Systematic analysis of phishing websites

Michelle Krejci, mickr592
Linköpings Universitet
mickr592@student.liu.se

Erik Mattfolk, erima882
Linköpings Universitet
erima882@student.liu.se

***Abstract*—Social engineering, more specifically phishing is not always easy to detect. The techniques and methods for this are always developing - but so is the phishing. The aim of this project is to extract data one from on form of phishing, namely phishing websites, and briefly analyse the collected data. This is done semi-manually through the help of a script that collects verified phishing websites though the PhishTank API and then asks the user several questions about the website.**

I. Introduction

Phishing websites are a specific form of phishing, they try to impersonate another real website to make the user interact with it in a specific way that allows the site to collect information on the user [1]. Phishing websites are always evolving, just like phishing filters improve to be able to catch more phishing websites the websites in turn update how they work to avoid the filters. To be able to keep up with these changes data must be collected on active phishing websites. Then the data can be analysed for similarities and patterns. But to be able to do that in a efficient way one needs a way to identify active phishing websites and a method to extract information about what that websites requests as well as what website it pretends to be - if any. This is the goal of the project. To summarize it can be defined by three questions.

- How can active phishing websites be identified?
- How can data from the phishing websites be extracted efficiently?
- Additionally, what does the collected data say about the phishing websites?

To answer the questions a script will be created that collects active phishing websites through the PhishTank API and then collects data about the websites semi-manually by asking the user several questions about the website. The hope is that this script makes it easier to collect information on these websites and thus makes it easier to keep up with changes and protect users from being tricked.

II. Background

A. Phishing

To understand what this report and project is about we first need to know what phishing is. Phishing is a form of social engineering attack that focuses on extracting some important information from the user. This can be done in many different ways such as emails containing malicious links or website that masquerade as another (legit) website and ask you to login. A real world example could be a text message that appears to be coming from the bank telling the person in question that they need to verify their account by clicking on the link in the message and signing in [2] [3].

Today attackers have started exploiting CAPTCHA features to carry out their attacks. This is a form of human verification where the user is expected to identify some obfuscated letters in pictures. It is considered to be effective in protecting against bots, but it also seems to be able to protect phishing websites from being identified by anti-phishing websites [4]. A study done in 2020 on this topic submitted 35 URLs of phishing websites using Google reCAPTCHA to anti-phishing websites and none of these were detected as malicious [5].

In the Anti-Phishing Working Groups (APWG) 3rd quarterly trend report from 2021 they reported that the number of phishing attacks had doubled from early 2020. Most phishing attacks were towards the software-as-a-service and webmail sector that made up 29.1% of the attacks. Attacks towards financial institutions and payment providers made up 24% of the attacks and the crypto market made up 5.5% of the attacks [6].

B. PhishTank

PhishTank is a community site for submitting, verifying, tracking and sharing phishing data. Users submit suspected phishes and others users can vote on if it is an actual phishing website or not. Voting is based on the actual people voting, less people are necessary to verify a website if those people have had a high success rate before [7].

III. Method

This section of the report goes over the theoretical and practical methods used during the project.

A. Theoretical Methods

This project consists of mainly practical work. The only theoretical method that was used was reading several scientific papers on the subject of phishing and phishing blacklists. This was done to gain a basic understanding of what the goal of a phishing website is and what typically characterizes a phishing website.

Some research was also done to find a good anti-phishing website to use. There are three big ones out there: PhishTank, Google Safe Browsing and OpenPhish. For this project PhishTank was chosen because it is freely and easily available.

B. Practical Methods

The first practical step of the project was to look at several phishing websites and determine what they had in common. From this several yes-or-no questions were created.

1) Is the site still accessible or has it been blocked?
2) Is the site a duplicate of one you have analysed before?
3) Does the site look like:
   - Bank, Post service, Rewards site, Cryptocurrencies, Stock trading, University, Other
4) Does the site contain a form? If yes, what information does the form ask for?
   - Name, ID-number, Bank information, E-mail, Password, Other
5) Does the site want you to sign in using some sort of personal identification such as BankID?
6) Other

The second step was to create a script that used the PhishTank API and the questions to identify websites and extract information from them. The script was to function as such: The script collects a large number of verified phishing websites through the PhishTank API in an array. The website at index zero is then selected first and opened in a web browser and the the user is asked the questions above. For questions 1, 2 and 5 the user is prompted to answer "yes" by pressing "y" then "Enter" or "no" by only pressing "Enter". For questions 3 and 4 the user can select answers from a pre-made list by typing the list index and pressing "Enter", or by typing in a description and pressing "Enter". Question 6 is answered by optionally typing in something and pressing "Enter".

