Gophish: A Phishing Campaign Toolkit

Noshin Nawal - 1805061
Maneesha Rani Saha - 1805076

April 18, 2025

Contents

1 Introduction 2
2 Installation 2
3 Steps to Launch a Phishing Attack 4
3.1 Sending Profile 4
3.2 Landing Page 6
3.3 Email Template 8
3.4 Group 10
3.5 Launch Campaign 11
4 Viewing Campaign Results 13
5 Getting Notification Using Webhooks 15
6 High Level Overview of Source Code 18
7 Video Demonstration Link 21

1 Introduction

Gophish is a phishing toolkit that makes the simulation of real-world phishing attacks dead-simple. The idea behind gophish is simple – make industry-grade phishing training available to everyone. "Available" in this case means two things –

- **Affordable** : Gophish is open-source software that is completely free for anyone to use.
- **Accessible** : Gophish is written in the Go programming language. This has the benefit that gophish releases are compiled binaries with no dependencies. In a nutshell, this makes installation as simple as "download and run"!

2 Installation

In this section we will see the installation process of gophish step by step.

1. Deploy the phishing server in the cloud so that it can be accessible from the victim. For this purpose we used **Microsoft Azure** to create our virtual machine with public ip.
2. We used **Linux (ubuntu 20.04)** as our OS.
3. Start your virtual machine and ssh into it from your local machine.
4. Run the following commands in your VM's terminal.

```
mkdir gophish
cd gophish

wget https://github.com/gophish/gophish/releases\
    /download/v0.12.1/gophish-v0.12.1-linux-64bit.zip

unzip gophish-v0.12.1-linux-64bit.zip
```

   Here we installed `gophish-v0.12.1-linux-64-bit`

5. Setup `config.json` before starting the server.

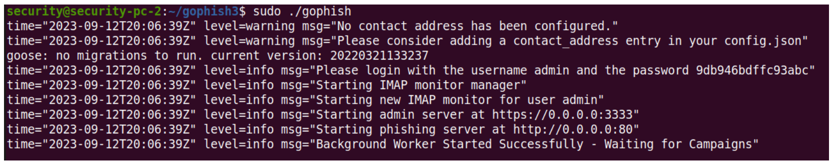
```
{
        "admin_server": {
                "listen_url": "0.0.0.0:3333",
                "use_tls": true,
                "cert_path": "gophish_admin.crt",
                "key_path": "gophish_admin.key",
                "trusted_origins": []
        },
        "phish_server": {
                "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": ""
        }
}
```

We changed `admin_server.listen_url` to 0.0.0.0:3333 as 127.0.0.1:3333 (default setup) is the loopback address and will only accept incoming connections from the same machine where the service is running.

Getting Started

Now let's start the server:

```
chmod +x gophish
sudo ./gophish
```



```
security@security-pc-2:~/gophish3$ sudo ./gophish
time="2023-09-12T20:06:39Z" level=warning msg="No contact address has been configured."
time="2023-09-12T20:06:39Z" level=warning msg="Please consider adding a contact_address entry in your config.json"
goose: no migrations to run. current version: 20220321133237
time="2023-09-12T20:06:39Z" level=info msg="Please login with the username admin and the password 9db946bdffc93abc"
time="2023-09-12T20:06:39Z" level=info msg="Starting IMAP monitor manager"
time="2023-09-12T20:06:39Z" level=info msg="Starting new IMAP monitor for user admin"
time="2023-09-12T20:06:39Z" level=info msg="Starting admin server at https://0.0.0.0:3333"
time="2023-09-12T20:06:39Z" level=info msg="Starting phishing server at http://0.0.0.0:80"
time="2023-09-12T20:06:39Z" level=info msg="Background Worker Started Successfully - Waiting for Campaigns"
```

Figure 1: Gophish server is started.

Here, we can see gophish admin server is started at `https://0.0.0.0:3333` and gophish phishing server is started at `http://0.0.0.0:80`.

Since our servers are hosted on a cloud VM with a public IP, we can access them from our local machine. Ensure that the IP rules on your cloud VM permit access to the required ports from external machines.

