Year 2022-2023

Project Report

Malware Project: “Bypass antivirus with malware programming and forensics methodology”

Project realised by

Locqueneux Owen

Felicio Thomas

Fen-Chong Arthur

Under the supervision of

Nathan Duverger

To

ESME Sudria - Engineering School of Mechanics and Electricity

Table des matières

Acknowledgements

First of all, we would like to express our special thanks and appreciation to the following people who gave us the opportunity to carry out this cybersecurity project:

• Mr. Nathan Duverger, our supervisor, for his patience and valuable advice, who helped us completing this first step of our project and who taught us many useful things throughout these three months.

• Mr. Driss ESSAYED MESSAOUDI, Major Cybersecurity Manager, for his professional cooperation and giving us the opportunity to achieve this project.

Introduction and context

Introduction

The last couple of years have been far from ordinary, both for cybersecurity and business in general. The COVID-19 pandemic has permanently changed how business is done, and cybercriminals have adapted to these changes, tailoring their tactics to the new reality.

While 2021 and 2022 have been exceptional years for cyberattacks, there is little indication that things will return to “normal” in 2023. Cyber threat actors have tried new tactics and techniques, found them to be successful, and added them to their core arsenal.

In 2022, several cyberattack campaigns and cyber threat actors became household names as the impacts of cyberattack were felt far beyond their target companies. The modern threat landscape is composed of bigger, flashier, and higher-impact attacks as cybercrime becomes increasingly professionalized and cyber threat actors look to extract maximum value or impact from their attacks.

Context

As engineering students majoring in Cybersecurity, we have to master different aspects of this field. For that, this project is perfect for us because it covers malwares, programming; forensics; antiviruses and phishing attacks. So, it’s an interesting project with many works and research to do.

The end goal of this project is to bypass windows defender with a malware. It was carried out in previous years by students from ESME Sudria without being completed. Our goal is to finish it completely by the end of March.

Our group consists of three students in their last year of engineering school: Arthur FEN-CHONG; Owen LOCQUENEUX; Thomas FELICIO.

Time management

During those first 3 months, we organized our work in different tasks:

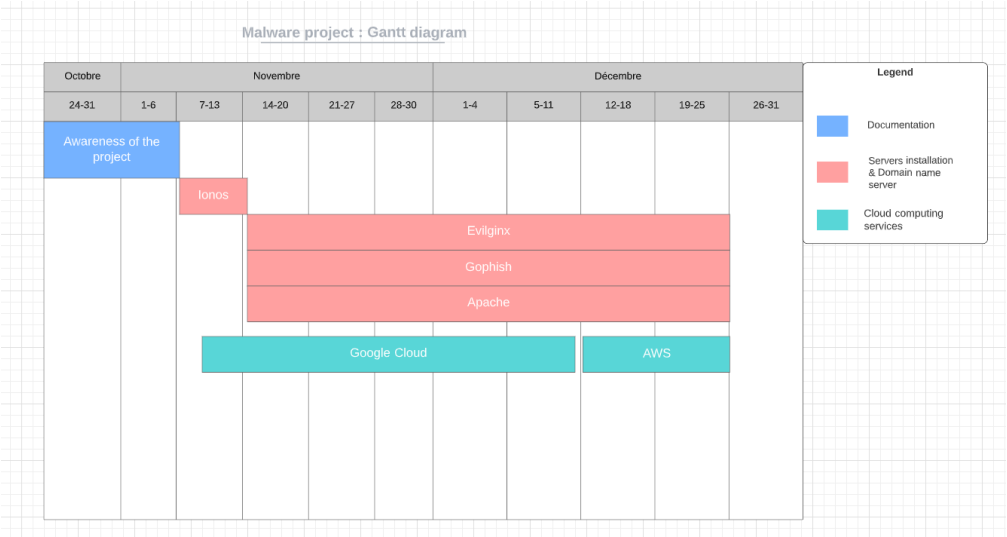
First, we had to get to know the project, its topic, and the different steps to do.

Then, once the awareness of the project was done, we created an account on IONOS and we purchased a domain name that we named « aot-project.com » After that, we did some research about DNS records; apache2, gophish server and evilginx2 configurations.

We had to install a Linux machine with Google Cloud at first, unfortunately we've met some issues with google cloud, so we switch to another cloud platform named AWS. We also searched information about PE format Files and Windows API concerning the malware.

Finally, the last step of this first part consisted of trying to send an email that redirects to a fake evilginx2 office365 page to our supervisor with our AWS instance.

Below is the repartition of these tasks during those 3 months:



Objectives and Issues

Objectives

The main goal of this project is to bypass a strong antivirus (here windows defender) with a malware. To do so we will use phishing attack that will steal credentials from our target.

The last three months our goal was to complete the first part given to us by our supervisor:

Part 1:

- Installation of an Amazon EC2 instance with Ubuntu 20.04 because we had some errors with kali on Google Cloud.

- Learn about GitHub and git commands, what's DNS and DNS records and what's SSL/TLS.

- Find out about the SMTP, IMAP, POP protocols.

- Take a domain name (1&1Onos - 1€ with ssl certificate): "aot-project.com" in our case

- Configure the DNS records with Ionos

- Install and configure apache, a smtp server (postfix), a gophish server and evilginx2 (evilgophish) on our instance.

- Install an SSL/TLS certificate on our web server (the one from IONOS).

- Try to send me an email that redirects to a fake evilginx2 office365 page to our supervisor.

Part 2 :

- Search what's the structure of a PE file.

- What's a windows API

- Search how to compile in C++ a program that uses a windows api to make a messagebox pop and how to compile this program as exe and dll.

Issues

Throughout this first part of the project, we encountered various problems with the tasks requested by our supervisor Nathan Duverger.

The first problem encountered was to work all three of us on one machine, whether on Google Cloud or AWS. To solve this problem, we each had to generate a key so that we could connect in SSH to the machine and work remotely. For this we had to warn the owner of the machine to turn it on and provide us with its IP or combine working hours. The reason we opted for this approach was to avoid wasting too much time doing the same things each on our own machine and thus favor group work and speed.

Another problem encountered on Google Cloud first then on AWS with our EC2 instance was the configuration of evilgophish as seen below in the screen. We had the same error on the Kali even with our supervisor's help. So, following our supervisor's instructions, we decided to install an Ubuntu 20.04 and evilgophish on it, instead of a kali and at that time the problem was solved. If that hadn't been the case, we would have opted to use evilginx2 and gophish individually rather than evilgophish which combines the two directly.

```
sqlite3-binding.c: In function ‘sqlite3SelectNew’:
sqlite3-binding.c:128049:10: warning: function may return address of local variable [-Wreturn-local-ad
dr]
128049 |   return pNew;
       |          ^~~~
sqlite3-binding.c:128009:10: note: declared here
128009 |   Select standin;
       |          ^~~~~~~
[evilgophish setup] [+] Configured gophish!
[evilgophish setup] [*] Configuring evilginx2
Failed to stop systemd-resolved.service: Unit systemd-resolved.service not loaded.
go: github.com/chzyer/logex@v1.1.10: Get "https://proxy.golang.org/github.com/chzyer/logex/@v/v1.1.10.
mod": dial tcp: lookup proxy.golang.org on [::1]:53: read udp [::1]:37073->[::1]:53: read: connection
refused
go: downloading github.com/chzyer/readline v0.0.0-20180603132655-2972be24d48e
go: downloading github.com/elazarl/goproxy v0.0.0-20190911111923-ecfe977594f1
go: downloading github.com/fatih/color v1.7.0
go: downloading github.com/go-acme/lego/v3 v3.1.0
go: downloading github.com/inconshreveable/go-vhost v0.0.0-20160627193104-06d84117953b
go: downloading github.com/miekg/dns v1.1.22
go: downloading github.com/mwitkow/go-http-dialer v0.0.0-20161116154839-378f744fb2b8
go: downloading github.com/spf13/viper v1.4.0
go: downloading golang.org/x/net v0.0.0-20200707034311-ab3426394381
go: downloading github.com/tidwall/buntdb v1.1.0
go: github.com/chzyer/logex@v1.1.10: Get "https://proxy.golang.org/github.com/chzyer/logex/@v/v1.1.10.
mod": dial tcp: lookup proxy.golang.org on [::1]:53: read udp [::1]:37073->[::1]:53: read: connection
refused
[evilgophish setup] [+] Configured evilginx2!
[evilgophish setup] [+] Installation complete! When ready start apache with: systemctl restart apache2
[evilgophish setup] [*] It is recommended to run all servers inside a tmux session to avoid losing the
m over SSH!

┌──(root㉿kali)-[/home/kali/evilgophish]
└─#
```

