The Figure presents a toolbar with four main sections of the report. The community section is where users can make comments. These signed users can also vote, which leads to the community score. The links tab is where a list of outgoing links is presented. Details is a report on information concerning the history of submissions, category of the link, HTTP response data, and HTML info. Finally is the detection section, the one shown on the image, and the one we retrieve information for the results of our pipeline. The antivirus and blacklists we can see here are the ones that make the reputation score. This reputation score, in this case, is 14/88, which is the entry for the results report on our pipeline.

Indicators of Compromise		
IP data	**Geographical data**	**Server info**
Domain name	Country	Creation date
IP address	State/Region	Expiration date
ISP		

Table 4.1: IoCs classification

The other tool used is YALIH, a low interaction honeyclient. This honeyclient uses static detection techniques, such as a signature detection engine (ClamAV, an antivirus) in combination with de-obfuscation of JavaScript code to improve the detection of attack signatures by the YARA rules employed. Therefore, we input the same URL as above in YALIH using the following command as root: *python honeypot.py –url* `https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com/`. The results are summarized into two parts, the ClamAV Antivirus scan and the YARA rules scan. Both scans for this link have returned with no indication of malign features or infected files, as shown in Figure 4.4.



```
root@yalih:/home/yalih/yalih# python honeypot.py --url https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com/
/home/yalih/yalih/tmp
root,   INFO,   1,      https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com, Visiting
****https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com/ui/javascript/homepage-ui/ps-homepage.js
mechanize,      INFO,   1,      https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com, JS retrieve,    https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com
/ui/javascript/homepage-ui/ps-homepage.js
***ps-homepage.js***
****https://static.wellsfargo.com/assets/js/wfui/container/wfui-container-bottom.js
mechanize,      INFO,   1,      https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com, JS retrieve,    https://static.wellsfargo.com/assets/js/wfui/container
/wfui-container-bottom.js
***wfui-container-bottom.js***
****https://connect.secure.wellsfargo.com/auth/login/static/js/general_alt.js?single
mechanize,      INFO,   1,      https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com, JS retrieve,    https://connect.secure.wellsfargo.com/auth/login/stati
c/js/general_alt.js?single
***general_alt.js?single***
****https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com/ui/javascript/homepage-ui/homepage_iaoffer.js
mechanize,      INFO,   1,      https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com, JS retrieve,    https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com
/ui/javascript/homepage-ui/homepage_iaoffer.js
***homepage_iaoffer.js***
****https://static.wellsfargo.com/assets/js/wfui/appdynamics/appdEUMConfig.js
mechanize,      INFO,   1,      https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com, JS retrieve,    https://static.wellsfargo.com/assets/js/wfui/appdynami
cs/appdEUMConfig.js
***appdEUMConfig.js***
****https://static.wellsfargo.com/tracking/hp/utag.js
mechanize,      INFO,   1,      https://www--wellsfargo--com--cp49329d48d6c.wsipv6.com, JS retrieve,    https://static.wellsfargo.com/tracking/hp/utag.js
***utag.js***

=============== ClamAV Antivirus Engine is running! Please Wait ===============

----------- SCAN SUMMARY -----------
Known viruses: 8628770
Engine version: 0.105.1
Scanned directories: 26
Scanned files: 59
Infected files: 0
Data scanned: 16.58 MB
Data read: 7.68 MB (ratio 2.16:1)
Time: 26.715 sec (0 m 26 s)
Start Date: 2022:08:20 02:19:33
End Date:   2022:08:20 02:19:59
================================================================================

===================================== Yara =====================================
================================================================================
root@yalih:/home/yalih/yalih# █
```

Figure 4.4: YALIH execution summary

4.2.3 Retrieve IoCs information

In this section, as the name suggests, the Indicators of Compromise result from the execution. These are addressed on Table 4.1.

The first step in this phase is checking if the URL is shortened. A URL expander will obtain the effective link if this is the case. However, the option we use in our pipeline is retrieving this from the analysis we perform in the next phase, but another option can be using available online tools, such as [64] and [65]. If it is not shortened, we can proceed with the execution.

For the second part, packages like *whois*, *traceroute*, and *dnsutils* (contains the commands *dig*, *nslookup* and *host*). Another tool, just to double check is the website located on [61]. Using `https://clck.ru/tAPtV` we obtain this report:

- **Effective URL**: `https://dilscordisgix.com/lt`
- **Domain name**: DILSCORDISGIX.COM
- **IP address**: 172.67.161.146/104.21.65.99
- **ISP**: CLOUDFLARENET
- **Country**: NOT FOUND
- **State/Region**: NOT FOUND
- **Creation date**: 2022-08-20
- **Expiration date**: 2023-08-20

These can be warnings on the classification of the link. For example, looking at a domain that has been up for some time (looking at the creation and expiration date), we conclude that this is probably a benign URL, since malicious domains usually do not last very much.

4.2.4 Dynamic Analysis

This stage of the pipeline is the most important. The crucial requirement for this work is the classification and categorization of the type of attack it performs if any. Therefore, a dynamic analysis of the website is the case here. It is partitioned into three parts: the execution of cuckoo sandbox, of Thug, a low interaction honeyclient, and finally, the running of Zeek, the network traffic analysis software.

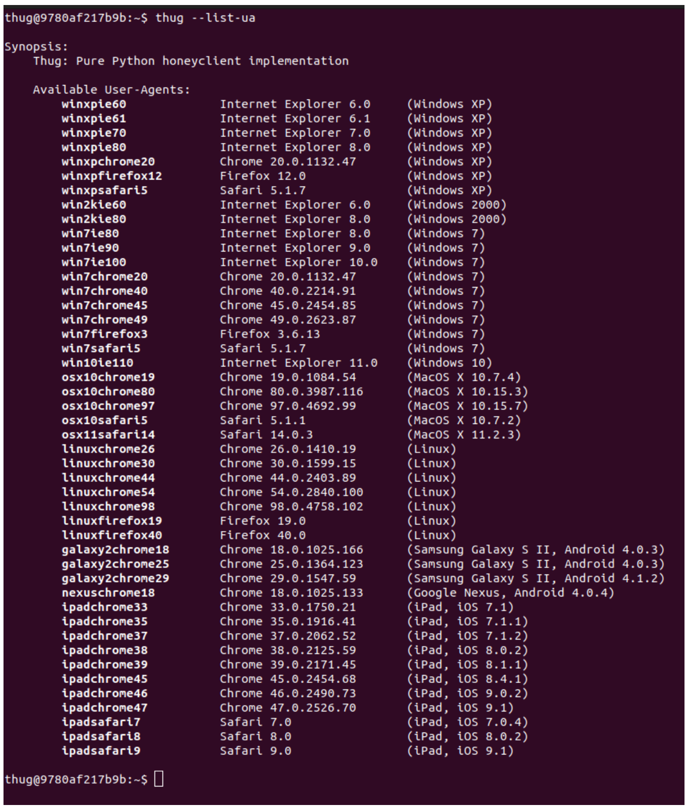
4.2.5 Thug Execution

First is the execution of Thug. Thug simulates a user agent to retrieve as much information from the URL execution as possible. Primarily, we must select the user agent from the list shown in Figure 4.5.

