

ThePhish: an Automated Open-Source Phishing Email Analysis Platform

Emanuele Galdi¹, Gaetano Perrone^{1,*} and Simon Pietro Romano¹

¹Department of Information Technology and Electrical Engineering, University of Naples Federico II, Naples, Italy

Abstract

Phishing, and specifically phishing emails, are becoming the most pervasive cyberattack and the most widely used infection vector. As a consequence, SOCs, CERTs, and CSIRTs are overwhelmed by the number of emails that they need to analyze every day, with the majority of them being false positives. The manual email analysis is a huge waste of effort. Thus, finding approaches to the full or at least partially automated analysis is crucial. This work aims to present ThePhish, an open-source phishing email analysis platform capable of automating the entire email analysis process, starting from the extraction of the observables from the header and the body of the email to the elaboration of a verdict, which is final in most cases. The framework leverages the effectiveness of important open-source projects, namely, MISP, TheHive and Cortex, to filter out a significant number of false positives. If ThePhish is sure about the maliciousness of the email, it scores it as “malicious”. However, an email sometimes can only be considered suspicious and need further analysis. In these cases, ThePhish offers several features that allow analysts to speed up the analysis process and obtain further details on the suspicious emails.

Keywords

Phishing, Email, Cybersecurity, Malware, Automation

1. Introduction

The number of cyberattacks is growing faster and faster, and cyberattacks have been rated seventh and eighth in the ranking of the top 10 risks of 2020 in terms of likelihood and impact respectively [1]. This increment has surely been exacerbated by the COVID-19 outbreak, which on the one hand has triggered a massive digital transformation of all the companies and organizations around the world, but on the other hand has led to a bigger attack surface for the attackers to exploit. A cybersecurity incident can cause a business a lot of damage, such as financial losses, loss of productivity, reputation damage, legal liability, or business continuity problems. Among the sheer number of typologies of cyberattacks, the ones that are becoming the most pervasive are those involving *social engineering* [2]. Social engineering is the psychological manipulation of people into performing actions or divulging confidential information [3], so it does not depend on the technological measures used by an organization to protect its assets, but it is based on human error and, as it is said by Bruce Schneier in [4]:

ITASEC'22: Italian Conference on Cybersecurity, June 20–23, 2022, Rome, Italy

*Corresponding author.


✉ emanuele.galdi@secsi.io (E. Galdi); gaetano.perrone@unina.it (G. Perrone); spromano@unina.it (S. P. Romano)

🌐 <https://github.com/emalderson/> (E. Galdi); <https://github.com/giper45/> (G. Perrone);

<https://www.docenti.unina.it/simonpietro.romano/> (S. P. Romano)

🆔 0000-0002-1607-1095 (E. Galdi); 0000-0001-8238-6426 (G. Perrone); 0000-0002-5876-0382 (S. P. Romano)

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

“People often represent the weakest link in the security chain and are chronically responsible for the failure of security systems”. Social engineering itself is a broad category of attacks, of which the best known is definitely *phishing*. It consists of an attacker persuading a victim to take actions that make it possible for the attacker himself to access the victim’s device, account, or personal information, for example, by pretending to be a person or organization the victim trusts [5]. Although the methods to carry out a phishing attack are diverse, when one thinks about phishing, the first thing that comes to his mind is surely a situation where the victim receives an email that tries to convince him to click on a malicious link or downloads a malicious attachment. The problem is that while some phishing emails are fairly simple to spot for someone with a trained eye, some are definitely not. With the number of cyberattacks growing so rapidly, another problem arises: SOCs (Security Operations Centers), CERTs (Computer Emergency Response Teams), and CSIRTs (Computer Security Incident Response Teams) are struggling to face all the alerts that require their attention. Moreover, the number of accurate alerts is only a small percentage, with the rest being false positives. This leads to a massive waste of effort by the analysts who have to manually take several actions to filter out the false positives instead of focusing on only the alerts. This problem is a natural consequence of a lack of automation, which is critical for a SOC that wants to do its job in a more efficient way [6]. Clearly, the concept of automation inside a SOC can be also applied to alerts related to potential phishing emails. In fact, analyzing an email is a time-consuming task that can take hours, so the full or at least partial automation of the analysis to filter out false positives or simplify the analyst’s job reveals to be crucial. In this work, we present ThePhish, an open-source phishing email analysis platform that makes analysts’ activities faster and more efficient. The remainder of this paper is structured as follows. In Section 2 we explore several open-source similar solutions and show their limitations. Section 3 describes the Cyber Threat Intelligence concept and introduces the frameworks used by ThePhish, i.e., MISP, TheHive and Cortex. We also show that it is possible to integrate these frameworks to benefit from all their features, as we did in ThePhish. Sections 4 and 5 describe ThePhish modules and explain how it is possible to integrate the framework into security analysis processes. The last section shows possible future evolutions of the platform.

2. State of the art

Tools that help the analyst do his job when it comes to phishing email analysis can be implemented following different approaches and provide different automation levels to the analysis. In this section, some already existent tools will be presented, with a focus on free online tools and open-source projects that are available on GitHub, leaving out the various commercial and paid products, which are usually implemented as comprehensive security solutions that are tailored on the needs of a particular organization with the objective of offering protection against many types of attacks, phishing included, and that also provide automated ways to respond to such attacks if necessary. That is because the aim of this work is to implement an open-source solution that is specifically focused on being an aid to an analyst that is dealing with phishing emails and anyone can use that for free.

2.1. MxToolbox

MxToolbox supports global Internet operations by providing free, fast, and accurate network diagnostic and lookup tools [7]. For example, it is possible to test SMTP, HTTP/HTTPS, and DNS servers, perform different types of DNS queries, or obtain information about domains and IP addresses. What is more is that it has some tools that can be helpful when analyzing an email. In fact, it is possible to check if a domain has correctly set-up SPF, DKIM, and DMARC records or if an IP address is blacklisted, but the most useful tools are certainly the *Email Header Analyzer* and the *Spam Analyzer*. The former makes it possible to submit the header of an email to get it in a human-readable format and with the possible problems highlighted, like a failed DMARC authentication. The latter allows submitting a full email (header and body) and analyzing it. It uses the SpamAssassin [8] software to analyze a message and return a spam score. SpamAssassin is an intelligent email filter that uses a diverse range of tests to identify spam emails. These tests are applied to both the header and the content of the email to classify it using advanced statistical methods.

2.2. EmailAnalyzer

EmailAnalyzer is an open-source command-line tool written in Python that is available on GitHub. It is able to extract data such as email addresses, IP addresses, URLs and attachments from an email provided as an EML or MSG file [9]. It supports the expansion of shortened URLs and also allows scanning URLs and hashes of attached files with VirusTotal [10]. Furthermore, it allows the use of a whitelist through a configuration file that accepts regular expressions. At the end of the analysis this tool creates the `extracted-attachments` folder, which contains the attachments found in the email, and different files containing:

- the extracted email addresses (i.e., `emails.txt`);
- the extracted IP addresses (i.e., `ips.txt`);
- the extracted URLs (i.e., `urls.txt`);
- the extracted *Received* lines (i.e., `received.txt`).

If the command-line option `-vt` is used, the tool will scan the URLs that are located in the `urls.txt` file and the files that are located in the `extracted-attachments` folder with VirusTotal. If a file is found as malicious, it will be renamed by appending the word `_malware` at its end, while if a URL is found as malicious, it will be added to a file named `malware_urls.txt`. Obviously, an API key for VirusTotal is needed to use this feature. However, it has been noted that the extraction of the observables it makes through the use of regular expressions is not 100% precise. For instance, those regular expressions may fail and extract URLs that end with an HTML tag, or erroneous header lines (e.g., *Received-SPF* among the *Received* ones). Moreover, email *Message-ID* header lines may be mistaken for the email addresses as well.

2.3. Sooty

Sooty is an open-source command-line tool written in Python that is available on GitHub. It aims to aid SOC analysts with automating part of their workflow. One of the goals of Sooty is to

perform as many of the routine checks as possible, allowing the analyst to spend more time on deeper analysis within the same time frame [11]. It offers the following features that allows to:

- transform a URL in a way that prevents a user from clicking on it by mistake. This is particularly useful when dealing with malicious URLs.
- choose one of many different transformations for URLs or strings, such as URL unshortening or base64 decoding).
- perform reputation checks for IP addresses, email addresses or URLs on VirusTotal, AbuseIPDB [12] and Tor exit nodes
- perform DNS, reverse DNS and WHOIS lookups.
- generate a hash for a string or a text and check it for known malicious activity against VirusTotal.
- extract IP addresses, email addresses, URLs, and some basic header lines from an email provided as an MSG file format (EML is currently not supported). Then, it can automatically check email addresses against emailrep.io [13] for known malicious activity and also against HaveIBeenPwned [14] to see if those addresses are included in a past data breach. Moreover, it allows submitting URLs to PhishTank [15] to see if they are related to phishing. Lastly, it is possible to create a dynamic response template based on the result of the email analysis.
- get a reputation report from urlscan.io [16] for a URL.

Obviously, most of the external tools used to analyze the information provided by the analyst require an API key. It should be noted that the tool searches for those pieces of information only inside the body of the email, ignoring the header lines. Also, it does not extract all the header lines but only the ones usually displayed by the mail client application. The only email address that is automatically analyzed is the one found in the *From* field of the email. All the other pieces of information that have to be analyzed must be provided manually to the tool.

2.4. IsThisLegit

IsThisLegit is an open-source tool that is available on GitHub [17]. It makes it easy to receive, analyze and respond to phishing reports and consists of two parts:

- *Dashboard*: a Google App Engine application and is the analyst's window into phishing reports from the organization. Analysts can use the dashboard to view, categorize and respond to phishing emails.
- *Chrome extension*: a button in Gmail application that makes it easier for users to report phishing emails to the dashboard.

Once an email is submitted, a report is shown in a table on the initial dashboard page. An analyst can view a particular report and perform the following actions:

- Classify the report as "Benign", "Malicious" or "Pending".
- Respond to the user who submitted the report.

