“Gophishing” to Boost your Security Awareness Training

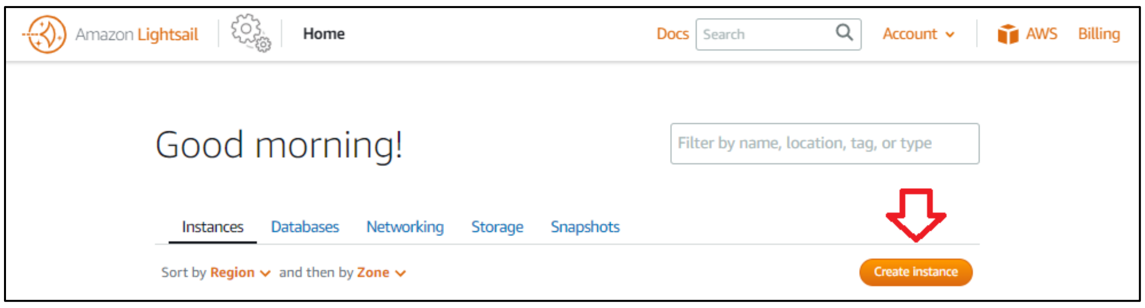
Any information security program must consider how users react to phishing campaigns. It is undeniable that constant training helps to defend against deceptive tactics. Many of these tactics entice employees to click on malicious links or provide credentials. Gathering credentials is a simple method for an attacker to establish a foothold in a targeted organization.

In most cases, the desirable training is obtained through third-party vendors that offer pre-established phishing campaigns with limited customization. However, for organizations looking for a more flexible and affordable process to train their employees, freely available tools can be easily integrated into the network infrastructure.

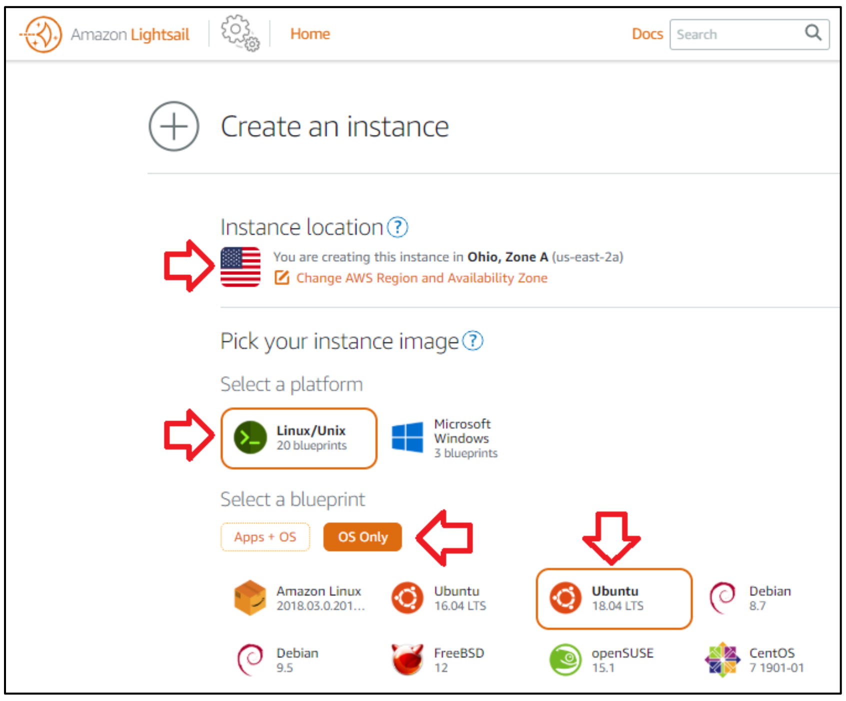
Gophish is an excellent example of these types of open-source tools. As described in its documentation portal, “Gophish is a powerful, easy-to-use, open-source phishing toolkit designed to help pentesters and businesses conduct real-world phishing simulations.”[1] It also provides all the flexibility sought by system administrators to launch in-house phishing campaigns.