In most cases, the selection is between this three options: Windows 7 using Firefox (win7firefox3), Windows 7 using Chrome 49.0 (win7chrome49), and Linux using Firefox 40.0 (linuxfirefox40). If some error appears during the execution, other user agents are considered, especially the ones simulating Android and iOS, because the URL can target only mobile devices.

The next step before the execution is to consider the flags to use. We have narrowed it down to some specific to detecting the type of attack:

- **File logging**: -F



```
thug@9780af217b9b:~$ thug --list-ua

Synopsis:
    Thug: Pure Python honeyclient implementation

    Available User-Agents:
        winxpie60             Internet Explorer 6.0     (Windows XP)
        winxpie61             Internet Explorer 6.1     (Windows XP)
        winxpie70             Internet Explorer 7.0     (Windows XP)
        winxpie80             Internet Explorer 8.0     (Windows XP)
        winxpchrome20         Chrome 20.0.1132.47       (Windows XP)
        winxpfirefox12        Firefox 12.0              (Windows XP)
        winxpsafari5          Safari 5.1.7              (Windows XP)
        win2kie60             Internet Explorer 6.0     (Windows 2000)
        win2kie80             Internet Explorer 8.0     (Windows 2000)
        win7ie80              Internet Explorer 8.0     (Windows 7)
        win7ie90              Internet Explorer 9.0     (Windows 7)
        win7ie100             Internet Explorer 10.0    (Windows 7)
        win7chrome20          Chrome 20.0.1132.47       (Windows 7)
        win7chrome40          Chrome 40.0.2214.91       (Windows 7)
        win7chrome45          Chrome 45.0.2454.85       (Windows 7)
        win7chrome49          Chrome 49.0.2623.87       (Windows 7)
        win7firefox3          Firefox 3.6.13            (Windows 7)
        win7safari5           Safari 5.1.7              (Windows 7)
        win10ie110            Internet Explorer 11.0    (Windows 10)
        osx10chrome19         Chrome 19.0.1084.54       (MacOS X 10.7.4)
        osx10chrome80         Chrome 80.0.3987.116      (MacOS X 10.15.3)
        osx10chrome97         Chrome 97.0.4692.99       (MacOS X 10.15.7)
        osx10safari5          Safari 5.1.1              (MacOS X 10.7.2)
        osx11safari14         Safari 14.0.3             (MacOS X 11.2.3)
        linuxchrome26         Chrome 26.0.1410.19       (Linux)
        linuxchrome30         Chrome 30.0.1599.15       (Linux)
        linuxchrome44         Chrome 44.0.2403.89       (Linux)
        linuxchrome54         Chrome 54.0.2840.100      (Linux)
        linuxchrome98         Chrome 98.0.4758.102      (Linux)
        linuxfirefox19        Firefox 19.0              (Linux)
        linuxfirefox40        Firefox 40.0              (Linux)
        galaxy2chrome18       Chrome 18.0.1025.166      (Samsung Galaxy S II, Android 4.0.3)
        galaxy2chrome25       Chrome 25.0.1364.123      (Samsung Galaxy S II, Android 4.0.3)
        galaxy2chrome29       Chrome 29.0.1547.59       (Samsung Galaxy S II, Android 4.1.2)
        nexuschrome18         Chrome 18.0.1025.133      (Google Nexus, Android 4.0.4)
        ipadchrome33          Chrome 33.0.1750.21       (iPad, iOS 7.1)
        ipadchrome35          Chrome 35.0.1916.41       (iPad, iOS 7.1.1)
        ipadchrome37          Chrome 37.0.2062.52       (iPad, iOS 7.1.2)
        ipadchrome38          Chrome 38.0.2125.59       (iPad, iOS 8.0.2)
        ipadchrome39          Chrome 39.0.2171.45       (iPad, iOS 8.1.1)
        ipadchrome45          Chrome 45.0.2454.68       (iPad, iOS 8.4.1)
        ipadchrome46          Chrome 46.0.2490.73       (iPad, iOS 9.0.2)
        ipadchrome47          Chrome 47.0.2526.70       (iPad, iOS 9.1)
        ipadsafari7           Safari 7.0                (iPad, iOS 7.0.4)
        ipadsafari8           Safari 8.0                (iPad, iOS 8.0.2)
        ipadsafari9           Safari 9.0                (iPad, iOS 9.1)

thug@9780af217b9b:~$
```

Figure 4.5: List of user agents Thug emulates

- **Features logging**: -W
- **Screenshot capturing**: -f
- **Image processing analysis**: -a
- **Maximum pages to fetch**: If the analysis is stuck on a loop, we use -t
- **User agent selection**: -u or –useragent=
- **ElasticSearch logging**: -G
- **JSON logging**: -Z
- **Specify the address and port of the MongoDB instance**: -D, format is host:port

These parameters were carefully selected between the many options thug can provide of results. The options are listed next:

- Payload Files
- Other content Files, like screenshots and page content code
- Visited URLs

- MongoDB output
- ElasticSearch output
- HPFeeds
- Native Report Format

Thug can be considered a "wolf in sheep's clothing". This system emulates an actual browser with all its plugins to trigger the URL to act maliciously. The same can, and will be, executed on a sandbox environment (Section 4.2.6), but some malwares detect this type of tools and does not behave as they should, therefore, not being flagged. For that reason, a honeypot can trigger this behavior and register in the form of outputs the actions.

Executing Thug is a simple command on the terminal. Running the command: *thug <flags> <url>*, will run the analysis. Our focus in this step is on if any Payload Files exists, this is, if the honeypot triggered any dropped files in the execution of the URL. Then, 2 outcomes can come of this:

- First, there are no payload files in which the option of the site being a drive-by is eliminated. Then we examine the rest of the outputs, like the screenshots and the HTML and application files, to check for any irregularities and advance on to the next phase.
- Second, there are payload files. These files are passed on to Cuckoo for a better analysis.

4.2.6 Cuckoo Sandbox

As is visible from the diagram in section 3.2, from the analysis of Thug, the executables are retrieved and passed on to cuckoo for analysis. Not only that, but cuckoo also has the feature to submit a URL for analysis, which we also execute as a security measure.

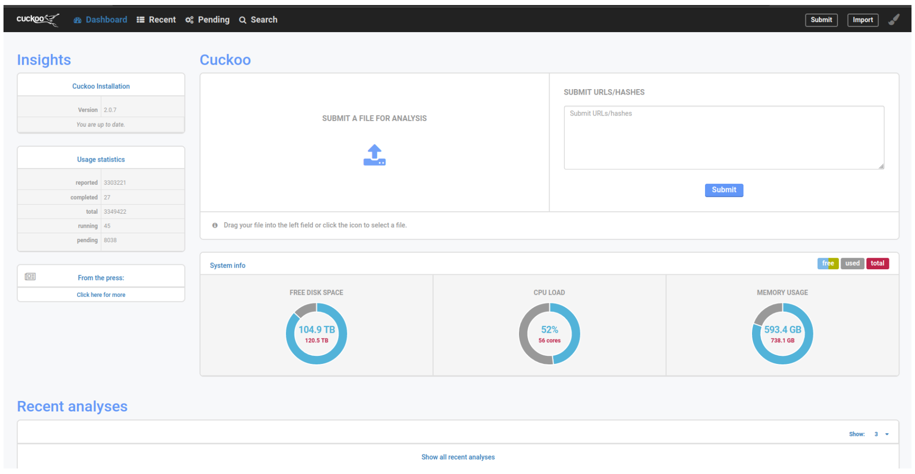
Cuckoo Sandbox GUI can be seen in Figure 4.6. The dashboard tab is divided into four parts: submission section, system info, insights on the tool, and recent analysis.

