Alma Mater Studiorum · Università di Bologna

SCUOLA DI INGEGNERIA E ARCHITETTURA

Corso di Laurea Magistrale in Ingegneria Informatica

DEVELOPMENT OF A FRAMEWORK FOR THE ANALYSIS OF DOCUMENT-BASED PHISHING CAMPAIGNS

Relatore:
Chiar.mo Prof.
MARCO PRANDINI

Correlatore:
Prof.
GIANCARLO PELLEGRINO

Correlatore:
Dott.
DAVIDE BERARDI

Correlatore:
Dott.
ANDREA MELIS

Candidato:
ANDREA MENGASCINI

III Sessione
Anno Accademico 2020/2021

Nothing will come of nothing.
William Shakespeare

Introduction

Phishing is a type of social engineering where an attacker sends a fraudulent message to a human victim to obtain sensitive information or deploy harmful software on the victim's device, such as ransomware. Phishing attacks have evolved to the point that they now often transparently mirror the victim site. As stated in the 2020 Internet Crime Report redacted by the FBI [17] Phishing is the most common attack performed by cyber-criminals, with over twice as many incidents by any other type of computer crime.

Also, with the Covid-19 pandemic and the widespread use of lockdowns and work-from-home solutions, cybercriminals exploited this increasing users' need for the Internet. Multiple studies [9, 1]found out that the phishing attack skyrocketed up to 220% of its pre-COVID-19 rate. Those analysis highlisht the attackers use of a range of highly targeted scams to take advantage of the victims' confusion during the epidemic, including novel scam types for which current defenses are insufficient, as well as standard Phishing.

In recent years new categories of Phishing emerged. One of those is Document-based Phishing, where the attacks use PDF files containing fraudulent content to lure the victims into performing an action.

We will analyse a newly emerging type of phishing attack based on documents. This study will be focused on retrieving and studying those document-based phishing campaigns by developing a pipeline to collect data about the infrastructure and the techniques used to avoid detection.

Contents

List of Figures

Listings

Chapter 1

Phishing landscape

Despite document-based phishing being a novelty in the research community until the last period, the phishing ecosystem has been studied for more than a decade, and multiple solutions to mitigate those attacks exist and are currently deployed.
The Anti-Phishing Working Group (APWG) [7], PhishTank [36], and the newly formed COVID-19 Cyber Threat Coalition [14] are just a few examples of research and industry efforts to tackle phishing. Nevertheless, according to the latest IBM X-force report [56] phishing is one of the most common initial infection vector used among attackers.

The increasing variety of products present in the various black market remove the already low technical barrier to perform successful attacks, leaving the only remaining constraint the financial cost [4]. Moreover, we will be able to observe how in some scenarios, this financial barrier is non-present due to the presence of a free live phishing kit, undoubtedly increasing the popularity of this type of attack [13].
Part of the problem is that phishing is not only a technical challenge; it also presents a social engineering nature. Attackers have become experts at manipulating human psychology to persuade users to click on links and hand over their credentials. Multiple research [21] has also been done to study the psychological part of phishing and in most companies is quite common

to have phishing training awareness campaigns. Frameworks for classifying phishing emails based on a natural language scheme [52] were studied and has existed for quite some time.
This continuously increasing awareness against phishing attacks is surely a driving factor for the attackers to find new channels to reach the victim. Documents are not a new threat for carrying out attacks, but only recently we saw them used for phishing attacks on a large scale.
Moreover, with the COVID-19 Pandemic and the rise of the smart-working, digital document become the normality and its volume increased rapidly, making them even more appealing to Phishers.

1.1 Anti-Phishing Ecosystem

The anti-phishing ecosystem has been involved in a cat-and-mouse game with attackers for a long time. Despite the ecosystem's evolving defenses, the volume of phishing websites has steadily increased over time, recently reaching record-high levels.
Due to the support of illicit underground services, phishing is prevalent among criminals due to its scalability and low entry barrier, even for sophisticated and highly evasive attacks.
The popularity and risk of phishing attacks led to the creation of an extensive anti-abuse ecosystem that is relentlessly tested and enhanced by studies and research.

Anti-phishing techniques have existed since phishing came to the surface, and therefore multiple layers of defense are available.
Defense techniques vary from email classification filter [15] to direct report of malicious site [41], but also includecredential leaks analytics [23], blocklisting, URL and content classification [59, 54], malware scanning by web hosts [11], DNS and domain Intelligence [20], and content-take-down [6].

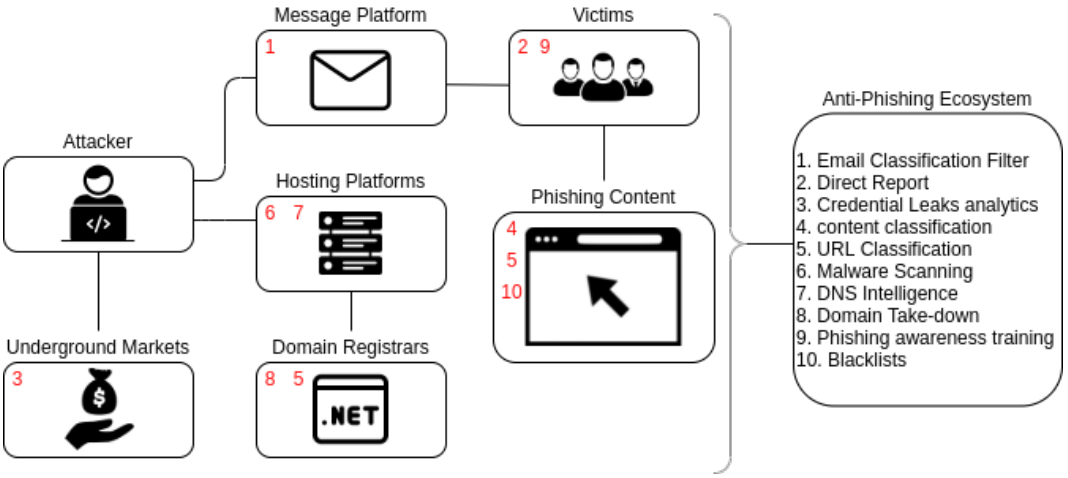


Figure 1.1: Components of the Phishing and anti-phishing ecosystem

1.1.1 Blocklist

Blocklist can be found in all major web browsers and prevent users from visiting malicious web pages via local filtering.
Considering that blocklists are implemented in every major browser[33], including mobile one's, it is not an understatement that they become the user's main line of defense against malicious websites. Those blocklists are a crucial line of defense against malicious sites that is enabled by default to protect even unaware users. This last technical barrier between the user and the sites is highly effective at stopping attacks like phishing, showing victims a prominent warning like the one that can be seen in Figure 1.2. Studies have shown that these warning messages greatly reduce user interaction with the malicious websites [46].
Before visiting an URL, the browser makes a call to a backend used for detection, like a URL blocklist or a heuristic classifier, and displays to the user a visual catching warning if the URL is classified as harmful. Those backend infrastructures collect suspected phishing URLs and, to avoid false-positive, verify the maliciousness of the content before adding it to the blocklist.

Chrome, Safari, Firefox and Chromium, which retain most of the global

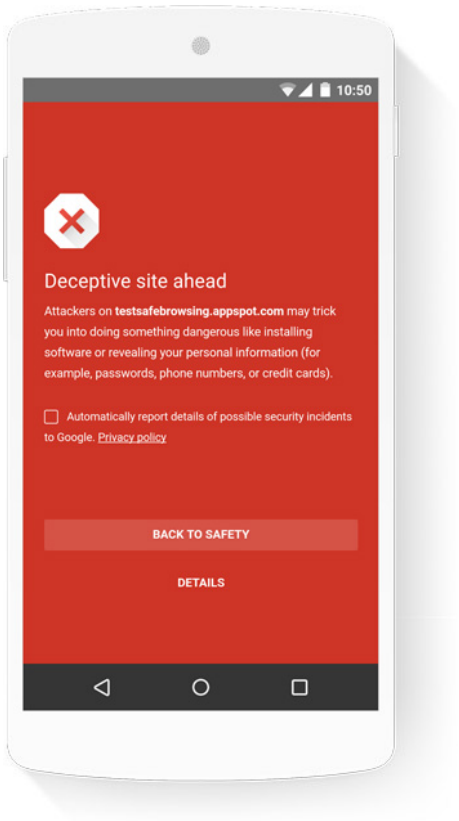


Figure 1.2: Deceiptive prominent warning in chrome browser from a mobile device

browser market share, are protected by *Google Safe Browsing* (GSB)[10]. Used by three of the most widespread browser, this blocklist is the most impactful, protecting more than 80% and 90% in the desktop and mobile ecosystem respectively by 2019 [10]. *Microsoft Smart Screen* is used on Internet Explorer and Edgeand accounts for approximately 13% of desktop users, and others solution exists like Opera's fraud and malware protection[10] While Google and Microsoft blocklists are documented and standalone [29, 18] , Opera does not disclose the source for its blocklist but experimental result [33] suggests that they use data from their third-party partners, Phisthank and Netcraft.

But from the creation of blocklists [46], the major weakness emerged: they

are a reactive solution to the problem.
Recent studies [34] found that an average phishing attack lasts less than 24 hours between the first and the last victim, and the detection of the malicious website occurs on average nine hours after the visit of the first victim. These time windows correspond to the most lucrative hours of the phishing campaign : the longer the phishing page remains online and accessible to victims, the more the attacker profit from it. After the detection, the page visits drop drastically because most users are informed by the browser that the site is deceptive. Nevertheless, up until the final takedown of the page, the campaign still earned visits and profits.
For this reason, sophistication in phishing sites, such as evasion techniques to evade blocklists, continue to proliferate. Attackers want to maximize their return-on-investment (ROI), maximizing the longevity of their campaign while remaining as stealthy as possible [19].

1.2 Evasion Techniques

As said previously, the longer the malicious websites remain online and accessible, the more the attacker gains. Blocklist detection needs to rely on content verification to minimize false positives; therefore, they are vulnerable to evasion techniques that try to delay or prevent the analysis of the page. [33].
Attackers (Phishers) evolve their techniques in response to the ecosystems standards and include new components in their attacks to circumvent existing mitigations in a cat and mouse fashion game. For example, most phishing and malicious pages adopted HTTPS and certificates, which helps give the visitors a false sense of security, avoiding in-security indicators.

1.2.1 Redirection Links

Another well-established trend is redirection links, which allow the attacker to distribute an URL different from the actual phishing webpage.

For the Phisher, hiding the URL is extremely important because it is the most direct indicator of the attack. It is common for the attacker to use a typo-squatted domain that resembles the targeted domain name via misspelt letter or using Punycode characters that the browser renders similar to regular characters. Given that the URL is one of the most crucial pieces of information that the blocklist leverage, Phishers try to promote their attacks using different easily replaceable indirection links.
Redirection links are used to evade detection by Phishers that distribute looking benign links, making it impossible to utilize URL heuristic detection methods and make it challenging to correlate URLs that are part of the same redirection chain [21].
Redirection chains commonly consist of multiple hops, using a URL shortener service or an open redirection vulnerability. This technique makes the number of unique phishing links higher than the actual number of phishing pages, causing some of the ending pages to slip through the various detection systems. The use of a well-known redirection service like bit.ly may fool victims to trust the links and allow intermediate to mitigate the attacks and give researcher possible data to study the phenomenon [8].
Redirection can be implemented with different techniques other than the simple HTTP redirect response, and attackers leverage all the types of redirection techniques to make their landing page less reachable by automatic analysis.
As Mozilla explain in their documentation [40] the main method for redirecting are:

- **HTTP Redirection**. The server can trigger the redirection by sending a special *redirect* response to a request. Those types of responses have a status code that starts with three and a `location` header that holds the URL to redirect to. The status code of redirection starts with 3 and is followed by other numbers indicating the redirection's cause. However, attackers misuse this technique only to provide a fast loading redirect, so we will refer to it as 3XX redirection.

- **HTML Redirections**. HTML Redirection are common when the attackers don't have control over the server. They consist of a `<meta>` element with the `http-equiv` attribute set to `Refresh` in the `<head>` of the page. The `content` attribute should start with a number indicating how many seconds should pass before the user is redirected to the new URL, and the new URL. This method can only be used with HTML and cannot be used for other types of content

- **JavaScript Redirections**. Attackers can also leverage the browser support to Javascript to redirect to other pages. The standard method used to redirect is setting the new URL string in the `window.location` property. Given that attackers can use this technique or similar only on a browser that supports Javascript is extremely popular because it eliminates automated visit from bots.