In this article, we’ll walk through the setup of the Gophish toolkit. The goal is to design a phishing platform that will help network administrators to conduct tailored phishing campaigns in a short amount of time.

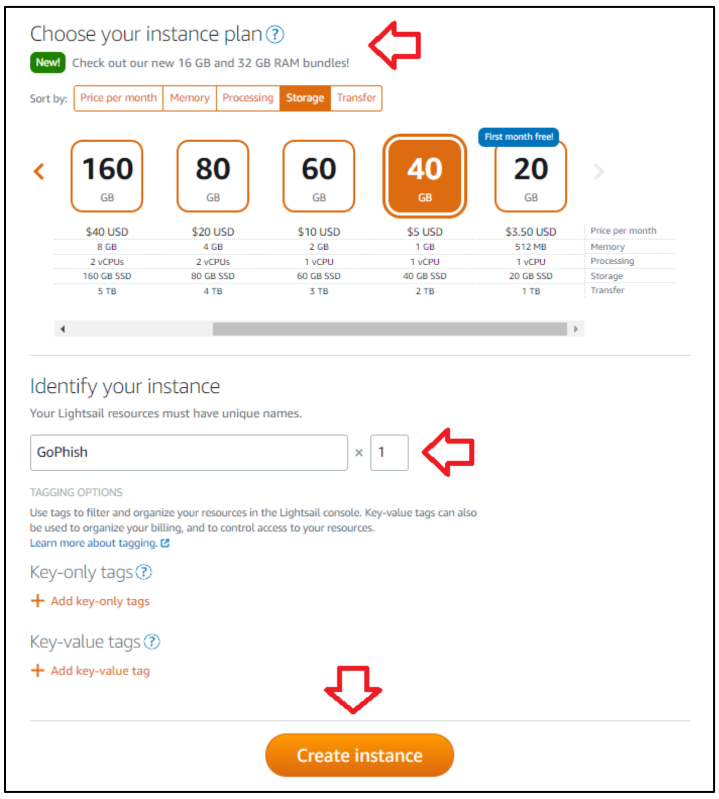
1. For easy deployment, we will be using an AWS LightSail instance. This is also known as a virtual private server (VPS). After signing up for AWS, go to the **Home** page and choose “**Create Instance.**”



2. Then select the location of the instance (**AWS Region and Availability Zone**). Under Pick instance image, select **Linux/Unix** and **OS only** – Ubuntu 18.04.1 LTS.



3. Choose an instance plan and label the instance. For our example, we selected a 40 GB Storage plan and labeled it “Gophish.” Next, click on “**Create instance.**” For more info on how to create an instance, see a reference here. Once the process is complete, make a note of the provided ***PublicIP_AWS_Instance.***


Choose your instance plan

New! Check out our new 16 GB and 32 GB RAM bundles!

Sort by: Price per month | Memory | Processing | Storage | Transfer

First month free!

160 GB	80 GB	60 GB	40 GB	20 GB	
$40 USD	$20 USD	$10 USD	$5 USD	$3.50 USD	Price per month
8 GB	4 GB	2 GB	1 GB	512 MB	Memory
2 vCPUs	2 vCPUs	1 vCPU	1 vCPU	1 vCPU	Processing
160 GB SSD	80 GB SSD	60 GB SSD	40 GB SSD	20 GB SSD	Storage
5 TB	4 TB	3 TB	2 TB	1 TB	Transfer

Identify your instance

Your Lightsail resources must have unique names.

GoPhish × 1

TAGGING OPTIONS

Use tags to filter and organize your resources in the Lightsail console. Key-value tags can also be used to organize your billing, and to control access to your resources.
Learn more about tagging.