When we submit a URL or a file on cuckoo, the report gives us a score for if the link is malicious or not. Other outputs cuckoo provides: the Static analysis, the dropped files, HTML report, JSON report, PCAP file, the memory image and behavioral analysis. The main page of the report can be seen on Figure 4.7.

After Thug, Cuckoo analyzes or an executable file downloaded by the link or the link itself.Either way, the classification it attributes in the execution is then registered in the Database for the final report.

As mentioned before, Cuckoo has a very good reporting method, presenting every detail separately in a very simple and clean interface. So, besides the classification it provides, we are giving special attention to:

- The Signatures of the file or URL. It can present some very straightforward and important characteristics.



Version	2.0.7

You are up to date.

reported	3303221
completed	27
total	3349422
running	45
pending	8038

Drag your file into the left field or click the icon to select a file.

Figure 4.6: Cuckoo Sandbox GUI

- The Behavioral Analysis tab.
- The Network Analysis tab. It contains the PCAP file to be uploaded to Zeek.
- The VM Memory Dump.

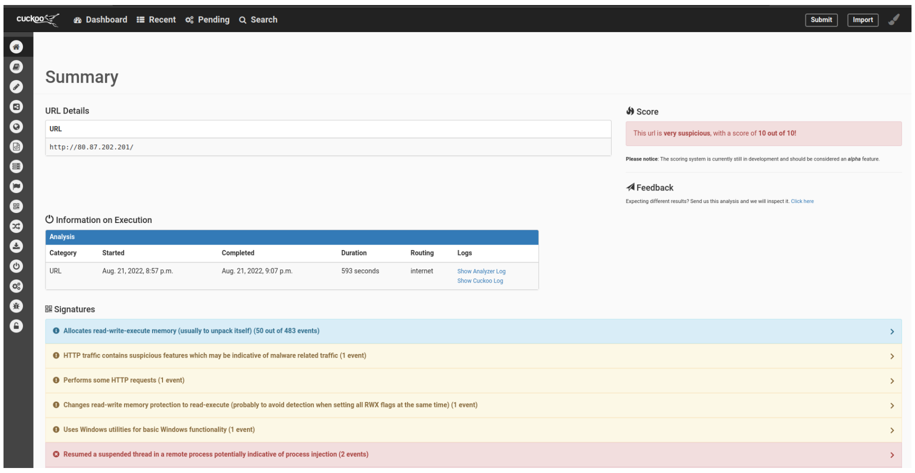
In this dynamic analysis phase, a decision on how to categorize a URL can be taken. However, we will get into a more detailed explanation of this result interpretation in Section 4.3.

4.2.7 Zeek Network Traffic Analyzer

The last analyzer we use is the network traffic manager, Zeek, which in this case we use has an IDS (Intrusion Detection System). The previous URL analysis downloaded a PCAP file containing all the traffic from the link execution. Now, this PCAP is uploaded to Zeek for analysis using the command *zeek -r <path to the PCAP file>*, where the flag *-r* is used to read the file.

Zeek is chosen because it can divide the PCAP file into multiple ones that contain specific logs for each package, such as:

- **conn.log**: It contains
- **dns.log**: File for every DNS request;
- **files.log**: It is a list of files;
- **http.log**: It has every http transaction;
- **pe.log**
- **ssl.log**
- **x509.log**



Summary

URL Details

URL

http://80.87.202.201/

Score

This url is **very suspicious**, with a score of **10 out of 10!**

Please notice: The scoring system is currently still in development and should be considered an *alpha* feature.

Feedback

Expecting different results? Send us this analysis and we will inspect it. Click here

Information on Execution

Analysis

Category	Started	Completed	Duration	Routing	Logs
URL	Aug. 21, 2022, 8:57 p.m.	Aug. 21, 2022, 9:07 p.m.	593 seconds	internet	Show Analyzer Log Show Cuckoo Log

Signatures

- Allocates read-write-execute memory (usually to unpack itself) (50 out of 483 events)
- HTTP traffic contains suspicious features which may be indicative of malware related traffic (1 event)
- Performs some HTTP requests (1 event)
- Changes read-write memory protection to read-execute (probably to avoid detection when setting all RWX flags at the same time) (1 event)
- Uses Windows utilities for basic Windows functionality (1 event)
- Resumed a suspended thread in a remote process potentially indicative of process injection (2 events)

Figure 4.7: Cuckoo Sandbox analysis report

These log files are an example. Zeek can format a pcap into different types of files depending on the types of packets it encounters. For example, if there is a **pe.log** it means someone download an executable, which is a strong indicator that this is a malicious link.

This component has more of a forensic side to it. We need to check each log looking for IoCs, organize and compare the different files in order to make sense of what is happening. Here we try to locate log files with packets that Zeek does not recognize the protocol, for example, or has mentioned before, check for a pe.log that contains information about the download of an executable. Then, we can, for instance, try to correlate the uid of these packets in order files to understand it better using the command **grep**. After gathering some information, together with the interpretation taken from the rest of the tools, a final decision on how to categorize the URL can be made.

With this part over, the execution of the pipeline has ended and we must now have the results in the database. The following section will present how the dynamic analysis results are interpreted so we can get the results we pretend.

4.3 Result Interpretation

All of section 4.2 portray a picture of the execution of every tool. The first phases of the analysis are straightforward to understand how we get the results stored in the database. However, the dynamic analysis can be more complex and further explanation is needed. That is the aim of this section, where some examples will be considered to better comprehend the interpretation of the results.

First, the results of Thug. This results are stored in a folder of our choice, and divided like:

- **Application**: where the captured javascript files are stored.
- **Image**: here the images from the webpage is stored, like screenshots, logos, among others.

- **Text**: the HTML and css files are stored here.
- **Unkown**: It can contain important files that are not recognized by Thug, but still stored for analysis.

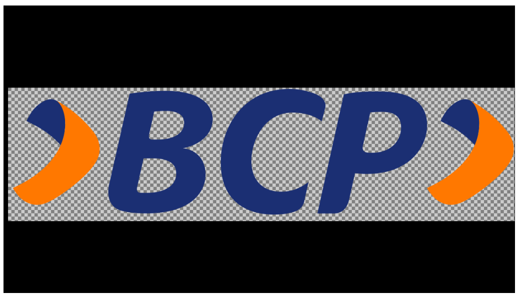
The second step is the analysis utilizing Cuckoo Sandbox. Here a more straightforward interpretation of the results is made since the tool uses a graphical interface for reporting (Figure 4.7).