Error during evilgophish installation

Concerning the configuration of the servers we also had a problem with postfix, the electronic mail server. We always had a connection that failed when we tried to send an email to the esme mail for example in anticipation of the attack to be carried out, as seen below:

```
2022-11-24T11:26:19.185507+00:00 kali postfix/smtp[324020]: connect to esme-fr.mail.protection.outlook.com[104.47.0.36]:25: Connection timed out
-11-24T11:26:19.185889+00:00 kali postfix/smtp[324019]: connect to esme-fr.mail.protection.outlook.com[104.47.0.36]:25: Connection timed out
2022-11-24T11:26:19.189887+00:00 kali postfix/smtp[324019]: 66B6FB007A: to=<arthur.fen-chong@esme.fr>, relay=none, delay=967, delays=907/0.01/60/0, dsn=4.4.1, st
2022-11-24T11:26:19.191353+00:00 kali postfix/smtp[324020]: 6721A8007E: to=<arthur.fen-chong@esme.fr>, relay=none, delay=600, delays=540/0.02/60/0, dsn=4.4.1, st
2022-11-24T11:30:52.222756+00:00 kali postfix/qmgr[321616]: 4FB4CB0087: from=<arthur@kali>, size=316, nrcpt=1 (queue active)
2022-11-24T11:30:52.236974+00:00 kali postfix/error[325438]: 4FB4CB0087: to=<arthur.fen-chong@esme.fr>, relay=none, delay=365, delays=365/0.01/0/0.01, dsn=4.4.1,
25: Connection timed out)
2022-11-24T11:34:23.413616+00:00 kali postfix/postfix-script[326309]: stopping the Postfix mail system
```

Error : connection failed

Finally, the last problem encountered was the installation of a graphical user interface on our EC2 AWS instance to display evilgophish. We had to delete our ubuntu and reinstall it with a graphical interface because on the previous one it didn't work.

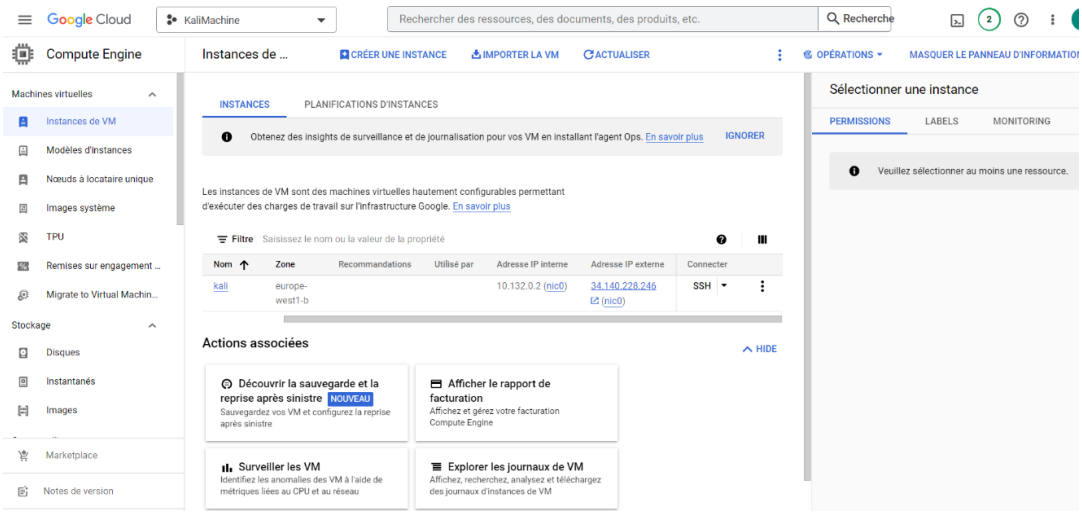
1. Servers installation and configuration

1.1. Google Cloud

At the beginning of our project, we decide to use google cloud platform:

Google Cloud is a suite of public cloud computing services offered by Google. The platform includes a range of hosted services for compute, storage and application development that run on Google hardware. Google Cloud services can be accessed by software developers, cloud administrators and other enterprise IT professionals over the public internet or through a dedicated network connection.

We create our VM instance which contains all servers



Google plateform

To access to our VM, we use SSH protocol

SSH connection

Once we have access to the shell, the steps below are necessary packages, to install, to have a Kali machine on the cloud:

Kali linux on GCP

1. Add repo /etc/apt/source.list (https://www.kali.org/docs/general-use/kali-linux-sources-list-repositories/)
2. apt update
3. gpg --keyserver pgpkeys.mit.edu --recv-key ED444FF07D8D0BF6
4. gpg -a --export ED444FF07D8D0BF6 | sudo apt-key add -
5. apt upgrade
6. intsall metapackage (https://www.kali.org/docs/general-use/metapackages/)

Thus, we can access as root to our machine.

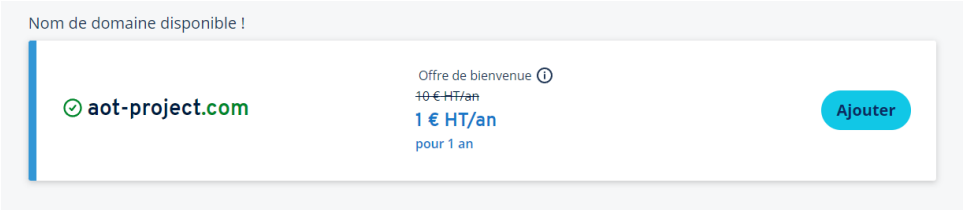
```
Linux kali 5.10.0-19-cloud-amd64 #1 SMP Debian 5.10.149-2 (2022-10-21) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Nov  4 22:18:36 2022 from 35.235.243.224
┌─(Message from Kali developers)
│
│ This is a minimal installation of Kali Linux, you likely
│ want to install supplementary tools. Learn how:
│ ⇒ https://www.kali.org/docs/troubleshooting/common-minimum-setup/
│
│ This is a cloud installation of Kali Linux. Learn more about
│ the specificities of the various cloud images:
│ ⇒ https://www.kali.org/docs/troubleshooting/common-cloud-setup/
│
└─(Run: "touch ~/.hushlogin" to hide this message)
owenlocqueneux@kali:~$ sudo -i
┌─(Message from Kali developers)
│
│ This is a minimal installation of Kali Linux, you likely
│ want to install supplementary tools. Learn how:
│ ⇒ https://www.kali.org/docs/troubleshooting/common-minimum-setup/
│
│ This is a cloud installation of Kali Linux. Learn more about
│ the specificities of the various cloud images:
│ ⇒ https://www.kali.org/docs/troubleshooting/common-cloud-setup/
│
└─(Run: "touch ~/.hushlogin" to hide this message)
┌──(root㉿kali)-[~]
└─# 
```