Because different ways of redirection can be used at the same time, browsers use this order of execution:

1. HTTP redirect as they exist even before the page is transmitted.
2. HTML redirects via the `<meta>` tag.
3. Javascript redirects that execute last, if Javascript is enabled.

1.2.2 Cloacking Technique

To prevent the anti-phishing ecosystem from discovering and blocklisting phishing pages and malicious content, the webpages use cloaking to display benign content or an error page if they detect that crawlers are visiting them [58]. Since most anti-phishing blocklists leverage crawlers to inspect the page's content cloaking those pages, those cloaking protection techinques are widely spread and effective.
Before being used in phishing attacks, cloaking has been used to trick search engines to influence rankings by showing different content to bots and humans.

Filtering can be implemented in server folders, server-side scripts, or Javascript executed in the user's browser as can be seen in the sempified cheme in Figure 1.3.
The techniques used on malicious web pages is only a fraction of the browser fingerprinting technique found by researcher. Cloaking needs just to be working against the current anti-phishing tools, and without external pressure, there is no need for cloaking developers to implement more complex approaches.

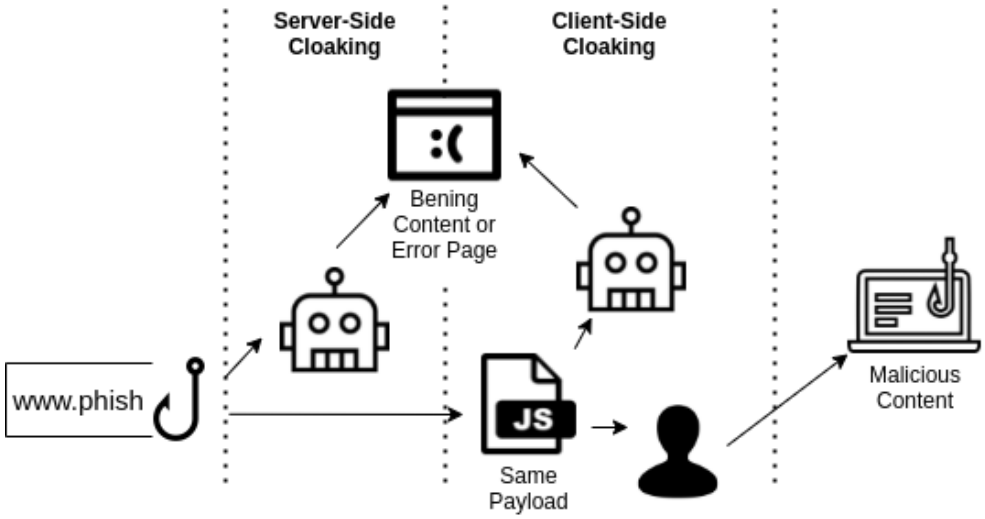


Figure 1.3: Redirection Flow with Server-side and/or Client-side cloaking technique

Server-Side Cloacking

A category of cloaking techinques focuses on allowing or denying access based on information the browser gives to the server, hence the name server-side cloaking. Request filtering is supported by many phishing kits and requires meeting some conditions before the phishing websites are displayed. Those conditions are an interesting topic of study because they explain what the attacker is trying to evade.

Those web cloakings are already well known and can be divided into three macro-categories based on filtered information [25].

1. Network Fingerprinting

    - **IP Address**: Some of the crawlers in the anti-phishing ecosystem re-use the same IP addresses range, allowing the attacker to enumerate and block the IP of those bots easily. Those lists also include multiple hosting providers, clouds, registrar and proxy and overlay networks such as Tor that rarely rapresent organic traffic. Those lists could also include entire ISPs if they are known to crawl websites or entire IP spaces like the one of academic or research institutions. Studies [25] highlight that those IP lists are updated regularly more than one time a day. It also allows the attacker to cloak the page even to a user after being visited several times, making it harder to report.
    - **Reverse DNS**: rDNS lookup of a visitor IP's are done and compared to a list of domain substrings of the major anti-phishing entities. If the check gives positive results, they are added to the IP blocklists.
    - **Geolocation**: Geographic targeting at country level granularity is widespread and allows to avoid blocklisting and offer specific geo-targeted phishing.

2. Browser Fingerprinting

    - **User-Agent**: Most search and advertisement crawlers use standard known strings, for example, Google with `googlebot` or Yahoo with `slurp`. All of those strings are blocked, and the checks extend to any user-agent that matches the substring `spider`, `crawler` or `bot` and any translation of those. On top of this, the user-agent string is used to present a different type of phishing campaign or

malicious page based on the Operating System (OS) and browser used.

- **Javascript**: The lack of Javascript is a commonly used fingerprinting technique both in the cloaking and the redirection links, as we have already seen. Almost all modern browsers support Javascript, and the lack of this functionality indicates that the visit is from an automated browser.

3. Contextual Fingerprinting

    - **HTTP Referer**: An evergreen technique is observing the HTTP Referer request header, which gives information about where the traffic originated. It can detect if the victim comes from a search engine or has followed the redirection chain. This technique blocks the crawler from visiting the page out of the context it appeared.
    - **Time Window**: With server-side logs, it is possible to prevent visitors from accessing the page more than once in a specific time window. This also prevents crawlers without a large pool of IP from being slowed down significantly.
    - **Order of operation**: Until now, the cloaking techniques illustrated rely on a single page to block crawlers, but this cloaking method takes advantage of multiple hops. When visiting a page, a cookie is handed to the browser and in the following redirection, the server checks if this browser is present and not expired. This allows the enforcement of a specific sequence of action like visiting multiple links or clicking to the next page that the crawler cannot easily mimic and often schedule a visit to a later time or different machine.

Client-Side Cloaking

Evasion techniques are also implemented client-side with the help of browser capabilities. Client-side cloaking leverage Javascript to require com-

plex interaction between victims and the phishing webpage, preventing automated mitigation by crawlers.

Unlike server-side cloaking, it is possible to retrieve the code of the client-side technique by crawling the website. Hence Phishers use extensively code obfuscation methods to hide their strategy, posing a challenge for static code analysis and detection. Cloaking can also target dynamic analysis by blocking fingerprinted automated browsers used in those types of investigation.

Client-side cloaking techniques have been studied [58] and similary to the server-side counterpart semantically categorized:

1. User Interaction

    - **Pop-up**: The phishing website presents a button in a pop-up window to the user, who has to click it to access the real page. Presenting the visitor an `alert` window evades anti-phishing bots visits because the page does not show any malicious content but waits for an `onclick` event to load the content. Currently, alert box can be closed by automation framework, so a new trend is to use Web Notification API [48] that are not yet currently supported by major automated browsers.
    - **Mouse Detection**: With this technique, the attacker tries to identify if the visitor is a human or a bot waiting for mouse movement before displaying the content. The cloaking page waits for events like `onmousemove` or `onmouseleave`. Attackers use long waiting animation to induce the user frustration and make mouse movements [22], that bots usually does not emulate.
    - **Click Through**: Like the pop-up technique, the phishing website requires the user to click on a specific item on the page before displaying the content. While a simple phishing page presents a button like the one in the pop-up, more sophisticated variants are emerging where the user is presented with a fake Captcha similar to the Google one. The increasing use of ReCaptcha verification

in legitimate sites makes phishing sites more difficult to identify for potential victims.

2. Bot behaviour

   - **Timing**: Showing phishing content only at a specific time may keep the anti-phishing detection system from flagging the page as malicious. Similarly, making the page load slow or artificially rendering it slow by using APIs like `setTimeout()` may surpass bot time thresholds evading those systems. On the other hand, the actual user might wait for the complete render of the page, especially if some indication of the page's loading is given. [22]
   - **Randomization**: This technique uses a non-deterministic mechanism for choose to show the phishing page or not. By generating a random number, if it is greater than a threshold, it shows the websites.

3. Fingerprinting

   - **Cookie**: Similar to the server-side technique, client-side cloaking also leverages cookies. By checking if the browser has cookie enabled, it can reveal if it is a human, given that crawlers often disable them not to be tied to a single session.
   - **Referrer**. In the same way as the server-side cloaking technique, Javascript can be instrumented to check the HTTP Referrer header to see if the page is reached via its redirection chain or just opened without context.
   - **User-agent**: With the use of `navigator.userAgent` it is possible to check the user agent and deny access if it contains a known crawler substring, as done in the server-side cloaking method.

Phishers can use multiple client-side techniques together to further increase the evasiveness against the anti-phishing ecosystem.

1.3 Technolgies

To better understand evasion techniques, it is crucial to explain the possible approach for phishing detection.
In either the approach used for phishing detection, the phishing page needs to be opened, and the main categories of detection differ from who open the page.
Phishing detection system involves either client-side or server-side approaches.

1.3.1 Client-Side Detection

The client-side approach leverages detection systems as browser add-ons or extensions to directly access the page content during the end-user visit.
With this approach, cloaking is less problematic due to the organic visit the user made, and it is possible to collect features and decide on the maliciousness of the page either client-side or later if data is sent to a system server-side.
The client-site method can access the same content as users because it is installed on top of the browsers.
However, lack of support of add-ons for different browsers, the absence of solutions in the mobile ecosystem and the privacy concern of a third-party extension capable of observing every content affect the popularity of client-side solutions.

1.3.2 Server-Side Detection

The server-side detection systems, e.g. Google Safe Browsing, receives possible suspect URLs by human report or automated crawling and collect data to detect the maliciousness.
This approach can be used by every browser interacting via an API to the detection system. Privacy is better preserved for the user as the URL is hashed before is sent to the server.

However, there is no guarantee that the URL will present the same content to the user and the server, especially if it uses cloaking techniques.

Automated Browser

While most anti-phishing detection systems are black-box to not let the attacker understand and exploit their weakness, it is fair to assume they use a sort of instrumented browser.
While most search engines announce their crawling by inserting a well-known string in the user-agent, in the anti-phishing ecosystem, the crawlers try to hide the automated nature and stay as close as possible to an organic visit.
For this reason, most anti-phishing studies and research use automated browsers to mimic the visit from a user.
Puppeteer [39] for example is a nodejs library that provides a high-level API to control Chrome, either in its full version or in headless mode.
Other automated solutions like Selenium exists and are not limited to the use of a single browser. Selenium is, in fact, a project that includes a variety of tools and libraries that enable web browser automation: it provides an infrastructure for the W3C WebDriver specification [49], a platform compatible with all major web browsers. Available to use with Selenium are all major browser webdrivers, and also custom made one's, like `phantomjs` [35] or `undetected_chromedriver` [50] focused on not triggering anti-bot services.

Forceful Execution

While automated browsers cope with some of the cloaking techniques, others are hard to avoid or change too rapidly and make the automation process hard to keep updated.
With techniques that leverage order of operation or user interaction, it becomes hard for traditional automation tools to reach the phishing page.
For this reason, new techniques like J-force [27] which rely on a forceful execution of the Javascript to reach the final landing page, are being used to

study cloaking techniques on phishing pages.
As we have seen, once we escape the server-side techniques, the server will give us the page with the Javascript code that can contain the client-side cloaking technique, maybe even obfuscated.
In the web page given by the server, even if some cloaking techniques is present, a branch of the execution of the Javascript will lead us to the actual phishing page. With this assumption, this technique relies on forcefully executing all branches of the Javascript code and comparing the result to choose one outcome of the Javascript code as the final phishing page. We can later infer the cloaking techniques used based on the difference between the results we obtained.
While it seems the most accurate solution on paper, it presents some drawbacks. The biggest is computational power limits: executing all execution paths of the Javascript present in the malicious page take a long time, making it impossible to run in real-time scenarios. Even if used in server-side detection, it needs to be limited in the number of paths it explores, and time thresholds must be set to avoid infinite loops or resource wasting as observed by Invernizzi et al. [58]. In the process of making their page undetectable, attackers often put artificial time constraints in their Javascript code, in the form of timeout and also use functions to decrypt at runtime piece of code evaluated by the `eval` function. Those cloaking techniques threaten the Javascript forceful execution method, wasting the limited resource anti-phishing systems have.

1.4 Document Attack

The use of malicious document files for attacks is known and can be traced back to more than a decade. Those attacks consist of leveraging famous document extensions and related software to deliver malware or execute arbitrary code.
During the years, the Adobe file format, PDF, got multiple security updates,

so their readers, which have been found vulnerable to different attacks.