The third and last step is the analysis of the PCAP files. Here is where a deeper explanation will take place, for it introduces more manual labor on our part. For the interpretation, we will take a look at 2 cases:

- **Case 1**: The interpretation of the analysis on a Phishing URL.
- **Case 2**: The interpretation of the analysis on a Drive-by Download URL.

4.3.1 Case 1: Phishing URL

We start by retrieving the URL (`http://jardimdosavos.com/`) from a popular Database available online OpenPhish [20]. This URL is inputted in Thug and the results are divided into three folders: analysis, image and text. The first conclusion is that the link is not a Drive-by Download, since it did not trigger any executables download. Next, we examine the image folder and are presented with Figure 4.8b and Figure 4.8a.



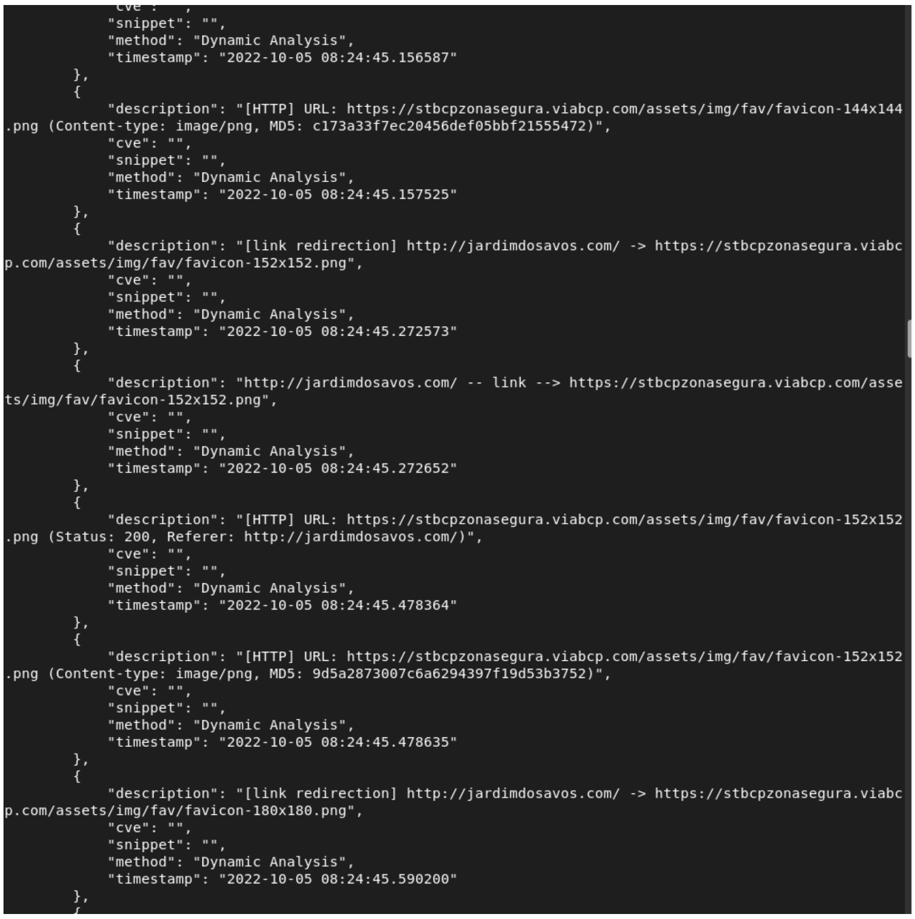
(a) Image in the *svg+xml* folder

(b) Image in the *png* folder

Figure 4.8: Images captured by Thug

Looking at these images, we can suppose this can be some business website. After a search on Google, we can see that the logo belongs to the **Banco de Crédito del Perú**. If we consider the domain name (*jardimdosavos*), there seems to be something wrong. They do not match. So, after another search, we encounter the legitimate website: `https://www.viabcp.com/`. This is the first indication that we can be dealing with a possible Phishing URL that tries to steal users information.

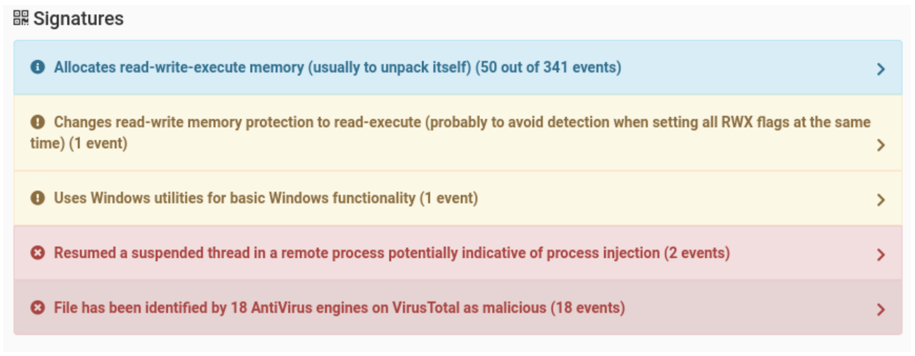
Next, we look into the file with the JSON analysis and discover some link redirections to the legitimate website (Figure 4.9), which can be considered another IoC. Taking a look at the HTML, javascript and CSS files can also raise some red flags (like some grammatical errors in the HTML file). In this case, we find no indicator, especially since the CSS files are retrieved from the legitimate URL to make the site the most graphically synonymous with the latter.



```
            "snippet": "",
            "method": "Dynamic Analysis",
            "timestamp": "2022-10-05 08:24:45.156587"
        },
        {
            "description": "[HTTP] URL: https://stbcpzonasegura.viabcp.com/assets/img/fav/favicon-144x144
.png (Content-type: image/png, MD5: c173a33f7ec20456def05bbf21555472)",
            "cve": "",
            "snippet": "",
            "method": "Dynamic Analysis",
            "timestamp": "2022-10-05 08:24:45.157525"
        },
        {
            "description": "[link redirection] http://jardimdosavos.com/ -> https://stbcpzonasegura.viabc
p.com/assets/img/fav/favicon-152x152.png",
            "cve": "",
            "snippet": "",
            "method": "Dynamic Analysis",
            "timestamp": "2022-10-05 08:24:45.272573"
        },
        {
            "description": "http://jardimdosavos.com/ -- link --> https://stbcpzonasegura.viabcp.com/asse
ts/img/fav/favicon-152x152.png",
            "cve": "",
            "snippet": "",
            "method": "Dynamic Analysis",
            "timestamp": "2022-10-05 08:24:45.272652"
        },
        {
            "description": "[HTTP] URL: https://stbcpzonasegura.viabcp.com/assets/img/fav/favicon-152x152
.png (Status: 200, Referer: http://jardimdosavos.com/)",
            "cve": "",
            "snippet": "",
            "method": "Dynamic Analysis",
            "timestamp": "2022-10-05 08:24:45.478364"
        },
        {
            "description": "[HTTP] URL: https://stbcpzonasegura.viabcp.com/assets/img/fav/favicon-152x152
.png (Content-type: image/png, MD5: 9d5a2873007c6a6294397f19d53b3752)",
            "cve": "",
            "snippet": "",
            "method": "Dynamic Analysis",
            "timestamp": "2022-10-05 08:24:45.478635"
        },
        {
            "description": "[link redirection] http://jardimdosavos.com/ -> https://stbcpzonasegura.viabc
p.com/assets/img/fav/favicon-180x180.png",
            "cve": "",
            "snippet": "",
            "method": "Dynamic Analysis",
            "timestamp": "2022-10-05 08:24:45.590200"
        },
```

Figure 4.9: JSON analysis file

The next step is the Cuckoo Sandbox analysis. Here are the definitive details that classify the URL as Malicious. The first is the score: **9.7 out of 10**. So, now that the URL is classified, we start looking for other indicators that can support our hypothesis of this being a Phishing URL. Examining the signatures, we can see that cuckoo sinalyzes a suspended thread in a remote process was resumed. It considers this an indicative of process injection (Figure 4.10). Figure 4.11a and Figure 4.11b represent the comparation between the homepage of the legitimate and malicious URL's.As mentioned in Section 4.2.6 some other sections can be inspected more closely, but in this case we are advancing to the PCAP file analysis for other indicators that can support the previously referred hypothesis.