- Obtain an overview about the report, which includes information such as the person who submitted the report, the time at which the report was submitted and the current status. It also contains a timeline, which tracks any changes to the report.
- View the list of all the header fields in the email.
- View the text and HTML content of the email.

It is also possible to create rules that check for a match against a new phishing submission and perform an action if they match. These rules can either match on header fields or on the body content. The actions are short Python code snippets that can be coded by the analyst at will. For example, a rule can be used to automatically classify a report if it matches that rule. Thus, by using this tool an analyst is able to manage the reports in a simple and polite way, but he has to write Python snippets that allow automating part of the analysis, otherwise he has to analyze all the pieces of information manually.

3. Cyber threat intelligence

In order to analyze an email, it is important to extract many pieces of information, such as IP addresses, email addresses, domains, and URLs, and analyze them one by one. From now on, the term that will be used to refer to such information is the term *observable*. An observable is defined as an event (benign or malicious) on a network, or a system [18]. This event can be, for instance, the sighting of a specific IP address that might require a further analysis if considered malicious. If an observable is related to malicious activity, it is called *Indicator of Compromise (IoC)*. An IoC is defined as an artifact that, with high confidence, indicates a computer intrusion [19]. In the case of email analysis, IoCs can be IP addresses, email addresses, domains, URLs deemed related to phishing or spam activity, and dangerous attachment files. To find out whether an observable is malicious, an analyst can use online tools such as VirusTotal to analyze a URL or a file or check an IP address or domain against a blocklist. In addition, he/she can use sandbox services to run an executable file (e.g., an email attachment) into an automated, virtualized, and safe environment to observe its behavior and obtain additional IoCs. After IoCs have been identified via a process of incident response and computer forensics, they can be used for early detection of future attack attempts using intrusion detection systems and antivirus software.

Although IoCs found inside an organization might be useful to prevent future similar attacks, they are not enough to prevent attacks that are currently in the wild and have never targeted the organization. Also, IoCs alone are not able to identify who is behind an attack, their motivations, or the ultimate sponsor of the attack itself [20]. If an organization had this information, it would be able to make important decisions that go far beyond the simple addition of a rule in a firewall. What is needed is the concept of *Cyber Threat Intelligence (CTI)*. Gartner defines CTI as “evidence-based knowledge, including context, mechanisms, indicators, implications and actionable advice, about an existing or emerging menace or hazard to assets that can be used to inform decisions regarding the subject’s response to that menace or hazard” [21]. CTI is not only the raw data or the single IoC but requires rich contextual information that can only be created with the application of human analysis. This contextual information includes the linkage between the technical indicators, adversaries, their motivations and intents, and

information about who is being targeted. Analysts, in fact, should not only focus on the result of an attack, like an IoC but also look at the tactics and techniques that indicate that an attack is in progress. The real advantage provided by CTI is, however, the concept of *cyber threat information sharing*. It provides access to threat information that might otherwise be unavailable to an organization, which in this way can benefit from the knowledge, experience, and capabilities of other organizations to gain a complete understanding of the threats that it may face. This allows one organization's detection to become another's prevention [22]. In the remainder of this section we will describe three of the most important CTI open-source frameworks, namely, MISP, Cortex and TheHive.

3.1. MISP

Malware Information Sharing Platform (MISP) is a free and open-source software helping information sharing of threat intelligence including cyber security indicators [23]. Its aim is to help improve the countermeasures used against targeted attacks and set up preventive actions and detection [24]. It makes it possible to store IoCs in a structured manner, and thus enjoy the correlation, automated exports for IDSes or SIEMs, in STIX, OpenIOC and many other formats including CSV and plain text, and synchronize to other MISP servers. It also makes it easier to share with, but also to receive from trusted partners and trust groups so as to enable fast and effective detection of attacks. MISP provides an intuitive web interface, but also a REST API that can be used for automation and feeding devices. Moreover, there is also a Python library called PyMISP that allows easy access via the API [23]. The basic building block of MISP is the *event*. Each event is made up of a list of attributes, which are atomic pieces of data that could be IoCs like for example an IP address, a URL or a file. Each time an attribute is created, MISP will check whether that attribute already exists in the system so as to find a correlation. Not only does a correlation manifest itself with an exact match, but also with the presence of attributes that are believed to be related in some way, for example there may be a correlation between an IP address and a range of IP addresses to which it belongs. Moreover, it is possible to add information to an event by either using *tags* or more advanced features. Apart from being a self-contained repository of attacks and malware, one of the main features of MISP is its ability to connect to other MISP servers (also called *MISP instances*) and share its information. The exchange of data between two or more MISP instances is called *synchronization*. Another way of getting events from a remote source in MISP is by using *feeds*. Feeds are remote or local resources containing indicators that can be automatically imported into MISP at regular intervals. A great advantage of MISP is the fact that it is supported by many open-source and proprietary tools. An example of such a tool is TheHive, which is an open-source Security Incident Response Platform (SIRP) [25].

3.2. TheHive

TheHive is a scalable, open-source and free Security Incident Response Platform (SIRP) created by TheHive Project, tightly integrated with MISP, designed to make life easier for SOCs, CSIRTs, CERTs and any information security practitioner dealing with security incidents that need to be investigated and acted upon swiftly. It provides a web interface from which it is possible to

manage alerts related to security events coming from a multitude of sources, like a SIEM, an IDS, an email report or a MISP event. Alerts can be ignored, marked as read, previewed and imported. When an alert is imported, it becomes a case that needs to be investigated [26][27][28]. It also offers a REST API and most of its endpoints are accessible through TheHive4py, which is the Python API client for TheHive [29]. The core construct of TheHive is the case, which is also the core construct of most security investigations. A case is characterized by a title, a description and a date. Moreover, it is also characterized by several elements, some of which are outlined below [26][27][28][30]:

- *Tasks*: They are used to track the actions taken to answer the investigative questions, but also to track containment, eradication and remediation events. They can contain multiple *logs*, which are text entries used to describe an analyst's progress, attach pieces of evidence or noteworthy files and even password-protected ZIP archives containing malware or suspicious data.
- *Observables*: They can be of different types, for example IP addresses, email addresses, URLs and domains. In addition, custom observable types can be defined if needed. They can be tagged, flagged as IoC and analyzed. If an observable in a case has already been seen in other cases, it is automatically marked as sighted, and cases that share common observables are considered related.
- *Tags*: They are another way of adding information to a case and can be used for quick searching and filtering. They are labels that can be attached to cases, but also to many other TheHive objects like alerts and observables. For instance, it is possible to add the source that provided or generated an observable by using a tag.

In order to reduce the time wasted on the creation of cases that share the same structure, TheHive supports the creation of pre-defined *case templates*. Those templates can also be used to outline the steps to follow so as to drive the team's activity. [26][27]. A case can be generated from an *alert* or created from scratch. The alert is another important TheHive construct that shares many properties with the case construct, including the observables that have been observed in the security event that generated that alert. All those fields will be directly mapped to the correspondent case fields once the alert is imported. Moreover, case templates can also be used to create cases from alerts when they are imported. A key feature of TheHive is *collaboration*. In fact, each analyst has his own account with its set of permissions and every action he performs is logged in a real-time *live feed*, which is visible by other analysts. Both cases and tasks can be assigned to an analyst and it is possible for multiple analysts to work on the same case but on different tasks, so as to share the responsibilities. In order to keep track of the progress of the investigation, cases and tasks can be in different states. For example, a case can be in the "Open" state, but it can transit in the "Resolved" state when the analysis is terminated and the case gets closed, specifying a series of fields like the possible impact of the incident. Similarly, tasks can be in a "Waiting" state when they have not been assigned to an analyst yet, then they can transit in the "InProgress" state when they are started and they can transit in the "Completed" state when they are closed.

3.3. Cortex

Cortex is a powerful open-source observable analysis and active response engine [26] created by TheHive Project that allows analyzing observables at scale by querying a single tool instead of several [31]. It provides a web interface from which it is possible to analyze observables one by one or in bulk mode, but it can also be used to automate these operations and submit large sets of observables from TheHive or through the Cortex REST API. Moreover, most of the endpoints offered by the Cortex REST API are accessible through Cortex4py, which is the Python API client for Cortex [32]. The usage of Cortex is based on *neurons*, which are autonomous applications managed by and run through the Cortex core engine [33]. They can be of one of two types:

- *Analyzers*: They allow analyzing different types of observables by automating the interaction with a service or a tool so as to speed up the analysis and make it possible to contain threats before it is too late. Cortex comes with more than a hundred analyzers for popular services such as VirusTotal, emailrep.io, urlscan.io, AbuseIPDB and PhishTank. It should be noted that while many analyzers are free to use, some require special access and others need a valid service subscription or product license [26][33].
- *Responders*: They are installed along with the analyzers. Unlike analyzers, they are only useful when Cortex is used in conjunction with TheHive, in fact they perform different actions and apply to alerts, cases, tasks, task logs and observables [33].

Analyzers and responders can be enabled, disabled and configured from the web interface. For each of them it is possible to define many parameters such as a rate limits, usernames, passwords and API keys. When an observable is submitted for analysis, Cortex creates a job. That job will generate an analysis report in JSON format if it terminates successfully [28]. Moreover, these job reports can be cached so that if an analyzer is launched against the same observable, the previous report can be returned without re-executing the analyzer. The cache is used only if the second job occurs within a configurable amount of time, where the default is 10 minutes [33]. A job is also created when a responder is started. Also in this case a JSON report regarding the result of the performed action is provided.

3.4. Integrating TheHive, Cortex and MISP

Cortex is a great tool on its own, but its real potential is only unlocked when used in conjunction with TheHive. Indeed, TheHive can connect to one or multiple Cortex instances in order to have access to neurons. When TheHive is connected to a Cortex server, it is possible to launch one or more analyzers against the observables in a case so as to obtain additional information about them. The output of each analyzer is a report in JSON format that is viewable from TheHive. Moreover, it is also possible to launch responders against cases, observables, tasks, task logs and alerts to execute an action. For example, the *Mailer* responder can be launched against a task to automatically send an email containing the description of the task itself, which can be for example the result of the analysis [34].

