Nerrzo Infra Solutions

PPE

GOPHISH

CAMPAGNE DE SENSIBILITATION

Presenter par
MOCQUILLON
Lucas

Dossier Technique – Infrastructure GoPhish (Lab Local → Entreprise)

Auteur : *MOCQUILLON Lucas*– Apprenti Administrateur Systèmes, Réseaux & Sécurité
Date : 28 janvier 2026
Contexte : Projet de sensibilisation au phishing avec l'outil open source GoPhish, d'abord en environnement local (homelab / BTS), puis réutilisation dans un contexte réel.

Table des matières

---

1. Introduction

1.1. Contexte général

Le phishing (hameçonnage) reste aujourd'hui l'un des vecteurs d'attaque les plus utilisés par les cybercriminels. Une grande majorité des cyberattaques ciblant les organisations commence par un courriel de phishing, visant à obtenir des identifiants, des données sensibles ou à faire exécuter un code malveillant par l'utilisateur.

Les protections techniques (antivirus, filtrage web, pare-feu, EDR, etc.) sont nécessaires mais ne suffisent plus : **l'utilisateur final est souvent le dernier rempart**. La sensibilisation et la formation deviennent donc des composants essentiels de la sécurité globale.

Dans ce contexte, l'outil **GoPhish** est utilisé comme framework de simulation de phishing pour :

- Concevoir des scénarios réalistes (e-mails + pages d'atterrissage).
- Lancer des campagnes de test auprès de collaborateurs.
- Mesurer les comportements : ouverture des mails, clics, saisies éventuelles.
- Débriefer et transformer les erreurs en apprentissage.

1.2. Objectif du projet

L'objectif du projet est de **concevoir, déployer et documenter** une infrastructure complète permettant :

- De tester GoPhish dans un environnement **local isolé** (homelab/école) en toute sécurité.
- De **préparer la montée en charge** vers une campagne de sensibilisation destiné a des étudiant de l'école Saint Félix la salle.
- De rester **100 % sur des composants gratuits / open source**.
- D'anticiper les principaux problèmes techniques (DNS, SMTP, SPF/DKIM, HTTPS), organisationnels (validation RSSI) et juridiques (RGPD, éthique).

Ce dossier technique doit pouvoir servir de base à une mise en production contrôlée.

1.3. Présentation de GoPhish

GoPhish est un **framework open source de simulation de phishing**, développé en Go, et distribué sous licence MIT.

Fonctionnalités principales :

- Interface web d'administration.
- Création de templates d'e-mails.
- Création de landing pages (pages web d'atterrissage).
- Gestion des listes de cibles.

- Gestion des campagnes (lancement, suivi, statistiques en temps réel).
- Export des résultats.

GoPhish ne fournit **pas** de serveur SMTP intégré : il doit être associé à un **relais de messagerie** (local ou externe), ce qui impose de réfléchir à l'architecture mail (Postfix, Exchange, service cloud, etc.).

---

2. Cahier des charges

2.1. Besoins fonctionnels

- Permettre la création et l'envoi de campagnes de phishing depuis une interface web GoPhish.
- Pouvoir envoyer des e-mails vers des boîtes aux lettres de test (environnement local) puis vers des boîtes professionnelles (en environnement entreprise).
- Proposer des landing pages pédagogiques montrant les indices de phishing après clic, sans jamais voler de mots de passe réels.
- Fournir des statistiques détaillées : e-mails délivrés, ouverts, clics, données saisies (optionnel et encadré).
- Permettre la configuration de scénarios variés (alerte mail, note RH, mise à jour de sécurité, etc.).