Signatures

- Allocates read-write-execute memory (usually to unpack itself) (50 out of 341 events)
- Changes read-write memory protection to read-execute (probably to avoid detection when setting all RWX flags at the same time) (1 event)
- Uses Windows utilities for basic Windows functionality (1 event)
- Resumed a suspended thread in a remote process potentially indicative of process injection (2 events)
- File has been identified by 18 AntiVirus engines on VirusTotal as malicious (18 events)

Figure 4.10: Signatures of the Cuckoo analysis.

After all these indicators, this final step can be considered just a precaution in this case. So, after executing zeek to read the PCAP file, we will focus on the following log files: conn.log, http.log and files.log. First, we examine the connections log. At first glance, no abnormality stands out until we reach the end of the file and see two unrecognized services (Figures 4.12 and 4.13). We can further explored using tools like Wireshark, but we will focus on the download of a file that is present both on the http and files log files (Figure 4.14). A compressed file is downloaded using the GET method from `www.download.windowsupdate.com`. Considering all the evidence gathered before, we can safely categorize this URL as Phishing.

4.3.2 Case 2: Drive-by Download URL

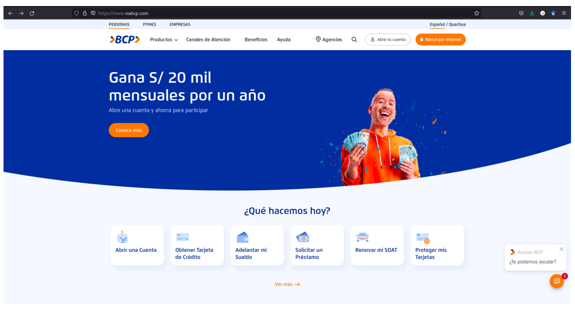
Now basing ourselves on the URL: `http://107.172.206.118/oi/gud.exe`, retrieved from [66] we follow the same steps as before.

We start the analysis by running Thug to check if it triggers any malicious behavior. So, after the execution, the results are two folders: analysis and application. In this analysis, what stands out is the PE file that is downloaded after retrieving the webpage. This file is stored in the application folder. So, Thug does trigger the download of an executable that, in order case, would be downloaded without the user's permission. This is an indication of malicious behavior, so to determine that, we submit the file to Cuckoo for analysis.

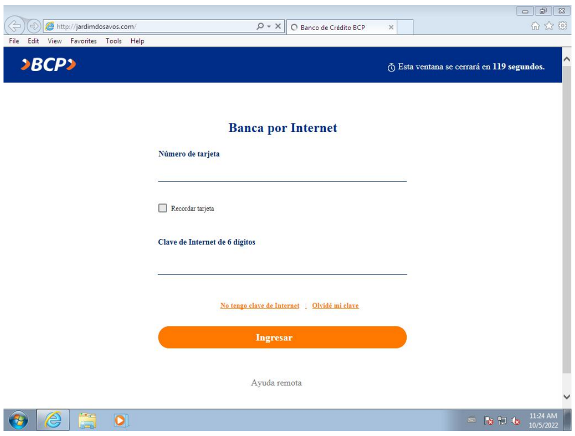
The results of the Cuckoo analysis are very incriminating. The classification of the file is 10. By looking at the PE file's signatures (Figure 4.15), we can see that this is malware and that the URL can be categorized as a Drive-by Download type.

Another interesting part of the Cuckoo results is the dropped buffers. Here we can see the file is running the following:

- network_smtp_dotNet - Communications smtp
- keylogger - Run a keylogger
- win_hook - Affect hook table



(a) Homepage of the legitimate URL


http://jardimdosavos.com/
Banco de Crédito BCP
File Edit View Favorites Tools Help
BCP
Esta ventana se cerrará en 119 segundos.
Banca por Internet
Número de tarjeta
Recordar tarjeta
Clave de Internet de 6 dígitos
No tengo clave de Internet
Olvidé mi clave
Ingresar
Ayuda remota
11:24 AM
10/5/2022

(b) Homepage of the malicious URL

Therefore, after all these conclusions, there is no need to execute the PCAP analysis to determine the categorization of the file. In conclusion, the URL is classified as malicious and categorized as a Drive-by Download.

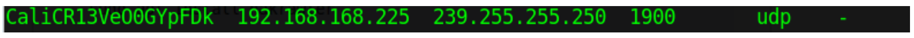
4.4 Reporting to the user

Executing the entire pipeline leads to a set of entries on the website's database that need to be explained to the user. So, the last part of the practical validation is the report we build for the user to understand what is the intent of the URL.

The website presents the following information to the user;

- **Private Information Detection**
- **Reputation Score of the URL**
- **Indicators of Compromise (IoC's)**
- **Classification and Categorization of the URL**

Now, we report this information, but in a certain way. The private information detection variable in the database is a boolean. It can only be 0 or 1 (True or False). Therefore, the report can present two phrases to the user: *There was no private information detected during the analysis* or *There was private information detected during the analysis*.



```
CaliCR13VeO0GYpFDk  192.168.168.225  239.255.255.250  1900        udp    -
```

Figure 4.12: First packet with missing service type

The reputation score of the link has two columns in the database: the reputation from YALIH and VirusTotal. So, this part will have two results printed on the report. One boolean, the reputation score from YALIH, which means the URL was detected as malicious or not. The other is numerical and uses a scale of **0 to 89** (the number of antivirus and databases VirusTotal uses for detection). For example, `http://capensis.online/i.exe` has a score of **11/89** and the `http://roblox.com.kz/users/190556601/profile` a score of **14/89** (these links were retrieved from URLhaus [66] and OpenPhish [20], respectively).

The third section of the report contains the ICOs presented in table 4.1.Thus, the report will have this five text information and two dates printed. For example, looking at the database entries for `http://roblox.com.kz/users/190556601/profile`, we have the following information reported to the user:

- **Domain Name:** roblox.com.kz
- **IP address:** 172.67.217.8
- **ISP:** CloudFlare
- **Country:** Kazakhstan Republic
- **State/Region:** Riyadh
- **Creation Date:** 2022-08-16
- **Expiration Date:** NULL

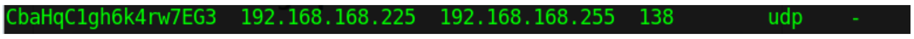
Finally, the last and most essential information about this analysis is the URL's classification and categorization. As mentioned in this last section 4.3, the classification is a numerical attribute with a scale from **0 to 10**. As for the categorization, it is a text attribute of the database and can have three possible outcomes: *Benign*, *Phishing URL* or *Drive-By Download URL.*

4.5 Discussion

This chapter focused on the more practical mode of the dissertation. Here, we explained how the end goal of having a site directed to a more technical knowledge community is achieved.