Even though many organizations can share information about cases and observables through TheHive and Cortex, the actual support for cyber threat information sharing is provided by the integration with MISP. In fact, by integrating TheHive with MISP it is possible to automatically

import MISP events as alerts and also export cases to MISP as events. Moreover, also Cortex can be integrated with MISP so as to allow searching observables within a MISP instance. This is possible thanks to a *MISP Search* analyzer that is available for Cortex that returns the number of events where the observable has been found and a list of links to those events with additional data. This analyzer is very useful when it is launched against an observable in a case from TheHive, so as to further enrich the information on the investigation.

4. ThePhish

Many tools can be used to automate or facilitate some of the activities to perform to analyze an email, but none of them provides complete automation for the analysis process. The aim of this work is to take advantage of the huge potential of TheHive, Cortex and MISP, and develop an application that is able to automate the entire analysis cycle of an email. ThePhish is a web application written in Python that allows the analyst to choose the email to analyze and obtain a verdict, which can be final or not. Figure 1 shows an overview of how the application works [35][36]. The scenario depicted in Figure 1 is composed of the following steps:

1. An attacker starts a phishing campaign and sends a phishing email to a user.
2. A user who receives such an email can send that email as an attachment to the mailbox used by ThePhish.
3. The analyst interacts with ThePhish and selects the email to analyze.
4. ThePhish extracts all the observables from the email and creates a case on TheHive. The observables are analyzed thanks to Cortex and its analyzers.

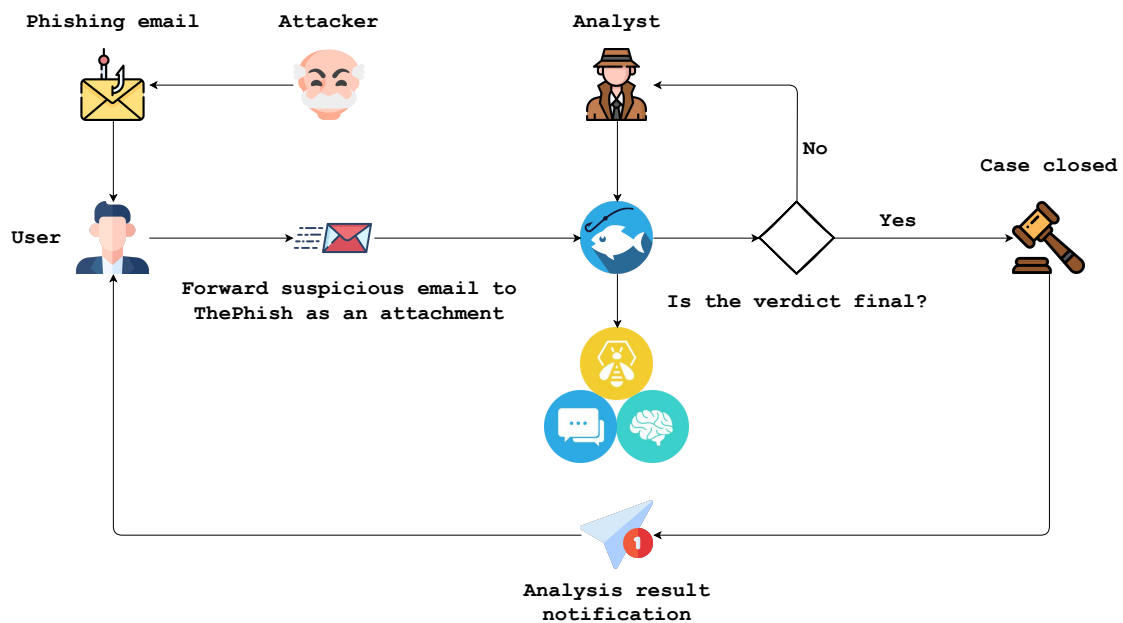


Figure 1: ThePhish overview

5. ThePhish calculates a verdict based on the verdicts of the analyzers.
6. If the verdict is final, the case is closed and the user is notified. In addition, if it is a malicious email, the case is exported to MISP.
7. If the verdict is not final, the analyst's intervention is required. He must review the case on TheHive along with the results given by the various analyzers to formulate a final verdict, then it can send the notification to the user, optionally export the case to MISP and close the case.

ThePhish relieves the analyst from manually extracting all the observables from the header and the body of the email and adding them one by one in a case on TheHive. Moreover, he does not need to start the various analyzers on each observable, send notifications to users, nor interacting with MISP. Even in the case in which his intervention is required, the majority of the work will have already been performed so that he can focus only on things that matter to elaborate a final verdict.

In order to automate the analysis process, ThePhish communicates with TheHive, Cortex and MISP through the REST APIs made available by TheHive and Cortex. The sequence diagram in Figure 2 and Figure 3 shows the high-level interactions among all the components. Supposing that at least one user has already sent a suspicious email as an attachment in EML format to ThePhish, the workflow is as follows:

1. The analyst interacts with ThePhish to obtain a list of suspicious emails to analyze. ThePhish obtains those emails by selecting all the unread emails in the mailbox that have an email message in EML format as an attachment.
2. The analyst interacts with ThePhish to make the analysis of the selected email start.
3. ThePhish parses both the header and the body of the attached email to extract all the observables.
4. ThePhish creates a case on TheHive and adds all the previously extracted observables to it.
5. ThePhish starts the *Mailer* responder to send a notification email to the user in order to let him know that the analysis of the email he sent has started.
6. ThePhish waits for the responder job to complete.
7. For each observable, ThePhish starts all the analyzers that have been configured on Cortex for that observable type. It should be noted that the control returns to ThePhish once an analyzer job is started, without waiting for the analysis to terminate.
8. If the analyzer is the *MISP Search* analyzer, Cortex checks the presence of the analyzed observable among MISP events, while for the other analyzers the interaction is with an external service and is not represented in the diagram.
9. ThePhish waits until all the previously started analyzer jobs are terminated.
10. Once the results of all the analyzers are available, ThePhish elaborates the verdict on the email.
11. If the verdict is "malicious", ThePhish interacts with TheHive, which in turn interacts with MISP, in order to export the case as an event to MISP.

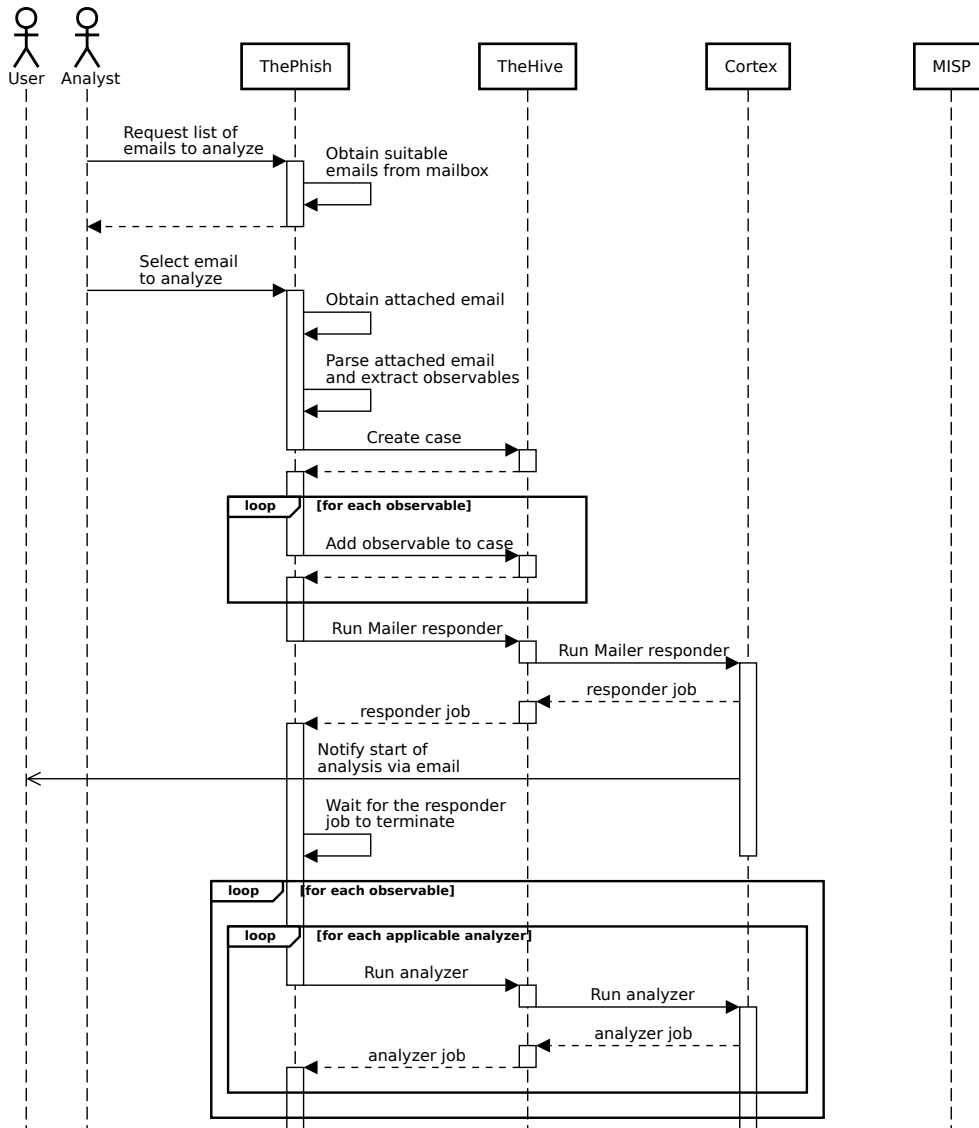


Figure 2: ThePhish sequence diagram (part 1)

12. If the verdict is either “malicious” or “safe”, the *Mailer* responder is started to send the verdict via email to the user and then, when the responder job terminates, the case is closed. The verdict is then shown to the analyst, which can then start a new analysis.
13. If the verdict is “suspicious”, the analyst can review the case on TheHive. In that case, the results of all the analyzers will already be available and the analyst only has to make the final decision and close the case. Moreover, if the email is classified as “malicious” in this phase, the analyst can also export the case to MISP from TheHive in just one click. This allows ThePhish to classify the next email that has some observables in common with this email to be classified as “malicious” as well, thanks to the *MISP Search* analyzer.