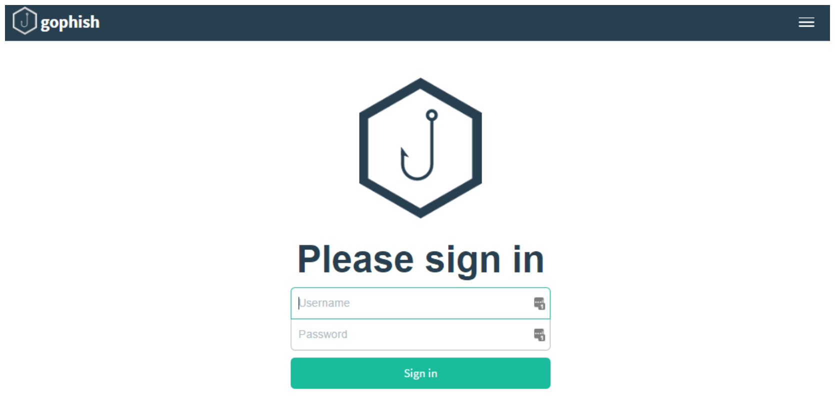


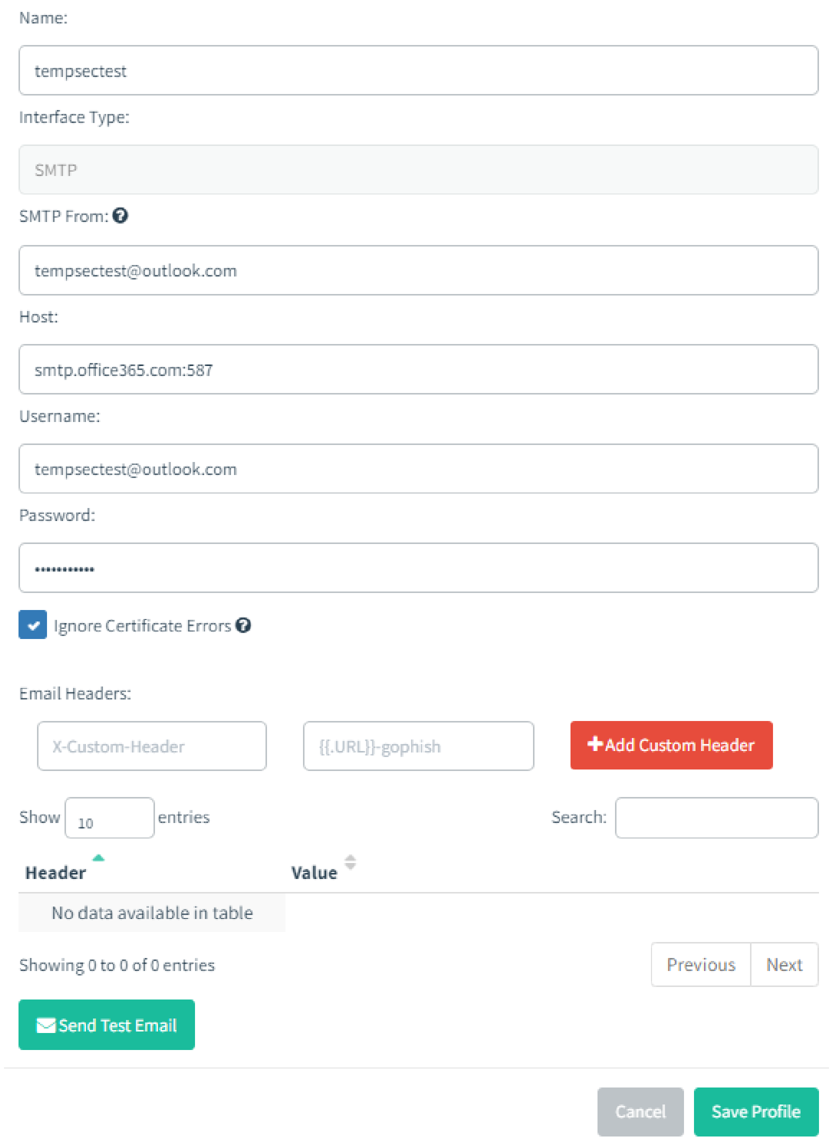
Figure 2: Login page

When you visit the admin server, you will encounter a login page. You should use the credentials obtained after starting the server (as shown in Figure 1, line 5) to log in. Once logged in, you will need to reset the password. After completing this step, you can begin using Gophish!

3 Steps to Launch a Phishing Attack

3.1 Sending Profile

To setup a sending profile, click the "Sending Profiles" navigation entry in the sidebar and click the "New Profile" button.


Name:

tempsectest

Interface Type:

SMTP

SMTP From:

tempsectest@outlook.com

Host:

smtp.office365.com:587

Username:

tempsectest@outlook.com

Password:

•••••••••••

✓ Ignore Certificate Errors

Email Headers:

Figure 3: Setting up sending profile.

Description of the fields:

- **Name** : Name of the sending profile
- **SMTP From** : Reciever will see this mail id as the sender (use less secure SMTP server if this field differs from "username")
- **Host** : SMTP server and port of mail client

- **Username** : Sender mail id
- **Password** : Password of sender mail id

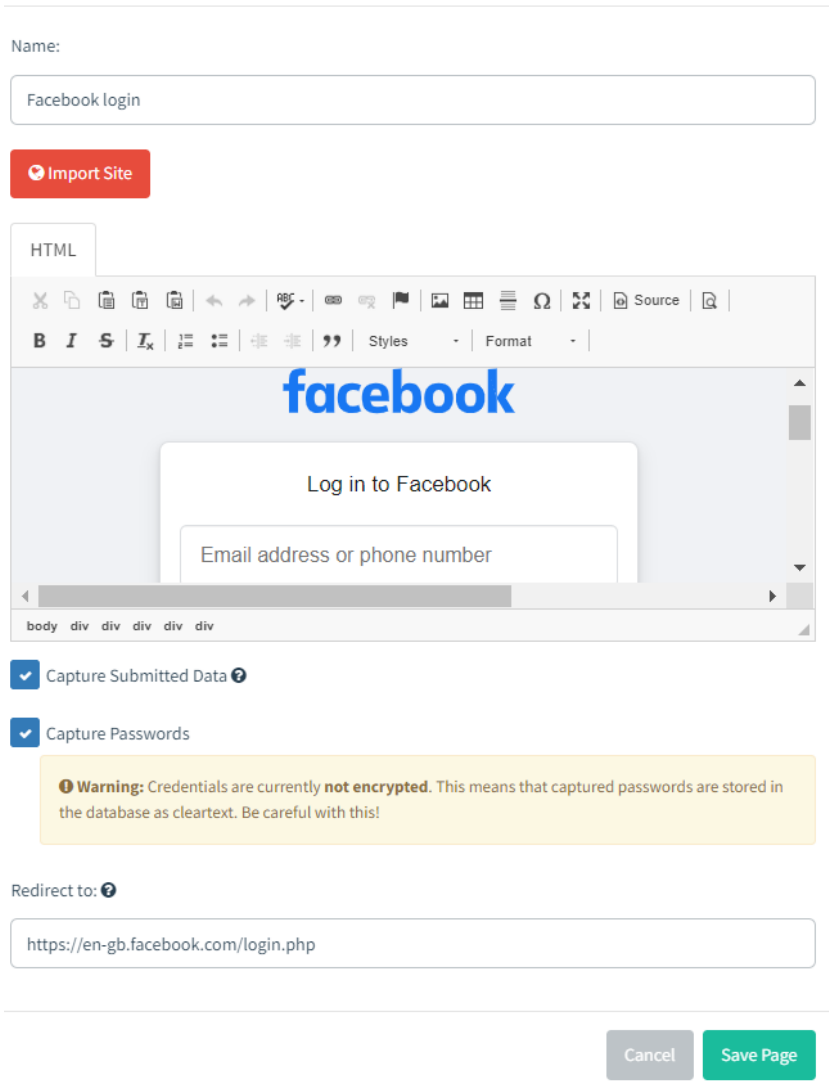
You can test everything by clicking "Send Test Email" before saving the profile.

3.2 Landing Page

Landing pages are the actual HTML pages that are returned to the users when they click the phishing links they receive.

Landing pages support templating, capturing credentials, and redirecting users to another website after they submit their credentials.

To create a landing page, click on the "Landing Pages" entry in the sidebar and click the "New Page" button.


Name:

Facebook login

✓ Capture Submitted Data

✓ Capture Passwords

Warning: Credentials are currently **not encrypted**. This means that captured passwords are stored in the database as cleartext. Be careful with this!