2.2. Besoins techniques

- Infrastructure hébergée dans un **homelab** type Proxmox / pfSense, sur réseau isolé.
- Utilisation de systèmes **Linux Debian/Ubuntu** pour les serveurs.
- Utilisation de **Docker** pour l'exécution de GoPhish (facilite la mise à jour et la portabilité).
- Mise en place d'un **reverse proxy Nginx** pour publier GoPhish en HTTPS.
- Mise en place d'un **serveur SMTP Postfix** comme relais pour l'envoi des mails de test.
- Configuration d'un **DNS local** pour les noms de domaine utilisés dans les scénarios.
- Compatibilité avec une future intégration dans l'infrastructure en condition réel. (Exchange, DNS publics, etc.).

2.3. Contraintes

- Utiliser uniquement des solutions **gratuites / open source**.
- Ne pas impacter la production réelle durant la phase de test en local.
- Limiter l'exposition de la plateforme vers Internet (filtrage firewall, DMZ, segmentation réseau).
- Respecter les contraintes **RGPD** et l'éthique de la sensibilisation : transparence, absence de collecte abusive, données minimisées, suppression après débriefing.

---

3. Architecture cible

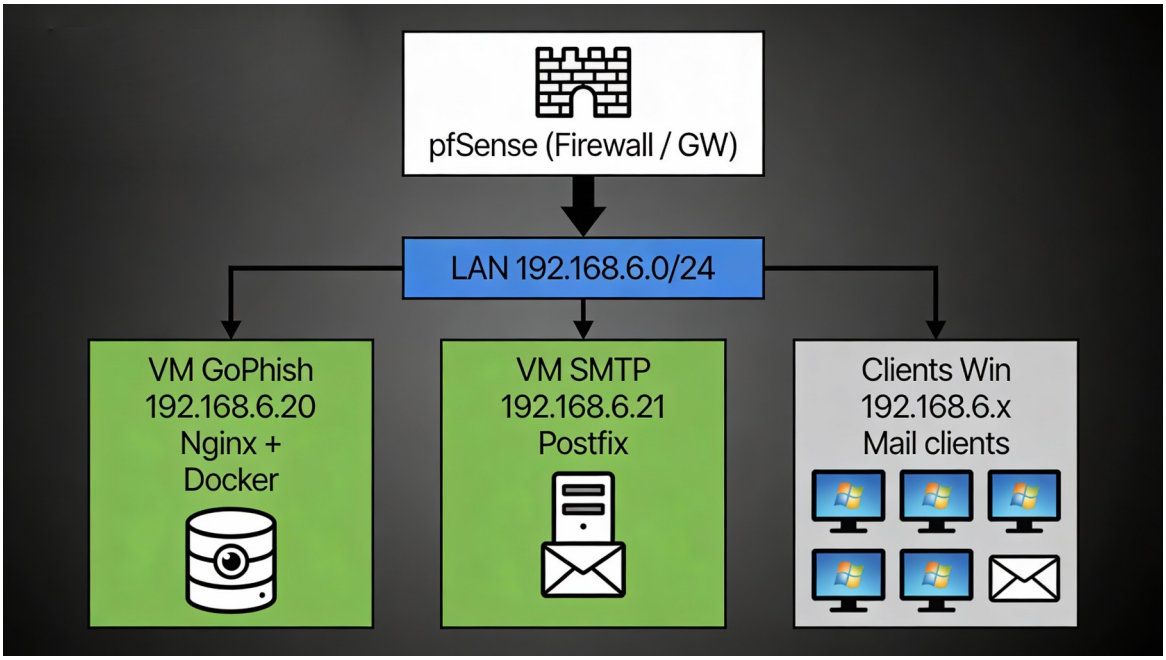
3.1. Architecture logique en environnement local (Lab)

Réseau de lab type : `192.168.6.0/24` (LAN homelab).