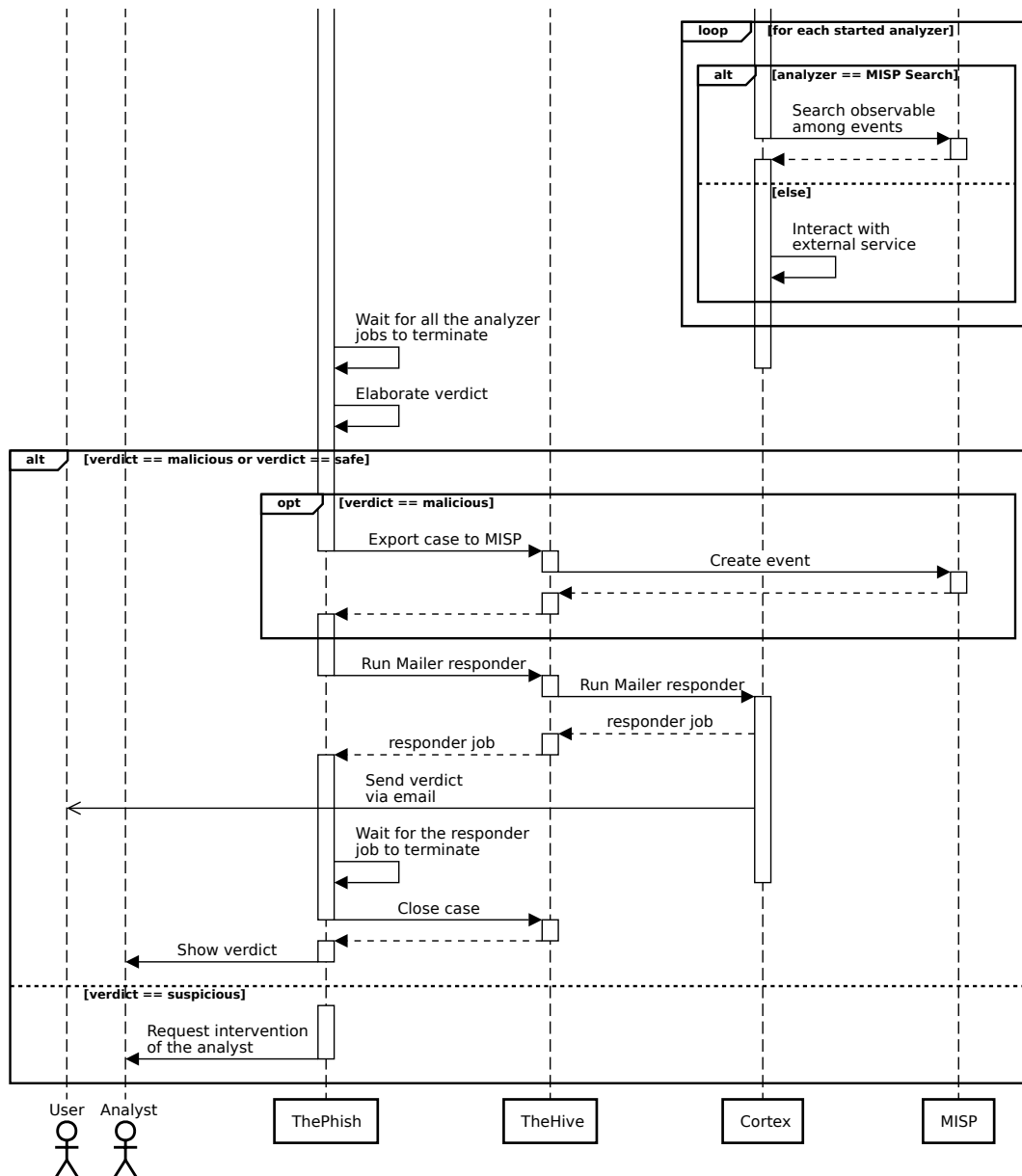


Figure 3: ThePhish sequence diagram (part 2)

5. ThePhish Implementation

This section will describe implementation details and illustrate the main ThePhish modules. Finally, we explain the communication between the front-end and the back-end component. ThePhish is a web application written in Python 3. The web server is implemented using *Flask*, which is a lightweight Web Server Gateway Interface (WSGI) web application framework [37].

In contrast, the front-end part of the application, which is the dynamic page written in HTML, CSS, and JavaScript, is implemented using the front-end framework *Bootstrap*. Apart from the webserver module, the back-end logic of the application is constituted by three Python modules that encapsulate the logic of the application itself and a Python class used to support the logging facility through the WebSocket protocol. Moreover, there are several configuration files used by the aforementioned modules that serve various purposes.

5.1. Obtaining the list of emails to analyze

The `list_emails` module allows analyst to visualize the emails suitable for the analysis. Figure 4 shows the activity diagram of this module. The first two actions that the module performs are needed to connect to the IMAP server. Then, it retrieves all the unread emails from the mailbox on the IMAP server using the `imaplib` module and walks through their multipart structure using the `email` Python standard library module to find out whether they contain an EML attachment. For each email that satisfies this condition, *From*, *Subject*, the date and email body will be extracted, along with the *Subject* field of the attached email. These pieces of information, together with the *Unique Identification Number (UID)* of the email, will be added to the list to

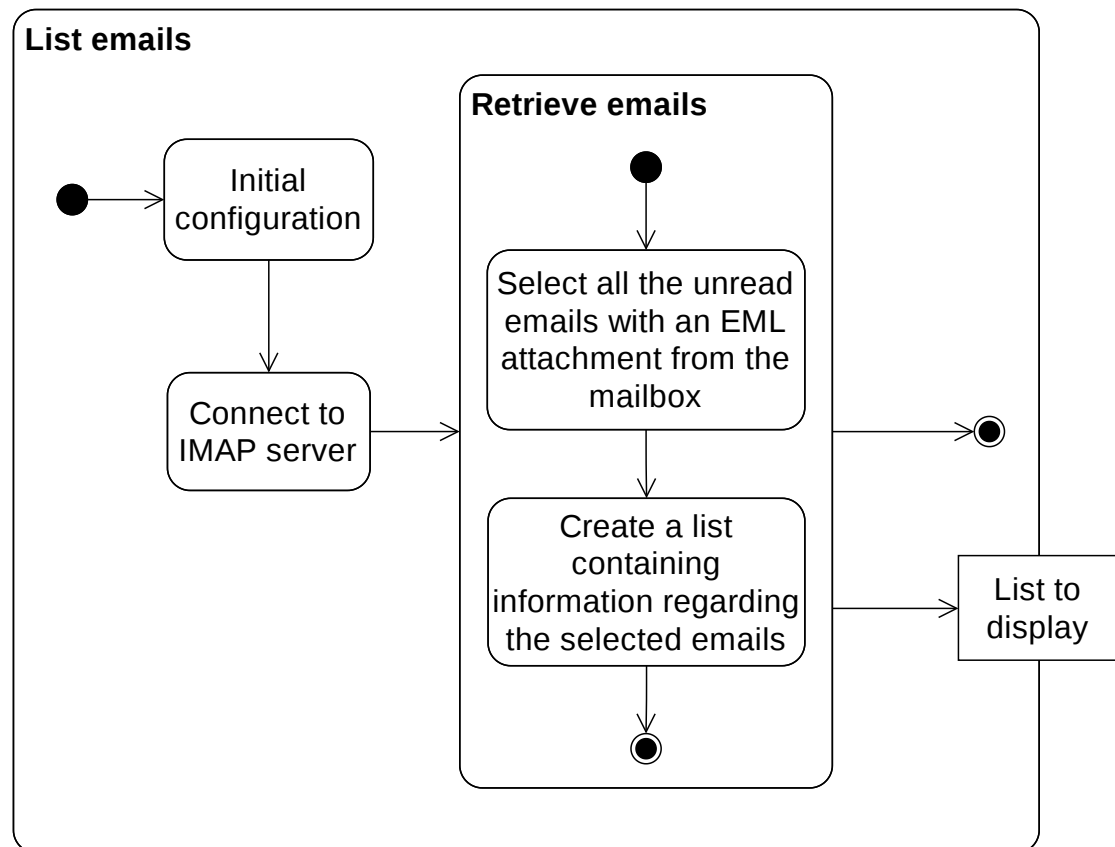


Figure 4: Activity diagram of the `list_emails` module

return so that the analyst can visualize information regarding the emails to analyze on the web interface. The UID is a fundamental piece of information. Indeed it uniquely identifies the email in an IMAP folder and is used when the analyst selects the email to process during the analysis. When the execution of the module terminates, information regarding the emails that can be analyzed is returned. It is displayed to the analyst on the web interface, giving him the possibility to choose the email to analyze.