Multiple malicious Javascript and malware have been found in PDF files, and frameworks have been created to analyse those types of documents.
The same occurred with other widespread document extensions, like the Microsoft renowned `docx` extension. Malware has targeted the Microsoft Word editor to install malware, so solutions to analyse malicious documents with static and dynamic analysis emerged [16].

1.4.1 Phishing PDF

Recently, new use of those documents appeared, a category of attack where the Phisher sends fraudulent content via PDF files, luring the victim into performing some actions. While document attacks still exist, this new category of social engineering attacks does not rely on bugs and exploits in the software but uses social engineering techniques to trick the victims. Document-based phishing is a new topic in the research community [47], but quickly become a global and persistent threat at the moment neglected and unopposed by the anti-phishing ecosystem that haven't implemented any form of content filtering or detection yet.
Phisher lures the victim with visual bait in the document, often impersonating known companies or using a well-known template to make actions, like clicking on a hidden link that redirects them to the phishing page.

1.5 Scam Landscape

Attackers do not always try to steal user's login credentials but use deceptive products and services to obtain money, personal data, or even install unwanted software. Those services swing from health and beauty products to financial services. They are commonly referred as *scam* and has been studied for a long time [30, 12].

Search Engine Optimization

Those scams are not always sent via email, but attackers use social media platforms or search engines to arrive at the end-users. Popular sites like YouTube, Pinterest and Facebook are used to host many URLs of those pages. However, attackers also leverage advertisements for tricking users or even ranking their service through a reputable search engine. Many techniques are used to rank in the Search Engines, from SEO poisoning [24], to buy PBN backlinks [51] and even use highly ranked defaced websites [57].

Affiliate Advertisement Network

An increasing amount of Social Engineering scam campaigns are delivered via malicious advertisement. Those ads are served via an AD network which does not always verify the ad content they serve. Those AD networks use an affiliate market business model by collecting services or products the publisher wants to show to the users and pays the affiliates who deliver them to the victims. With this strategy, the AD network relieves the campaign's responsibility to attract users to the affiliates, who use deceptive methods to promote the goods.

Chapter 2

Research Question

We decided to develop a pipeline to collect document-based phishing campaigns data and information starting from the document.

2.1 Dataset

The dataset is the one presented in [47], still in review, provided by two industrial partners. Those sources constantly gave us daily feeds of new documents. The first partner collects data from VirusTotal [53] filtering by file type and engines that flag the document as malicious. The second partner uses multiple feeds of malicious documents from various sources, one of which is VirtusTotal, and send the data only when confirmed by a second source. PDFs are then processed by an already built pipeline that extracts features from them. The data from which our analysis starts are the URLs extracted from the document, which is our entry point for our analysis.

2.1.1 Clustering

The features extracted from the already existing framework was used by Stivala et al. to identify different clusters of malicious campaigns. They identified 37 malicious campaigns with more than 80 clusters. When looking at the duration and volume of the campaign, it is clear that two extensive

campaigns capitalize the dataset, accounting for almost 80% of the total dataset and lasting for the entire study.

The dataset created from those feeds of Documents contains files written in 56 different languages. They are not evenly distributed, but English is the most popular, followed by Russian and Spanish. This indicates that some campaigns target a specific country and may not be accessible from different geo-localized IP range.

Visual Deceipt

As studied in the research of Stivala et al., who presented this dataset, there are two main aspects in the visual deceits used in the documents. The first is the content of the message, and the two categories that emerged are the promotion of goods, services or illegal content and the impersonification of a legit service. While most of the promotion is done with non-elaborate layout documents, those who try to pass as legit, reproducing parts of communications like emails from colleagues or present a step in a well-known online process, like a Captcha procedure. The second aspect of visual deceit is the visual elements and structure of the document. Unlike email-based phishing, only small campaigns presented a business document or a cloud or email notification layout. Most of the documents in the dataset contain some UI elements typical of web pages or are just filled with plain text with sometimes the use of images.

2.2 Research questions

As we have seen, extensive work has been done to fight phishing, and different studies have focused on classifying it. Any of those works focus on the attacks behind Document-based phishing. This work wants to create a Pipeline starting for those documents, capable of collecting data to analyse those attacks.

We wanted to be able to collect data to later respond to different questions:

- *Which types of attacks are delivered with phishing documents?*
  Phishing is recently used to indicate a wider variety of attacks. Traditional credential phishing makes up only a portion of phishing attacks. New types of Social Engineering attacks have emerged and are capitalizing on the Phishing landscapes. With our pipeline, we want to gather enough data to classify the range of attacks delivered with phishing documents.

- *Does they target the same companies and use the same deceptive technique as a standard phishing landing page?*
  Phishing has always used deceptive methods and techniques to induce the victim to complete an action. We want to know if those techniques are the same with document-based phishing or more or less elaborate.

- *Which types of infrastructure are behind those types of attacks?*
  We want to gather information about the attacker's infrastructure to comprehend what measures could be used to block those types of attacks. The data gathered will be used to compare those infrastructure complexities to the regular phishing ones. We are also interested in the techniques used to defend those infrastructures: we want to collect information about their client and server-side cloaking.

- *How fast is the burn rate of the infrastructure? Where can those PDF be found? Are the domain in which they are hosted higher enough to be ranked?*
  Since we have a daily feed of new documents, we want to study those campaigns' longevity and the anti-phishing ecosystem's latency to this new type of attack. We also want to know if some of the document increases in popularity and shows up on the top-visited websites.

Chapter 3

Pipeline

We decided to develop a pipeline to collect document-based phishing campaigns data and information starting from the documents.

We developed the pipeline to automatically start every day in a Ubuntu-based LXC container hosted in our local server. This decision gives us a wide range of options for the tool we could choose and the configuration we could apply. We choose Python3 as the programming language for its wide range of libraries and online documentation for the crawler and scraping.

3.1 Automated Browser

Some of the more straightforward requests in the pipeline are implemented via the `pyrequest` HTTP library [42] that allowed us to send HTTP requests easily.

3.1.1 Selenium

However, to support Javascript, we choose Selenium [45]: an open-source framework used for testing and scraping web applications that enables user impersonification with the most famous web browsers. A critical aspect that let us choose Selenium over other solutions was the ability to use most modern browsers in the form of web drivers.

For this reason, we created the class `SeleniumDriverManager`, responsible for creating the `webDriver` object with the specified parameter selected.
When launching the pipeline is indeed possible to specify between different browser and version and add optional arguments.
The currently supported browsers are:

- **Chrome**. Using *chromedriver* is possible to run an instrumented version of chrome with the same feature as normal. Furthermore is possible to specify *Capabilities*, options to customize and configure a `ChromeDriver` session.

- **Firefox**. Using *geckodriver* [32] is possible to choose Firefox as webdriver. Mozilla released this Proxy for using W3C WebDriver compatible clients to interact with gecko-based browsers.

- **Undetected_chrome**. Optimized and patched version of *chromedriver*, which does not trigger anti-bot services. It download automatically the driver binary and patches it [50].

Furthermore, it is possible to specifically pick a version of the various browsers or use the latest version by default. After specifying the parameter, the class will download the chosen version and instrument the pipeline to use the selected webdriver.
The pipeline is currently set to run using chrome webdriver, and we finetuned specific additional options to customize our browsing experience.
Given the absence of a display in our configuration, we use the `headless_browser` command to instruct the webdriver to not display anything. We forcefully set the virtual resolution to 1920x1080, the full HD standard to not be fingerprinted in case of cloaking. In the same way, we disable the GPU when launching the pipeline with the argument `disable-gpu` when creating the webdriver.
Another helpful option that Selenium gave us is the page load timeout, a threshold to not waste our resources on an unresponsive page that we conveniently set to 15 seconds after our first experiments.

Screenshots

To later study the visual catch of the final landing page, we decided to take a screenshot using the method provided by the webdriver. All the final landing pages we collect need to have the screenshot, and in case of error, a job is scheduled to re-take the screenshot. Multiple runs of the pipeline for the same URL are possible and may have different final screenshots, so we added a field with the hash in the database for the ease of comparing them.

Page Download

To study the final landing page, we decided to save a static copy of the page in our filesystem. This decision exposed different problems, starting from the method we need to use to the fidelity and completeness of the downloaded pages.
We tried different techniques to download the page:

- **pywebcopy**: A web scraping and archiving tool to save a complete webpage in Python. Unfortunately, cloaking technique or simple external imports let the tool hang on indefinitely.
- **chromedriver page download**: This technique mimics a user downloading the page with the tool the chrome browser offer. By using the macro `CTRL + S` a popup opens to save a static version of the page. While this technique worked reliably, in the headless mode, we had to rely on a virtual screen, and the python library for the mouse movement was not quite accurate, causing the download button not to be clicked and the pipeline to be stuck.
- **wget**: We tried and used for the first days the popular network utility `wget`. We used high customization and recursive function, instrumented it to wait a random time between the page visit and shared custom headers with the webdriver used at that moment. However, even if the GNU tool worked, we replaced it with a more straightforward solution.

3.1.2 Selenium-wire

Initial experiments showed how saving only the last page did not permit us to measure and analyse the complete redirection chain and the intermediate page used. Using the methods mentioned above for downloading the page caused too much overhead, so we decided to use a man in the middle proxy to access all traffic easily. We decided to use Selenium wire [55] as a proxy since it extends Selenium python binding giving access to the request made by the browser. Those extra APIs for inspecting requests and responses were used for logging all traffic.

Accessing Requests

Selenium Wire captures all HTTP and HTTPS[1] traffic made by the browser and gives API to access it. These APIs were crucial in developing the flow of the redirection chain and dynamically resolve some cloaking techniques.
The driver objects capture all the requests in chronological order in the `driver.requests` attribute and it is able to wait for a request matching a specific pattern with the `driver.wait_for_request(pat, timeout=10)` method, where `pat` is a sub-string or a regex.
Those API enabled us to cycle through the requests and wait for a response to asynchronous action.

Request and Response Objects

The easy access to the request object and the ability to modify it on the fly allows us to highly customize our headers during the pipeline execution. It is possible, indeed, to access different attributes of the request object starting from the body. A crucial attribute in our development was the `headers` : a dictionary-like object of request headers that can be cycled through and

[1] via the OpenSSL package

edited on the fly.
We also heavily relied on the response object and its attributes, stuctured in the same ways og the request object, that allowed us to easily access the body of the message.

Upstream Proxy

An essential function of selenium wire was the ability to use an upstream proxy. As we will see in the following pages, our analysis and method of escaping server-side cloaking heavily rely on the use of a proxy. In the mentioned above class `SeleniumDriverManager`, if an upstream proxy is passed as parameter, it will be added to the Selenium options as can be seen in listing 3.1.

```
if proxy:
    options = {
        'proxy': {
            'http': 'http://' + proxy['http'],
            'https': 'http://' + proxy['https'],
            'no_proxy': 'google.*'
        },
        'exclude_hosts': ['google.*']
    }
```

Listing 3.1: UpStream Proxy options in Selenium

3.1.3 Residential Proxy

As we started launching the pipeline, it was clear that while most phishing pages in the various phishing datasets reached the final page, the URLs in the PDFs of our study were redirected to a benign page or a different page as was promised on the document.
Comparing this to the initial manual analysis we made of those document-based campaigns, it was evident that we were victims of some cloaking technique.

We could not arrive for example at the promised software or the promised streaming service, but we were redirected to fake news articles or offline pages.
The problem revealed to be the server IP from where we were launching the pipeline. The machine indeed appears to be on a network of known IP for the phishing community, and it has been commonly added to the blocklists of not allowed IP in the phishing kits [25].
Launching the pipeline from different residential IP changed the pipeline's outcome, which reached most of the final landing page or some client-side cloaking technique based on human action.
The preliminary study showed us that the burn rate of those residential IP is very high, especially for one of the most prominent document-based campaigns, limiting our ability to crawl their various landing pages.
Moreover, using different IP from various countries in our analsyis, we found that the landing page gratly changed based on the geolocalization of our IP. While for some campaigns, the content is only translated and for others, the site is unavailable, some landing pages present a geo-localized tailored attack to the end-user.
For this reason, we decided to use a residential proxy service. This service let us use a pool of real residential IP provided by Internet Service Provider (ISP) instead of directly connecting to the server. In this way, the phishing pages identify us as organic internet users, not revealing the nature of our analysis. As can be seen in figure 3.1 all the traffic to the internet, no matter which browser is chsoen, will be routed to the residential proxy.
The large pool of IP those services offer lets us select the country of the IP we are using as a proxy to gather information about the various different landing pages.