Composants :

- **VM "GoPhish"**
    - OS : Debian 12 ou Ubuntu Server LTS.
    - Services :
        - Docker + container GoPhish.
        - Nginx en reverse proxy HTTP/HTTPS.
    - IP : `192.168.6.20`.
- **VM "SMTP/Postfix"** (optionnel si on sépare)
    - OS : Debian 12.
    - Service : Postfix (relais SMTP local).
    - IP : `192.168.6.21`.
- **DNS interne (Bind9 ou pfSense)**
    - Résolution des noms :
        - `phishing.lab` → landing pages GoPhish.
        - `admin.phishing.lab` → interface d'administration GoPhish.
    - Permet d'éviter l'utilisation d'IP brutes dans les liens.
- **Clients de test**
    - VMs Windows 10/11.
    - Client mail : Outlook, Thunderbird ou webmail interne.

Schéma logique (simplifié) :



3.2. Architecture envisagée en production (entreprise)

En production la même logique peut être adaptée :

- Hébergement sur un **VPS** (Oracle Cloud Free Tier, par exemple) ou en DMZ de l'entreprise.
- Nom de domaine public dédié :
  - Exemple : `sensibilisation.entreprise.fr`.
- GoPhish + Nginx sur une VM Linux.
- Relais SMTP :
  - Soit via le serveur de messagerie interne (Exchange/Office 365) avec règles adaptées.
  - Soit via un serveur SMTP dédié.
- Configuration DNS publique + enregistrements SPF, DKIM, DMARC.

L'architecture locale sert alors de **maquette fonctionnelle** pour préparer le passage en production.

---

4. Mise en place de l’infrastructure locale

4.1. Préparation des machines virtuelles

4.1.1. Paramétrage Proxmox (ou équivalent)

- Création de deux VMs :
    - **VM GoPhish** :
        - 2 vCPU, 4 Go RAM, 40 Go disque.
        - OS : Debian 12.
    - **VM SMTP** :
        - 1 vCPU, 2 Go RAM, 20 Go disque.
        - OS : Debian 12.
- Chaque VM est raccordée au réseau virtuel correspondant au LAN de lab (vmbr0, par exemple).

4.1.2. Configuration réseau statique

Exemple pour la VM GoPhish (Debian 12) via Netplan :

```
network:
version: 2
ethernets:
ens18:
dhcp4: no
addresses:
- 192.168.6.20/24
gateway4: 192.168.6.1
nameservers:
addresses: [192.168.6.1, 8.8.8.8]
```

Commande d’application :

```
sudo netplan apply
```

Même principe pour la VM SMTP (ex. 192.168.6.21).

4.1.3. Mise à jour et paquets de base

Sur chaque VM :

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y wget curl unzip vim
```

4.2. Installation de GoPhish

Deux approches possibles : installation directe ou via Docker. Nous privilégions **Docker** pour la portabilité.

4.2.1. Installation directe (binaire)

Documentation officielle : GoPhish User Guide – Installation.

```
cd /opt
sudo mkdir gophish && cd gophish
sudo wget https://github.com/gophish/gophish/releases/download/v0.12.1/gophish-v0.12.1-linux-64bit.zip
sudo unzip gophish-v0.12.1-linux-64bit.zip
sudo chmod +x gophish
```

Lancement de test :

```
sudo ./gophish
```

Par défaut :

- Interface admin : `https://localhost:3333`.
- Serveur “phishing” : `http://localhost:80`.

Les identifiants sont affichés lors du premier démarrage.

4.2.2. Installation via Docker (recommandée)

1. Installer Docker :

```
sudo apt install -y docker.io docker-compose
sudo systemctl enable --now docker
```

2. Créer un répertoire dédié :

```
sudo mkdir -p /opt/gophish
cd /opt/gophish
```

3. Exemple de `docker-compose.yml` minimal :

```
version: "3"
services:
gophish:
image: gophish/gophish:latest
container_name: gophish
ports:
- "3333:3333" # Admin
- "80:80" # Phishing HTTP
- "443:443" # Phishing HTTPS (option)
volumes:
```

```
- ./config:/opt/gophish/config
restart: unless-stopped
```

4. Lancer GoPhish :