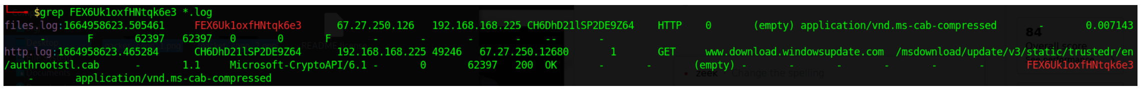
The pipeline execution description is presented, with the role of each tool explained. Here, we give more importance to the dynamic analysis part, because it is the more complex step.

To clarify how the classification and categorization of the URL are done, there is a result interpretation section in this chapter. To culminate the work, the report is built from the information stored in the database along the execution of the pipeline.



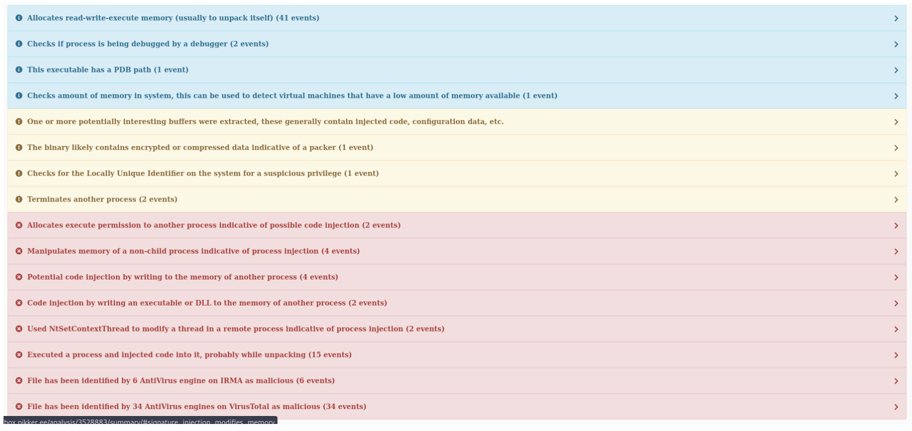
```
CbaHqC1gh6k4rw7EG3  192.168.168.225  192.168.168.255  138        udp    -
```

Figure 4.13: First packet with missing service type



```
└──╼ $grep FEX6Uk1oxfHNtqk6e3 *.log
files.log:1664958623.505461     FEX6Uk1oxfHNtqk6e3      67.27.250.126   192.168.168.225 CH6DhD21lSP2DE9Z64    HTTP    0       (empty) application/vnd.ms-cab-compressed       -       0.007143
        -       F       62397   62397   0       0       F       -       -       -       -       --      -
http.log:1664958623.465284      CH6DhD21lSP2DE9Z64      192.168.168.225 49246   67.27.250.12680       1       GET     www.download.windowsupdate.com  /msdownload/update/v3/static/trustedr/en
/authrootstl.cab        -       1.1     Microsoft-CryptoAPI/6.1 -       0       62397   200  OK     -       -       (empty) -       -       -       -       -       -       FEX6Uk1oxfHNtqk6e3
  -     application/vnd.ms-cab-compressed
```

Figure 4.14: Packets that describe the Download of the compressed file



Allocates read-write-execute memory (usually to unpack itself) (41 events)
Checks if process is being debugged by a debugger (2 events)
This executable has a PDB path (1 event)
Checks amount of memory in system, this can be used to detect virtual machines that have a low amount of memory available (1 event)
One or more potentially interesting buffers were extracted, these generally contain injected code, configuration data, etc.
The binary likely contains encrypted or compressed data indicative of a packer (1 event)
Checks for the Locally Unique Identifier on the system for a suspicious privilege (1 event)
Terminates another process (2 events)
Allocates execute permission to another process indicative of possible code injection (2 events)
Manipulates memory of a non-child process indicative of process injection (4 events)
Potential code injection by writing to the memory of another process (4 events)
Code injection by writing an executable or DLL to the memory of another process (2 events)
Used NtSetContextThread to modify a thread in a remote process indicative of process injection (2 events)
Executed a process and injected code into it, probably while unpacking (15 events)
File has been identified by 6 AntiVirus engine on IRMA as malicious (6 events)
File has been identified by 34 AntiVirus engines on VirusTotal as malicious (34 events)
box.pikker.ee/analysis/3528883/summary/#signature_injection_modifies_memory

Figure 4.15: Signatures of the Executable

Chapter 5

Discussion and Conclusion

Having presented a guideline for malicious URL analysis, our proposed method for analysis, and the execution and results, we now draw conclusions regarding the work performed (Section 5.2), some difficulties(Section 5.1), and points that can be explored past this point (Section 5.3).

5.1 Difficulties encountered

This thesis's fundamental goals can be divided into three main parts: The overall pipeline, the classification and categorization process, and the website implementation. Consequently, some difficulties were encountered in each respective area.

The overall pipeline was a difficult decision to make. There are plenty of approaches for different types of goals in the phishing and malware area of studies. The goal was to design a concept others could adopt in future works. Therefore, it was necessary that the process division was based on principles and not on tools or results. It took some time to find the perfect number of steps and what could be applied in each, but the final result was achieved, as we can see in chapter 3.

The second difficulty was how to design the website and implement it together with the rest of the work. The website design was a secondary goal, and some designs and techniques to make it available only were considered but, in the end, discarded. A simple graphical interface was the choice.

Lastly, the big problem and, consequently, the most critical part of this thesis was the classification and categorization process. Many tools were considered for this process, and many backs and forth situations of the chosen. The first choice was cuckoo, but it only classified the URL alone. Therefore, we changed the tool to FlareVM, then Maltego, and finally, we returned to cuckoo and paired it with some other packages to categorize. Other problems were the installation. Some tools had problems with the python version, and others were not connecting to the honeypot device (Thug). Thus, the desired results were obtained in the end, and all the requirements were satisfied.

5.2 Conclusions

The main goal of this dissertation was developing a website where students can submit a URL for malicious intent analysis and receive a detailed report they can trust.

We first present the requirements for the process. We then describe the phases for accomplishing these. A lot of particular ways of examining web pages were investigated. This brought a sense of how the URL work, the attacker side operations, and the defender side tasks for uncovering this.

The execution of the process is then exhibited, how each tool works, and the result they provide. The last step was mainly explained due to its complexity and the variety of results it can output. Here a sense of our analysis tools operate was collected.

Finally, the results are gathered and examined. Then, after trimming, they are inputted into the website's database. For the last stage, the front-end developer part was combined with the knowledge of the URL analysis, and the data was retrieved from the database, the report was built, and delivered to the user.