Redirect to:

https://en-gb.facebook.com/login.php

Figure 4: Setting up landing page.

Description of the fields:

- **Name** : Name of the landing page
- **Import Site** : URL of the actual page we want to replicate as our landing page
- **HTML** : After importing, the HTML of the actual page will be loaded here
- **Redirect to** : URL of the page victim will be redirected to after submitting data to our phishing server

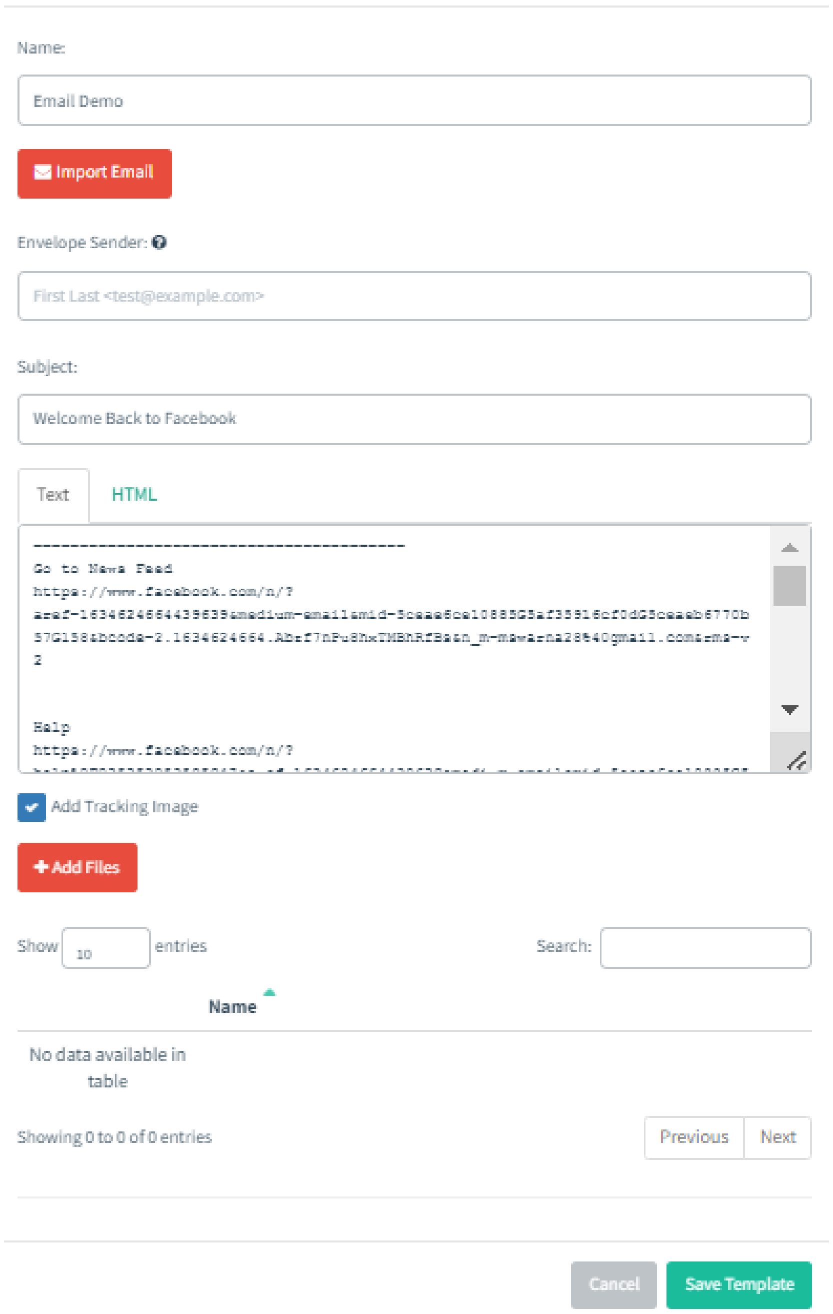
Make sure to check ”Capture Submitted Data” and ”Capture Password” before saving the page.

3.3 Email Template

A ”Template” is the content of the emails that are sent to targets. They can be imported from an existing email, or created from scratch. They also support sending attachments.

Additionally, templates can contain tracking images so that gophish knows when the user opens the email.

To create a template, first navigate to the “Email Templates” page and click the “New Template” button.


Name:

Email Demo

Envelope Sender:

First Last <test@example.com>

Subject:

Welcome Back to Facebook

```
----------------------------------------
Go to News Feed
https://www.facebook.com/n/?
aref=1634624664439639&medium=email&mid=5caae6ca10885G5af35916cf0dG5caaab6770b
57G158&bcode=2.1634624664.Abzf7nPu8hxTMBhRfBaan_m=mawarna28%40gmail.com&rms=v
2


Help
https://www.facebook.com/n/?
```

✓ Add Tracking Image

Show 10 entries Search:

Name
No data available in table

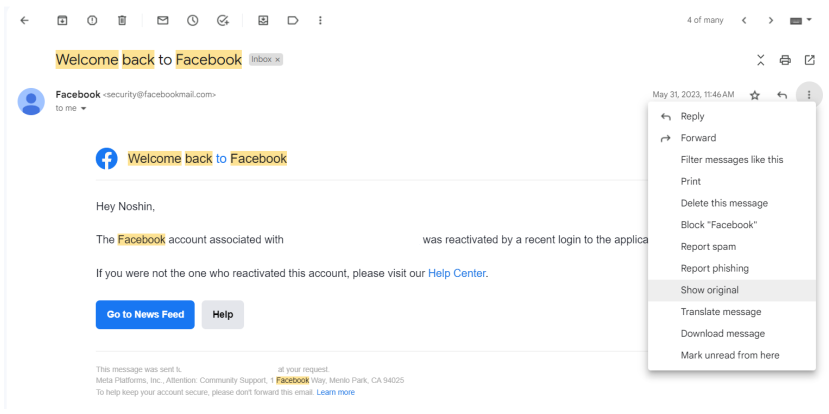
Showing 0 to 0 of 0 entries

Figure 5: Setting up email template.

Description of the fields:

- **Name** : Name of the email template

- **Import Email** : Go to the original email you want to copy.