Time spent analyzing is recorded from when it is a known valid page (after the second question) until the final question about the page is answered. The results from the questions is saved locally and if the website is accessible - i.e not blocked by Google or such - a copy of the website is saved locally as well. When the script is exited it parses the information collected and outputs the statistics, for example how many websites contained forms or were pretending to be banks. The answers to all questions asked are stored and parsed.

The third and final step was to, with the help of the script, extract information from a large number of websites and do a preliminary analysis on the resulting data. The script can be seen as a whole in appendix A.

IV. Results and Analysis

As can be seen from figure 1 there were a lot of inaccessible pages, 141 of the 248 pages were actually inaccessible. Another 49 were duplicates of already processed pages. In the end only 58 of the processed pages were viable. Out of the viable pages 48 had a form on the website and 46 had a known appearance. The most asked information is these forms was email and password as can be seen in figure 3. Name and crypto wallet information also occurred on around ten websites. From figure 2 it can be deduced that the most popular appearances that the websites tried to mimic were cryptocurrencies, banks and social media. A total of 1214 seconds was spent analyzing and recording data for the 58 valid pages. A visual representation of the distribution of the results can be found in appendix B.

Statistic	Amount
Total pages	248
Total valid pages	58
Total duplicate pages	49
Total inaccessible pages	141
Total pages with forms	48
Total pages with known appearance	46
Total time analyzing pages	1214s

Fig. 1. Statistics

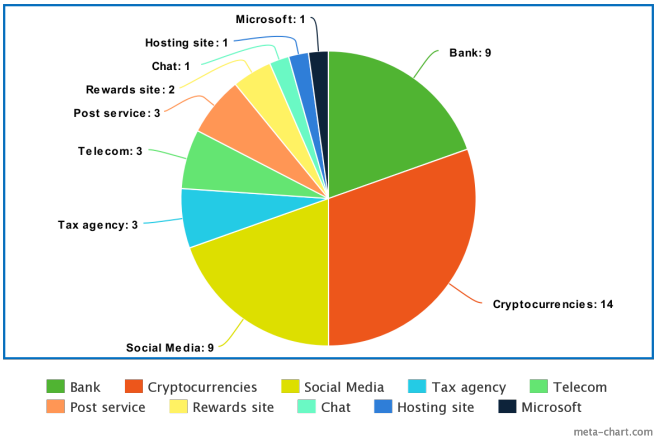

Fig. 2. The frequency of known appearances

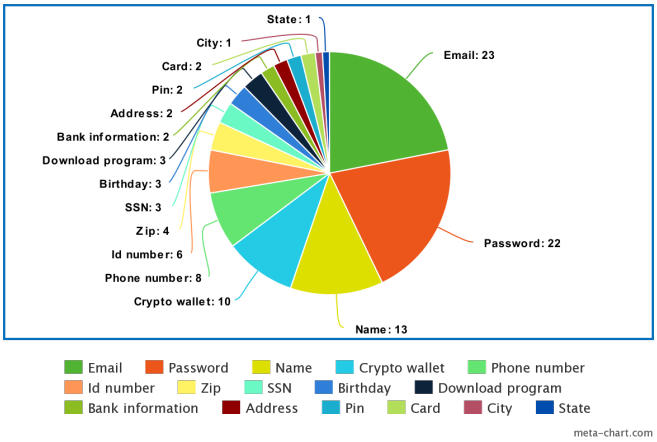

Fig. 3. Form statistics

From the results it can be concluded that a majority of the viable websites included a form that in most cases asked for a e-mail and a password. This seems logical since the goal of phishing is to get some information out of the user. For example their login to an website. As mentioned before it can also be seen that out of the valid websites with a known

appearance the most popular appearance was some sort of cryptocurrency website. Gaining access to a crypto wallet is currently not very difficult—only the secret key is needed—and thus it makes sense that this appearance is popular.

V. Discussion

This section deals with analyzing the results in the previous section. The efficiency and possible improvements of the script will also be discussed.

It should be stated that the collected statistics does not indicate phishing sites in general. For example, once a phishing site ends up in the PhishTank database, the hosting site for the phishing site may shut it down (likely due to monitoring of the PhishTank API). This leads to a inflated number of inaccessible pages and possibly a larger proportion of self-hosted phishing sites among the valid pages recorded.

Still, many pages can be processed using the script in a timely manner. On average the time spent analyzing a valid page was 20.9 seconds. These 20.9 seconds does not include loading the page in a browser, clicking through various warnings displayed in the browser, answering whether the site is accessible or a duplicate and downloading the page afterwards. This number should not be confused with the time it takes to find and process a valid page, as that time may be up to an order of magnitude larger.