5.3 Future work

Since some of the objectives of this thesis were not fully accomplished, the top priority for future tasks would be the see this fulfilled.

One thing that task that can be performed in the future is limiting the website access to students of FEUP only. As mentioned, the goal of this tool was to be available to the faculty staff only so that they could have a reliable method of detecting a malicious link at any time. However, there was no implementation on how to prevent other users from joining.

One other modification that could be performed in the future is the automation of the entire process. This work was solely focused on describing a process for retrieving and analyzing links. However, now that this goal has been accomplished, the automatization of the entire process could be done to decrease the response time of the website. Pipeline automation has limitations since a lot of the project involves human interaction with the dynamic analysis results. Therefore, using some rule-based system can be helpful in this option.

References

[1] Francois Mouton, Mercia M Malan, Louise Leenen, and Hein S Venter. Social engineering attack framework. In *2014 Information Security for South Africa*, pages 1–9. IEEE, 2014.

[2] Koceilah Rekouche. Early phishing. *CoRR*, abs/1106.4692, 2011. URL: http://arxiv.org/abs/1106.4692, arXiv:1106.4692.

[3] Fırat Dalgıç, Ahmet Bozkir, and Murat Aydos. Phish-iris: A new approach for vision based brand prediction of phishing web pages via compact visual descriptors. 10 2018. doi:10.1109/ISMSIT.2018.8567299.

[4] Rohit Shukla and Maninder Singh. Pythonhoneymonkey: Detecting malicious web urls on client side honeypot systems. In *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization*, pages 1–5. IEEE, 2014.

[5] Masood Mansoori, Ian Welch, and Qiang Fu. Yalih, yet another low interaction honeyclient. In *Proceedings of the Twelfth Australasian Information Security Conference-Volume 149*, pages 7–15, 2014.

[6] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Identifying suspicious urls: An application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 681–688, New York, NY, USA, 2009. Association for Computing Machinery. URL: https://doi.org/10.1145/1553374.1553462, doi:10.1145/1553374.1553462.

[7] What is phishing?, Aug 2022. URL: https://nordvpn.com/pt/blog/what-is-phishing/.

[8] Published by Statista Research Department and Jul 7. Phishing: Most targeted industries 2022, Jul 2022. URL: https://www.statista.com/statistics/266161/websites-most-affected-by-phishing/.

[9] Paypal phishing email scam: How to spot it, Aug 2022. URL: https://nordvpn.com/pt/blog/beware-of-this-paypal-phishing-scam/.

[10] Google docs phishing scam: What you need to know, Aug 2022. URL: https://nordvpn.com/pt/blog/what-you-need-to-know-about-the-new-google-docs-scam/.

[11] The worst amazon scams and how to avoid them, Aug 2022. URL: https://nordvpn.com/pt/blog/amazon-scams/.

[12] Fatima Salahdine and Naima Kaabouch. Social engineering attacks: A survey. *Future Internet*, 11(4), 2019. URL: https://www.mdpi.com/1999-5903/11/4/89, doi:10.3390/fi11040089.

[13] Surbhi Gupta, Abhishek Singhal, and Akanksha Kapoor. A literature survey on social engineering attacks: Phishing attack. In *2016 International Conference on Computing, Communication and Automation (ICCCA)*, pages 537–540, 2016. doi:10.1109/CCAA.2016.7813778.

[14] Steven Rees-Pullman. Is credential stuffing the new phishing? *Computer Fraud Security*, 2020(7):16–19, 2020. URL: https://www.sciencedirect.com/science/article/pii/S1361372320300762, doi:https://doi.org/10.1016/S1361-3723(20)30076-2.

[15] What is phishing?, Aug 2022. URL: https://nordvpn.com/blog/what-is-phishing/.

[16] Weili Chen, Xiongfeng Guo, Zhiguang Chen, Zibin Zheng, and Yutong Lu. Phishing scam detection on ethereum: Towards financial security for blockchain ecosystem. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4506–4512. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Special Track on AI in FinTech.

[17] Simon Bell and Peter Komisarczuk. An analysis of phishing blacklists: Google safe browsing, openphish, and phishtank. In *Proceedings of the Australasian Computer Science Week Multiconference*, ACSW '20, New York, NY, USA, 2020. Association for Computing Machinery. URL: https://doi.org/10.1145/3373017.3373020, doi:10.1145/3373017.3373020.

[18] Join the fight against phishing. URL: https://www.phishtank.com/.

[19] URL: https://safebrowsing.google.com/.

[20] Phishing intelligence. URL: https://openphish.com/.

[21] Martin Husák and Jakub Cegan. Phigaro: Automatic phishing detection and incident response framework. In *2014 Ninth International Conference on Availability, Reliability and Security*, pages 295–302, 2014. doi:10.1109/ARES.2014.46.

[22] Manuel Sánchez-Paniagua, Eduardo Fidalgo, Enrique Alegre, and Rocío Alaiz-Rodríguez. Phishing websites detection using a novel multipurpose dataset and web technologies features. *Expert Systems with Applications*, 207:118010, 2022. URL: https://www.sciencedirect.com/science/article/pii/S0957417422012301, doi:https://doi.org/10.1016/j.eswa.2022.118010.

[23] Q1 2022 phishing and malware report: Malware skyrockets, microsoft is the most impersonated brand. URL: https://www.vadesecure.com/en/blog/q1-2022-phishing-and-malware-report.

[24] SANYA GOEL, MUDIT BANSAL, ATUL KUMAR SRIVASTAVA, and NEHA ARORA. Web crawling-based search engine using python. In *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 436–438, 2019. doi:10.1109/ICECA.2019.8821866.

[25] David Mathew Thomas and Sandeep Mathur. Data analysis by web scraping using python. In *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 450–454, 2019. doi:10.1109/ICECA.2019.8822022.

[26] Davide Canali, Marco Cova, Giovanni Vigna, and Christopher Kruegel. Prophiler: A fast filter for the large-scale detection of malicious web pages. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, page 197–206, New York, NY, USA, 2011. Association for Computing Machinery. URL: https://doi.org/10.1145/1963405.1963436, doi:10.1145/1963405.1963436.

[27] Hyunsang Choi, Bin B. Zhu, and Heejo Lee. Detecting malicious web links and identifying their attack types. In *Proceedings of the 2nd USENIX Conference on Web Application Development*, WebApps'11, page 11, USA, 2011. USENIX Association.

[28] Chaitrali Amrutkar, Young Seuk Kim, and Patrick Traynor. Detecting mobile malicious webpages in real time. *IEEE Transactions on Mobile Computing*, 16(8):2184–2197, 2017. doi:10.1109/TMC.2016.2575828.

[29] Orestis Christou, Nikolaos Pitropakis, Pavlos Papadopoulos, Sean McKeown, and William J. Buchanan. Phishing URL detection through top-level domain analysis: A descriptive approach. *CoRR*, abs/2005.06599, 2020. URL: https://arxiv.org/abs/2005.06599, arXiv:2005.06599.

[30] Justin Ma, Lawrence K. Saul, Stefan Savage, and Geoffrey M. Voelker. Learning to detect malicious urls. *ACM Trans. Intell. Syst. Technol.*, 2(3), may 2011. URL: https://doi.org/10.1145/1961189.1961202, doi:10.1145/1961189.1961202.