Our VM instance is launched

1.2 IONOS

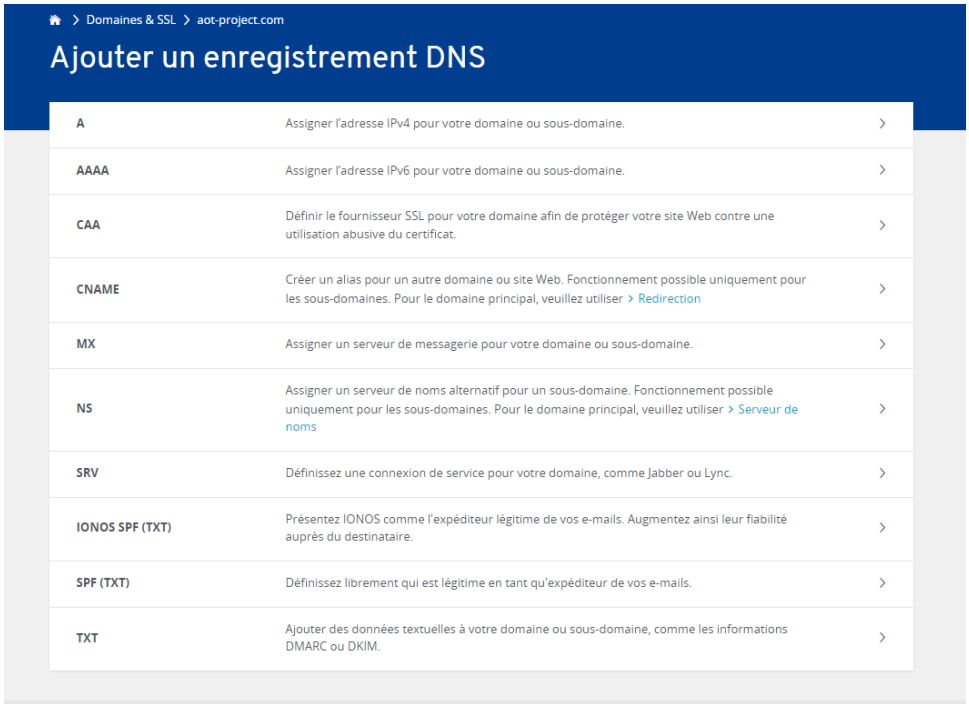
Ionos (formerly 1&1 IONOS and 1&1 Internet) is a web hosting company. It was founded in Germany in 1988 and is currently owned by United Internet. In addition to web hosting, it also provides domain registration, SSL certificates, email services, website builder packages, and cloud hosting, ...

For our project we use IONOS to create and name “aot-project.com” as our domain name:



Our domain name for the project

Then to redirect the domain name to the EC2 Apache server, we must modify the DNS records with the correct IP address. We can choose from the common records described below.



Finally, we assigned 3 A records to our VM ip address that will connect us to the Apache web page. We also created a TXT record that contains a certificate token.

TYPE	NOM D'HÔTE	VALEUR	SERVICE ▲	ACTIONS
A	@	100.24.99.197	-	
TXT	_acme-challenge	"ZM36u_aYpGZJ9Ojv4amZ5RKzoF1Z7lSpQLF...	-	
A	login	100.24.99.197	-	
A	www	100.24.99.197	-	

DNS records configured

1.3 Apache2

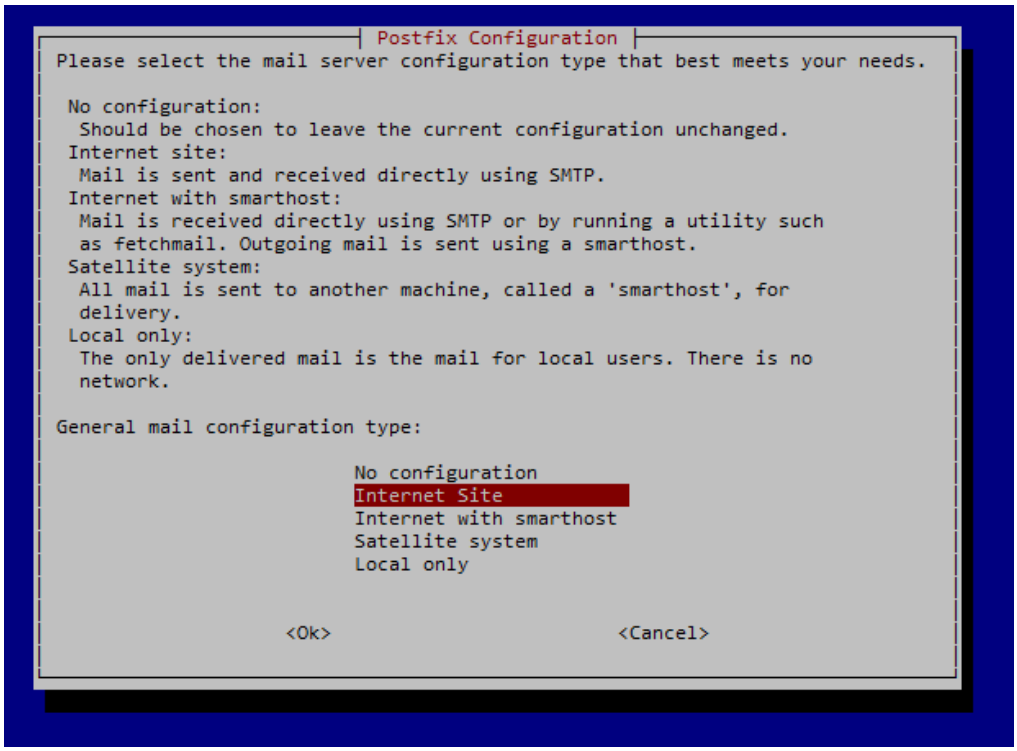
The Apache HTTP Server is a collaborative software development effort aimed at creating a robust, commercial grade, featureful, and freely available source code implementation of an HTTP (Web) server. It's easy to launch apache2:

```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl start apache2

┌──(kali㉿kali)-[~]
└─$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; disabled; preset: disabled)
     Active: active (running) since Mon 2022-12-26 17:03:54 UTC; 1s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 747 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 764 (apache2)
      Tasks: 55 (limit: 4674)
     Memory: 11.3M
        CPU: 38ms
     CGroup: /system.slice/apache2.service
             ├─764 /usr/sbin/apache2 -k start
             ├─765 /usr/sbin/apache2 -k start
             └─766 /usr/sbin/apache2 -k start

Dec 26 17:03:54 kali systemd[1]: Starting The Apache HTTP Server...
Dec 26 17:03:54 kali systemd[1]: Started The Apache HTTP Server.

┌──(kali㉿kali)-[~]
└─$
```

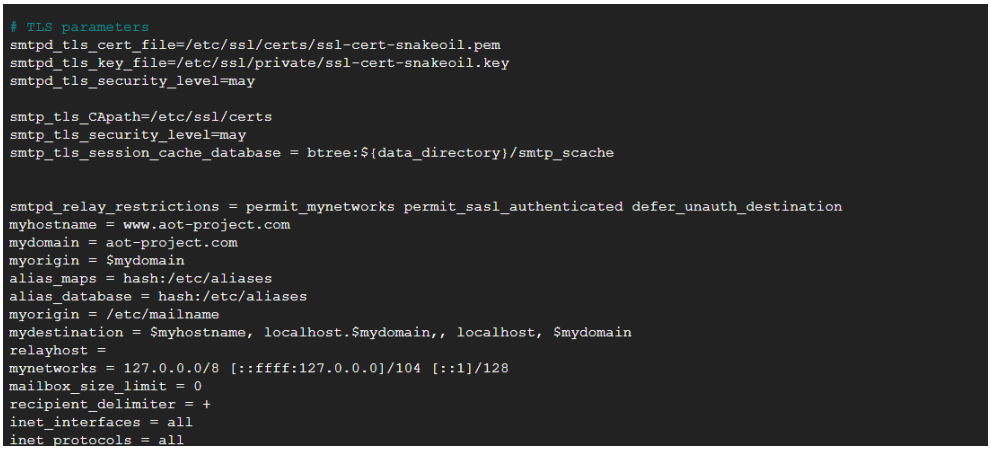
1.4 Postfix

Postfix is a Mail Transfer Agent (MTA) designed to determine routes and send emails. This cross-platform server is open-source, free, and suitable for installation on the majority of UNIX-like operating systems.

Postfix installation and configuration



Postfix initialisation



```
# TLS parameters
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
smtpd_tls_security_level=may

smtp_tls_CApath=/etc/ssl/certs
smtp_tls_security_level=may
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache


smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_destination
myhostname = www.aot-project.com
mydomain = aot-project.com
myorigin = $mydomain
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = $myhostname, localhost.$mydomain,, localhost, $mydomain
relayhost =
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
inet_protocols = all
```

Configuration file main.cf

Sending email within the command line use the command below:

Echo “message” | -s “mail subject” <adresse mail>

For our attack, Postfix is used to send an email with the gophish configuration to start the campaign, sending an email with a fake address to someone.

1.5 GoPhish

Gophish is an open-source phishing framework that makes it easier to perform phishing campaigns.