Key-only tags

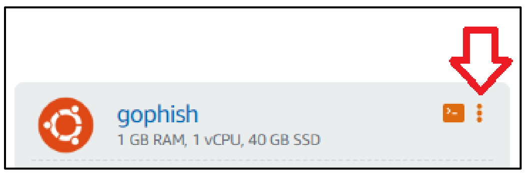
+ Add key-only tags

Key-value tags

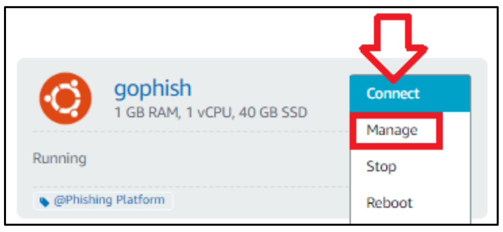
+ Add key-value tag

Create instance

4. To allow HTTPS communication (Hypertext Transfer Protocol Secure), it is necessary to enable port 443 on the newly created instance. Click on ":" :

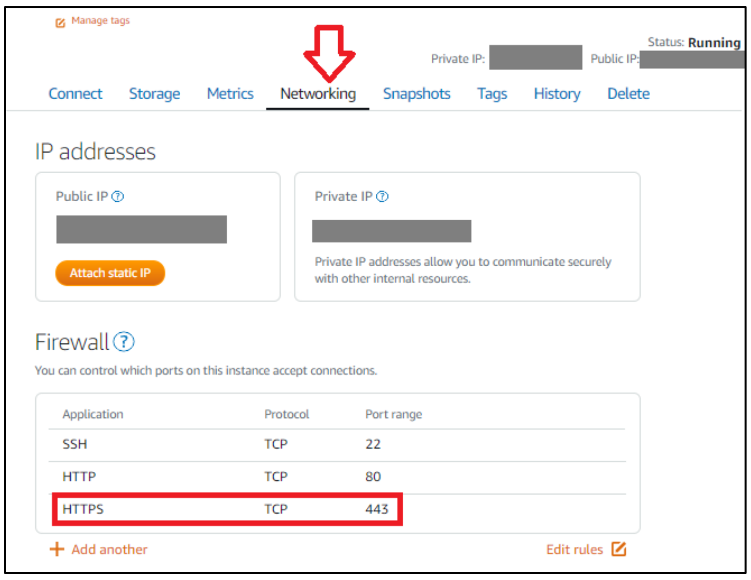

gophish
1 GB RAM, 1 vCPU, 40 GB SSD

Then **Manage**:


gophish
1 GB RAM, 1 vCPU, 40 GB SSD
Running
@Phishing Platform

Connect
Manage
Stop
Reboot

Under the **Networking** tab, set the port number (443) in the **Firewall** section:


Manage tags

Status: Running
Private IP: [redacted] Public IP: [redacted]

Connect | Storage | Metrics | Networking | Snapshots | Tags | History | Delete

IP addresses

Public IP
[redacted]
Attach static IP

Private IP
[redacted]
Private IP addresses allow you to communicate securely with other internal resources.

Firewall

You can control which ports on this instance accept connections.

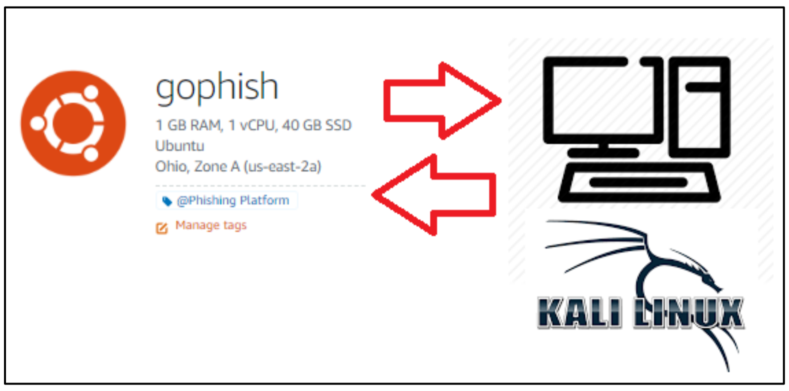
Application	Protocol	Port range
SSH	TCP	22
HTTP	TCP	80
HTTPS	TCP	443

+ Add another
Edit rules

5. (OPTIONAL) To avoid dealing with SSH keys during the configuration process, SSH-Password authentication can be enabled (Also, we can allow logging as root). Use the following link for details: https://serverpilot.io/docs/how-to-enable-ssh-password-authentication/

Note: If you enable this option, use secure passwords – If logging as root was also enabled, **DO NOT FORGET** to revert this setting upon finishing the steps on this blog post.