5.2. Creating the case on TheHive

When the analyst selects an email, the `case_from_email` module manages the observable extraction and the case creation on TheHive. Figure 5 shows the activity diagram of this module. The first two actions that the module performs are needed to connect to the IMAP server and the TheHive instance. Moreover, the file `whitelist.json` is also considered. It is used to configure the lists of observables that have to be excluded from the analysis to reduce the occurrence of false positives. When the user selects the email to analyze from the web interface,

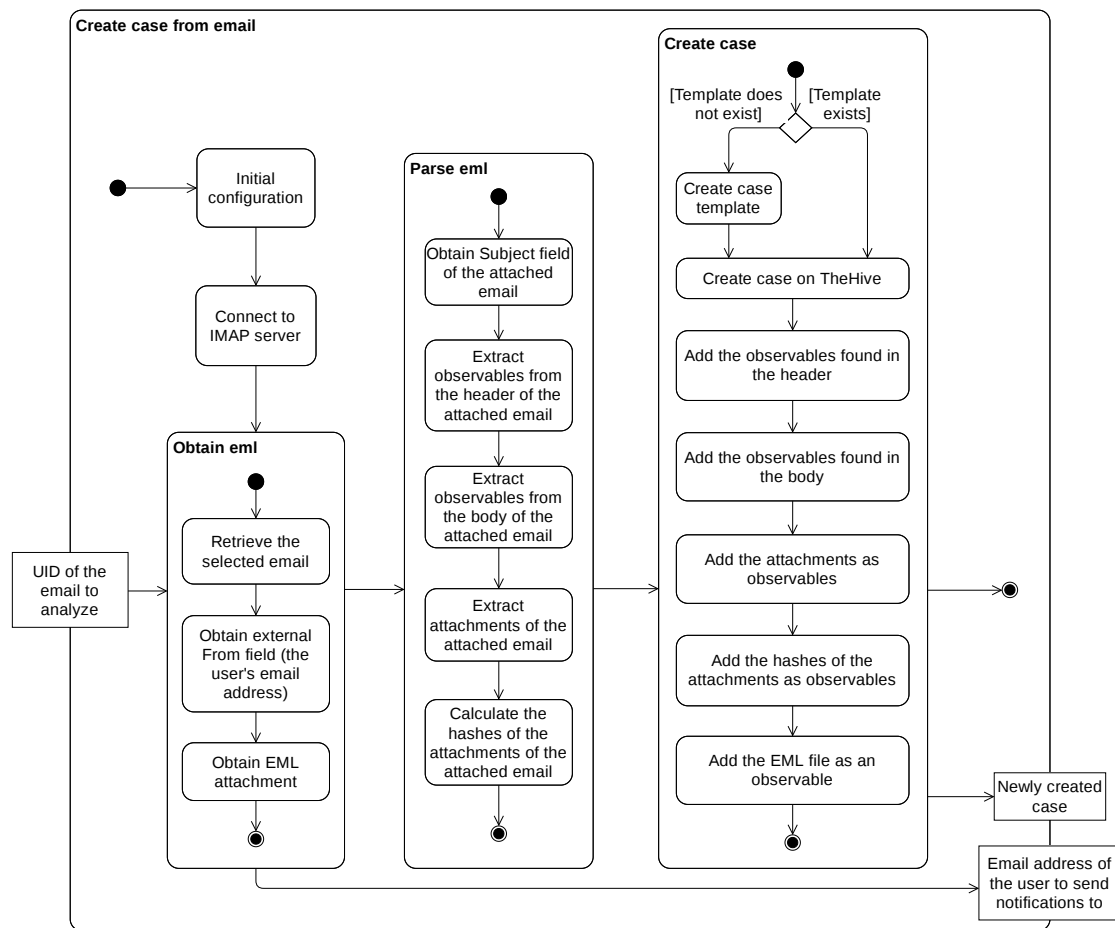


Figure 5: Activity diagram of the `case_from_email` module

the UID previously returned by the `list_emails` module is sent back to the server and passed to this module. The `email` module utilizes the UID to fetch the email from the mailbox and extract its *From* field, i.e., the email address of the user who sent the email to ThePhish for analysis. This email will be used to send him the email notifications. The module indeed also extracts the EML attachment to parse. Once the EML attachment is available, it is an email itself that has to be parsed to extract the observables it contains. The extraction logic uses the `ioc_finder` Python module [38] to extract email addresses, IP addresses, URLs and domains from that buffer. To decide whether an observable should be considered or not and avoid analyzing that may cause false positives, ThePhish allows creating a whitelist. Through the whitelist, the analyst can also customize the observables deemed relevant for the email analysis. The whitelist is contained in a file named `whitelist.json`. It is constituted by many different lists to offer great flexibility in terms of observable types to match and matching modes. It supports the following matching modes:

- Exact string matching for email addresses, IP addresses, URLs, domains, file names, file types and hashes
- Regex matching for email addresses, IP addresses, URLs, domains and file names
- Regex matching for subdomains, URLs and email addresses that contain the specified domains

While both the parts related to exact matching and regex matching are used without any modification, the remaining parts are used to create three more lists of regular expressions. It is not required for the analyst to design complex regular expressions to enable those features, but he only needs to add the domains to the right lists. These regular expressions have been designed to avoid some unwanted behaviors. For instance, if the intention is to whitelist the domain “paypal.com”, they prevent domains like “paypal.com.attacker.com” from being mistakenly whitelisted. The observables are extracted from the values of some carefully chosen header fields and the body of the email. Then, the attachments are extracted, and their SHA256 hashes are calculated in order to add them as observables to the case as well. Finally, the content of the EML file itself is saved in a variable so that it can be added to the case as an observable named after the *Subject* field of the email it contains. Then, TheHive4py is used to create a case and add observables, which are also tagged to indicate the relevant email sections. *Subject* field of the email under analysis is used to name the case. Before doing that, TheHive4py also creates a case template.

5.3. Running the analysis

The `run_analysis` module automates the entire analysis procedure and calculates a verdict that is shown to the analyst and sent via email to the user. Thanks to the `case_from_email` module, it knows the case under analysis and the user email address to send notifications to. Figure 6 shows the activity diagram of this module. The first two actions that the module performs are needed to connect to the IMAP server and the TheHive, Cortex, and MISP instances. Moreover, the file `whitelist.json` is also considered. Then, this module uses TheHive4py to obtain the IDs of the three tasks present in the case and handles their life cycle. Each task is handled by one of the remaining three actions. The three tasks are described below.

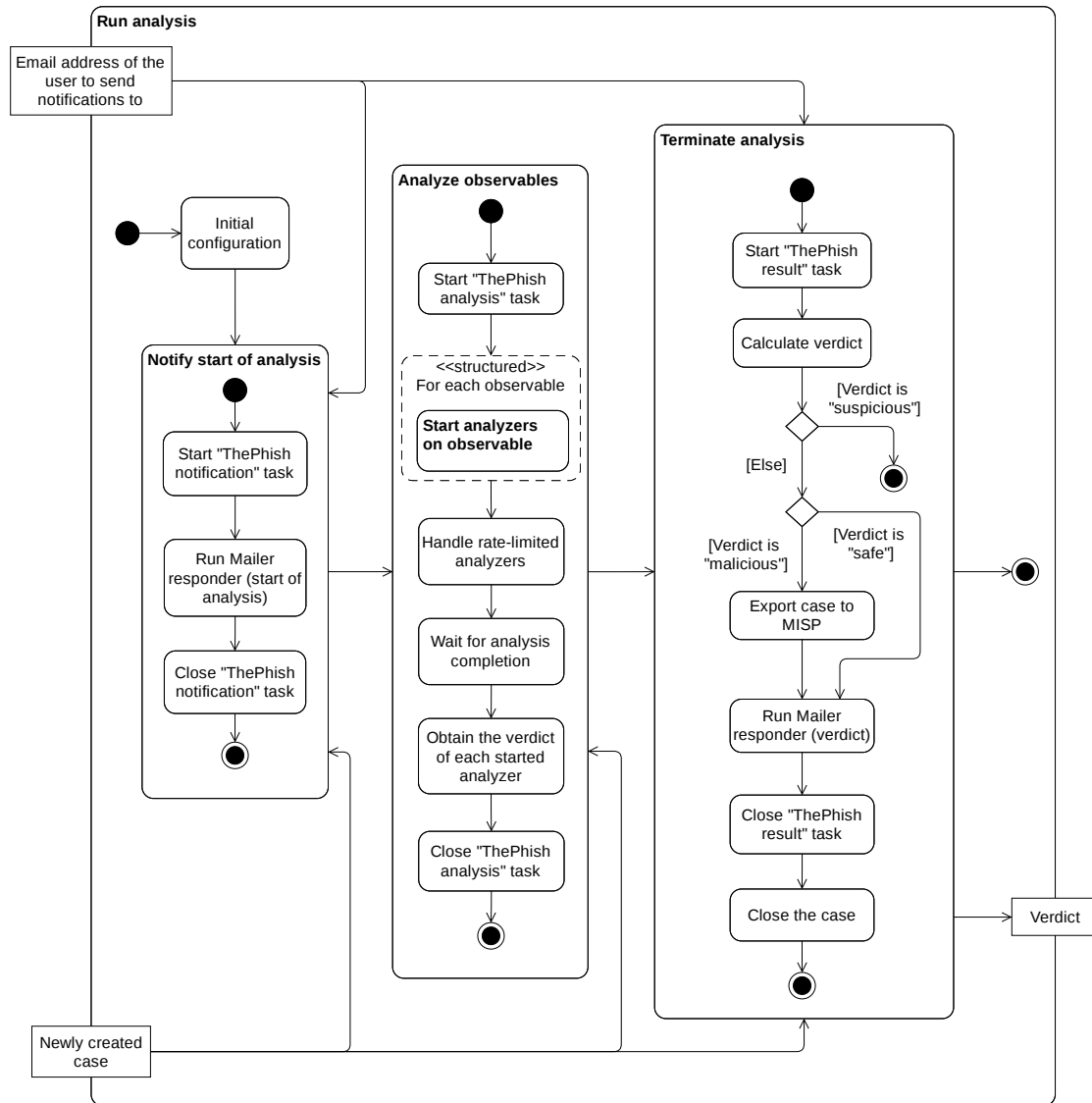


Figure 6: Activity diagram of the run_analysis module

- *ThePhish notification:* This task is used to send the notification that the analysis has been started to the user. This is obtained by starting the *Mailer* responder using Cortex4py after having filled the description of the task with the user's email address and the body of the email.
- *ThePhish analysis:* This task is used to analyze the observables. This is obtained by starting all the analyzers that are applicable to each observable using Cortex4py. Since this task is a fairly complex procedure, it has been modeled as an activity itself that is repeated for each observable, and its activity diagram is shown in Figure 7. Each analyzer outputs a JSON report containing a maliciousness level for an observable that can be one