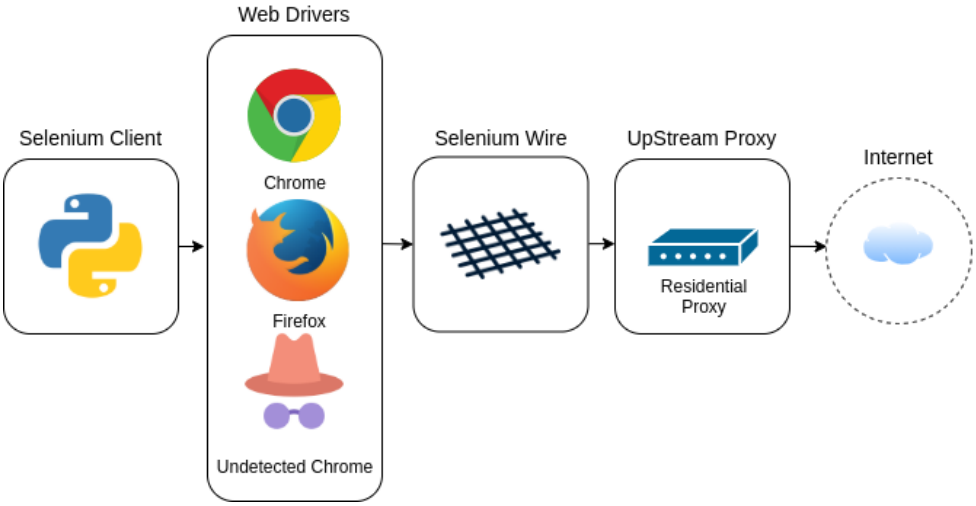


Figure 3.1: Selenium-wire architecture overview

Chapter 4

Data Collection

To study the document-based phishing and social engineering phenomenon, we decided to collect different types of data. On the following pages, we listed the data our pipeline collects and the implementation challenges and solutions we adopted.

4.1 Redirection chain

As we explained, redirection links are among the most well-established trends in phishing and social engineering campaigns. This technique is not only used as a cloaking mechanism, but the attackers also gain a link that seems not malicious that can be posted on social media platforms. Those types of redirection can also be used for collecting metrics about one campaign leveraging analytics services.
To collect information of a single link of the redirection chain, we designed a `Phish` object that can be instantiated given the URL and contain the data we want to collect. Chaining multiple Phish objects in an ordered list gives us a redirection chain that we can quickly cycle.

4.1.1 Control Flow

As we can see in Figure 4.1 that represent the control flow of our pipeline, the initial request is made by the python library to assess the status of the page quickly. The request is indistinguishable from the one made by Selenium because they share the same headers. If a `5xx` error code status is found, the pipeline stops and declares the URL offline, saving the result in the database.

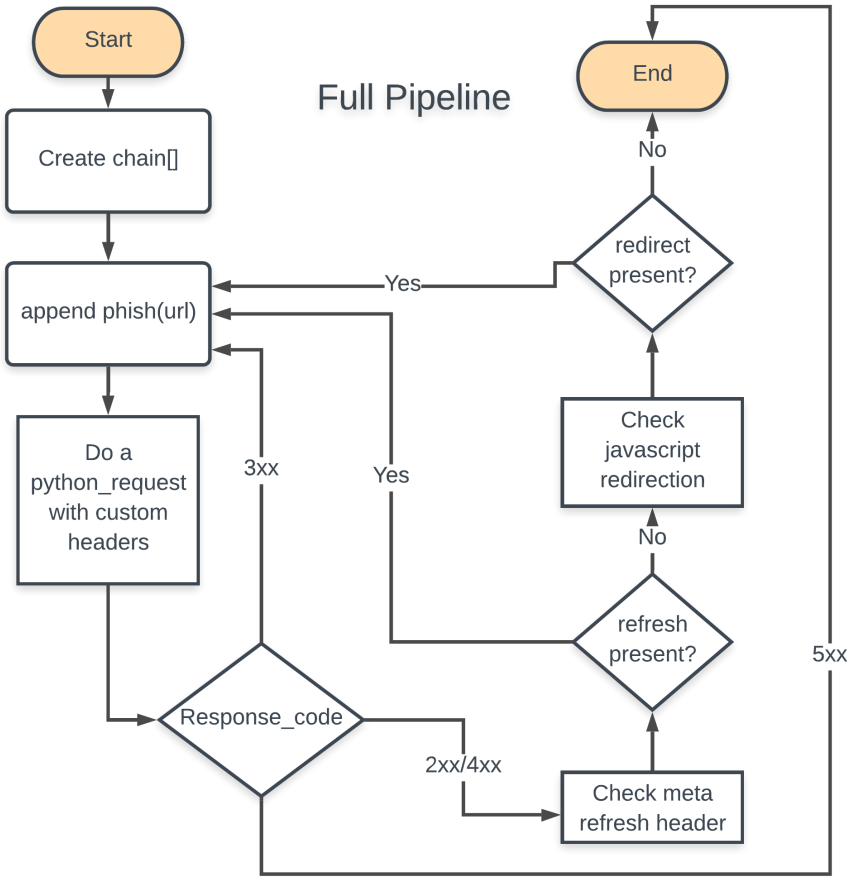


Figure 4.1: Flowchart of the redirection order resolution

We choose to not start using Selenium at this initial stage to not overload our server. If the pipeline is launched with the upstream proxy, the same

proxy is used by the python requests library as well by Selenium for the sake of consistency.

By checking if the page presents a `5XX` or a `3XX` HTTP response status code, we are sure we do not miss any Javascript redirection because it is the last one executed given the order of priority of execution of the browser. If a redirection response code or a refresh meta attribute with refresh time equivalent to 0 is found, we append a new Phish object to the chain object and start analysing from that link.
Only when the HTTP response status code is `2XX` or `4xx` the pipeline start using Selenium. We also chose to collect data and continue analysing the landing page even if it presents us with the `4xx` error status code. We found in our initial investigation that some phishing land pages show malicious content with a `4XX` code. This technique allows the attacker to evade some crawling technique while remaining unnoticed by the user: not an error page is shown but the actual phishing page.
With Selenium, we continue to analyse the page, try to evade the cloaking techniques, and wait for a threshold before capturing a screenshot of the final phishing page.

4.1.2 Server-Side Cloacking

To escape server-side cloaking, we apply various techniques. We resolved our IP range problem relying on the residential proxy, letting us avoid IP cloaking, Geolocalization and look upon our DNS entry.
Pyrequest and Selenium share most of the information used by server-side cloaking techniques to detect a bot during the pipeline execution. The User-agent string and the cookie, set in the Session, are shared and added to Selenium when the use of python-requests end and the Selenium Session starts. To avoid being fingerprinted at the start of an URL analysis session the user-agent is modified with a perturbation obtained by changing the OS version and adding fake information about the plugins installed on the browser. This perturbation is obtained by the permutation of OS verion and

plugin information we stored in local saved list.

4.1.3 Client-Side Cloacking

Given the nature of the pipeline, all the client-side cloaking techniques will appear during the Selenium session, being the only tool capable of using a webdriver that supports Javascript.
We developed some techniques to avoid the most common cloaking technique:

- **Mouse Movements**. We added a bezier curve mouse movement with the `pyautogui` library done during the waiting time of the loading of the page to simulate a real-human behaviour.
- **Pop-up**. We instrumented `chromedriver` to close any alert box automatically. For the new technique of cloaking that uses Web Notification that can be seen in Figure 4.2, we accept by default any, simulating an unsuspicious victim.
- **Referrer**. The referrer, as the user-agent and the session, is carried during the transition between Selenium and python-request not to fail any check in the redirection chain.

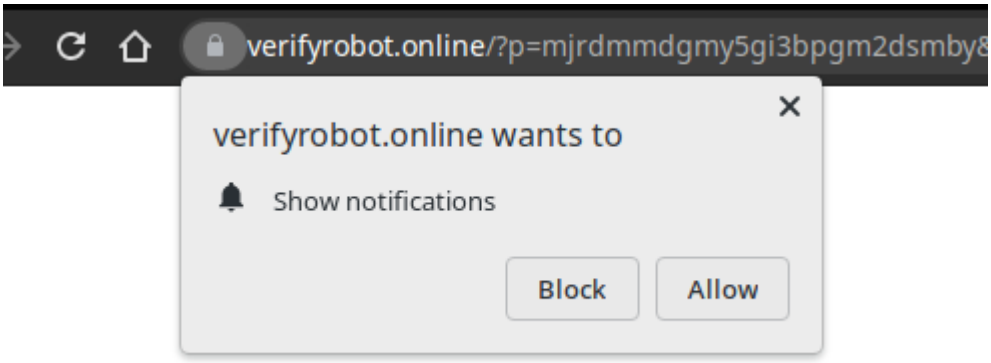


Figure 4.2: Cloaking techinque leveraging Web Notification API to show a pop-up

4.2 Blocklist

To study the response of the existing anti-phishing ecosystem, we decided to monitor if the URLs we visited appeared to be flagged in blocklists.
We decided to analyse the most widespread Blocklist, Google Safe Browsing (GSB), because, as we have seen, it has the most significant market share.
After the redirection chain has been visited and saved to the database, the online URLs are checked with the GSB Database every hour. If they appear to be blocklisted, we log the timestamp and the cause for which the URL has been flagged. From the moment we found them flagged, a script runs daily to check if the URL has been unblocked and is no longer present in the Blocklist.

4.2.1 REST service

To have always an updated version of Google Safe Browsing, we used a dockerized REST Service to look up URLs in Google Safe Browsing v4 API based on gglsbl [3] and Flask [31].
This solution let us overcome the problem of updating the local SQLite database, which made the pipeline stuck for different minutes. The SQLite database used by default is locked during writes, so the downtime caused was noticeable, and a race condition could happen.
The REST version, however, set the database to write-ahead logging mode so readers and writers can work concurrently, We modified the scheduled task that updates the database and perform a full checkpoint to run every 15 minutes.

4.3 Documents

Most of the PDFs in our dataset present a similar distribution of URLs they contain. Almost the totality of the PDFs presents only one link that redirects us to the final landing page, and all the other URLs point to other

online hosted PDFs.
We wanted to study the other PDFs as well as the domain used for hosting. We surprisingly found that even for not so newly collected URLs, many PDFs are still online, and for the new ones, almost every cross-linked PDF is online. In order to not add overhead to the pipeline, URLs which path ends with `.pdf` extensions (fragment excluded) are first analysed to see if their content type match the one of a PDF file, `application/pdf`. If they match, the PDF is downloaded with python-requests, and if the file hash of the PDF is not present in the dataset, they are sent back to the first pipeline as new data.
Using this simplified method, we found out that some websites used to host those PDFs, mostly free web hosting providers, applied cloaking techniques to disable crawler and scraper if they detected the lack of Javascript.
In case of failure downloading the PDF with the python-request library, we automatically launch the pipeline and handle the URL as a standard phishing one, but using selenium-wire as proxy, we can immediately capture a PDF in the traffic and stop the pipeline to save them and conclude the current analysis.
Those types of cross-related URLs are used on the web and are known as private blog networks (PBN) [51]. These ecosystems of web pages are a type of black-hat method to boost ranking across various search engines through artificially created backlinks. Although little is known about ranking algorithms in major search engines, it is strongly suspected that incoming links significantly influence the position.

4.3.1 Threshold

Using our daily feeds of documents, we analysed the provided phishing URLs, but downloading other PDFs and adding them to the pipeline caused us to deplete our resources quickly. On the first days of testing the pipeline, we collected as many documents as the initial dataset, most pointing at the same URL or domain but with a different path or query.

Analysing the PDFs, we found that most campaign templates are identical and text slightly changes but always presents the same keywords.
For this reason, we decided to limit the downloading capability of our pipeline. We set a total daily threshold of 10,000 documents and set a scoring system to assign priority. The more URLs with unseen domains a PDF contains, the higher score gets. Then out of these ranks, we select the first 200 PDF and launch the pipeline for the URLs contained. Then we collect those PDFs, and if their hash is not present in our database, we extract the URLs and add them to the ranking. Given the average documents cross-linked to one PDF, in the first iteration of this algorithm, we obtain on average 4000 new documents.
In order to see and possibly study the number of those cross-linked PDFs and their ephemerality, we deplete our remaining available documents download before reaching the thresholds by visiting the URLs in the rank order.