Even though phishing websites were analyzed beforehand, there were some missing and unused alternatives when it came to answering questions three and four. Some notable missing alternatives were social media and telecom for question three and crypto wallet and phone number for question four. Question three had two unused alternatives: stock trading and university. The clarity of these options can also be discussed as their meaning is ambiguous in some cases. For instance, when the phishing site asks for a zip code, is it an address or only a zip? Similarly, if the phishing site has the appearance of Facebook, should it be recorded as "Facebook" or just "social media"? It is for the user of the script to decide, which leads to increased complexity merging answers from different people as they may differ. This also affects the scalability of using the script.

The script could be improved by gathering data from a larger number of anti-phishing websites to possibly increase the number of still accessible sites. Another aspect that could be further improved is the categorization of questions three and four to increase clarity and reduce the effect of human bias on the results. This bias cannot be removed completely since the parsing of the phishing websites is done by a human. Greater objectivity could be achieved by decreasing the human interaction needed with the script. This could be done by parsing the html structure of the websites and extracting form- and input-tags from it.

VI. Related work

A paper published in 2019 looked at detecting phishing websites trough the use of deep convectional neural networks. This method could automatically detect phishing websites during the earlier stage and had a accuracy of 99% [8].

Another approach proposed in a paper released in 2021 looked at using Fuzzy logic in order to detect phishing websites. It suggests the use of smooth logic and machine learning algorithms to define several factors of the phishing websites. According to the paper there are a total of 30 attributes of phishing websites that can be used for phishing detection with a high success rate [9].

VII. Conclusions

Active phishing websites can be identified by gathering websites from anti-phishing websites that have already identified them trough their filters, such as PhishTank.

Data from these websites can then be efficiently extracted trough the creation of a semi-manual script that opens phishing pages automatically for the user and asks them several questions. There are still improvements to be made on the scripts formulation, such as drawing data from more sources to gather a larger information pool since websites that are registered by PhishTank and tagged as phishing attempts are quickly shut down, or are duplicates of each other. Refining the categorization of the websites would also be advised to reduce human bias.

From the analysed websites the conclusion can be drawn that forms are common amongst phishing websites. Most common are forms that require the users e-mail and password. In addition a large amount of the websites were copies of legitimate websites. The most popular appearances that the websites tried to mimic were cryptocurrencies, banks and social media.

References

[1] Oluwatobi Ayodeji Akanbi, Iraj Sadegh Amiri, and Elahe Fazeldehkordi. Chapter 1.2 - problem background. In Oluwatobi Ayodeji Akanbi, Iraj Sadegh Amiri, and Elahe Fazeldehkordi, editors, *A Machine-Learning Approach to Phishing Detection and Defense*, pages 1–8. Syngress, Boston, 2015.

[2] Imperva. Learning Center. What is phishing: Attack techniques & scam examples, June 2021.

[3] Athulya A.A. and Praveen K. Towards the detection of phishing attacks. In *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*, pages 337–343, 2020.

[4] Waqas. Threat actors using captcha to evade phishing, malware detection. Hackred, August 2021.

[5] Andrzej Duda Sourena Maroofi, Maciej Korczyński. Are you human? resilience of phishing detection to evasion techniques based on human verification. ACM digital library, October 2020.

[6] Greg Aaron. Apwg phishing activity trends report, 3rd quarter 2021. APWG, November 2021.

[7] PhishTank. Frequently asked questions.

[8] K. M. Zubair Hasan, Md Zahid Hasan, and Nusrat Zahan. Automated prediction of phishing websites using deep convolutional neural network. In *2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2)*, pages 1–4, 2019.

[9] M. D. Bhagwat, P. H. Patil, and T. S. Vishawanath. A methodical overview on detection, identification and proactive prevention of phishing websites. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pages 1505–1508, 2021.