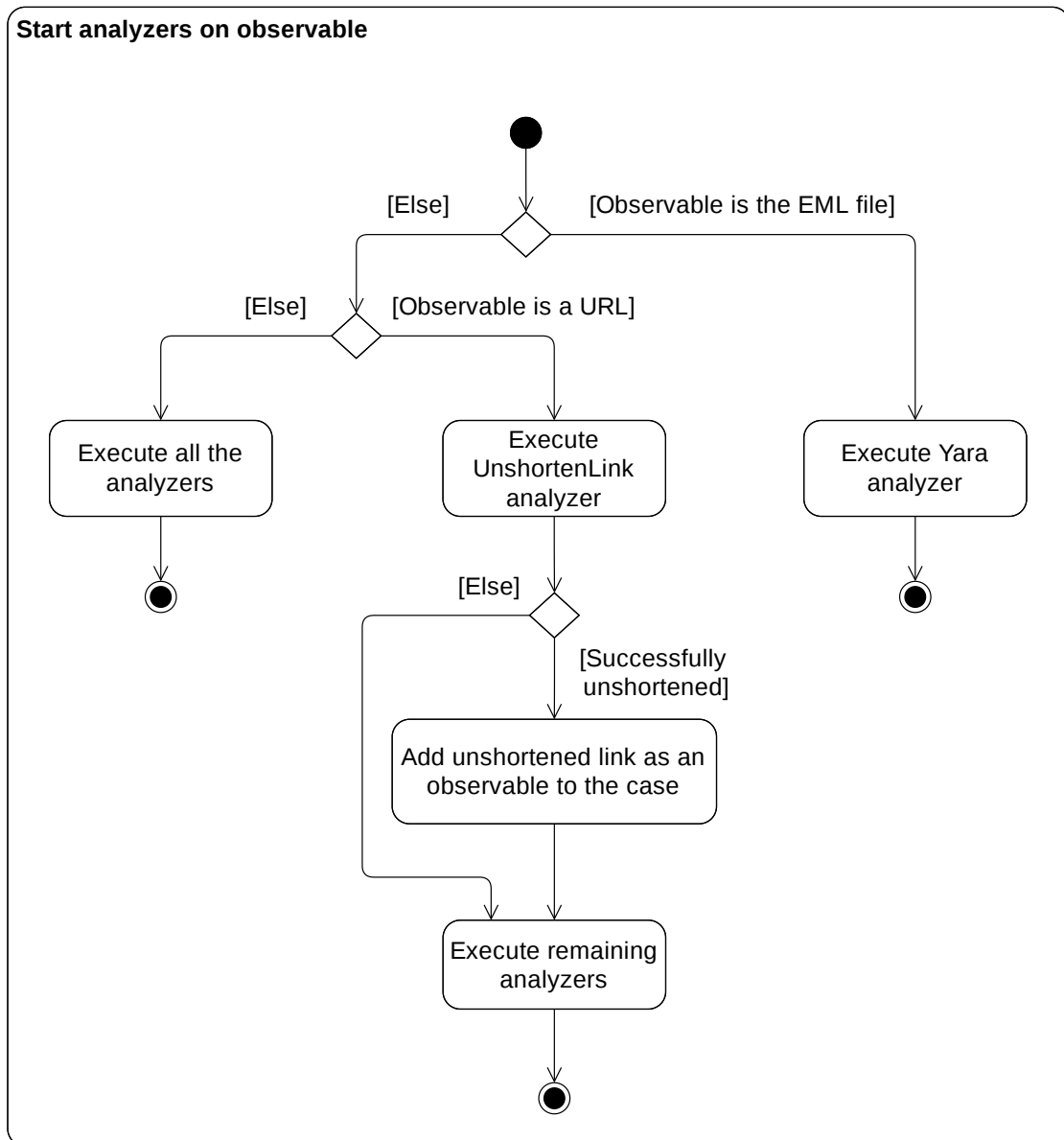


Figure 7: *Start analyzers on observable* activity diagram

of “info”, “safe”, “suspicious”, or “malicious”. However, even though the report structure usually follows a convention, this convention is not always respected. Moreover, after analyzing the code of many analyzers and several tests, some analyzers have been found to contain bugs. For this reason, we made various tweaks and workarounds to obtain the maliciousness levels provided by these analyzers anyway or to prevent the application from crashing due to those bugs. Furthermore, these levels do not always represent the real maliciousness level of an observable. Since this depends on how the analyzers

themselves have been programmed, ThePhish comes with another configuration file called `analyzers_level_conf.json`, with which it is possible to create a mapping between the actual maliciousness levels provided by any analyzer and the levels decided by the analyst. Besides that, this file allows the analyst to choose which observable types should be modified.

- *ThePhish result*: This task calculates the verdict and sends it to the user. The judgment is calculated so that if at least one analyzer has given a malicious maliciousness level to at least one observable, then the email is marked as malicious, and all the observables that satisfy this condition are marked as IoC. If that is not the case, then the email is marked as “suspicious” if there is at least one analyzer that has given a “suspicious” maliciousness level to at least one observable. Otherwise, the email is marked as “safe”. The cases containing “malicious” emails, are exported to MISP along with all the observables marked as IoC. Finally, the task is updated with a description, including the final verdict and the email address of the user to send it to by using the *Mailer* responder. Both the task and the case are closed, and the verdict is returned to the analyst. However, if the verdict is “suspicious”, the analyst’s intervention is required, so neither the task nor the case is closed.

5.4. Interaction between front-end and back-end

The user interacts with a web application front-end to perform email analysis activities. The front-end component establishes a bi-directional connection with the server to manage asynchronous tasks. This is done by using the *Socket.IO* JavaScript library [39] in the web page that enables real-time, bi-directional, and event-based communication between the browser and the server. This connection is established with a WebSocket connection whenever possible and will use HTTP long polling as a fallback. For this to work, the server application uses the *Flask-SocketIO* Python library [40], which provides a Socket.IO integration for Flask applications. ThePhish then uses this connection to display the progress of the analysis on the web interface. Once the connection has been established, a unique identifier for the socket session (SID) is available both on the client and the server. Then, when the analyst requests the analysis of a specific email, that SID is sent as a parameter in the request so that the server can associate the request with one of the connected clients. Whenever the analyst acts on the web interface, an AJAX request is sent to the server, an asynchronous HTTP request that permits the exchange of data with the server in the background and updates the page without reloading it. If the analyst wants to visualize the list of emails to analyze, he must click on the “List emails” button on the web interface, which triggers an asynchronous HTTP GET request to the correct endpoint. Once the response is obtained, the HTML DOM is updated to show the list of emails. At this point, the analyst can select the email to analyze by clicking on the corresponding “Analyze” button. Again, an asynchronous HTTP request is triggered, but this time it is a POST request to another endpoint. Once the response is obtained, the HTML DOM is updated to show the result of the analysis. It should be noted that the payload of the request contains the UID of the selected email and the SID used to identify the client to send log messages to.

6. Conclusions

As of today, phishing emails are the most widely used infection vector. The natural consequence of this fact is that SOCs, CERTs, and CSIRTs are becoming overwhelmed by the number of emails they need to analyze every day, with the majority being false positives. To avoid wasting time and effort, many commercial or open-source solutions have been proposed to automate, at least partially, the long and tedious process of email analysis. In this work, we presented ThePhish, an open-source phishing email analysis platform. It is based on three open-source platforms, namely TheHive, Cortex, and MISP, and allows automating the entire analysis process starting from the extraction of the observables from the header and the body of the email to the elaboration of a verdict, which is the final in most cases. In addition, it allows the analyst to intervene in the analysis process and obtain further details on the email being analyzed if necessary. The platform has been released under the AGPL license and made available on GitHub so that anyone can contribute to improving it over time. The development of ThePhish will, in fact, not stop here, as there is significant room for improvements. Further changes will be made in the future to add new functionalities to ThePhish, support any new feature introduced by TheHive and Cortex, support new analyzers, and fix bugs that might be present in the current release or any future release of ThePhish. In future works, we are going to compare ThePhish with other phishing email analysis approaches against a known dataset, such as [41]. This dataset is composed of 303 phishing emails. Preliminary studies seem promising, as ThePhish was able to parse 90% of the emails in the dataset and classify those parsed emails correctly. In particular, 84.6% of the parsed emails were classified as malicious, while 15.4% of them as suspicious. Anyway, we also would like to evaluate the performance of ThePhish against non-phishing emails.