4 of many

Welcome back to Facebook Inbox ×

Facebook <security@facebookmail.com>
to me

May 31, 2023, 11:46 AM

Welcome back to Facebook

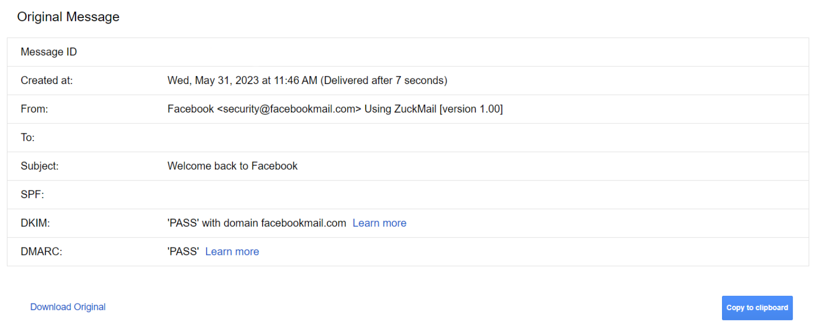
Hey Noshin,

The Facebook account associated with was reactivated by a recent login to the applica

If you were not the one who reactivated this account, please visit our Help Center.

This message was sent tc at your request.
Meta Platforms, Inc., Attention: Community Support, 1 Facebook Way, Menlo Park, CA 94025
To help keep your account secure, please don't forward this email. Learn more

Figure 6: Click "Show original".



Original Message

Message ID	
Created at:	Wed, May 31, 2023 at 11:46 AM (Delivered after 7 seconds)
From:	Facebook <security@facebookmail.com> Using ZuckMail [version 1.00]
To:	
Subject:	Welcome back to Facebook
SPF:	
DKIM:	'PASS' with domain facebookmail.com Learn more
DMARC:	'PASS' Learn more

Figure 7: Click "Copy to clipboard".

Paste it here and click "Import".

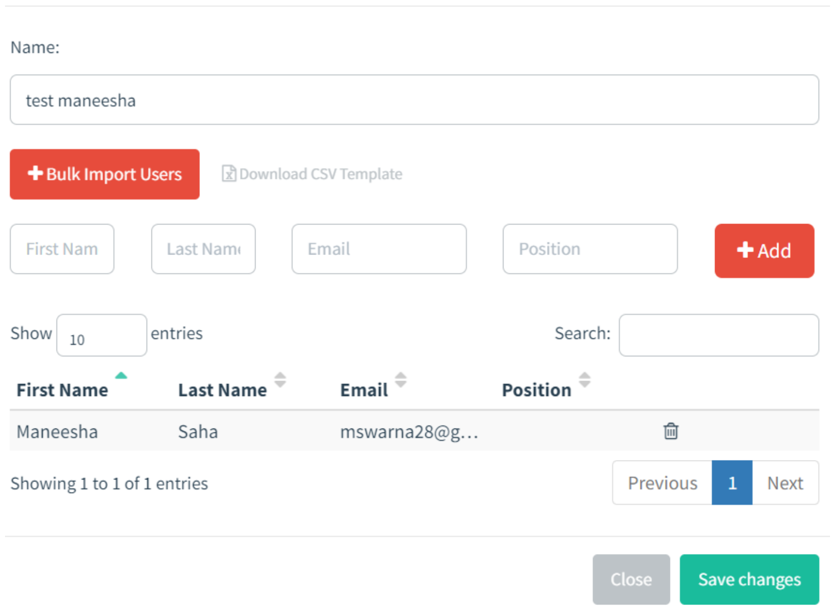
- **Envelope Sender** : Reciever will see this mail id as the sender (use less secure SMTP server if this field differs from Sending profile's "username")
- **Subject** : Subject of the email

Check "Add Tracking Image" to track the email (if clicked or not) and then save the template.

3.4 Group

Gophish lets you manage groups of users targeted in campaigns.

To create a group, first navigate to the "Users & Groups" page in the navigation menu and click the button


Name:

test maneesha

First Name	Last Name	Email	Position
Maneesha	Saha	mswarna28@g...	

Showing 1 to 1 of 1 entries

Figure 8: Creating a group.

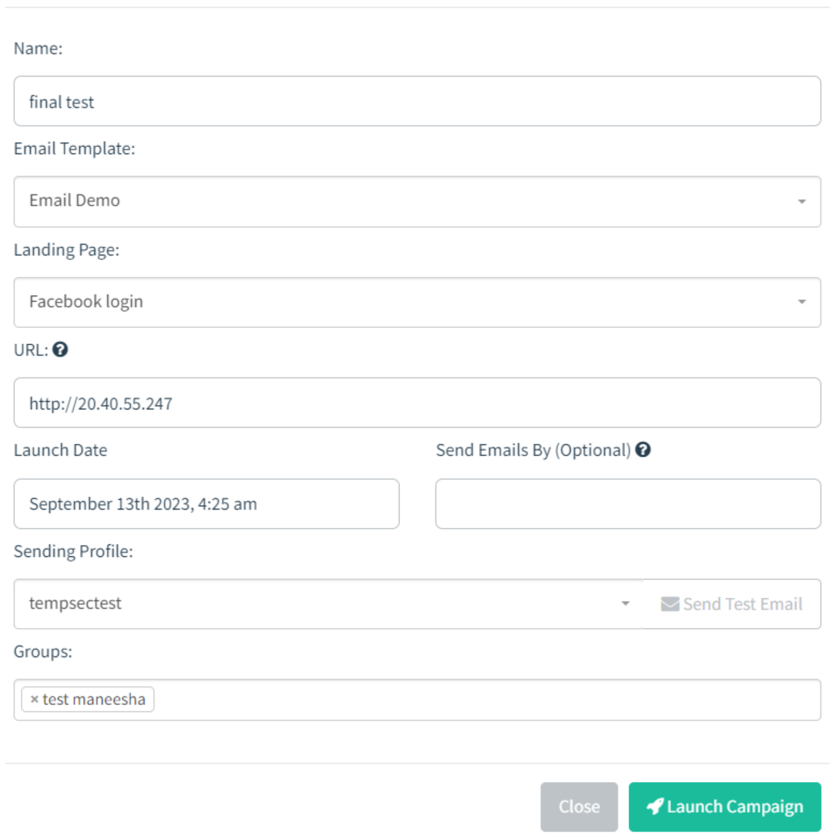
Description of the fields:

- **Name** : Name of the group
- **[First Name, Last Name, Email, Position]** : First name, last name, email and position of the phished person

You can add multiple individuals and even perform bulk imports using a CSV template. Afterward, you have the option to save the entire group.