4.3.2 Filesystem and naming convention

To save and access those PDF easily in our filesystem, we heavily relied on the SHA256 file hash. To overcome the limitation and slowness of our filesystem that became unresponsive when we surpassed the order of 100.000 documents in one folder, we decided to use a structured naming convention. After some tests, we decided to divide our files with a 4 level directory structure, giving the directories the name of the consecutive two characters of the hash.
While the original filename was kept in the database, the path of those PDF could be found, adding the base path and the hash of the file as it can be seen in the filesystem structure in figure 4.3.

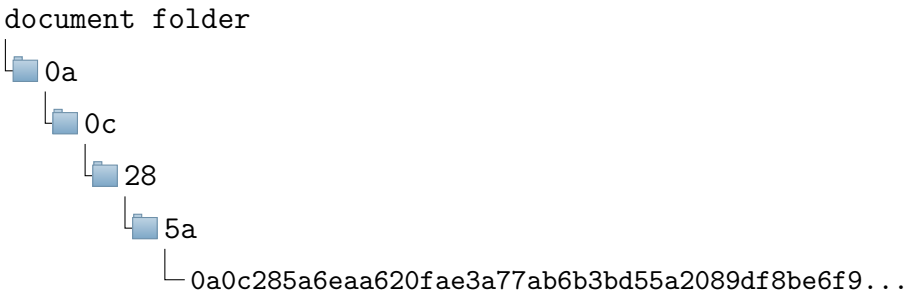


Figure 4.3: Folder path of documents with 4 subfolder named after the starting characters of the hash.

4.4 Sites ranking

To understand and measure the popularity of the domain used in this attack, we decided to check for each domain we have the position in top sites ranking. To get a better and more unbiased metric we selected two popularity ranking :

- **Tranco** [28], a Research-Oriented Top Sites Ranking Hardened Against Manipulation.

- **Alexa** [5], a global ranking system that ranks millions of websites in order of popularity, estimated average daily unique visitors.

With those two data, we can study if the ranking of some domains was manipulated to be better ranked to gather traffic from different search engines. This inexpensive analysis is done on every redirection chain: the computational cost is minimal because we kept a daily updated local version of these lists. This analysis will also help check if some open redirection or URL laundering is done using a popular domain, inducing the user to trust the URL more.

4.5 DNS Information

In order to later study the taxonomy of the phishing infrastructure and its various chain, we decided to query DNS for the various domain we found in our analysis.

We decided to look up those entries specifically:

- **A**. The record that points a domain to an IPv4 address.
- **AAAA**. In the same way the A records point to the IPV4, the AAAA records point to an IPV6 Address.
- **CNAME**. This record is used to link a subdomain to a domain's A or AAAA record instead of making 2 A records. For example, it is possible to link `test.phishing.com` with a CNAME record to an A record set on `phishing.com`, and they would both point to the same server.
- **CERT**. CERT resource records are used for storing certificates in DNS.
- **NS**. DNS Name Server (NS) record specifies the domain name of the name server servicing a particular domain.
- **TXT**. A TXT record (short for text record) is an informational DNS record used to associate arbitrary text with a host or other name.
- **SOA**. The DNS start of authority (SOA) record stores important information about a domain or zone, such as the email address of the administrator, when the domain was last updated, and how long the server should wait between refreshes.
- **DNAME**. DNAME-record is used to map or rename an entire subtree of the DNS namespace to another domain. It differs from the CNAME-record, which maps only a single node of the namespace.

- **CAA**. CAA stands for Certification Authority Authorization and its record is used to specify which certificate authorities (CAs) are allowed to issue certificates for a domain.

- **MX**. The MX record identifies the mail server that receives email messages on behalf of a domain name.

In order to analyse those DNS entries, we used the `dns.resolver` package offered by python.

4.6 Autonomous system information

In order to have a complete view of the infrastructure of these attacks, we also instrumented the pipeline to collect information about the entity behind it.
We regularly check the domains we found with Whois, a widely used Internet record listing that identifies who owns a domain and how to get in contact with them.
We also used the Whois daemon to ask bulk information to the Cymru database [26], which maps IP to Autonomous System Number (ASN). An autonomous system (AS) is a large network or group of networks with a unified routing policy, and each of them is assigned an official number as it can be seen in figure 4.4. By collecting these pieces of information, we will be able to map the IP to the network operator.

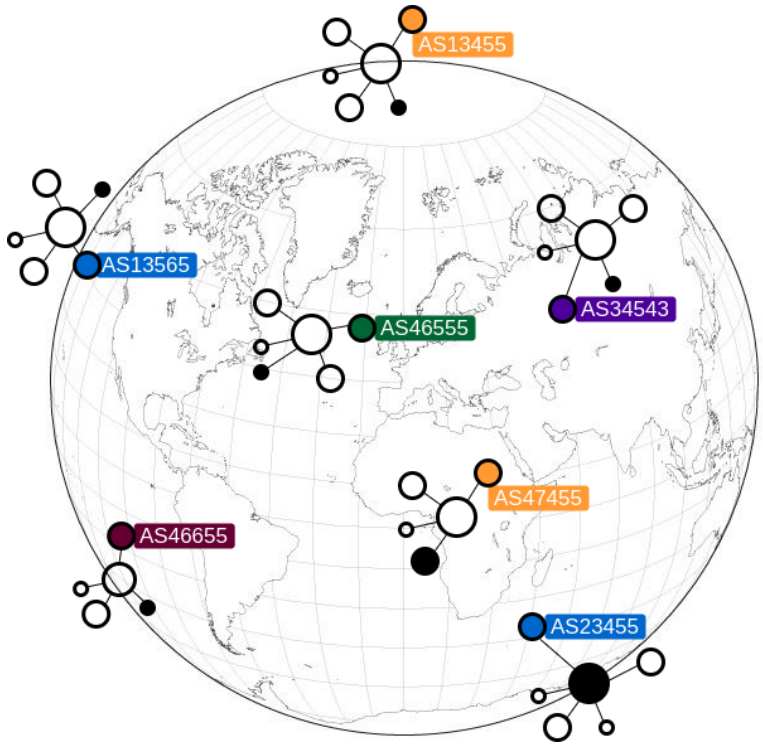


Figure 4.4: autonomous systems (AS) is a large network or group

4.6.1 Database

All the data we collect is stored in different tables shown in figure 4.5 on a PostgreSQL database, an open-source object-relational database system known for reliability and performance [37].

ip	as_num	bgp_prefix	cc	registry	allocated	as_name
34.102.176.152	15169	34.100.0.0/14	US	arin	2018-09-28	GOOGLE, US
52.217.96.102	16509	52.217.96.0/20	US	arin	2015-09-02	AMAZON-02, US
52.216.233.85	16509	52.216.233.0/24	US	arin	2015-09-02	AMAZON-02, US
143.204.98.21	16509	143.204.96.0/21	US	arin	2018-01-05	AMAZON-02, US
162.13.135.168	15395	162.13.0.0/16	GB	ripencc	1992-06-30	RACKSPACE-LON, GB
95.217.28.189	24940	95.217.0.0/16	DE	ripencc	2009-02-24	HETZNER-AS, DE
185.215.113.14	51381	185.215.113.0/24	SC	ripencc	2020-11-13	ELITETEAM-PEERING-AZ1, SC
167.114.126.65	16276	167.114.0.0/17	CA	arin	2014-08-29	OVH, FR

Table 4.1: AS Network analysis data

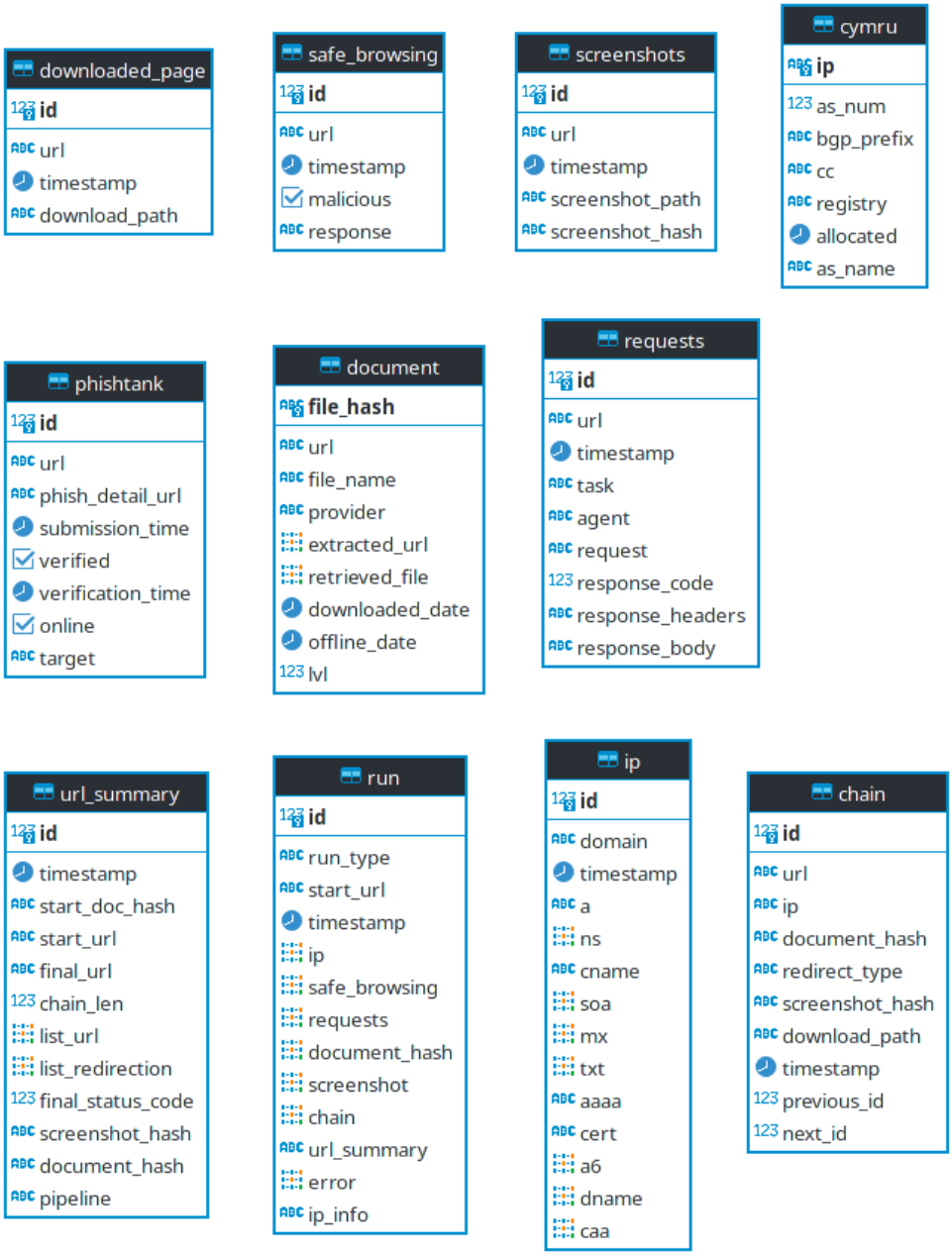


Figure 4.5: Database ER Diagram

In order to make query faster we created indexes with the most used data, like filehash, IP and domain. Indexes are special lookup tables that can be

used by the database search engine to speed up data retrieval. It helped speeding up `SELECT` and `WHERE` clauses and not slowed down too much the data input.

We easily accesses the database with the pipeline via the `psycopg2` package [38], a database adapter for Python.

Chapter 5

Analysis

We used the pipeline and its initial results to study the most prominent campaign, correcting our pipeline to collect the most meaningful data.
We found results that validate previous research but we have to point out that our analysis is just preliminary, and the outcome could change when we have completed our data collection.
Indeed, studied phishing and social engineering attacks in various research have been volatile and rapidly changing [34].

5.1 Preliminary analysis on PhishTank

In order to test the pipeline and its ability to overcome the various cloaking techniques, we decided not to use the URLs extracted from the PDF. As we said, document-based phishing is an emergent threat, and we did not want to alarm the Phisher by visiting their infrastructure with an incomplete crawler, letting them notice we are investigating the phenomenon.
In our analysis, we used the publicly available dataset Phishtank [36] which is continuously updated with new and online phishing examples. Only when the success rate of the pipeline was high enough, we launched our pipeline to a small number of URLs retrieved from the PDFs.
As we explained before, our initial analysis on the URLs from the documents

failed because we did not expect such fierce server-side redirection and IP blocklisting technique. Once we overcame this difficulty, it was clear that the results were different from the ones we found on Phishtank.
Most of the PDFs landing pages we visited did not contain phishing attacks but more Social Engineering campaigns.
While this was clear from the visual bait for some PDFs, we did not expect this high percentage of non-phishing attacks.