References

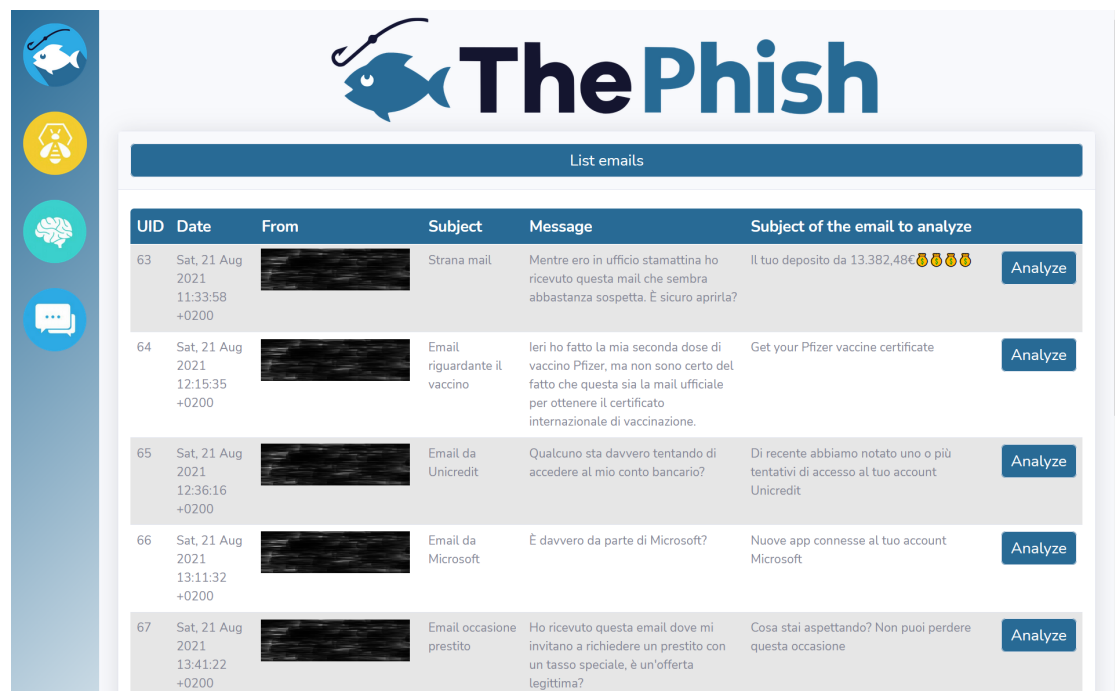
- [1] World Economic Forum (WEF), The Global Risks Report 2020, Technical Report, 2020.
- [2] 2021 Must-Know Cyber Attack Statistics and Trends, 2021. URL: <https://www.embroker.com/blog/cyber-attack-statistics/>.
- [3] Wikipedia contributors, Social engineering (security) – Wikipedia, The Free Encyclopedia, 2021. URL: [https://en.wikipedia.org/wiki/Social_engineering_\(security\)](https://en.wikipedia.org/wiki/Social_engineering_(security)).
- [4] Bruce Schneier, Secrets and Lies: Digital Security in a Networked World, Wiley, 2015.
- [5] All About Phishing Scams & Prevention: What You Need to Know, 2021. URL: <https://www.kaspersky.com/resource-center/preemptive-safety/phishing-prevention-tips>.
- [6] Sam Erdheim, A SOC Under Siege: Alert Overload and Cyber Skills Shortage, 2018. URL: <https://fidelissecurity.com/threatgeek/threat-detection-response/industry-professional-shortage/>.
- [7] MxToolbox, 2021. URL: <https://mxtoolbox.com/>.
- [8] SpamAssassin, 2021. URL: <https://spamassassin.apache.org/>.
- [9] MrCalv1n, EmailAnalyzer, 2020. URL: <https://github.com/MrCalv1n/EmailAnalyzer>.
- [10] VirusTotal, 2021. URL: <https://www.virustotal.com/>.
- [11] TheresAFewConors, Sooty, 2021. URL: <https://github.com/TheresAFewConors/Sooty>.

- [12] AbuseIPDB, 2021. URL: <https://www.abuseipdb.com/>.
- [13] emailrep.io, 2021. URL: <https://emailrep.io/>.
- [14] HaveIBeenPwned, 2021. URL: <https://haveibeenpwned.com/>.
- [15] PhishTank, 2021. URL: <https://www.phishtank.com/>.
- [16] urlscan.io, 2021. URL: <https://urlscan.io/>.
- [17] duo-labs, IsThisLegit, 2020. URL: <https://github.com/duo-labs/isthislegit>.
- [18] Observable - Glossary, 2021. URL: <https://csrc.nist.gov/glossary/term/observable>.
- [19] Wikipedia contributors, Indicator of compromise — Wikipedia, The Free Encyclopedia, 2021. URL: https://en.wikipedia.org/wiki/Indicator_of_compromise.
- [20] Wikipedia contributors, Cyber threat intelligence — Wikipedia, The Free Encyclopedia, 2021. URL: https://en.wikipedia.org/wiki/Cyber_threat_intelligence.
- [21] Different Definitions of Threat Intelligence and Gartner's Perspective, 2016. URL: <https://socradar.io/different-definitions-of-threat-intelligence-and-gartners-perspective/>.
- [22] ITL, CYBER-THREAT INTELLIGENCE AND INFORMATION SHARING, Technical Report, 2017.
- [23] MISP - Open Source Threat Intelligence Platform & Open Standards For Threat Information Sharing, 2021. URL: <https://www.misp-project.org/>.
- [24] MISP - Open Source Threat Intelligence Platform, 2021. URL: <https://www.circl.lu/services/misp-malware-information-sharing-platform/>.
- [25] User guide of MISP intelligence sharing platform, 2021. URL: <https://www.circl.lu/doc/misp/>.
- [26] TheHive-Project, TheHive Project, 2021. URL: <https://thehive-project.org/>.
- [27] TheHive-Project, TheHive, 2021. URL: <https://github.com/TheHive-Project/TheHive>.
- [28] Saâd Kadhi, TheHive, Cortex and MISP: How They All Fit Together, 2017. URL: <https://blog.thehive-project.org/2017/06/19/thehive-cortex-and-misp-how-they-all-fit-together/>.
- [29] TheHive-Project, TheHive4py Documentation, 2021. URL: <https://thehive-project.github.io/TheHive4py/>.
- [30] Chris Sanders, Investigation Case Management with TheHive, 2017. URL: <https://chrissanders.org/2017/03/case-management-the-hive/>.
- [31] TheHive-Project, Cortex, 2021. URL: <https://github.com/TheHive-Project/Cortex>.
- [32] TheHive-Project, Cortex4py, 2021. URL: <https://github.com/TheHive-Project/Cortex4py>.
- [33] TheHive-Project, Cortex Docs, 2021. URL: <https://github.com/TheHive-Project/CortexDocs>.
- [34] Arnaud Loos, Open Source SIRP with Elasticsearch and TheHive - Part 6 - Case Management, 2019. URL: <https://arnaudloos.com/2019/open-source-sirp-part-6-case-management/>.
- [35] Flaticon, 2021. URL: <https://www.flaticon.com/>.
- [36] Free Icons, Clipart Illustrations, Photos, and Music, 2021. URL: <https://icons8.com>.
- [37] pallets, Flask, 2021. URL: <https://github.com/pallets/flask>.
- [38] fhightower, ioc-finder, 2021. URL: <https://github.com/fhightower/ioc-finder>.
- [39] SOCKET.IO, 2021. URL: <https://socket.io/>.
- [40] miguelgrinberg, Flask-SocketIO, 2021. URL: <https://github.com/miguelgrinberg/flask-socketio>.
- [41] J. Nazario, Phishing Corpus, 2018. URL: <https://monkey.org/~jose/phishing/phishing-2015>.

- [42] Docker docs, 2021. URL: <https://docs.docker.com>.
- [43] TheHive-Project, thehive4-cortex3-misp-shuffle, 2021. URL: <https://github.com/TheHive-Project/Docker-Templates/tree/main/docker/thehive4-cortex3-misp-shuffle>.
- [44] Emanuele Galdi (emalderson), ThePhish, 2021. URL: <https://github.com/emalderson/ThePhish>.

A. ThePhish example usage

In order to start the analysis process, the analyst must first navigate to the web page of ThePhish and obtain the list of emails to analyze, as shown in Figure 8. It should be noted that the emails must be forwarded by the users as attachments in EML format so as to prevent the contamination of the email header. The analyst can then select the email to analyze and start the analysis, the progress of which is shown in Figure 9. In the meantime, ThePhish extracts the observables from the email and then interacts with TheHive. Figure 10 shows the case populated with the extracted observables. At the end of the analysis, ThePhish calculates the verdict. Since the verdict is “malicious”, all the observables that are found to be “malicious” are marked as IoC. In this case only one observable is marked as IoC, as shown in Figure 11. The case is then exported to MISP as an event, with a single attribute represented by the observable mentioned above. Figure 12 shows the event created on MISP and Figure 13 shows its attribute. It should be noted that, due to how TheHive implements the interaction with MISP, the event will not be



List emails						
UID	Date	From	Subject	Message	Subject of the email to analyze	
63	Sat, 21 Aug 2021 11:33:58 +0200	[REDACTED]	Strana mail	Mentre ero in ufficio stamattina ho ricevuto questa mail che sembra abbastanza sospetta. È sicuro aprirla?	Il tuo deposito da 13.382,48€ 🤖🤖🤖	Analyze
64	Sat, 21 Aug 2021 12:15:35 +0200	[REDACTED]	Email riguardante il vaccino	Ieri ho fatto la mia seconda dose di vaccino Pfizer, ma non sono certo del fatto che questa sia la mail ufficiale per ottenere il certificato internazionale di vaccinazione.	Get your Pfizer vaccine certificate	Analyze
65	Sat, 21 Aug 2021 12:36:16 +0200	[REDACTED]	Email da Unicredit	Qualcuno sta davvero tentando di accedere al mio conto bancario?	Di recente abbiamo notato uno o più tentativi di accesso al tuo account Unicredit	Analyze
66	Sat, 21 Aug 2021 13:11:32 +0200	[REDACTED]	Email da Microsoft	È davvero da parte di Microsoft?	Nuove app connesse al tuo account Microsoft	Analyze
67	Sat, 21 Aug 2021 13:41:22 +0200	[REDACTED]	Email occasione prestito	Ho ricevuto questa email dove mi invitano a richiedere un prestito con un tasso speciale, è un'offerta legittima?	Cosa stai aspettando? Non puoi perdere questa occasione	Analyze

Figure 8: List of emails to analyze



Figure 9: Analysis progress

Observable List (14 of 14)

Flags	Type	Value/Filename	Dates	S. C. U.	Actions
<input type="checkbox"/>	file	Il tuo deposito da 13[,382,48€ 🏠🏠🏠🏠].jemi 📧 email: email_sample ⚙️ No reports available	S. Saturday 21/08/2021 17:06:31 +02:00 C. Saturday 21/08/2021 17:06:31 +02:00		⚙️
<input type="checkbox"/>	url	hxxp://beautifulpartialthings[.]com/a1e1f2c2kav7ipn0rb/6fedd97[.]png 📧 email: email_body ⚙️ No reports available	S. Saturday 21/08/2021 17:06:31 +02:00 C. Saturday 21/08/2021 17:06:31 +02:00		⚙️
<input type="checkbox"/>	url	hxxp://beautifulpartialthings[.]com/6KJ4K6[.]jsp?sdVvOEqAXqWtB=QVRdcBDBiLosj09s0y8800xcal01gZr20107n08013reeu345t 📧 email: email_body ⚙️ No reports available	S. Saturday 21/08/2021 17:06:31 +02:00 C. Saturday 21/08/2021 17:06:31 +02:00		⚙️
<input type="checkbox"/>	url	hxxp://beautifulpartialthings[.]com/6KJ4K6[.]phtml?CSLUjpdqhbRDH=IFawnUXOzCABc19s0y8800xcal01gZr20107n08013reeu345t 📧 email: email_body ⚙️ No reports available	S. Saturday 21/08/2021 17:06:29 +02:00 C. Saturday 21/08/2021 17:06:29 +02:00		⚙️