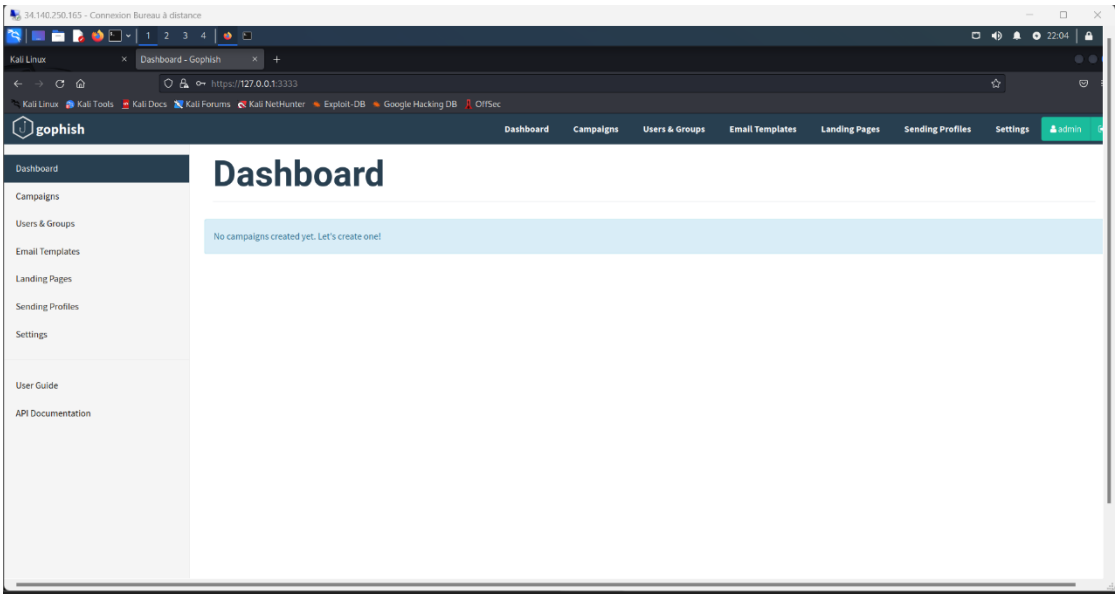
GoPhish Installation :

```
┌──(root㉿kali)-[/home]
└─# ls
gophish-v0.7.1-linux-64bit.zip  leagu  owenl  owenlocqueneux  test  thomas

┌──(root㉿kali)-[/home]
└─# unzip gophish-v0.7.1-linux-64bit.zip -d /home/owenlocqueneux/gophish/
Archive:  gophish-v0.7.1-linux-64bit.zip
   creating: /home/owenlocqueneux/gophish/static/js/dist/app/
  inflating: /home/owenlocqueneux/gophish/static/js/dist/vendor.min.js
  inflating: /home/owenlocqueneux/gophish/static/js/dist/app/users.min.js
  inflating: /home/owenlocqueneux/gophish/static/js/dist/app/dashboard.min.js
  inflating: /home/owenlocqueneux/gophish/static/js/dist/app/templates.min.js
  inflating: /home/owenlocqueneux/gophish/static/js/dist/app/campaigns.min.js
  inflating: /home/owenlocqueneux/gophish/static/js/dist/app/landing_pages.min.js
  inflating: /home/owenlocqueneux/gophish/static/js/dist/app/gophish.min.js
  inflating: /home/owenlocqueneux/gophish/static/js/dist/app/campaign_results.min.js
  inflating: /home/owenlocqueneux/gophish/static/js/dist/app/settings.min.js
  inflating: /home/owenlocqueneux/gophish/static/js/dist/app/sending_profiles.min.js
   creating: /home/owenlocqueneux/gophish/static/js/src/vendor/ckeditor/plugins/
   creating: /home/owenlocqueneux/gophish/static/js/src/vendor/ckeditor/lang/
   creating: /home/owenlocqueneux/gophish/static/js/src/vendor/ckeditor/skins/
```

```
┌──(root㉿kali)-[/home/owenlocqueneux]
└─# cd gophish/

┌──(root㉿kali)-[/home/owenlocqueneux/gophish]
└─# ls
LICENSE  README.md  VERSION  config.json  db  gophish  static  templates

┌──(root㉿kali)-[/home/owenlocqueneux/gophish]
└─# ./gophish
time="2022-11-21T21:57:27Z" level=info msg="Background Worker Started Successfully - Waiting for Campaigns"
time="2022-11-21T21:57:27Z" level=warning msg="No contact address has been configured."
time="2022-11-21T21:57:27Z" level=warning msg="Please consider adding a contact_address entry in your config.js
on"
goose: migrating db environment 'production', current version: 0, target: 20180830215615
OK    20160118194630_init.sql
OK    20160131153104_0.1.2_add_event_details.sql
OK    20160211211220_0.1.2_add_ignore_cert_errors.sql
OK    20160217211342_0.1.2_create_from_col_results.sql
OK    20160225173824_0.1.2_capture_credentials.sql
OK    20160227180335_0.1.2_store-smtp-settings.sql
OK    20160317214457_0.2_redirect_url.sql
OK    20160605210903_0.2_campaign_scheduling.sql
OK    20170104220731_0.2_result_statuses.sql
OK    20170219122503_0.2.1_email_headers.sql
OK    20170827141312_0.4_utc_dates.sql
OK    20171027213457_0.4.1_maillogs.sql
OK    20171208201932_0.4.1_next_send_date.sql
OK    20180223101813_0.5.1_user_reporting.sql
OK    20180524203752_0.7.0_result_last_modified.sql
OK    20180527213648_0.7.0_store_email_request.sql
OK    20180830215615_0.7.0_send_by_date.sql
time="2022-11-21T21:57:27Z" level=info msg="Starting phishing server at http://0.0.0.0:80"
time="2022-11-21T21:57:27Z" level=info msg="Creating new self-signed certificates for administration interface"
time="2022-11-21T21:57:27Z" level=info msg="TLS Certificate Generation complete"
time="2022-11-21T21:57:27Z" level=info msg="Starting admin server at https://127.0.0.1:3333"
```

We launch gophish on the browser of our machine by typing the given IP address: https://:127.0.0.1:3333

We log in with the admin login and the gophish password, which allows us to access the gophish dashboard:



GoPhish Dashboard

1.6 Evilginx2

Evilginx2 is a tool that allows you to recover authentication credentials through a malicious link. It acts as a proxy between the target site and the victim in order to intercept their login credentials (man-in-the-middle attack).

Before installing Evilginx2 you must install Golang.