APPENDIX A
CODE FOR GATHERING DATA FROM PHISHING WEBSITES

```
import os
import json
import signal

from time import time


def read_processed():
   try:
      with open("processed", "r") as f:
         processed = json.load(f)
   except:
      print("No processed pages")
      processed = {}
   return processed


def write_processed(processed):
   with open("processed", "w") as f:
      processed_pages = json.dump(processed, f)


def default_form():
   # Form
   # {
   #    "accessible": bool,
   #    "duplicate": bool,
   #    "appearance": str,
   #    "form": list[str],
   #    "personal_identification": bool,
   #    "other": str,
   # }
   return {
      "accessible": False,
      "duplicate": False,
      "time": -999999.0,
      "appearance": "",
      "form": [],
      "personal_identification": False,
      "other": "",
   }


processed_pages = read_processed()

with open("verified_online.json", "r") as f:
   to_process = json.load(f)


def interrupt_handler(_signum, _stackframe):
   valid_pages = list(
      filter(
         lambda p: p["accessible"] and not p["duplicate"], processed_pages.values()
      )
   )
   duplicate_pages = list(filter(lambda p: p["duplicate"], processed_pages.values()))
   inaccessible_pages = list(
      filter(lambda p: not p["accessible"], processed_pages.values())
   )
   pages_with_forms = list(filter(lambda p: p["form"] != [], processed_pages.values()))
   pages_with_personal_identification = list(
      filter(lambda p: p["personal_identification"], processed_pages.values())
   )
```

```
    total_time = sum(p["time"] for p in valid_pages)

    form_statistics = {}
    for p in pages_with_forms:
        for f in p["form"]:
            if f not in form_statistics:
                form_statistics[f] = 0
            form_statistics[f] += 1

    appearance_statistics = {}
    for p in valid_pages:
        f = p["appearance"]
        if f == "":
            continue

        if f not in appearance_statistics:
            appearance_statistics[f] = 0
        appearance_statistics[f] += 1

    # Appearence counts
    # Form counts
    # Personal identification counts
    # Other comments

    print()
    print(f"====== Statistics ======")
    print(f"Total pages: {len(processed_pages)}")
    print(f"Total valid pages: {len(valid_pages)}")
    print(f"Total duplicate pages: {len(duplicate_pages)}")
    print(f"Total inaccessible pages: {len(inaccessible_pages)}")
    print(f"Total pages with forms: {len(pages_with_forms)}")
    print(
        f"Total pages with personal identification: {len(pages_with_personal_identification)}"
    )
    print(f"Total time analyzing pages: {int(total_time)} sec")
    print(f"Form statistics:")
    for field, count in sorted(form_statistics.items(), key=lambda i: -i[1]):
        print(f"{field}: {count}")
    print(f"Appearance statistics:")
    for field, count in sorted(appearance_statistics.items(), key=lambda i: -i[1]):
        print(f"{field}: {count}")
    exit(0)


signal.signal(signal.SIGINT, interrupt_handler)

print(f"{len(processed_pages)} pages have been processed")
print(f"{len(to_process) - len(processed_pages)} pages to process")
print()
print("Starting site processing")

for page in to_process:

    phish_id = page["phish_id"]
    url = page["url"]

    if phish_id in processed_pages:
        continue

    os.system(f"brave {url} &> /dev/null")

    form = default_form()

    print()
    form["accessible"] = input("Is the site accessible? y/N\n> ") in ["y", "Y"]

    if not form["accessible"]:
```

```
    processed_pages[phish_id] = form
    write_processed(processed_pages)
    continue

print()
form["duplicate"] = input(
    "Is the site a duplicate of one you have analyzed before? y/N\n> "
) in ["y", "Y"]

if form["duplicate"]:
    processed_pages[phish_id] = form
    write_processed(processed_pages)
    continue

start_time = time()

print()
print("Does the site look like:")
print("1. Bank")
print("2. Post service")
print("3. Rewards site")
print("4. Cryptocurrencies")
print("5. Stock trading")
print("6. University")
print("If other, write a short description or leave blank")
appearance = input("> ")

form["appearance"] = {
    "1": "bank",
    "2": "post service",
    "3": "rewards site",
    "4": "cryptocurrencies",
    "5": "stock trading",
    "6": "university",
}.get(appearance, appearance.lower())

print()
print("Is there any form fields that matches the following?")
print("1. Name")
print("2. ID-number")
print("3. Bank information")
print("4. E-mail")
print("5. Password")
print("If other, write a short description. Leave blank when done")

while (field := input("> ")) != "":
    form["form"].append(
        {
            "1": "name",
            "2": "id number",
            "3": "bank information",
            "4": "email",
            "5": "password",
        }.get(field, field.lower())
    )

print()
form["personal_identification"] = input(
    "Does the site want you to sign in using some sort of personal indentification such as
        BankID? y/N\n> "
) in ["y", "Y"]

print()
form["other"] = input("Other relevant info about the site\n> ")

end_time = time()
```

```
    form["time"] = end_time - start_time

    processed_pages[phish_id] = form
    write_processed(processed_pages)

    print(f"Dowloading the site to folder {phish_id}")
    os.system(f"mkdir {phish_id}")
    os.system(
        f"cd {phish_id}; wget --recursive -np -nc -nH --cut-dirs=4 --timeout=1 --random-wait
            --wait 1 -e robots=off {url}"
    )
```