```
sudo docker-compose up -d
```

GoPhish est alors accessible en local sur `https://VM:3333` (admin) et `http://VM:80` (phishing).

4.3. Mise en place du reverse proxy Nginx + HTTPS

L'objectif est de publier GoPhish :

- En HTTPS (pour éviter les alertes de sécurité).
- Sous des noms de domaine lisibles (`phishing.lab`, `admin.phishing.lab`).

4.3.1. Installation de Nginx

Sur la VM GoPhish :

```
sudo apt install -y nginx
sudo systemctl enable --now nginx
```

4.3.2. Génération de certificats TLS

- **En environnement local** : utilisation d'un outil comme `mkcert` (certificats non publics, mais reconnus par les postes du lab).
- **En production** : utilisation de Let's Encrypt via Certbot. [

Exemple (production) :

```
sudo apt install -y certbot python3-certbot-nginx
sudo certbot --nginx -d phishing.lab -d admin.phishing.lab
```

4.3.3. Configuration Nginx

Fichier `/etc/nginx/sites-available/gophish` :

```
server {
listen 80;
server_name phishing.lab;
return 301 https://servername-request_uri;
}

server {
listen 443 ssl http2;
server_name phishing.lab;
```

```
ssl_certificate /etc/letsencrypt/live/phishing.lab/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/phishing.lab/privkey.pem;

location / {
    proxy_pass http://127.0.0.1:80;  # serveur phishing GoPhish
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
}

}

server {
listen 443 ssl http2;
server_name admin.phishing.lab;

ssl_certificate /etc/letsencrypt/live/phishing.lab/fullchain.pem;
ssl_certificate_key /etc/letsencrypt/live/phishing.lab/privkey.pem;

location / {
    proxy_pass http://127.0.0.1:3333;  # interface admin GoPhish
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
}

}
```

Activation :

```
sudo ln -s /etc/nginx/sites-available/gophish /etc/nginx/sites-enabled/gophish
sudo nginx -t
sudo systemctl reload nginx
```

4.4. Mise en place du serveur SMTP Postfix (lab)

La VM SMTP fournit un relais local pour l'envoi d'e-mails de test.

4.4.1. Installation de Postfix

```
sudo apt install -y postfix
```

Pendant l'installation, choisir :

- Type de configuration : **Site Internet** (ou équivalent).
- Nom de courrier : `lab.local` (par exemple).

4.4.2. Configuration de base

Éditer /etc/postfix/main.cf pour limiter le relais au réseau du lab :

```
myhostname = smtp.lab.local
mydomain = lab.local
myorigin = $mydomain
inet_interfaces = all
mynetworks = 127.0.0.0/8 192.168.6.0/24
relay_domains = $mydomain
smtpd_recipient_restrictions = permit_mynetworks, reject_unauth_destination
```

Redémarrage :

```
sudo systemctl restart postfix
```

4.4.3. Ajout de DKIM (préparation prod)

Même si ce n'est pas critique en lab, il est utile de préparer la configuration DKIM.

Installation :

```
sudo apt install -y opendkim opendkim-tools
```

Génération des clés :

```
sudo opendkim-genkey -t -s mail -d lab.local
sudo chown opendkim:opendkim mail.private
```

Configuration de opendkim.conf et lien avec Postfix, puis création de l'enregistrement DNS TXT correspondant (clé publique).

4.5. Mise en place du DNS local

Le DNS local permet aux clients de résoudre les noms utilisés pour GoPhish.

4.5.1. Bind9

Installation :

```
sudo apt install -y bind9
```

Fichier /etc/bind/named.conf.local :

```
zone "lab.local" {
type master;
file "/etc/bind/db.lab.local";
};
```

Fichier /etc/bind/db.lab.local (exemple minimal) :

```
$TTL 86400
@ IN SOA ns.lab.local. admin.lab.local. (
2026012801 ; Serial
3600 ; Refresh
1800 ; Retry
604800 ; Expire
86400 ) ; Negative Cache TTL