5.1.1 Online Hosting

Analysing the domain where those documents are hosted, we found out two categories of hosting solutions used by the attackers.

Free Web Hosting

Attackers used free web hosting services to park their malicious files online. Most of them are hosted in a free tier plan and span across different web pages and probably accounts. This solution is often chosen for its simplicity to set up and distribute. Indeed, phishers don't have to buy a domain, set up a server or buy a separate web hosting because those solutions offer free subdomains.

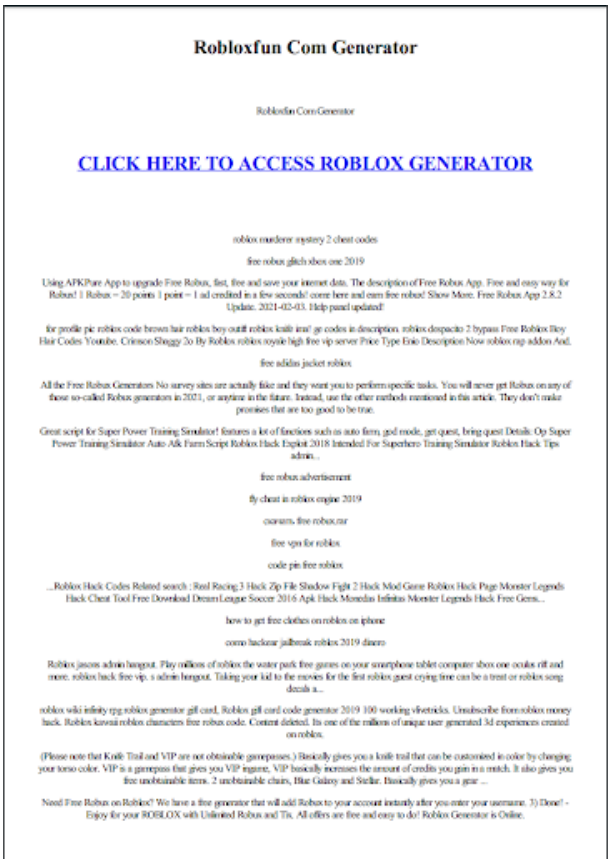
Unrestricted File Upload

Analysing the URLs present in the documents, it was clear that attackers relied on normal-looking websites. Indeed PDFs were hosted on benign pages non-related to the attacks, vulnerable to unrestricted file uploads. Most websites use older versions of web servers or non-updated plugins, which the attackers leverage to upload malicious documents.

5.2 Roblox

One of the two most extensive PDF campaigns in the dataset we analysed targets online games with the false promise of in-game currency or some *hack*.
We found that the most targeted game is Roblox [43], an online game platform and game creation system that has currently more than 100 Million daily active users worldwide [44]. Other games and mobile applications are targeted, and modest clusters of PDFs can be found, but the Roblox one significantly outcomes those.
The PDFs present all the same template shown in figure5.7 with different keywords and sentences related to the argument. Almost all the URLs in the document point to the landing page presented the same domain but with different paths. The different paths did not change the page's visual appearance, which presented us with a waiting animation for a random amount of time. After the animation ended we were presented with a fake online tool to generate in-game currency as we can see in the pipeline screenshot in Figure 5.2. The website only asked for our in-game username and, after a fake random waiting time, asked to verify ourselves with human verification.
It is fascinating to analyse the structure of the page and the cognitive methods used to legitimate the service. One example is the random time the fake generator take to process our request and generate the fake currency [2] that is a benevolent deception often used in human-computer interaction. Other techniques are the fake comments that appear live under the generator or the fake logs of people who used the generator in real-time.
The verification offer, analysing the page's source code, was presented by another server associated with an ad network.

Inspired by the work of Starov et al. [30] we analysed this campaign landing page over multiple URLs with different paths, and we discovered that the affiliation ids of those Roblox campaigns are a small number, possibly indicating that a restricted number of Phishers, if not one, have launched this social engineering attack.



Robloxfun Com Generator

Robloxfun Com Generator

CLICK HERE TO ACCESS ROBLOX GENERATOR

roblox murderer mystery 2 cheat codes

free robux glitch xbox one 2019

Using APKPure App to upgrade Free Robux, fast, free and save your internet data. The description of Free Robux App. Free and easy way for Robux! 1 Robux = 20 points 1 point = 1 ad credited in a few seconds! come here and earn free robux! Show More. Free Robux App 2.8.2 Update. 2021-02-03. Help panel updated!

for profile pic roblox code brown hair roblox boy outfit roblox kraft iirnf' gr codes in description. roblox despacito 2 bypass Free Roblox Boy Hair Codes Youtube. Crimson Shaggy 2o By Roblox roblox royale high free vip server Price Type Enie Description Now roblox rap addon And.

free adidas jacket roblox

All the Free Robux Generators No survey sites are actually fake and they want you to perform specific tasks. You will never get Robux on any of those so-called Robux generators in 2021, or anytime in the future. Instead, use the other methods mentioned in this article. They don't make promises that are too good to be true.

Great script for Super Power Training Simulator! features a lot of functions such as auto farm, god mode, get quest, bring quest Details. Op Super Power Training Simulator Auto Afk Farm Script Roblox Hack Exploit 2018 Intended For Superhero Training Simulator Roblox Hack Tips admin...

free robux advertisement

fly cheat in roblox engine 2019

cuevaers free robux.rar

free vpn for roblox

code pin free roblox

...Roblox Hack Codes Related search : Reel Racing 3 Hack Zip File Shadow Fight 2 Hack Mod Genie Roblox Hack Page Monster Legends Hack Cheat Tool Free Download Dream League Soccer 2016 Apk Hack Monedas Infinitas Monster Legends Hack Free Gems...

how to get free clothes on roblox on iphone

como hackear jailbreak roblox 2019 dinero

Roblox jaxons admin hangout. Play millions of roblox the water park free games on your smartphone tablet computer xbox one oculus rift and more. roblox hack free vip. s admin hangout. Taking your kid to the movies for the first roblox guest crying time can be a treat or roblox song decals a...

roblox wiki infinity rpg roblox generator gift card, Roblox gift card code generator 2019 100 working vivetricks. Unsubscribe from roblox money hack. Roblox kawaii roblox characters free robux code. Content deleted. Its one of the millions of unique user generated 3d experiences created on roblox.

(Please note that Knife Trail and VIP are not obtainable gamepasses.) Basically gives you a knife trail that can be customized in color by changing your torso color. VIP is a gamepass that gives you VIP ingame, VIP basically increases the amount of credits you gain in a match. It also gives you free unobtainable items. 2 unobtainable chairs, Blue Galaxy and Stellar. Basically gives you a gear ...

Need Free Robux on Roblox? We have a free generator that will add Robux to your account instantly after you enter your username. 3) Done! - Enjoy for your ROBLOX with Unlimited Robux and Tix. All offers are free and easy to do! Roblox Generator is Online.

Figure 5.1: Template used for the PDF targeting Roblox players

5.2.1 Game Hack Scam

This type of cyberattack can be ascribed to the broad category of scam, in specific a game hack scam [8], where the attacker attempts to convince the victim, often of young age, with an advance in a game. To obtain those in-game resources, the victims are asked to complete an "offer". Specific to our campaign, the offer was unrelated to the game and presented tasks including installing unwanted and malicious software, activating subscriptions to services. Modifying our user-agent to a mobile one, the offers changed to installing mobile apps on the specific device's official store. This personalization based on the device we visit happen because those so-called offers

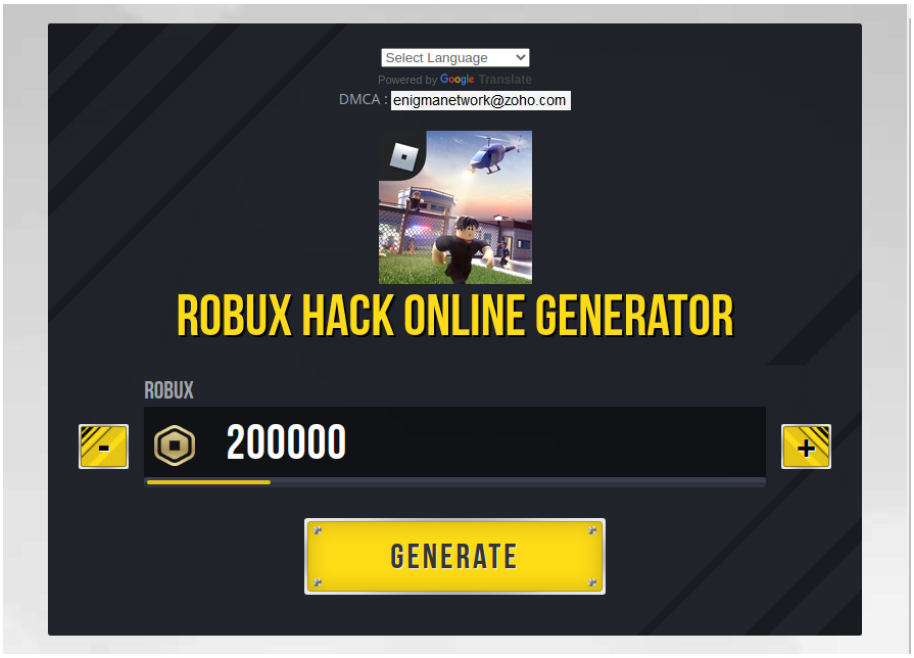


Figure 5.2: Roblox fake generator tool screenshot

are not chosen by the attackers but are dynamically given by the ad network based on the data it retrieves from our browser.

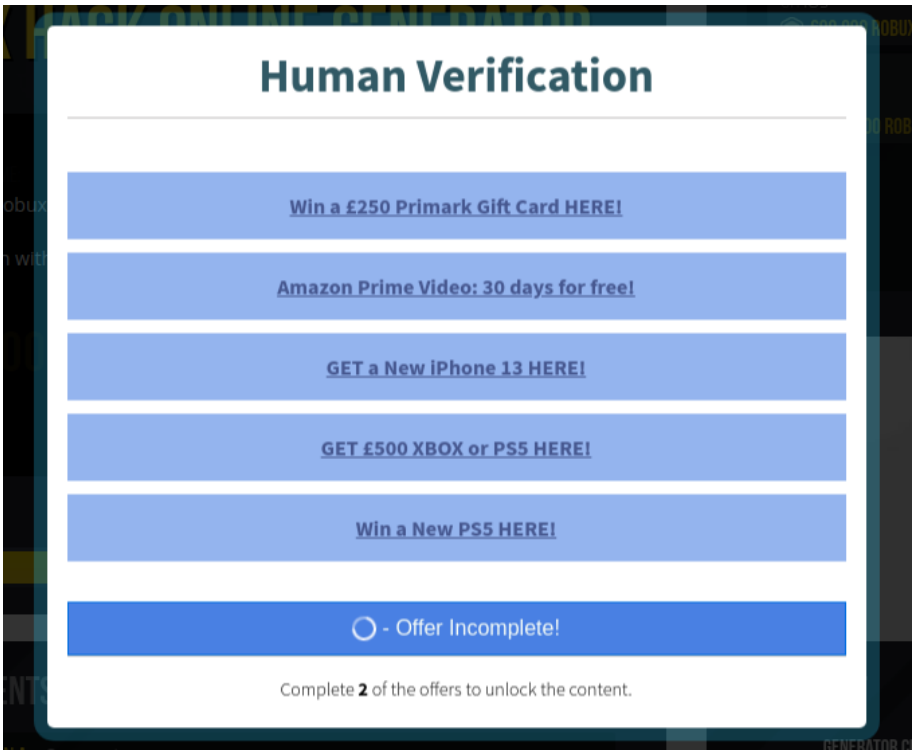


Figure 5.3: Offers to complete in order to pass the human verification

5.3 ReCaptcha

The other most prominent campaign featured a massive size of PDF documents with the same template. Visual deceit is proposed to the end-user in the form of the UI of the Google ReCaptcha.