3.5 Launch Campaign

Gophish is centered around launching campaigns. This involves sending emails to one or more groups and monitoring for opened emails, clicked links, or submitted credentials. To configure and launch a campaign, click the "Campaigns" entry in the navigation sidebar.


Name:

final test

Email Template:

Email Demo

Landing Page:

Facebook login

URL:

http://20.40.55.247

Launch Date

September 13th 2023, 4:25 am

Send Emails By (Optional)

Sending Profile:

tempsectest

Groups:

× test maneesha

Figure 9: Launching a campaign.

Description of the fields:

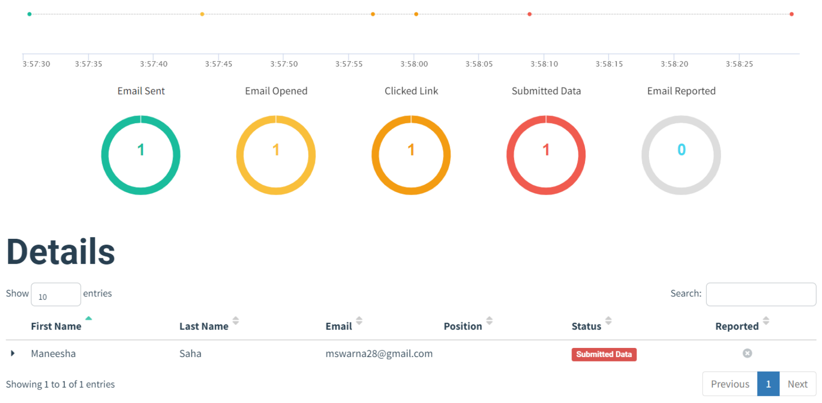
- **Name** : Name of the campaign
- **Email Template** : The email that is sent to campaign recipients. This is created in the section 3.3.
- **Landing Page** : The HTML that is returned when a recipient clicks the link in the email template. This is created in the section 3.2.
- **URL** : URL or IP address that points to the gophish phishing server and is reachable by the recipient.
- **Launch Date** : This is the date that the campaign will begin.
- **Send Emails By** : This is the date all emails will be sent by.
- **Sending Profile** : This is the SMTP configuration to use when sending emails. This is created in the section 3.1

- **groups** : This defines which groups of recipients should be included in the campaign.

After you have the campaign configuration ready to go, click the "Launch Campaign" button, click through the confirmation message, and you're good to go! Depending on your scheduling settings, gophish will either launch the campaign immediately or will schedule the campaign to be launched at a later date.

4 Viewing Campaign Results

When a campaign is launched, you are automatically redirected to the campaign results screen:



Email Sent	Email Opened	Clicked Link	Submitted Data	Email Reported
1	1	1	1	0

Details

Show 10 entries

Search:

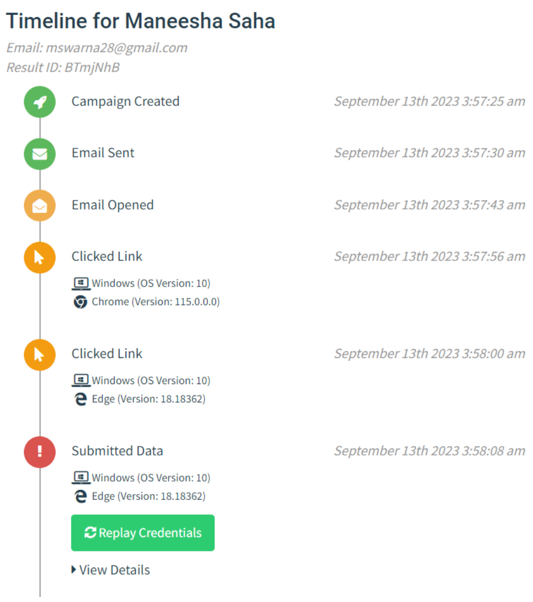
First Name	Last Name	Email	Position	Status	Reported
Maneesha	Saha	mswarna28@gmail.com		Submitted Data	⊗

Showing 1 to 1 of 1 entries

Previous 1 Next

Figure 10: Campaign overview

To view the timeline for each recipient, expand the row with the recipient's name.