@ IN NS ns.lab.local.

ns IN A 192.168.6.20
phishing IN A 192.168.6.20
admin IN A 192.168.6.20
smtp IN A 192.168.6.21
```

Redémarrer :

```
sudo systemctl restart bind9
```

Configurer les clients pour utiliser ce DNS (via parfeu ou manuellement).

4.6. Configuration GoPhish (interface)

Une fois GoPhish et Nginx en place, accéder à :

- Interface admin : https://admin.phishing.lab
- Login/mot de passe : ceux fournis (ou modifiés) au premier démarrage.

4.6.1. Configuration générale

- Modifier l'URL de base (base URL) pour correspondre au domaine réel utilisé (ex. https://phishing.lab).
- Configurer les **Sending Profiles** (profils d'envoi) en pointant vers le serveur SMTP (smtp.lab.local ou 192.168.6.21).
- Créer des groupes de cibles (adresses mails des VMs de test).

4.6.2. Création d'un template d'e-mail

Exemple : "Alerte quota boîte mail".

- Objet : [IT] Votre boîte mail a presque atteint son quota
- Corps : e-mail reprenant les codes graphiques d'un vrai message (logo, signature, etc.), mais en conservant quelques **indices de phishing** (adresse d'expéditeur suspecte, ton alarmiste, fautes, etc.). [1][3]

4.6.3. Création d'une landing page pédagogique

Plutôt que de voler un mot de passe, la landing page affichera un message de sensibilisation :

- Titre : `Exercice de sensibilisation à la sécurité`
- Texte :
  "Vous venez de participer à une simulation de phishing. Voici les indices qui auraient dû vous alerter : …"
- On peut y intégrer une liste des signaux d'alerte (URL, orthographe, urgence artificielle, etc.). [3]

4.7. Vérification et tests

4.7.1. Tests techniques

- **Ping / DNS** :
  - `ping phishing.lab` doit répondre.
  - `nslookup phishing.lab` doit renvoyer `192.168.6.20`.
- **SMTP** :
  - `telnet smtp.lab.local 25`
  - Envoi de mail de test en ligne de commande (`mail`, `swaks`, etc.).
- **GoPhish** :
  - Accès à l'interface admin.
  - Création d'une campagne test.
  - Vérification des logs (`/var/log/syslog`, `/var/log/mail.log`).

4.7.2. Tests fonctionnels

- Envoi d'une campagne vers les VMs clients.
- Réception des e-mails sur Outlook/Thunderbird.
- Clic sur le lien → redirection vers la landing page.
- Vérification des statistiques dans GoPhish :
  - Mails envoyés.
  - Mails ouverts.
  - Clics.
  - Saisies (si activées).

---

5. Anticipation des problèmes pour la future mise en production

5.1. Problèmes techniques fréquents

- **E-mails bloqués ou classés en spam** :
    - Cause : manque de SPF/DKIM/DMARC, réputation IP faible. [6][7]
    - Prévention :
        - Configurer SPF/DKIM/DMARC sur le domaine de production.
        - Faire valider par l'équipe Exchange/O365.
        - Tester auprès de quelques boîtes de test avant un déploiement plus large.
- **Landing page inaccessible depuis l'extérieur** :
    - Cause : Firewall/Proxy, NAT non configuré.
    - Prévention :
        - Publication en DMZ ou sur VPS avec pare-feu maîtrisé.
        - Test depuis un poste externe à l'entreprise.
- **Incompatibilités réseau** :
    - Cause : Proxy d'entreprise, filtrage HTTPS, DPI.
    - Prévention :
        - Collaboration avec l'équipe réseau pour whitelist du domaine de test.

5.2. Problèmes organisationnels

- **Refus ou réticence de la direction / RSSI** :
    - Risque d'être perçu comme une "chasse aux sorcières".
    - Solutions :
        - Présenter clairement les objectifs pédagogiques.
        - Insister sur la bienveillance : on ne "punit" pas les utilisateurs, on les forme. [3]
        - Proposer une première campagne limitée à un petit échantillon volontaire.