[31] Shraddha Parekh, Dhwanil Parikh, Srushti Kotak, and Smita Sankhe. A new method for detection of phishing websites: Url detection. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 949–952, 2018. doi:10.1109/ICICCT.2018.8473085.

[32] Mohammed Nazim Feroz and Susan Mengel. Phishing url detection using url ranking. In *2015 IEEE International Congress on Big Data*, pages 635–638, 2015. doi:10.1109/BigDataCongress.2015.97.

[33] ShymalaGowri Selvaganapathy, Mathappan Nivaashini, and HemaPriya Natarajan. Deep belief network based detection and categorization of malicious urls. *Information Security Journal: A Global Perspective*, 27(3):145–161, 2018. URL: https://doi.org/10.1080/19393555.2018.1456577, arXiv:https://doi.org/10.1080/19393555.2018.1456577, doi:10.1080/19393555.2018.1456577.

[34] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel. Phishstorm: Detecting phishing with streaming analytics. *IEEE Transactions on Network and Service Management*, 11(4):458–471, 2014. doi:10.1109/TNSM.2014.2377295.

[35] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel. Phishscore: Hacking phishers' minds. In *10th International Conference on Network and Service Management (CNSM) and Workshop*, pages 46–54, 2014. doi:10.1109/CNSM.2014.7014140.

[36] Routhu Srinivasa Rao, Tatti Vaishnavi, and Alwyn Roshan Pais. Catchphish: detection of phishing websites by inspecting urls. *Journal of Ambient Intelligence and Humanized Computing*, 11(2):813–825, 2020.

[37] Paul K. Mvula, Paula Branco, Guy-Vincent Jourdan, and Herna L. Viktor. Covid-19 malicious domain names classification. *Expert Systems with Applications*, 204:117553, 2022. URL: https://www.sciencedirect.com/science/article/pii/S0957417422008715, doi:https://doi.org/10.1016/j.eswa.2022.117553.

[38] Neha Gupta, Anupama Aggarwal, and Ponnurangam Kumaraguru. bit. ly/malicious: Deep dive into short url based e-crime detection. In *2014 APWG Symposium on Electronic Crime Research (eCrime)*, pages 14–24. IEEE, 2014.

[39] Raj Kumar Nepali and Yong Wang. You look suspicious!!: Leveraging visible attributes to classify malicious short urls on twitter. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 2648–2655. IEEE, 2016.

[40] Yu-Chen Chen, Yi-Wei Ma, and Jiann-Liang Chen. Intelligent malicious url detection with feature analysis. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–5, 2020. doi:10.1109/ISCC50000.2020.9219637.

[41] Marco Cova, Christopher Kruegel, and Giovanni Vigna. Detection and analysis of drive-by-download attacks and malicious javascript code. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 281–290, New York, NY, USA, 2010. Association for Computing Machinery. URL: https://doi.org/10.1145/1772690.1772720, doi:10.1145/1772690.1772720.

[42] Yue Zhang, Jason I. Hong, and Lorrie F. Cranor. Cantina: A content-based approach to detecting phishing web sites. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 639–648, New York, NY, USA, 2007. Association for Computing Machinery. URL: https://doi.org/10.1145/1242572.1242659, doi:10.1145/1242572.1242659.

[43] Matthew Dunlop, Stephen Groat, and David Shelly. Goldphish: Using images for content-based phishing analysis. In *2010 Fifth International Conference on Internet Monitoring and Protection*, pages 123–128, 2010. doi:10.1109/ICIMP.2010.24.

[44] Kang Leng Chiew, Ee Hung Chang, San Nah Sze, and Wei King Tiong. Utilisation of website logo for phishing detection. *Computers Security*, 54:16–26, 2015. Secure Information Reuse and Integration Availability, Reliability and Security 2014. URL: https://www.sciencedirect.com/science/article/pii/S0167404815001145, doi:https://doi.org/10.1016/j.cose.2015.07.006.

[45] Moruf A Adebowale, Khin T Lwin, Erika Sanchez, and M Alamgir Hossain. Intelligent web-phishing detection and protection scheme using integrated features of images, frames and text. *Expert Systems with Applications*, 115:300–313, 2019.

[46] Sungjin Kim, Jinkook Kim, Seokwoo Nam, and Dohoon Kim. Webmon: Ml- and yara-based malicious webpage detection. *Computer Networks*, 137:119–131, 2018. URL: https://www.sciencedirect.com/science/article/pii/S1389128618301142, doi:https://doi.org/10.1016/j.comnet.2018.03.006.

[47] Mahmood Moghimi and Ali Yazdian Varjani. New rule-based phishing detection method. *Expert systems with applications*, 53:231–242, 2016.

[48] Mahmoud T Qassrawi and Hongli Zhang. Detecting malicious web servers with honey-clients. *Journal of Networks*, 6(1):145, 2011.

[49] Jose Nazario. Phoneyc: A virtual client honeypot. *LEET*, 9:911–919, 2009.

[50] M. Veena, S. Upasana, S. Prathima, and Sudha Senthilkumar. *A Detection of Malware Embedded into Web Pages Using Client Honeypot*. 09 2020. doi:10.5772/intechopen.89646.

[51] Nurul Fariza Zulkurnain, Azli Fitri Rebitanim, and Noreha Abdul Malik. Analysis of thug: A low-interaction client honeypot to identify malicious websites and malwares. In *2018 7th International Conference on Computer and Communication Engineering (ICCCE)*, pages 135–140. IEEE, 2018.

[52] Routhu Srinivasa Rao and Alwyn R Pais. Detecting phishing websites using automation of human behavior. In *Proceedings of the 3rd ACM workshop on cyber-physical system security*, pages 33–42, 2017.

[53] Luke Stephens (@hakluke). Introducing hakrawler: A fast web crawler for hackers, Jan 2020. URL: https://hakluke.medium.com/introducing-hakrawler-a-fast-web-crawler-for-hackers-ff799955f134.

[54] URL: https://gchq.github.io/CyberChef/.

[55] Check if a website is malicious/scam or safe/legit: Urlvoid. URL: https://www.urlvoid.com/.

[56] Surbl. URL: http://www.surbl.org/.

[57] URL: https://www.malwaredomainlist.com/mdl.php.

[58] Should i click or not? URL: https://www.shouldiclick.org/.

[59] URL: https://www.virustotal.com/gui/home/url.

[60] HappyFunCorp. Happyfuncorp/socialinvestigator: Collection of utilities to see what's happening in your network. URL: https://github.com/HappyFunCorp/socialinvestigator.

[61] InfoByIP.com. Informações pelo endereço ip. URL: https://pt.infobyip.com/.

[62] Homepage. URL: https://www.maltego.com/.

[63] Mandiant. Mandiant/flare-vm. URL: https://github.com/mandiant/flare-vm.

[64] Get long url from hundreds of url shortening services. URL: http://checkshorturl.com/expand.php.

[65] Expand shortened urls - expandurl. URL: https://www.expandurl.net/expand.

[66] URLhaus. https://urlhaus.abuse.ch/browse/. Accessed: 2022-10-10.