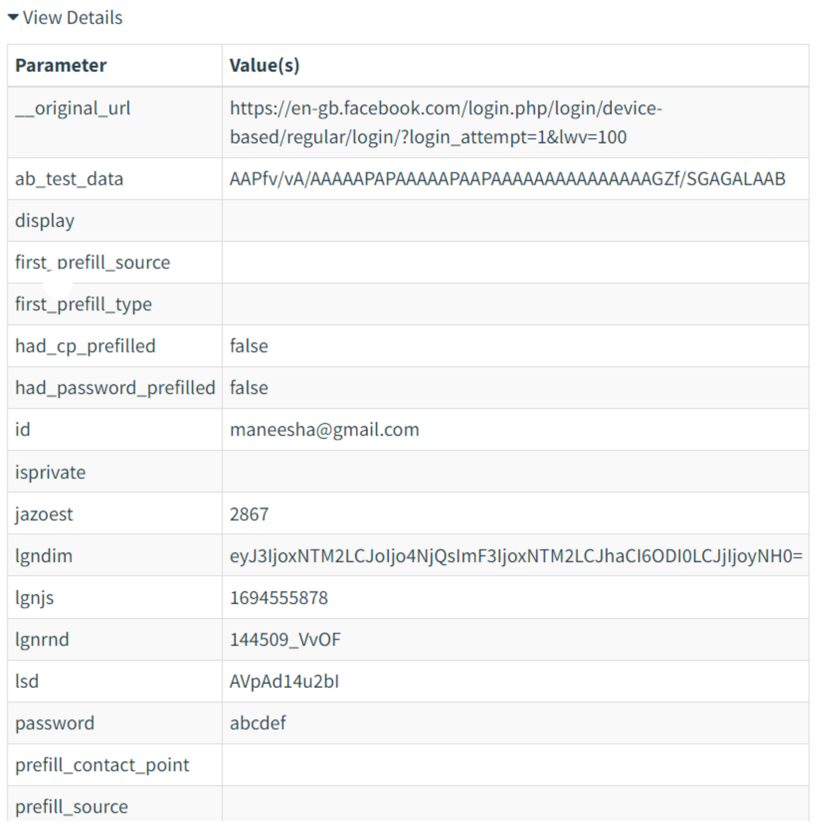
Timeline for Maneesha Saha

Email: mswarna28@gmail.com
Result ID: BTmjNhB

```
Campaign Created                      September 13th 2023 3:57:25 am
Email Sent                            September 13th 2023 3:57:30 am
Email Opened                          September 13th 2023 3:57:43 am
Clicked Link                          September 13th 2023 3:57:56 am
  Windows (OS Version: 10)
  Chrome (Version: 115.0.0.0)
Clicked Link                          September 13th 2023 3:58:00 am
  Windows (OS Version: 10)
  Edge (Version: 18.18362)
Submitted Data                        September 13th 2023 3:58:08 am
  Windows (OS Version: 10)
  Edge (Version: 18.18362)
```

Figure 11: Timeline for recipient

The results pane shows what a campaign recipient did, such as opening the email, clicking the link, or attempting to submit data from the landing page.



View Details

Parameter	Value(s)
__original_url	https://en-gb.facebook.com/login.php/login/device-based/regular/login/?login_attempt=1&lwv=100
ab_test_data	AAPfv/vA/AAAAAPAPAAAAAPAAPAAAAAAAAAAAAAAAAGZf/SGAGALAAB
display	
first_prefill_source	
first_prefill_type	
had_cp_prefilled	false
had_password_prefilled	false
id	maneesha@gmail.com
isprivate	
jazoest	2867
lgndim	eyJ3IjoxNTM2LCJoIjo4NjQsImF3IjoxNTM2LCJhaCI6ODI0LCJjIjoyNH0=
lgnjs	1694555878
lgnrnd	144509_VvOF
lsd	AVpAd14u2bI
password	abcdef
prefill_contact_point	
prefill_source	

Figure 12: View Details

By clicking "View Details" you can see the submitted credentials by the victim.

5 Getting Notification Using Webhooks

Monitoring events from the admin server's timeline is impractical because it's impossible for a human to continuously check for new events. However, we do need to receive event notifications promptly after they occur. Delayed responses could lead to victims becoming aware and changing their credentials.

For this issues, there is a feature called webhooks in gophish which helps to get realtime notification of campaign events.

Firstly we need a webhook server. We used `https://github.com/t94j0/gophish-notifier` for this purpose.

```
git clone https://github.com/t94j0/gophish-webhook-slack
cd gophish-webhook-slack
go build -o phishbot
```

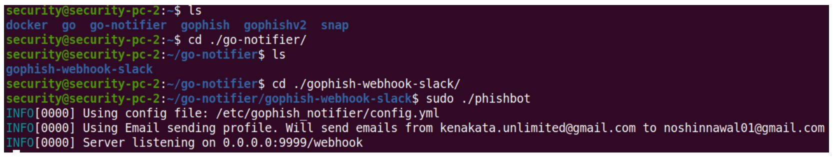
The configuration path is /etc/gophish_notifier/config.yml. Below is an example config:

```
# Host to listen on. If GoPhish is running on the same host, you can set this to 0.0.0.0
listen_host: 0.0.0.0
# Port to listen on
listen_port: 9999
# Webhook path. The full address will be http://<host>:<ip><webhook_path>. Ex: http://0.0.0.
webhook_path: /webhook
# Secret for GoPhish authentication
secret: secretpassword123
# Log level. Log levels are panic, fatal, error, warning, info, debug, trace.
log_level: debug
# (Optional) Base URL of GoPhish instance so that Slack notifications can link to campaign
base_url: https://0.0.0.0:3333
# Enables sending profiles. Options are `slack` and `email`. Make sure to configure the requ
profiles:
  - email

# Email Profile
# Email to send from
email:
  sender: test@test.com
  # Password of sender email. Uses plain SMTP authentication
  sender_password: password123
  # Recipient of notifications
  recipient: mail@example.com
  # Email host to send to
  host: smtp.gmail.com
  # Email host address
  host_addr: smtp.gmail.com:587

# You can also supply an email template for each notification
email_submitted_credentials_template: |
  Someone submitted credentials!
  Email ID - {{ .ID }}
  Email Address - {{ .Email }}
  IP Address - {{ .Address }}
  User Agent - {{ .UserAgent }}
  Username - {{ .Username }}
  Password - {{ .Password }}
```

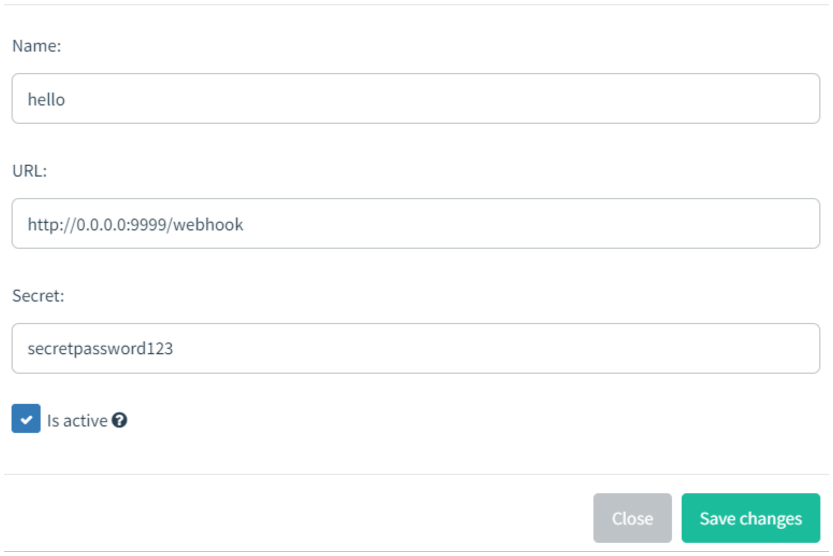
This webhook server worked for gmail clients.



```
security@security-pc-2:~$ ls
docker  go  go-notifier  gophish  gophishv2  snap
security@security-pc-2:~$ cd ./go-notifier/
security@security-pc-2:~/go-notifier$ ls
gophish-webhook-slack
security@security-pc-2:~/go-notifier$ cd ./gophish-webhook-slack/
security@security-pc-2:~/go-notifier/gophish-webhook-slack$ sudo ./phishbot
INFO[0000] Using config file: /etc/gophish_notifier/config.yml
INFO[0000] Using Email sending profile. Will send emails from kenakata.unlimited@gmail.com to noshinnawal01@gmail.com
INFO[0000] Server listening on 0.0.0.0:9999/webhook
```

Figure 13: Webhook server started.