5.3. Problèmes juridiques (RGPD)

- Collecte d'adresses mails professionnelles.
- Collecte potentielle de comportements (clics, saisies).
- Traçage des actions (logs).

Points à respecter :

- **Base légale** : intérêt légitime de sécuriser le SI, en lien avec les obligations de l'employeur en matière de sécurité. [3]
- **Minimisation** : ne collecter que ce qui est nécessaire (clic, ouverture, etc.).
- **Transparence** : informer les salariés (charte informatique, note interne).
- **Durée de conservation limitée** : suppression des données individuelles après le débrief.
- **Anonymisation / agrégation** : privilégier les statistiques anonymisées par service ou population.

---

6. Cadre éthique de la sensibilisation

Afin que la campagne ne soit pas vécue comme un "piège malveillant", les règles suivantes sont appliquées :

6.1. Pas de capture réelle de mots de passe

- Les landing pages ne stockent jamais les mots de passe saisis.
- Idéalement, le formulaire est remplacé par un texte pédagogique dès que l'utilisateur clique ou tente de se connecter.
- Si un formulaire est techniquement utilisé, les champs sont anonymisés et immédiatement détruits après la campagne.

6.2. Transparence et bienveillance

- Communication en amont (ou au minimum lors du débrief) expliquant :
    - Les objectifs des simulations.
    - Le rôle de chacun (DSI, RSSI, utilisateurs).
- Les résultats sont :
    - Présentés de manière **anonyme** (par département, profil, etc.).
    - Utilisés pour organiser des formations ciblées, non des sanctions individuelles.

6.3. Conformité interne

- Alignement avec :
    - La charte informatique de l'entreprise.
    - Le règlement intérieur.
    - Les directives du RSSI / DPO.

---

7. Conclusion

Ce projet a permis de :

- Concevoir une **infrastructure locale complète** autour de GoPhish (VMs, DNS, SMTP, Nginx, HTTPS), en environnement homelab, pour tester en toute sécurité un outil de simulation de phishing.
- Anticiper les problématiques techniques (SMTP, DNS, filtrage, authentification des mails), organisationnelles (validation par la direction et le RSSI) et juridiques (RGPD, cadre éthique).
- Préparer une **montée en charge progressive** vers une campagne de sensibilisation au phishing dans l'entreprise (Ugecam BRPL), tout en conservant une approche pédagogique et bienveillante.

Ce dossier technique peut servir :

- De base pour l'épreuve E4/E5 du BTS SIO SISR.
- De support pour présenter le projet à l'équipe SI / RSSI.
- De référentiel pour la mise en production et les évolutions futures (intégration avec SIEM, dashboards de suivi, etc.).

Références

[1] Wikipédia. (2025). *Hameçonnage*. https://fr.wikipedia.org/wiki/Hameçonnage

[2] Gophish Project. (2022). *Installation | Gophish User Guide*. https://docs.getgophish.com/user-guide/installation

[3] CNIL. (2023). *Sécurité des données personnelles et sensibilisation des employés*. https://www.cnil.fr

[4] Gophish GitHub Repository. (2025). *gophish/gophish: Open-Source Phishing Toolkit*. https://github.com/gophish/gophish

[5] It-Connect. (2024). *Publier GoPhish en HTTPS avec un reverse proxy Nginx*. https://www.it-connect.fr/gophish-reverse-proxy-nginx-et-certificat-lets-encrypt/

[6] SIDN. (2020). *Hands-on: implementing SPF, DKIM and DMARC in Postfix*. https://www.sidn.nl/en/news-and-blogs/hands-on-implementing-spf-dkim-and-dmarc-in-postfix

[7] Leosmith. (2022). *gophish setup tutorial*. https://leosmith.wtf/blog/gophish-setup-tutorial.html