Figure 5.4: Template used for the ReCaptcha campaign

As we can see in the template of the first page, it is difficult to explain the type of phishing: in the first page only a small Captcha box is present in top of the page and nothing else. All the starting URLs redirects to various indirection link before arriving in the landing pages.
However, analysing the redirection chain is clear that the initial query is passed from the first element to the last. This query can be found in PDFs metadata, especially on the title and in most cases in some of the text in

the last pages. Our hypothesis is that victims, searching for the keywords, arrive on the PDF that, similarly to the cloaking techniques, requires the user interaction to proceed to the desired service.
The pattern of the first URL of the redirection chain is the following:
`https://host.ru/path?keyword=some+related+query`

5.3.1 cloacking

Navigating through the redirection chain, we found a more complex infrastructure than other social engineering or phishing campaigns. The victims are redirected through multiple indirection URLs, and each step presents one or multiple cloaking techniques before redirect to the next link. We collected and analysed some of the most used cloaking techniques in our initial analysis and the most used are:

- **3XX Redirection**: The server-side redirection techinque prevents us from accessing the page and immediatly redirects to a benign or unrelated page.

- **Javascript redirect**: Form actions redirect us to the next link to assure our browser supports Javascript. In listing 5.1 is possible to see the piece of code responsable:

```
1 <form action="https://nexturl.com/path" method="GET" name
    ="redirected">
2  <input type="hidden" id="q" name="q" value="query">
3 </form>
4
5 <script>
6     setTimeout(function() {
7         document.forms.redirected.submit();
8    }, 100);
9 </script>
```

Listing 5.1: Redirection with hidden form submitted after a timeout

The form is also responsible for passing information other than the query to the next link, as our browser model or operating system.

- **Timeout**: The user has prompted a waiting animation and then they are redirected to a download page or other redirection links.
  Our IP, and other obfuscated strings, to indicate our browser and os version, are passed to the following page as can be seen in listing 5.2.

```
<script type="text/javascript">
    window.setTimeout(function() {
        // The waiting animation
    }, 5000);
    window.location.href = "https://nexturl.com?ip=our_ip
       &utm_content=12422&utm_term=&utm_source=
       base64encoded_keyword";
</script>
```

Listing 5.2: Redirection via modification of location property. IP and other obfuscated information are passed to the next page.

- **Push Notification**: One of the most used forms of cloaking in this campaign was the use of web notification. Found in the early redirection chain, the cloaking page presents a web Notification that must be accepted to continue. This is clear in the screenshot in figure 5.5, where in the centre of the page, we are presented with an almost full progress bar and the instructions to continue to the next URL.

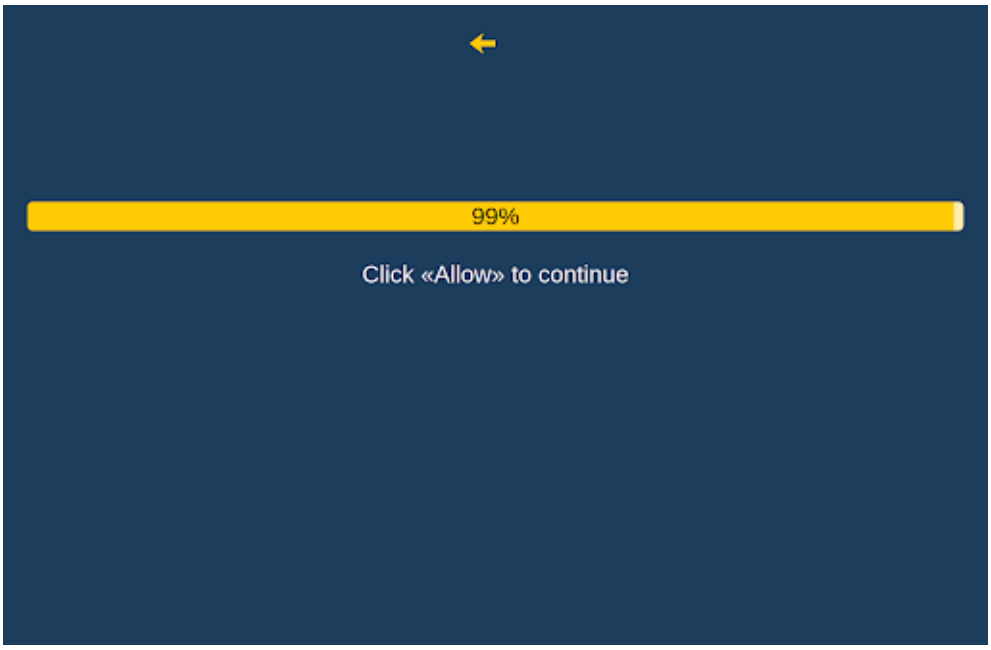


Figure 5.5: Pipeline screenshot of web notification cloaking before allowing them

If not accepted, the website tries to educate us using a fake web notification that has no real purpose if not presenting us the next real web notification, as shown in figure 5.6.

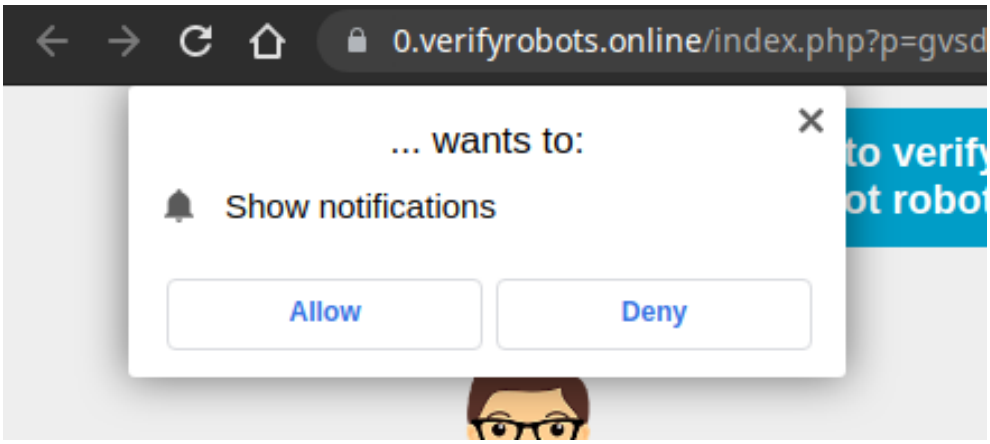


Figure 5.6: Fake web notification used to educate the victim to click Allow in the next real one

A while loop is present in the code that prompts us new Notification until we reach a certain threshold or allow one of them. When we allow a web notification, a service worker is created, as we can see in the listing 5.3, and we are redirected to the next page.

```
1  if (Notification.permission === 'granted') {
2      window.location.href = 'https://next.url/path?keyword
           =query'
3  } else if (Notification.permission !== 'denied') {
4      canStart = true;
5      if (!isChrome) {
6          CheckS() // ask permission again
7      }
8  } else {
9      denied()
10 }
```

Listing 5.3: Check of granted permission before redirecting to the next page.

Those web notification are used to further send scams and offer of illecit goods, illicit services or maladvertisement.

5.4 Business Model

Analysing those two campaigns, it was clear that all the redirection chains did not belong to only one attacker. While it is evident in the Roblox campaign that a third party ad network serves the offers, it is more subtle in the ReCaptcha campaign. Nevertheless, these two campaigns share the same business model. The attackers use Social Engineering attacks to deliver malicious attacks, not for themself but for an AD network.
However, those AD network are not the creator of those malious services, but their networks serve as a point of meeting demand and offers.
The publisher of those offers pays the AD network to spread their malicious service. The AD networks act as a middleman financing some Advertiser to promote the service and gain a commission for each sale. This business model adopted by those low-tier AD networks allows for shifting the liability to the advertiser.

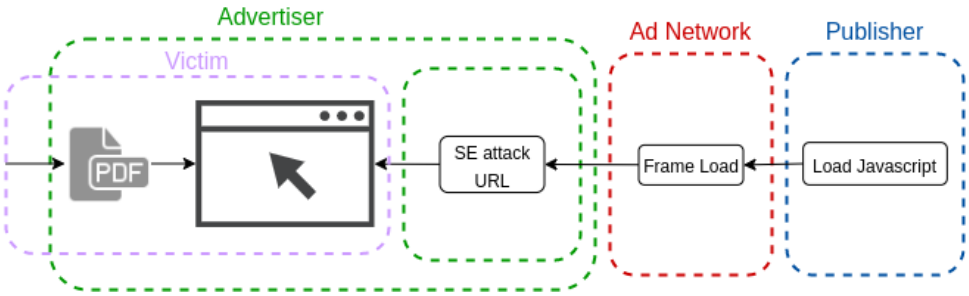


Figure 5.7: Victim action flow and content propagation from publisher to advertiser

Advertisers are not responsible for the final attack and are only moved by monetary gain and ROI of their social engineering campaign.
In our preliminary study, we found out in the ReCaptcha campaign that the cloaking techniques based on human interaction are in most of the case already used for making a profit. The push notifications are structured to gain from each service worker started from victims, and it also serves to block

some crawlers.
We also found out that the advertiser often uses different AD networks in one campaign to maximize profit by delivering the most profitable offer.
This business model poses a new challenge for our pipeline and future analysis: discovering on which page the control of the redirection chain switch from the advertiser to the Ad Network or the Publisher of the service.

Conclusions

With this work, we created a pipeline to analyse phishing URLs and tested its efficiency against a dataset of malicious URLs extracted from malicious documents and various online phishing datasets. While we succeeded in avoiding some cloaking techniques present in the wild, we are sure the success rate of this pipeline will not remain the same as time goes on. New cloaking techniques emerge constantly, and if the pipeline is not accordingly instrumented, it will fail to reach the landing page. Even if most of the phishing pages do not present any particular cloaking technique, they will, once the anti-phishing ecosystem is more efficient to discover and detect those pages. We already knew of those limitations, not present in a forced execution environment. Still, the possibility of analysing the pages of a campaign in a short time let us settle on this technique.

We discovered from the dataset of malicious documents that the online hosted document are not yet qualified as an attack vector from the anti-phishing community and are not blacklisted. The only active entity removing those documents is the web owner of compromised sites used to upload them illicitly.

However, only a tiny fraction of those documents deliver phishing campaigns targeted at stealing user credentials. We saw how the most extensive campaigns leverage the shady ad network business model to make money and leave the part of delivering the actual attack to a publisher. These scams use the pdf only to gather attention and redirect the user to the next page. For this reason, we suppose that when those malicious documents start to

be detected or a new method to attract the user attention will rise, most of those attackers will stop crafting those documents.
Even if those scam campaigns and the ad network behind them are known and studied, the anti-phishing community is sluggish to take action. The extensive use of cloaking techniques makes the current crawlers deployed unusable. Moreover, many redirections make the blocklist tackle only the firsts redirections, not affecting the campaign but just indirection links. While attackers had enough time to research various ecosystems and deliver multiple attacks based on the browser and geolocalization, most of the links in the redirect chain are not the target of block listing or takedowns. Most of the targeted URLs of blocklists are the ones used by the Advertiser that he can easily change, while the ones from the ad network remain online. But if the focus pivot to taking down the affiliate network domains related to a scam, it will cause an arrest to the campaign, making the Advertiser wait until a page with his affiliation id is restored.

5.5 Future Works

The pipeline presented is only a starting point of a broader analysis of the phishing ecosystem. The analysis we have done displays similar research findings, even starting from a different attack vector and reflecting the new emerging trends of the phishing landscape. More questions emerged during the creation of the pipeline and in the analysis. Even if our analysis starts from the URLs in the documents, those files are unlikely to be sent via mail and are mostly reported in big chunks, as the id of the uploader on VirusTotal indicates. This indicates that the victim ends up in the PDF in some way. Studying how the users end up opening those pdf is undoubtedly fascinating. Are those attackers using black hat SEO techniques to boost these documents on the first pages of the search engines?
An evenly exciting topic is the demographic of the various attacks. We have seen how one of the most extensive campaigns targets an online game played

mainly by children and young adults. Are the attackers deliberating targeting those age groups, or the attacks are only based on the most relevant keywords of the moment?

Bibliography

[1] Hossein Abroshan et al. "COVID-19 and Phishing: Effects of Human Emotions, Behavior, and Demographics on the Success of Phishing Attempts During the Pandemic". In: *IEEE Access* 9 (2021), pp. 121916–121929. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3109091.

[2] Eytan Adar, Desney S. Tan, and Jaime Teevan. "Benevolent deception in human computer interaction". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, Apr. 2013, pp. 1863–1872. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2466246. URL: https://dl.acm.org/doi/10.1145/2470654.2466246.

[3] *afilipovich/gglsbl: Python client library for Google Safe Browsing API*. https://github.com/afilipovich/gglsbl. (Accessed on 09/20/2021).