Gophish supports multiple webhooks. Only users with the Admin role are able to create webhooks by navigating to the "Webhooks" sidebar entry and clicking the "New Webhook" button.



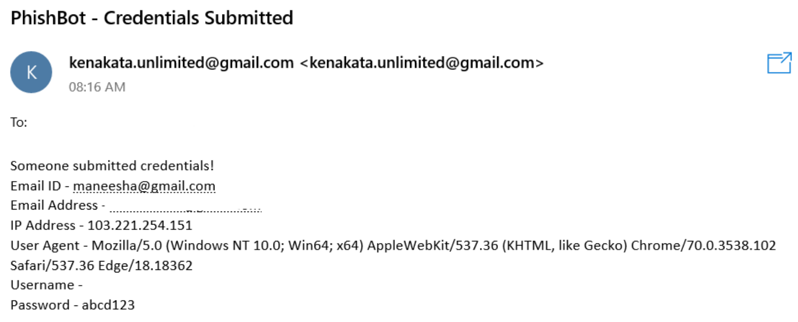
Name: hello

URL: http://0.0.0.0:9999/webhook

Secret: secretpassword123

✓ Is active

Figure 14: Setting up webhook.



PhishBot - Credentials Submitted

kenakata.unlimited@gmail.com <kenakata.unlimited@gmail.com>
08:16 AM

To:

Someone submitted credentials!
Email ID - maneesha@gmail.com
Email Address -
IP Address - 103.221.254.151
User Agent - Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.102 Safari/537.36 Edge/18.18362
Username -
Password - abcd123

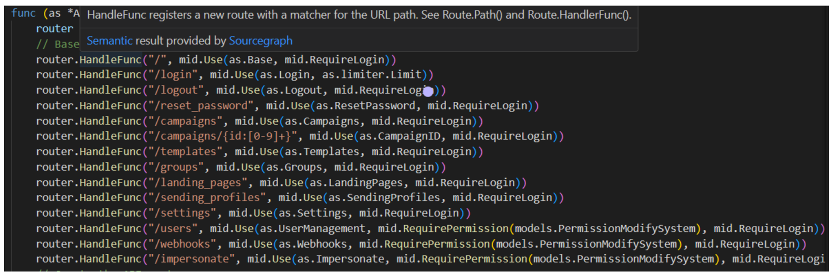
Figure 15: Getting notification by email.

6 High Level Overview of Source Code

You can find gophish source code here.

Gophish is written in golang. It uses MySQL as database. It is a monolithic website which uses MVC architectural pattern.

Flow of the website :
When we hit a URL it initally goes to the controller module and router.go process the request.



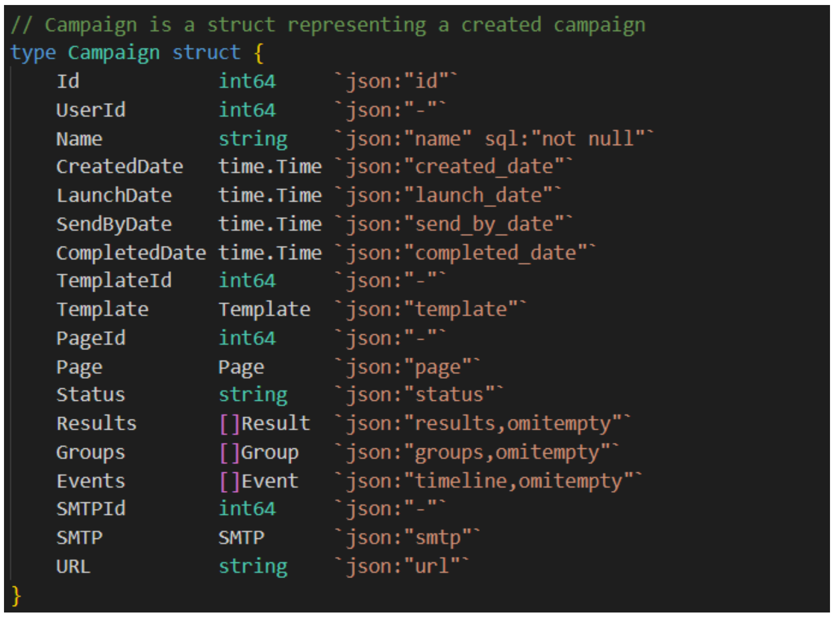
```
func (as *A  HandleFunc registers a new route with a matcher for the URL path. See Route.Path() and Route.HandlerFunc().
    router
    // Base  Semantic result provided by Sourcegraph
    router.HandleFunc("/", mid.Use(as.Base, mid.RequireLogin))
    router.HandleFunc("/login", mid.Use(as.Login, as.limiter.Limit))
    router.HandleFunc("/logout", mid.Use(as.Logout, mid.RequireLogin))
    router.HandleFunc("/reset_password", mid.Use(as.ResetPassword, mid.RequireLogin))
    router.HandleFunc("/campaigns", mid.Use(as.Campaigns, mid.RequireLogin))
    router.HandleFunc("/campaigns/{id:[0-9]+}", mid.Use(as.CampaignID, mid.RequireLogin))
    router.HandleFunc("/templates", mid.Use(as.Templates, mid.RequireLogin))
    router.HandleFunc("/groups", mid.Use(as.Groups, mid.RequireLogin))
    router.HandleFunc("/landing_pages", mid.Use(as.LandingPages, mid.RequireLogin))
    router.HandleFunc("/sending_profiles", mid.Use(as.SendingProfiles, mid.RequireLogin))
    router.HandleFunc("/settings", mid.Use(as.Settings, mid.RequireLogin))
    router.HandleFunc("/users", mid.Use(as.UserManagement, mid.RequirePermission(models.PermissionModifySystem), mid.RequireLogin))
    router.HandleFunc("/webhooks", mid.Use(as.Webhooks, mid.RequirePermission(models.PermissionModifySystem), mid.RequireLogin))
    router.HandleFunc("/impersonate", mid.Use(as.Impersonate, mid.RequirePermission(models.PermissionModifySystem), mid.RequireLogi
```