APPENDIX B
VISUALISATION OF RESULTS

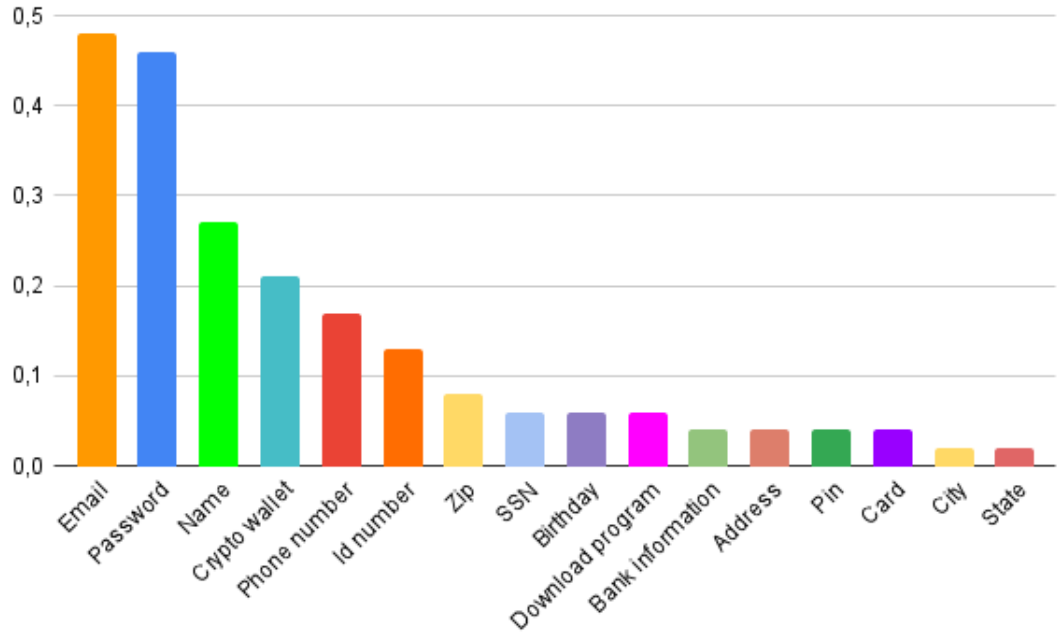

Fig. 4. The frequency of requested information in the identified forms

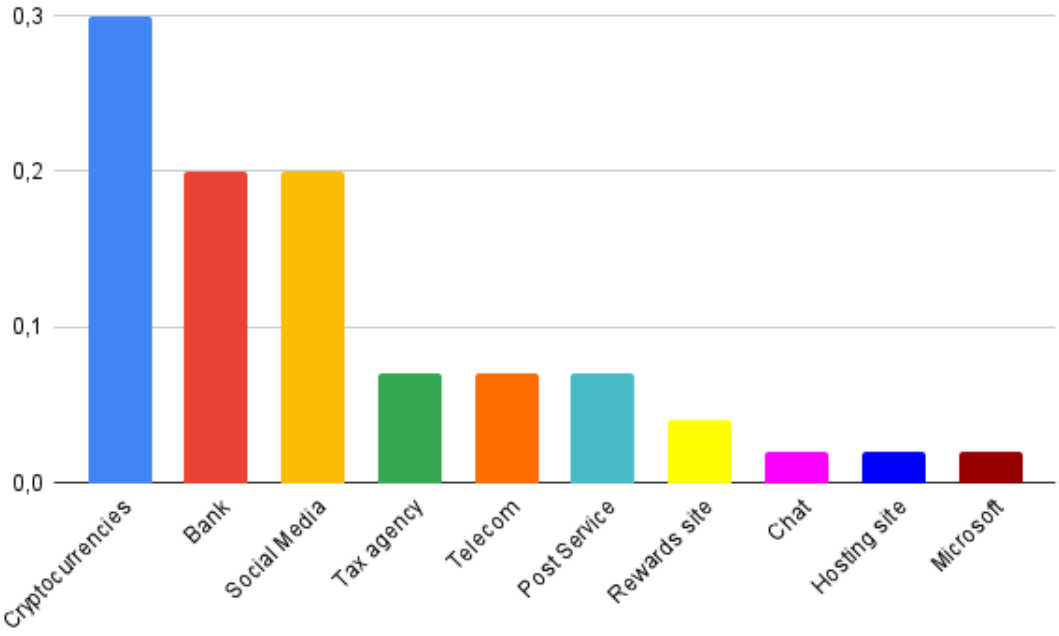

Fig. 5. The frequency of known appearances