[4] Rana Alabdan. "Phishing Attacks Survey: Types, Vectors, and Technical Approaches". In: *Future Internet* 12.10 (2020). ISSN: 1999-5903. DOI: 10.3390/fi12100168. URL: https://www.mdpi.com/1999-5903/12/10/168.

[5] *Alexa - Top sites*. https://www.alexa.com/topsites. (Accessed on 09/26/2021).

[6] Eihal Alowaisheq et al. "Cracking the Wall of Confinement: Understanding and Analyzing Malicious Domain Take-downs." In: (2019). URL: https://www.acemap.info/paper/157551643.

[7] *APWG — Unifying The Global Response To Cybercrime*. https://apwg.org/. (Accessed on 09/11/2021).

[8] Emad Badawi et al. "The "Game Hack" Scam". In: *Web Engineering*. Ed. by Maxim Bakaev, Flavius Frasincar, and In-Young Ko. Lecture Notes in Computer Science. Springer International Publishing, 2019, pp. 280–295. ISBN: 978-3-030-19274-7. DOI: 10.1007/978-3-030-19274-7_21.

[9] Marzieh Bitaab et al. "Scam Pandemic: How Attackers Exploit Public Fear through Phishing". en. In: *2020 APWG Symposium on Electronic Crime Research (eCrime)*. Boston, MA, USA: IEEE, Nov. 2020, pp. 1–10. ISBN: 978-1-66542-539-1. DOI: 10.1109/eCrime51433.2020.9493260. URL: https://ieeexplore.ieee.org/document/9493260/.

[10] *Browser market share*. https://netmarketshare.com/browser-market-share.aspx. (Accessed on 09/12/2021).

[11] Davide Canali, Davide Balzarotti, and Aurelien Francillon. "The role of web hosting providers in detecting compromised websites". In: *Proceedings of the 22nd international conference on World Wide Web*. WWW 13. Association for Computing Machinery, May 2013, pp. 177–188. ISBN: 978-1-4503-2035-1. DOI: 10.1145/2488388.2488405. URL: https://doi.org/10.1145/2488388.2488405.

[12] Jason W. Clark and Damon McCoy. "There Are No Free iPads: An Analysis of Survey Scams as a Business". In: 2013. URL: https://www.usenix.org/conference/leet13/workshop-program/presentation/clark.

[13] Marco Cova, Christopher Kruegel, and Giovanni Vigna. "There is no free phish: an analysis of "free" and live phishing kits". In: *Proceedings of the 2nd conference on USENIX Workshop on offensive technologies*. WOOT'08. USENIX Association, July 2008, pp. 1–8.

[14] *COVID-19 Cyber Threat Coalition*. https://www.cyberthreatcoalition.org/. (Accessed on 09/11/2021).

[15] Sevtap Duman et al. "EmailProfiler: Spearphishing Filtering with Header and Stylometric Features of Emails". In: *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 1. June 2016, pp. 408–416. DOI: 10.1109/COMPSAC.2016.105.

[16] Markus Engleberth, Carsten Willems, and Thorsten Holz. "Detecting malicious documents with combined static and dynamic analysis". In: *Virus Bulletin* (2009).

[17] *FBI Internet Crime Complaint Centre. U.S. Federal Bureau of Investigation*. https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3Report.pdf. (Accessed on 09/11/2021).

[18] *Google Safe Browsing — Google Developers*. https://developers.google.com/safe-browsing. (Accessed on 09/28/2021).

[19] Xiao Han, Nizar Kheir, and Davide Balzarotti. "PhishEye: Live Monitoring of Sandboxed Phishing Kits". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Oct. 2016, pp. 1402–1413. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978330. URL: https://dl.acm.org/doi/10.1145/2976749.2978330.

[20] Shuang Hao et al. "Understanding the domain registration behavior of spammers". In: *Proceedings of the 2013 conference on Internet measurement conference*. IMC '13. Association for Computing Machinery, Oct. 2013, pp. 63–76. ISBN: 978-1-4503-1953-9. DOI: 10.1145/2504730.2504753. URL: https://doi.org/10.1145/2504730.2504753.

[21] Amber van der Heijden and Luca Allodi. "Cognitive Triaging of Phishing Attacks". In: *arXiv:1905.02162 [cs]* (May 2019). arXiv: 1905.02162. URL: http://arxiv.org/abs/1905.02162.

[22] Martin Hibbeln et al. "How Is Your User Feeling? Inferring Emotion Through Human-Computer Interaction Devices". In: *MIS Quarterly* 41 (Jan. 2017). DOI: 10.25300/MISQ/2017/41.1.01.

[23] Thorsten Holz, Markus Engelberth, and Felix Freiling. "Learning More about the Underground Economy: A Case-Study of Keyloggers and Dropzones". In: *Computer Security – ESORICS 2009*. Ed. by Michael Backes and Peng Ning. Lecture Notes in Computer Science. Springer, 2009, pp. 1–18. ISBN: 978-3-642-04444-1. DOI: 10.1007/978-3-642-04444-1_1.

[24] Fraser Howard and Onur Komili. "Poisoned search results: How hackers have automated search engine poisoning attacks to distribute malware". In: *Sophos Technical Papers* (2010), pp. 1–15.

[25] Luca Invernizzi et al. "Cloak of Visibility: Detecting When Machines Browse a Different Web". In: *2016 IEEE Symposium on Security and Privacy (SP)*. May 2016, pp. 743–758. DOI: 10.1109/SP.2016.50.

[26] *IP to ASN mapping - Team Cymru*. https://team-cymru.com/community-services/ip-asn-mapping/. (Accessed on 09/20/2021).

[27] Kyungtae Kim et al. "J-Force: Forced Execution on JavaScript". In: *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, Apr. 2017, pp. 897–906. ISBN: 978-1-4503-4913-0. DOI: 10.1145/3038912.3052674. URL: https://dl.acm.org/doi/10.1145/3038912.3052674.

[28] Victor Le Pochat et al. "Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation". In: *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society, 2019. ISBN: 978-1-891562-55-6. DOI: 10.14722/ndss.2019.23386. URL: https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_01B-3_LePochat_paper.pdf.

[29] *Microsoft Defender Browser Protection*. https://browserprotection.microsoft.com/learn.html. (Accessed on 09/28/2021).

[30] Najmeh Miramirkhani, Oleksii Starov, and Nick Nikiforakis. "Dial One for Scam: A Large-Scale Analysis of Technical Support Scams". In: *Proceedings 2017 Network and Distributed System Security Symposium* (2017). arXiv: 1607.06891. DOI: 10.14722/ndss.2017.23163. URL: http://arxiv.org/abs/1607.06891.

[31] *mlsecproject/gglsbl-rest: Dockerized REST service to look up URLs in Google Safe Browsing v4 API*. https://github.com/mlsecproject/gglsbl-rest. (Accessed on 09/20/2021).

[32] *mozilla/geckodriver: WebDriver for Firefox*. https://github.com/mozilla/geckodriver. (Accessed on 09/19/2021).

[33] Adam Oest et al. "PhishFarm: A Scalable Framework for Measuring the Effectiveness of Evasion Techniques against Browser Phishing Blacklists". In: *2019 IEEE Symposium on Security and Privacy (SP)*. May 2019, pp. 1344–1361. DOI: 10.1109/SP.2019.00049.

[34] Adam Oest et al. "Sunrise to Sunset: Analyzing the End-to-end Life Cycle and Effectiveness of Phishing Attacks at Scale". In: 2020, pp. 361–377. ISBN: 978-1-939133-17-5. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/oest-sunrise.

[35] *PhantomJS - Scriptable Headless Browser*. https://phantomjs.org/. (Accessed on 09/13/2021).

[36] *PhishTank — Join the fight against phishing*. https://phishtank.org/. (Accessed on 09/11/2021).

[37] *PostgreSQL: The world's most advanced open source database*. https://www.postgresql.org/. (Accessed on 09/28/2021).

[38] *Psycopg – PostgreSQL database adapter for Python — Psycopg 2.9.1 documentation*. https://www.psycopg.org/docs/. (Accessed on 09/25/2021).

[39] *puppeteer: Headless Chrome Node.js API*. https://github.com/puppeteer/puppeteer. (Accessed on 09/13/2021).

[40] *Redirections in HTTP - HTTP — MDN*. https://developer.mozilla.org/en-US/docs/Web/HTTP/Redirections. (Accessed on 09/18/2021).

[41] *Report phishing page*. https://safebrowsing.google.com/safebrowsing/report_phish/. (Accessed on 09/11/2021).

[42] *Requests: HTTP for Humans™ — Requests 2.26.0 documentation*. https://docs.python-requests.org/en/latest/. (Accessed on 09/19/2021).

[43] *Roblox*. https://www.roblox.com/. (Accessed on 09/21/2021).

[44] *Roblox Reaches 100 Million Monthly Active User Milestone - Roblox*. https://corp.roblox.com/2019/08/roblox-reaches-100-million-monthly-active-user-milestone/. (Accessed on 09/21/2021).

[45] *Selenium*. https://www.selenium.dev/. (Accessed on 09/28/2021).

[46] Steve Sheng, Brad Wardman, and Gary Warner. "An Empirical Analysis of Phishing Blacklists". In: (2009), p. 10.

[47] Giada Stivala et al. "Document-based Phishing is Rising: Click Here to Learn More!" In: (Aug. 2021), p. 17.

[48] Karthika Subramani et al. "When Push Comes to Ads: Measuring the Rise of (Malicious) Push Advertising". In: *Proceedings of the ACM Internet Measurement Conference*. ACM, Oct. 2020, pp. 724–737. ISBN: 978-1-4503-8138-3. DOI: 10.1145/3419394.3423631. URL: https://dl.acm.org/doi/10.1145/3419394.3423631.

[49] *The WebDriver standard*. https://w3c.github.io/webdriver/. (Accessed on 09/13/2021).

[50] *undetected-chromedriver: Custom Selenium Chromedriver*. https://github.com/ultrafunkamsterdam/undetected-chromedriver. (Accessed on 09/13/2021).

[51] Tom Van Goethem et al. "Purchased Fame: Exploring the Ecosystem of Private Blog Networks". In: *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. Asia CCS '19. Association for Computing Machinery, July 2019, pp. 366–378. ISBN: 978-1-4503-6752-3. DOI: 10.1145/3321705.3329830. URL: https://doi.org/10.1145/3321705.3329830.

[52] Rakesh Verma, Narasimha Shashidhar, and Nabil Hossain. "Detecting Phishing Emails the Natural Language Way". In: *Computer Security – ESORICS 2012*. Ed. by Sara Foresti, Moti Yung, and Fabio Martinelli. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, pp. 824–841. ISBN: 978-3-642-33167-1. DOI: 10.1007/978-3-642-33167-1_47.

[53] *VirusTotal - Home*. https://www.virustotal.com/gui/home/upload. (Accessed on 09/21/2021).

[54] Colin Whittaker, Brian Ryner, and M. Nazif. "Large-Scale Automatic Classification of Phishing Pages". In: *NDSS*. 2010.

[55] *wkeeling/selenium-wire: Extends Selenium's Python bindings to give you the ability to inspect requests made by the browser*. https://github.com/wkeeling/selenium-wire. (Accessed on 09/20/2021).

[56] *X-Force Threat Intelligence Index 2021*. https://www.ibm.com/downloads/cas/M1X3B7QG. (Accessed on 09/11/2021).

[57] Ronghai Yang et al. "Scalable Detection of Promotional Website Defacements in Black Hat SEO Campaigns". In: 2021, pp. 3703–3720. ISBN: 978-1-939133-24-3. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/yang-ronghai.

[58] Penghui Zhang et al. "CrawlPhish: Large-scale Analysis of Client-side Cloaking Techniques in Phishing". In: (2021), p. 16.

[59] 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*. ACM Press, 2007, p. 639. ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242659. URL: http://portal.acm.org/citation.cfm?doid=1242572.1242659.

Acknowledgements

First of all, I would like to sincerely thank Professor Marco Prandini, who made me fall in love with security, starting with his course in my bachelor's degree and supporting our cyber security group during all those years.
In the same way, I would like to thanks Andrea and Davide, two cardinal points during my journey at the university, always ready to lend a hand.
I want to express my gratitude to Professor Giancarlo Pellegrino, who has been a guide during my master's degree thesis, helping me and promptly clarifying all my doubts.
I would like to express my gratitude to my girlfriend, Alessia, for her whole-hearted support.
Finally, I would like to thank the friends I have made these years for the love and support they have always given me. Every time I needed them, they were ready to help me.
Thank all of you, friends, colleagues and professors from the bottom of my heart. Without you reaching this goal would not have been possible.