6. It is necessary to understand the infrastructure and how to connect to it. The following **diagram** will help us to visualize the connection paths. Here, we use a Kali Linux box to connect to our AWS instance via SSH. We could use any other Linux distribution or Windows OS to establish an SSH connection to Gophish.



- **Gophish** = VPS hosted on AWS
- **Kali Linux** = The remote host to access the Gophish instance (used for Web Console and SSH access to Gophish).

7. Using the following command, SSH to the Gophish instance from the Kali host:

   ssh root@[PublicIP_AWS_Instance] -p22

8. Visit the Gophish repository and download the latest release. In our example, we downloaded the following zipped binary (gophish-v0.11.0-linux-64bit.zip):

   cd /tmp

   wget *https://github.com/gophish/gophish/releases/download/v0.11.0/gophish-v0.11.0-linux-64bit.zip*

9. Create a Gophish destination folder and unzip the downloaded binary:

   mkdir /opt/gophish-v0.11.0-linux-64bit

   unzip gophish-v0.11.0-linux-64bit.zip -d /opt/gophish-v0.11.0-linux-64bit/

10. Change the access permissions on the Gophish binary and run the application:

    cd /opt/gophish-v0.11.0-linux-64bit/

    cdmod +x gophish

    ./gophish

11. Here, we are ready to access the Gophish console. Since the console is only accessible through the browser, we need to establish a **Local Port Forwarding** to the AWS instance. Let's open another terminal in the Kali host and use the following command (**Gophish's default console port is 3333**):

    ssh -L3333:localhost:3333 root@[PublicIP_AWS_Instance] -p22

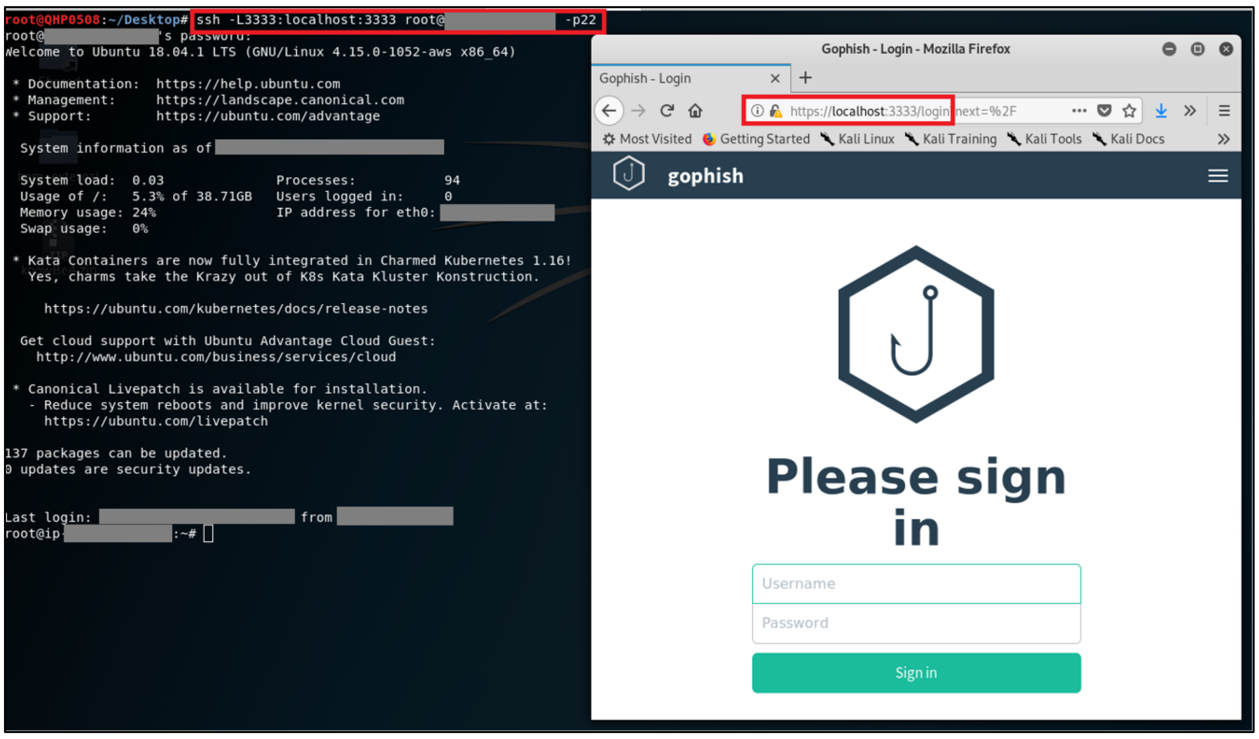
    Then, open the local browser (e.g., Firefox) and go to:

    https://localhost:3333/login

    *Default Credentials for Gophish: ***Admin / Gophish***

Note Gophish v0.11.0 :

This release adds a basic password policy for administrators, and removes the default password "gophish". Instead, an initial password is randomly generated and printed in the terminal when Gophish is launched for the first time. [2]



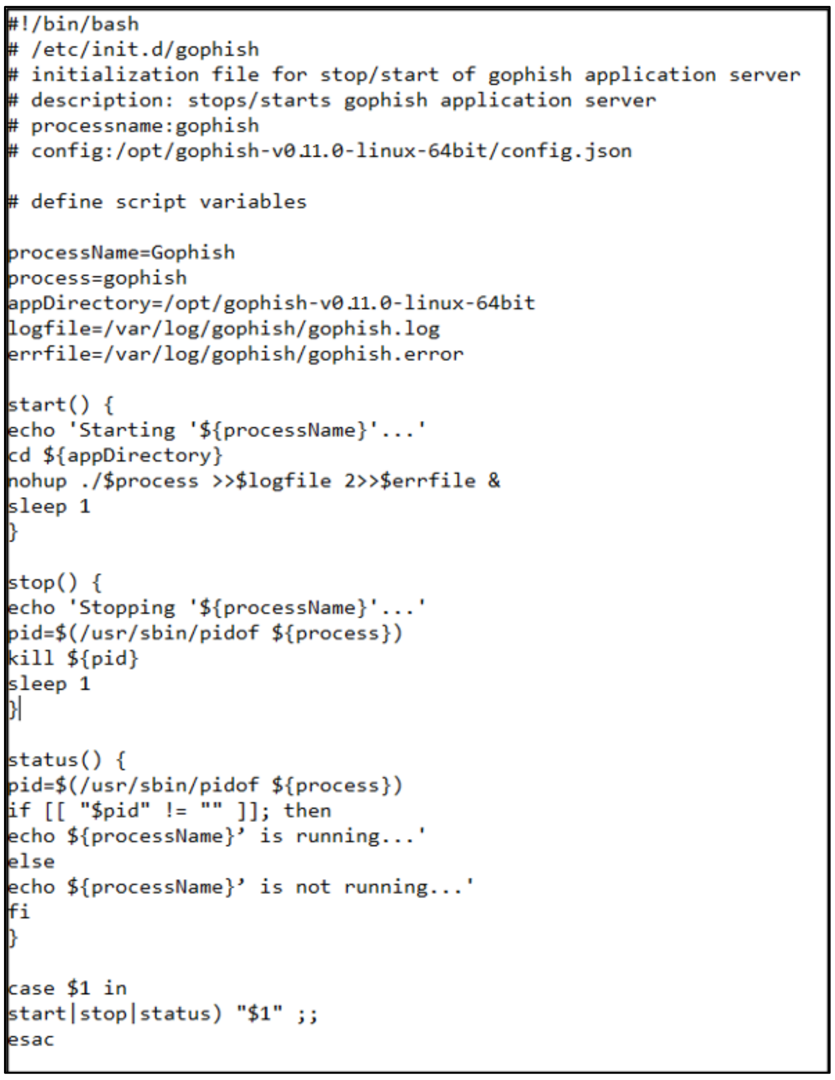
```
root@QHP0508:~/Desktop# ssh -L3333:localhost:3333 root@[redacted] -p22
root@[redacted]'s password:
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-1052-aws x86_64)

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

  System information as of [redacted]

  System load:  0.03               Processes:           94
  Usage of /:   5.3% of 38.71GB   Users logged in:     0
  Memory usage: 24%                IP address for eth0: [redacted]
  Swap usage:   0%

 * Kata Containers are now fully integrated in Charmed Kubernetes 1.16!
   Yes, charms take the Krazy out of K8s Kata Kluster Konstruction.

     https://ubuntu.com/kubernetes/docs/release-notes

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

137 packages can be updated.
0 updates are security updates.


Last login: [redacted] from [redacted]
root@ip-[redacted]:~#
```