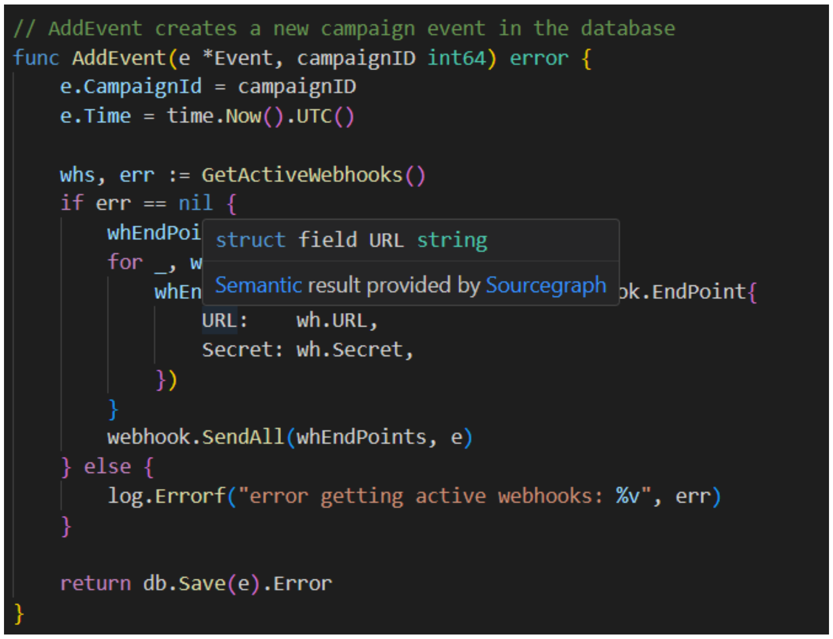
Figure 16: Request handling in router.go

In this context, you can observe that *router.HandleFunc* is used to establish associations between frontend routes and two functions: one function is responsible for delivering the corresponding HTML template, while the other serves as a middleware function, tasked with verifying the access control permissions of the authenticated user.

controller/api module acts like a bridge between *view* module and backend that is process the request from frontend and act accordingly on *models* module.

models module has the structure of every object and does the db operations. Example -


```go
// Campaign is a struct representing a created campaign
type Campaign struct {
	Id            int64     `json:"id"`
	UserId        int64     `json:"-"`
	Name          string    `json:"name" sql:"not null"`
	CreatedDate   time.Time `json:"created_date"`
	LaunchDate    time.Time `json:"launch_date"`
	SendByDate    time.Time `json:"send_by_date"`
	CompletedDate time.Time `json:"completed_date"`
	TemplateId    int64     `json:"-"`
	Template      Template  `json:"template"`
	PageId        int64     `json:"-"`
	Page          Page      `json:"page"`
	Status        string    `json:"status"`
	Results       []Result  `json:"results,omitempty"`
	Groups        []Group   `json:"groups,omitempty"`
	Events        []Event   `json:"timeline,omitempty"`
	SMTPId        int64     `json:"-"`
	SMTP          SMTP      `json:"smtp"`
	URL           string    `json:"url"`
}
```


```go
// AddEvent creates a new campaign event in the database
func AddEvent(e *Event, campaignID int64) error {
	e.CampaignId = campaignID
	e.Time = time.Now().UTC()

	whs, err := GetActiveWebhooks()
	if err == nil {
		whEndPoi struct field URL string
		for _, w
			whEn Semantic result provided by Sourcegraph ok.EndPoint{
				URL:    wh.URL,
				Secret: wh.Secret,
			})
		}
		webhook.SendAll(whEndPoints, e)
	} else {
		log.Errorf("error getting active webhooks: %v", err)
	}

	return db.Save(e).Error
}
```

Figure 17: Example from controllers/api/campaign.go

Other important modules - *templates* module has the html pages of the server, *logger* module writes the log of the admin server, *mailer and imap* module handles mailing related works, *dialer* module creates a network dialer, restricting outbound connections to specific allowed IP ranges while preventing connections to various internal and potentially unsafe IP addresses.

Important external packages :

- `github.com/jordan-wright/email` package is a third-party Go library for sending and managing email messages. It simplifies the process of creating, formatting, and sending email messages from your Go applications. This package is particularly useful when you want to send emails programmatically, such as for notifications, alerts, or transactional emails.

- `github.com/PuerkitoBio/goquery` package is a popular third-party Go library used for web scraping and parsing HTML documents. It provides a simple and flexible way to query and manipulate HTML content using jQuery-like selectors and methods. This library is particularly useful when you need to extract data from web pages, perform web scraping, or parse HTML documents in your Go applications.

- `github.com/jinzhu/gorm` package is a widely used third-party Go library for working with relational databases. GORM provides an Object-Relational Mapping (ORM) framework, which simplifies database interactions by allowing you to work with Go structs and objects instead of writing raw SQL queries. It supports various database systems, including MySQL, PostgreSQL, SQLite, and SQL Server

- `github.com/oschwald/maxminddb-golang` package is a third-party Go library that provides support for working with MaxMind's GeoIP2 and GeoLite2 databases. These databases contain geographical and location information for IP addresses, allowing you to determine the geographic location of an IP address, such as the country, city, and coordinates (latitude and longitude).

- `github.com/sirupsen/logrus` package is a widely used third-party Go logging library created by Simon Eskildsen. It provides a flexible and structured way to log messages in Go applications. Logrus is known for its simplicity and extensibility, making it a popular choice for logging in Go projects.

- `github.com/jordan-wright/unindexed` package is a third-party Go library created by Jordan Wright. It provides a middleware for HTTP servers that allows you to prevent search engines from indexing specific routes or paths on your website. This can be useful when you want to exclude certain content or pages from being indexed by search engines like Google, Bing, or others.

- `github.com/gorilla` GitHub organization hosts a collection of popular third-party Go packages that are commonly used for web development. These packages are widely adopted in the Go community and provide various tools and utilities for building web applications efficiently.

7 Video Demonstration Link

A video demonstration exploring all the features of Gophish can be found at:
`https://youtu.be/NolQ9THPsnc?si=gZO8lMXaCNBgd4OM`