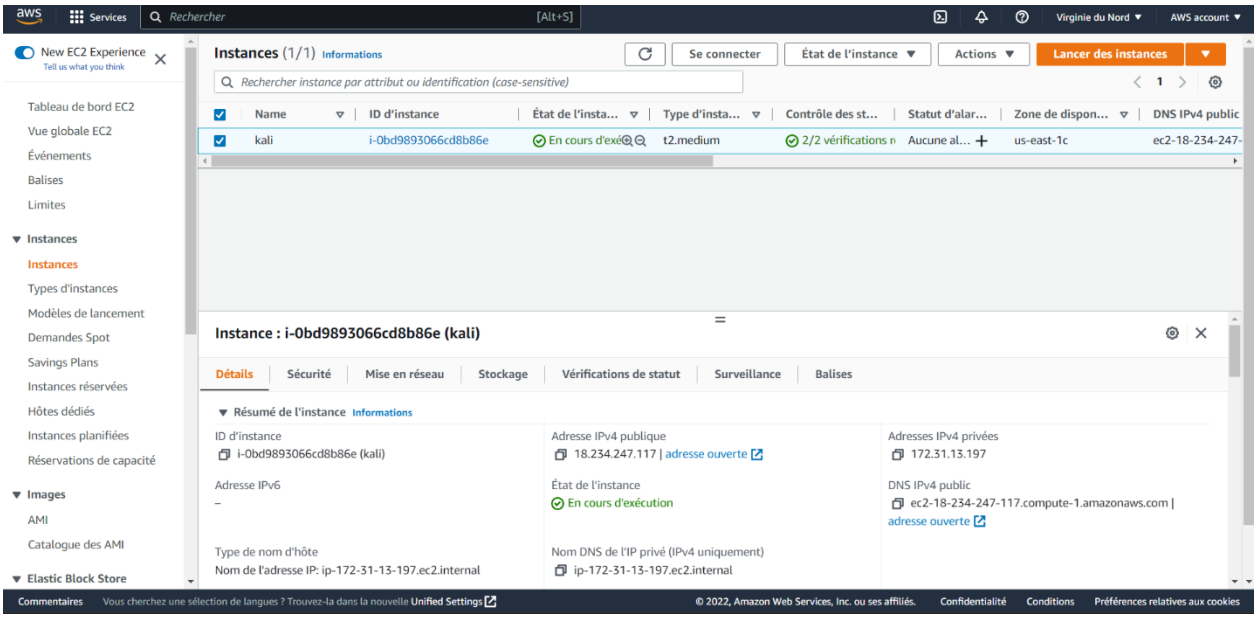
Installation of Evilginx2 on our EC2 instance (Google Cloud):

```
┌──(root㉿kali)-[/home/owenlocqueneux]
└─# git clone https://github.com/kgretzky/evilginx2.git
Cloning into 'evilginx2'...
remote: Enumerating objects: 2722, done.
remote: Total 2722 (delta 0), reused 0 (delta 0), pack-reused 2722
Receiving objects: 100% (2722/2722), 3.61 MiB | 6.08 MiB/s, done.
Resolving deltas: 100% (1409/1409), done.

┌──(root㉿kali)-[/home/owenlocqueneux]
└─# ls
evilginx2  go1.19.3.linux-amd64.tar.gz  gophish  gophish-v0.7.1-linux-64bit.zip

┌──(root㉿kali)-[/home/owenlocqueneux]
└─# cd evilginx2/
```

```
┌──(root㉿kali)-[/home/owenlocqueneux/evilgophish/evilginx2]
└─# evilginx

                                        - -- Gone Phishing -- -
                                  by Kuba Gretzky (@mrgretzky)      version 2.4.2

[09:21:42] [inf] loading phishlets from: /usr/share/evilginx/phishlets/
[09:21:42] [inf] loading configuration from: /root/.evilginx
[09:21:42] [inf] blacklist: loaded 0 ip addresses or ip masks

+-----------------+-------------------+-----------+-----------+----------+
|    phishlet     |       author      |  active   |  status   | hostname |
+-----------------+-------------------+-----------+-----------+----------+
| amazon          | @customsync       | disabled  | available |          |
| linkedin        | @mrgretzky        | disabled  | available |          |
| outlook         | @mrgretzky        | disabled  | available |          |
| twitter-mobile  | @white_fi         | disabled  | available |          |
| paypal          | @An0nud4y         | disabled  | available |          |
| protonmail      | @jamescullum      | disabled  | available |          |
| booking         | @Anonymous        | disabled  | available |          |
| citrix          | @424f424f         | disabled  | available |          |
| coinbase        | @An0nud4y         | disabled  | available |          |
| instagram       | @charlesbel       | disabled  | available |          |
| knowbe4         | @fin3ss3g0d       | disabled  | available |          |
| onelogin        | @perfectlylog...  | disabled  | available |          |
| reddit          | @customsync       | disabled  | available |          |
| tiktok          | @An0nUD4Y         | disabled  | available |          |
| facebook        | @charlesbel       | disabled  | available |          |
| github          | @audibleblink     | disabled  | available |          |
| google          | @fin3ss3g0d       | disabled  | available |          |
| o3652           | @kike,@bsmith...  | disabled  | available |          |
| okta            | @mikesiegel       | disabled  | available |          |
| airbnb          | @AN0NUD4Y         | disabled  | available |          |
| o365            | @jamescullum      | disabled  | available |          |
| twitter         | @white_fi         | disabled  | available |          |
| wordpress.org   | @meitar           | disabled  | available |          |
+-----------------+-------------------+-----------+-----------+----------+

: config domain aot-project.com
[09:22:13] [inf] server domain set to: aot-project.com
: config ip 217.160.0.148
[09:22:43] [inf] server IP set to: 217.160.0.148
: c
```

Now that we understand evilginx2 and gophish we have decided to use another tool called evilgophish which is a combination of evilnginx2 and Gophish but with AWS (cloud platform) due to some problems met before with GCP.

1.7 EC2 instance on AWS



1.8 Evilgophish

Evilgophish is a combination of Evilginx2 and GoPhish.

<u>Evilgophish installation</u>:

For this project we had to install Evilgophish.

Below is the installation of this server on our ubuntu (didn't work on a Kali with Google Cloud):

First, we need to clone the evilgophish github repository:

```
ubuntu@ip-172-31-83-218:~/Desktop$ git clone https://github.com/fin3ss3g0d/evilgophish.git
Cloning into 'evilgophish'...
remote: Enumerating objects: 2426, done.
remote: Counting objects: 100% (678/678), done.
remote: Compressing objects: 100% (279/279), done.
remote: Total 2426 (delta 416), reused 586 (delta 396), pack-reused 1748
Receiving objects: 100% (2426/2426), 30.98 MiB ¦ 20.12 MiB/s, done.
Resolving deltas: 100% (758/758), done.
```

Then launch the setup file:

```
ubuntu@ip-172-31-83-218:~/Desktop/evilgophish$ sudo ./setup.sh aot-project.com login,www true esme.fr true client_id false
[evilgophish setup] [*] Installing dependencies with apt
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Fetched 114 kB in 0s (259 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'certbot' instead of 'letsencrypt'
net-tools is already the newest version (1.60+git20180626.aebd88e-1ubuntu1).
apache2 is already the newest version (2.4.41-4ubuntu3.12).
build-essential is already the newest version (12.8ubuntu1.1).
git is already the newest version (1:2.25.1-1ubuntu3.6).
openssl is already the newest version (1.1.1f-1ubuntu2.16).
tmux is already the newest version (3.0a-2ubuntu0.3).
wget is already the newest version (1.20.3-1ubuntu2).
certbot is already the newest version (0.40.0-1ubuntu0.1).
jq is already the newest version (1.6-1ubuntu0.20.04.1).
0 upgraded, 0 newly installed, 0 to remove and 18 not upgraded.
[evilgophish setup] [+] Installed dependencies with apt!
[evilgophish setup] [*] Installing Go from source
--2022-12-29 19:54:51--  https://go.dev/dl/go1.19.4.linux-amd64.tar.gz
Resolving go.dev (go.dev)... 216.239.34.21, 216.239.36.21, 216.239.38.21, ...
Connecting to go.dev (go.dev)¦216.239.34.21¦:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://dl.google.com/go/go1.19.4.linux-amd64.tar.gz [following]
--2022-12-29 19:54:51--  https://dl.google.com/go/go1.19.4.linux-amd64.tar.gz
Resolving dl.google.com (dl.google.com)... 172.253.63.93, 172.253.63.136, 172.253.63.190, ...
Connecting to dl.google.com (dl.google.com)¦172.253.63.93¦:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 148931745 (142M) [application/x-gzip]
Saving to: 'go1.19.4.linux-amd64.tar.gz'

go1.19.4.linux-amd64.tar.gz          100%[===================================================================================>] 142.03M   109MB/s    in 1.3s

2022-12-29 19:54:52 (109 MB/s) - 'go1.19.4.linux-amd64.tar.gz' saved [148931745/148931745]

[evilgophish setup] [+] Installed Go from source!
[evilgophish setup] [*] Run the command below to generate letsencrypt certificates (will need to create two (2) DNS TXT records):
```