Figure 10: Observables added to the case

published by default and will have to be published manually by the analyst. Nevertheless, the event does not need to be published for the *MISP Search* analyzer to find a match between an observable in a case and an attribute of an event. Then, ThePhish sends the verdict via email to the user thanks to the *Mailer* responder. Finally, the case is closed. Figure 14 shows that the case has been closed after five minutes and resolved as “True Positive” with “No Impact”, which means that the attack has been detected before it could do any damage. Once the case is closed,

★ [DOMAIN]: imageplants[.]com

Cyberprotect:ThreatScore="not in database"
Malwares:Score="0 results"
Fortiguard:URLCat="Phishing"
Maltiverse:Report="neutral"
OTX:Pulses="0"
OPSWATMetadefender-Cloud.Reputation="0/0"
PT:Malware="False"
DomainMailSPF_DMARC:SPF="no"
DomainMailSPF_DMARC:DMARC="no"
Threatcrowd:votes="0"
CCF:C2 Search="0 hits"
URLhaus:Search="No results"
MISP:Search="0 events"
urlscan.io:Scan="Overall Score:0"
Onyphe:Threat="No threat found"

Basic Information Sharing Responders

TLP TLP:AMBER

Date added Saturday 21/08/2021 17:06:24 +02:00

Is IOC ★

Has been sighted 🔗

Ignored for similarity 🔗

Tags email email_header email_header_From

Description Found in the From field of the email header

Figure 11: Observable marked as IoC

[ThePhish] Il tuo deposito da 13.382,48€

Event ID	614
UUID	c0ae19ca-43d9-4750-84a8-27cbe5080de2 + 📄
Creator org	ThePhish
Owner org	ThePhish
Creator user	sync_user@thephish.test
Tags	🌐 tlp:amber x 🌐 + 👤 +
Date	2021-08-21
Threat Level	— Medium
Analysis	Initial
Distribution	Your organisation only 📄 🔔 🔗
Info	[ThePhish] Il tuo deposito da 13.382,48€
Published	No
#Attributes	1 (0 Objects)
First recorded change	2021-08-21 15:11:38
Last change	2021-08-21 15:11:38
Modification map	
Sightings	0 (0) - restricted to own organisation only. 🔑

Figure 12: Event created on MISP

Date ↑	Org	Category	Type	Value	Tags	Galaxies	Comment
2021-08-21		Payload delivery	domain	imageplants.com	tlp:amber		Found in the From field of the email header

Figure 13: Attribute of the MISP event

Case # 108 - [ThePhish] Il tuo deposito da 13.382,48€

ThePhish 08/21/21 17:06 08/21/21 17:11 as **True Positive with No Impact** 5 minutes

Sharing (0) | Reopen | Flag | Merge | Remove | Export (0) | Responders

Details | Tasks 3 | Observables 14 | TTPs | **ThePhish result**

Basic Information | Sharing (0) | Require Action | Flag | Reopen | Responders

Title	ThePhish result	Start date	Saturday 21/08/2021 17:11:00 +02:00
Group	default	Close date	Saturday 21/08/2021 17:11:44 +02:00
Assignee	ThePhish	Duration	Closed after <i>a few seconds</i>

Description

mailto: [REDACTED] Thanks for your submission. The e-mail with subject [Il tuo deposito da 13.382,48€] you submitted has been classified as Malicious

Figure 14: Task and case closed

the verdict is available for the analyst on the web interface together with the entire log of the analysis progress, as shown in Figure 15. The above-depicted case was related to a phishing email, but a similar workflow can be observed when the analyzed email is classified as “safe”. On the other hand, when an email is classified as “suspicious”, the verdict is only displayed to the analyst on the web interface. At this point the analyst needs to use the buttons on the left-hand side of the page to use TheHive, Cortex and MISP for further analysis. This is because the analysis has not been completed yet and so the user is only notified that the analysis of the email that he forwarded to ThePhish has been started. Indeed, the last task and the case have not been closed yet since they need to be closed by the analyst himself once he elaborates a final verdict. The analyst can view the full reports of all the analyzers on TheHive and Cortex, also the ones returned by the analyzers that only return an “info” maliciousness level. These analyzers, in fact, are not considered during the computation of the verdict but they can be

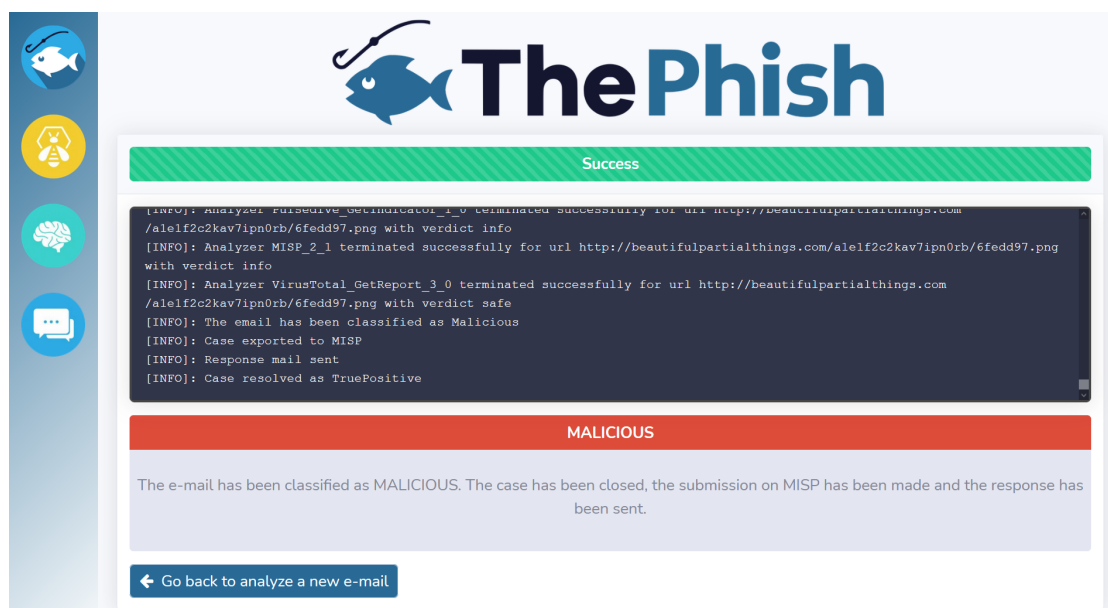


Figure 15: Final verdict (malicious) shown to the analyst

useful for getting more information about an observable. Moreover, in case this revealed not to be enough, the analyst could also download the EML file of the email and analyze it manually. When the analyst terminates the analysis, he can populate the body of the email to send to the user in the last task, start the *Mailer* responder, export the case to MISP if the verdict is “malicious” by clicking on the “Export” button and then close the case.

To do this demonstration, 45 different analyzers have been enabled, 12 of which included in the `analyzers_level1_conf.json` file to modify their maliciousness level. Moreover, the `whitelist.json` file has been populated to avoid false positives. Both these files have been populated based on many tests on different phishing and safe emails. However, changing the enabled analyzers and the content of the configuration files may make ThePhish give different results. This means that a universally correct configuration does not exist, but it is highly dependant on the organization’s needs and on whether the observables present in the email are known by the enabled analyzers. Hence, in order to use ThePhish, an analyst must first test it to configure it properly, but he must also keep this configuration always up to date.

B. Pull requests to TheHive4py

TheHive offers many API endpoints that allow performing the majority of the actions that can be performed from the web interface. However, the functions provided by TheHive4py do not cover all those API endpoints yet. In order to make ThePhish able to use all the functionalities it needs, the following two functionalities have been added to TheHive4py:

- *Export to MISP*: Exports a case to MISP with all of its observables marked as IoC.

- *Run a responder*: Launches a responder against alerts, cases, tasks, task logs or observables by its ID.

Since TheHive4py is an open-source project available on GitHub, two pull requests have been made to make these functionalities available to the entire community. The pull requests have been accepted and included in the 1.8.0 TheHive4py milestone.

C. ThePhish deploy

In order to use ThePhish, it is necessary to deploy TheHive, Cortex and MISP instances along with the services needed for them to work. The complete procedure needed to set up those services for a production-ready environment can be found in the official guides of TheHive, Cortex and MISP. However, in this case, Docker [42] and Docker Compose have been used to deploy the entire application. In order to facilitate the deployment procedure, TheHive Project has made available Docker images for both TheHive and Cortex. Moreover, several Docker templates have been made available as well. Those templates contain a `docker-compose.yml` file used to configure the containers that constitute the application. It should also be noted that the Cortex neurons are started as docker containers themselves, which means that their images are pulled from Docker Hub the first time they have to be executed and then every time a neuron is re-executed, a container is created based on that image and is destroyed at the end of the execution. ThePhish has been tested on a virtual machine running Ubuntu 20.04 LTS with Docker Engine 20.10.8 and Docker Compose 1.29.2, using a modified version of a Docker template provided by TheHive Project [43]. It not only uses TheHive and Cortex containers but also MISP, MySQL, Redis, Apache Cassandra and Elasticsearch containers, which are not provided by TheHive Project. Even though this template does not provide the full configuration options, it is enough to demonstrate how ThePhish works. The original `docker-compose.yml` file has been edited so as to remove unused services and add another container used to run ThePhish. The versions of the services used are listed below:

- Apache Cassandra 3.11
- TheHive 4.1.9
- Elasticsearch 7.11.1
- Cortex 3.1.1
- Redis 6.2.5
- MySQL 8.0.26
- MISP 2.4.148

ThePhish has been made available on GitHub as an open-source project under the AGPL license at the repository *emalderson/ThePhish* [44]. It is possible to refer to that repository for a complete installation and configuration guide.