To continue, we need a certificate that we will create with the command below:

```
ubuntu@ip-172-31-83-218:~$ sudo certbot certonly --manual --preferred-challenges=dns --email admin@aot-project.com --server https://acme-v02.api.letsencrypt.org/directory
 --agree-tos -d '*.aot-project.com' -d 'aot-project.com'
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator manual, Installer None
Obtaining a new certificate
Performing the following challenges:
dns-01 challenge for aot-project.com
dns-01 challenge for aot-project.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
NOTE: The IP of this machine will be publicly logged as having requested this
certificate. If you're running certbot in manual mode on a machine that is not
your server, please ensure you're okay with that.

Are you OK with your IP being logged?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: Yes

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please deploy a DNS TXT record under the name
_acme-challenge.aot-project.com with the following value:

K_w83rwu6OeTsD2dp-05Z0HTHzUlrg2ClSD5mtoulb4

Before continuing, verify the record is deployed.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Press Enter to Continue

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

This will give us a value to put in a DNS TXT record on IONOS

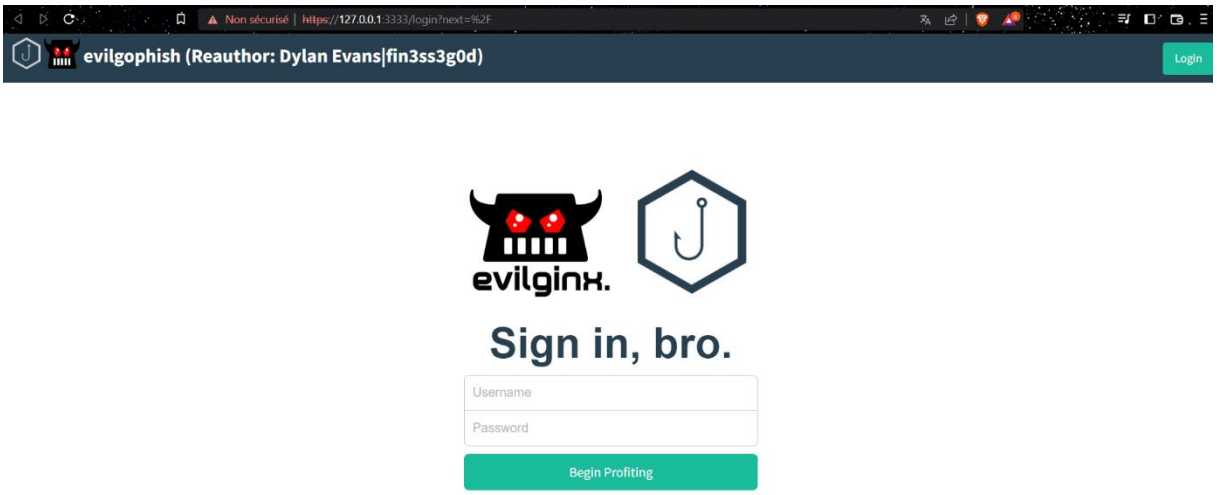
To start GoPhish server:

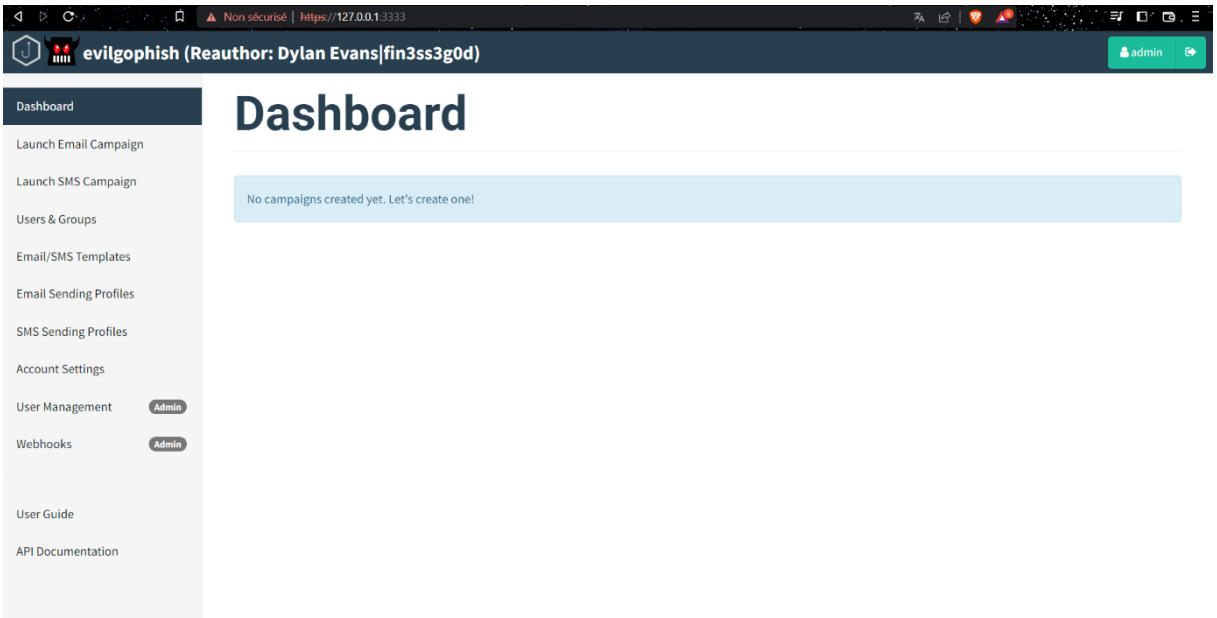
```
ubuntu@ip-172-31-83-218:~/Desktop/evilgophish/gophish$ sudo ./gophish
time="2022-12-29T19:56:49Z" level=warning msg="No contact address has been configured."
time="2022-12-29T19:56:49Z" level=warning msg="Please consider adding a contact_address entry in your config.json"
goose: no migrations to run. current version: 20220321133237
time="2022-12-29T19:56:49Z" level=info msg="Please login with the username admin and the password 606329a4ed047ab7"
time="2022-12-29T19:56:49Z" level=info msg="Starting admin server at https://127.0.0.1:3333"
time="2022-12-29T19:56:50Z" level=info msg="Background Worker Started Successfully - Waiting for Campaigns"
time="2022-12-29T19:56:50Z" level=info msg="Starting IMAP monitor manager"
time="2022-12-29T19:56:50Z" level=info msg="Starting new IMAP monitor for user admin"
time="2022-12-29T19:56:50Z" level=info msg="Starting phishing server at https://127.0.0.1:8080"
```

Then to access:

```
C:\Users\owenl\Downloads>ssh -L 3333:127.0.0.1:3333 -i kali.pem ubuntu@34.239.132.42
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-1026-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Dec 29 20:52:44 UTC 2022

  System load:  0.0                Processes:             184
  Usage of /:   28.2% of 19.20GB   Users logged in:       1
  Memory usage: 54%                IPv4 address for eth0: 172.31.83.218
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

18 updates can be applied immediately.
17 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

New release '22.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Thu Dec 29 20:52:11 2022 from 37.169.124.220
ubuntu@ip-172-31-83-218:~$
```

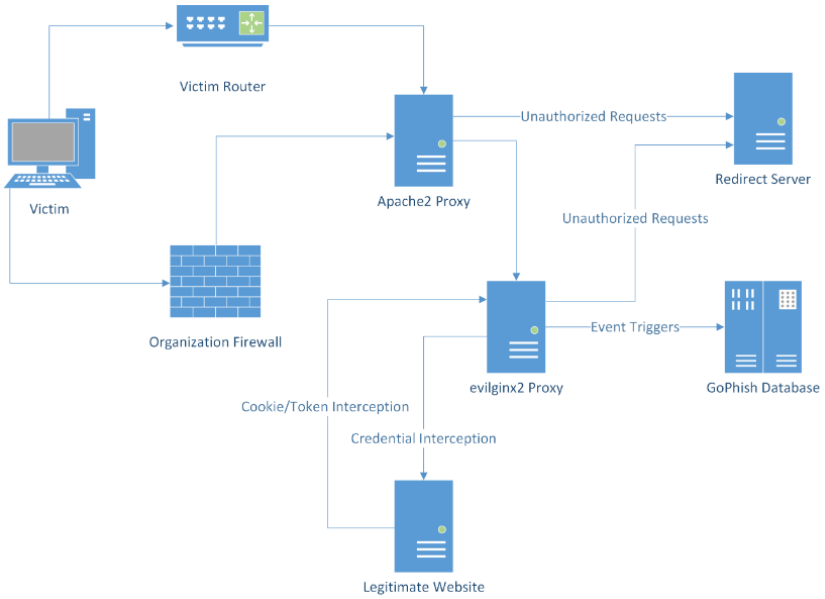
Below is our evilgophish dashboard:





Principle of the attack:

First, for the attack to word we need to start and configure GoPhish, Evilginx2 and Apache2. All those servers will be needed, and the reason why is that GoPhish, once configured with the email template, will be used to send emails, and provide a dashboard for evilginx2. The phishing links sent will point to an evilginx2 lure path and evilginx2 will be used for landing pages, which will provide us the ability to bypass 2FA/MFA with evilginx2. Apache2 will be needed to serve mainly as a proxy to the local evilginx2 server.



Attack setup

2.Documentation

2.1 GitHub and the git command

It’s commonly used to host open-source software development projects.

GitHub is an Internet hosting service for software development and version control using Git. It provides the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project.

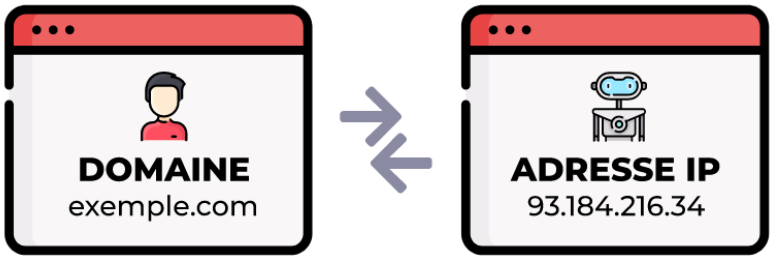
Screenshots below show the basics git commands to create a project:

```
PS C:\Users\owenl\Desktop\Cours Cyber\Projet Malware> git init
Initialized empty Git repository in C:/Users/owenl/Desktop/Cours Cyber/Proje
t Malware/.git/
PS C:\Users\owenl\Desktop\Cours Cyber\Projet Malware> git add README.md
fatal: pathspec 'README.md' did not match any files
PS C:\Users\owenl\Desktop\Cours Cyber\Projet Malware> git remote add origin
https://github.com/owen62/MalwareProject.git
```

```
PS C:\Users\owenl\Desktop\Cours Cyber\Projet Malware> New-Item README.md


    Répertoire : C:\Users\owenl\Desktop\Cours Cyber\Projet Malware


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        24/11/2022     14:14              0 README.md


PS C:\Users\owenl\Desktop\Cours Cyber\Projet Malware> ls


    Répertoire : C:\Users\owenl\Desktop\Cours Cyber\Projet Malware


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        04/11/2022     22:36            359 notes.txt
-a----        14/11/2022     23:28         409115 projectfirstpart.docx
-a----        24/11/2022     14:14              0 README.md


PS C:\Users\owenl\Desktop\Cours Cyber\Projet Malware> git add .\README.md
PS C:\Users\owenl\Desktop\Cours Cyber\Projet Malware> git commit -m "creatin
g a README.md file"
[master (root-commit) 8c3b701] creating a README.md file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
PS C:\Users\owenl\Desktop\Cours Cyber\Projet Malware> git push -u origin mas
ter
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 238 bytes | 238.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/owen62/MalwareProject.git
```

2.2 DNS et Record DNS

It's much easier for us to remember a domain name, exemple.com rather than a string of numbers, 93.184.216.34 (IP address) to exemple.com's website.



A **DNS** is a computer server that contains a database of many IP addresses and their associated domain names. It serves to translate a requested domain name into an IP address, so that the computer knows which IP address to connect to for the requested contents.

A great example is to see a DNS as a phone book, which matches a name to a telephone number. You can search for the name you want and find the corresponding phone number. Remembering domain names is easier for us than to remember a string of numbers. DNS helps us to do this by matching domain names to IP addresses and simplifies our web surfing experience significantly.

DNS **records** (aka zone files) are instructions that live in authoritative DNS servers and provide information about a domain including what IP address is associated with that domain and how to handle requests for that domain. These records consist of a series of text files written in what is known as DNS syntax.

DNS syntax is just a string of characters used as commands that tell the DNS server what to do.

The most common types of DNS record are:

- **A record** - The record that holds the IP address of a domain.
- **AAAA record** - The record that contains the IPv6 address for a domain (as opposed to A records, which list the IPv4 address).
- **CNAME record** - Forwards one domain or subdomain to another domain, does NOT provide an IP address.

- **MX record** - Directs mail to an email server.
- **TXT record** - Lets an admin store text notes in the record. These records are often used for email security.
- **NS record** - Stores the name server for a DNS entry.
- **SOA record** - Stores admin information about a domain.
- **SRV record** - Specifies a port for specific services.
- **PTR record** - Provides a domain name in reverse-lookups.

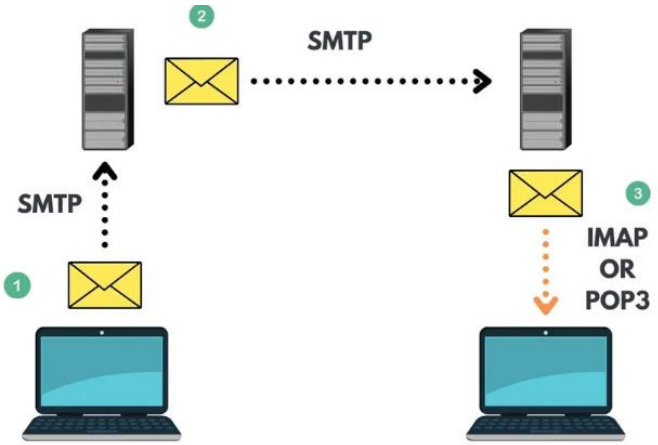
2.3 SSL/TLS

SSL (Secure Sockets Layer) encryption, and its more modern and secure replacement, TLS (Transport Layer Security) encryption, protect data sent over the internet or a computer network. This prevents attackers (and Internet Service Providers) from viewing or tampering with data exchanged between two nodes—typically a user's web browser and a web/app server.

SSL/TLS uses both asymmetric and symmetric encryption to protect the confidentiality and integrity of data-in-transit. Asymmetric encryption is used to establish a secure session between a client and a server, and symmetric encryption is used to exchange data within the secured session.

2.4 SMTP, IMAP, POP protocols

Simple Mail Transfer Protocol (SMTP), Post Office Protocol (POP), and Internet Message Access Protocol (IMAP) are specialized TCP/IP protocols used for sending and receiving emails. As developers, it is essential to understand these protocols to meet the modern application requirements.



- **POP protocol**

POP is a more user-friendly method of accessing mailboxes. POP transfers emails from the server to the client, allowing you to read them even if you are not connected to the internet.

When a user checks for a new email, the client makes a connection to the POP server. The email client then provides the server with its username and password for authentication. When the client connects, it issues text-based commands to retrieve all email messages. It then saves the downloaded messages as new emails on the user's local system, deletes the server copies, and disconnects from the server.

- **IMAP protocol**

IMAP is a protocol for receiving emails from a server. Since IAMP allows access to emails from multiple locations simultaneously, it keeps the email on the server after being delivered. Also, it doesn't;t download the entire email until the recipient opens it.

When using the IMAP protocol, the client connects to the server, checks for new messages, and saves them in the cache as temporary files. Only the date, sender, and subject are initially downloaded from the server. The content will only be downloaded when you open the message. So, it is possible to access the email's content without downloading the attached files using this protocol.

When an email is modified, deleted, or status changes from unread to read, the changes are reflected on the server. This process helps to reflect the status of emails on multiple devices in real-time.

- **SMTP protocol**

SMTP is a widely used TCP protocol for email sending. The SMTP protocol is mainly used by the clients to send emails to the servers or for the email communications between servers.

There are 2 types of SMTP servers: Relays and Receivers. Relays accept emails from users and route them to recipients, while Receivers deliver them to the mailbox after accepting the email from the Relay servers.

The SMTP workflow consists of 3 steps:

1- The SMTP client will connect to the SMTP server.

2- The email is transferred using that connection.

3- The client and the server terminate the connection.

SMTP client uses text-based commands such as HELLO, MAIL FROM, EHLO, and RCPT to send messages to the SMTP server. SMTP server responds to these messages using numeric codes like 220, 250, and 354

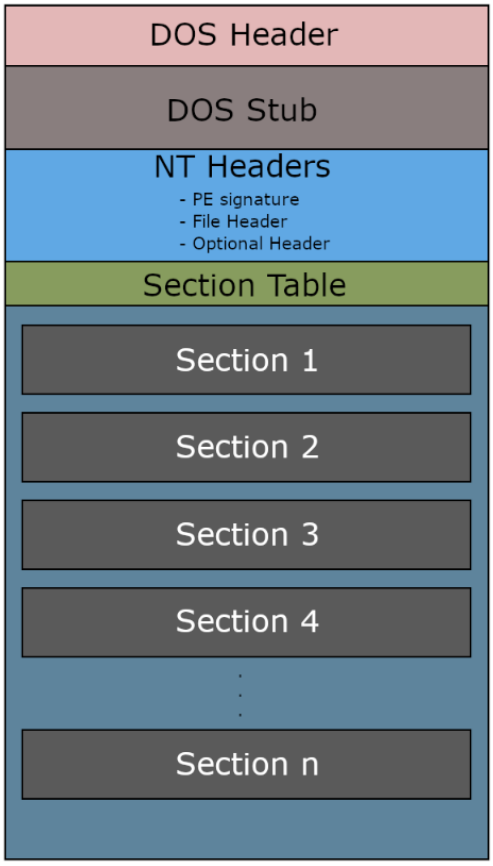
2.4 PE format file

The Portable Executable (PE) format is a file format for executables, object code, DLLs and others used in 32-bit and 64-bit versions of Windows operating systems. The PE format is a data structure that encapsulates the information necessary for the Windows OS loader to manage the wrapped executable code.

This includes dynamic library references for linking, API export and import tables, resource management data and thread-local storage (TLS) data.

On NT operating systems, the PE format is used for EXE, DLL, SYS (device driver), MUI and other file types. The Unified Extensible Firmware Interface (UEFI) specification states that PE is the standard executable format in EFI environments.

A typical PE file follows the structure outlined in the following figure:



PE file's structure

- **DOS Header**

Every PE file starts with a 64-bytes-long structure called the DOS header, it's what makes the PE file an MS-DOS executable.

- **DOS Stub**

After the DOS header comes the DOS stub which is a small MS-DOS 2.0 compatible executable that just prints an error message saying "This program cannot be run in DOS mode" when the program is run in DOS mode.

- **NT Headers**

The NT Headers part contains three main parts:

  - PE signature: A 4-byte signature that identifies the file as a PE file.
  - File Header: A standard COFF File Header. It holds some information about the PE file.
  - Optional Header: The most important header of the NT Headers, its name is the Optional Header because some files like object files don't have it, however it's required for image files (files like .exe files). This header provides important information to the OS loader.

- **Section Table**

The section table follows the Optional Header immediately, it is an array of Image Section Headers, there's a section header for every section in the PE file. Each header contains information about the section it refers to.

- **Sections**

Sections are where the actual contents of the file are stored, these include things like data and resources that the program uses, and the actual code of the program, there are several sections each one with its own purpose.

2.5 Windows API

The Windows API, informally WinAPI, is Microsoft's core set of application programming interfaces (APIs) available in the Microsoft Windows operating systems. The name Windows API collectively refers to several different platform implementations that are often referred to by their own names (for example, Win32 API). Almost all Windows programs interact with the Windows API.

On the Windows NT line of operating systems, a small number (such as programs started early in the Windows start up process) use the Native API.

The Windows API (Win32) is focused mainly on the programming language C in that its exposed functions and data structures are described in that language in recent versions of its documentation. However, the API may be used by any programming language compiler or assembler able to handle the (well-defined) low-level data structures along with the prescribed calling conventions for calls and call-backs.

The functions provided by the Windows API can be grouped into eight categories:

- **Base Services**

Provide access to the basic resources available to a Windows system. Included are things like file systems, devices, processes, threads, and error handling. These functions reside in kernel.exe, krnl286.exe or krnl386.exe files on 16-bit Windows, and kernel32.dll and KernelBase.dll on 32- and 64-bits Windows. These files reside in the folder \Windows\System32 on all versions of Windows.

- **Advanced Services**

Provide access to functions beyond the kernel. Included are things like the Windows registry, shutdown/restart the system (or abort), start/stop/create a Windows service, manage user accounts. These functions reside in advapi32.dll and advapires32.dll on 32-bit Windows.

- **Graphics Device Interface**

Provides functions to output graphics content to monitors, printers, and other output devices. It resides in gdi.exe on 16-bit Windows, and gdi32.dll on 32-bit Windows in user-mode. Kernel-mode GDI support is provided by win32k.sys which communicates directly with the graphics driver.

- **User Interface**

Provides the functions to create and manage screen windows and most basic controls, such as buttons and scrollbars, receive mouse and keyboard input, and other functions associated with the graphical user interface (GUI) part of Windows. This functional unit resides in user.exe on 16-bit Windows, and user32.dll on 32-bit Windows. Since Windows XP versions, the basic controls reside in comctl32.dll, together with the common controls (Common Control Library).

- **Common Dialog Box Library**

Provides applications the standard dialog boxes to open and save files, choose color and font, etc. The library resides in a file called commdlg.dll on 16-bit Windows, and comdlg32.dll on 32-bit Windows. It is grouped under the User Interface category of the API.

- **Common Control Library**

Gives applications access to some advanced controls provided by the operating system. These include things like status bars, progress bars, toolbars, and tabs. The library resides in a dynamic-link library (DLL) file called commctrl.dll on 16-bit Windows, and comctl32.dll on 32-bit Windows. It is grouped under the User Interface category of the API.

- **Windows Shell**

Component of the Windows API allows applications to access functions provided by the operating system shell, and to change and enhance it. The component resides in shell.dll on 16-bit Windows, and shell32.dll on 32-bit Windows. The Shell Lightweight Utility Functions are in shlwapi.dll. It is grouped under the User Interface category of the API.

- **Network Services**

Give access to the various networking abilities of the operating system. Its subcomponents include NetBIOS, Winsock, NetDDE, remote procedure call (RPC) and many more. This component resides in netapi32.dll on 32-bit Windows.

- **Web**

The Internet Explorer (IE) web browser also exposes many APIs that are often used by applications, and as such could be considered a part of the Windows API. IE has been included with the operating system since Windows 95 OSR2 and has provided web-related services to applications since Windows 98. Specifically, it is used to provide:

  - An embeddable web browser control, contained in shdocvw.dll and mshtml.dll.
  - The URL moniker service, held in urlmon.dll, which provides COM objects to applications for resolving URLs. Applications can also provide their own URL handlers for others to use.
  - An HTTP client library which also considers system-wide Proxy settings (wininet.dll); however, Microsoft has added another HTTP client library called winhttp.dll which is smaller and more suitable for some applications.
  - A library to assist multi-language and international text support (mlang.dll).
  - DirectX Transforms, a set of image filter components.
  - XML support (the MSXML components, held in msxml*.dll).
  - Access to the Windows Address Books.

Resources

Installation of Kali in Google Cloud & AWS:

https://www.youtube.com/watch?v=XRJMA67Beh4

https://www.learningjournal.guru/article/google-cloud/free-learning-virtual-machine/

https://www.youtube.com/watch?v=S0YZnY_4dlw

https://github.com/fin3ss3g0d/evilgophish#disclaimer

https://github.com/m0ns7er/GCP

https://www.xmodulo.com/how-to-set-up-ubuntu-desktop-vm-on-amazon-ec2.html

Evilgophish:

https://kalilinuxtutorial.com/install-evilginx2-on-kali-linux/

https://kalilinuxtutorial.com/install-golang-on-kali-linux/

https://go.dev/doc/install

https://github.com/fin3ss3g0d/evilgophish#disclaimer

IONOS:

https://www.ionos.fr/domaine/noms-de-domaine

DNS, DNS Records:

https://www.cloudflare.com/en-gb/learning/dns/dns-records/

https://www.webnic.cc/what-is-a-domain-name-server-dns-and-how-it-works/

SSL/TLS:

https://www.f5.com/glossary/ssl-tls-encryption

Gophish:

https://kifarunix.com/install-gophish-on-ubuntu-18-04-debian-9-8/

https://www.golinuxcloud.com/install-gophish-phishing-framework-tutorial/

Github:

https://www.developer.com/microsoft/c-sharp/creating-your-first-github-project/

https://en.wikipedia.org/wiki/GitHub

POP, SMTP, IMAP:

https://www.courier.com/guides/imap-vs-pop3-vs-smtp/

PE format file:

https://0xrick.github.io/win-internals/pe2/

https://en.wikipedia.org/wiki/Portable_Executable

Windows API:

https://en.wikipedia.org/wiki/Windows_API