12. After confirming console access, we need to set Gophish as a service. It will allow us to execute Gophish in the AWS instance background. For this, we set the following script in the **/etc/init.d** path:

    nano /etc/init.d/gophish

    Then add the following content – The script can be found here if you want to copy and paste.

    Note*: if you copy and paste the script from the repository, make sure to review its syntax. You may have to replace every ellipsis (…) with three periods and every quote (“) and half-quote (‘) manually. This is because the formatting might change during the copying process.*



```
#!/bin/bash
# /etc/init.d/gophish
# initialization file for stop/start of gophish application server
# description: stops/starts gophish application server
# processname:gophish
# config:/opt/gophish-v0.11.0-linux-64bit/config.json

# define script variables

processName=Gophish
process=gophish
appDirectory=/opt/gophish-v0.11.0-linux-64bit
logfile=/var/log/gophish/gophish.log
errfile=/var/log/gophish/gophish.error

start() {
echo 'Starting '${processName}'...'
cd ${appDirectory}
nohup ./$process >>$logfile 2>>$errfile &
sleep 1
}

stop() {
echo 'Stopping '${processName}'...'
pid=$(/usr/sbin/pidof ${process})
kill ${pid}
sleep 1
}

status() {
pid=$(/usr/sbin/pidof ${process})
if [[ "$pid" != "" ]]; then
echo ${processName}’ is running...'
else
echo ${processName}’ is not running...'
fi
}

case $1 in
start|stop|status) "$1" ;;
esac
```

CTRL + X to exit, then ‘y’ and ‘enter’ to save changes.

13. Set the Gophish log directory:

    mkdir /var/log/gophish

14. Now make the Gophish script file executable:

    chmod +x /etc/init.d/gophish

15. We have to add the Gophish service to “**update-rc.d**” to ensure it’s initiated every time the AWS server starts:

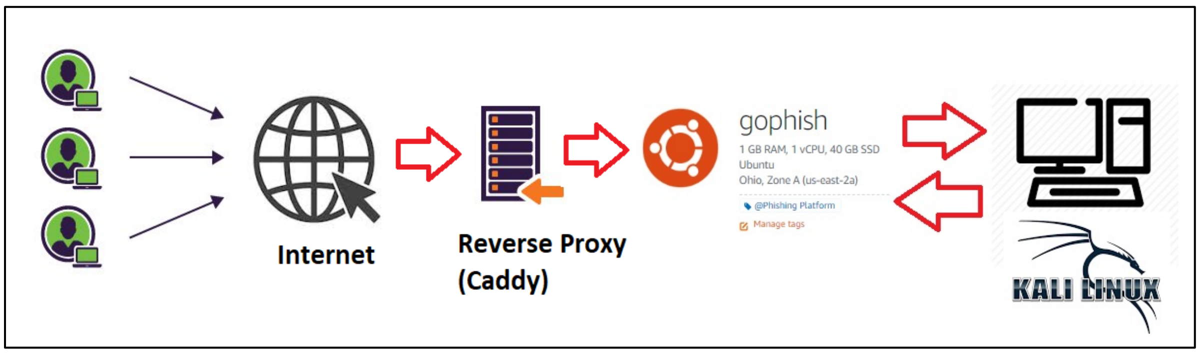
    update-rc.d gophish defaults

16. Start the Gophish service:

    service gophish start

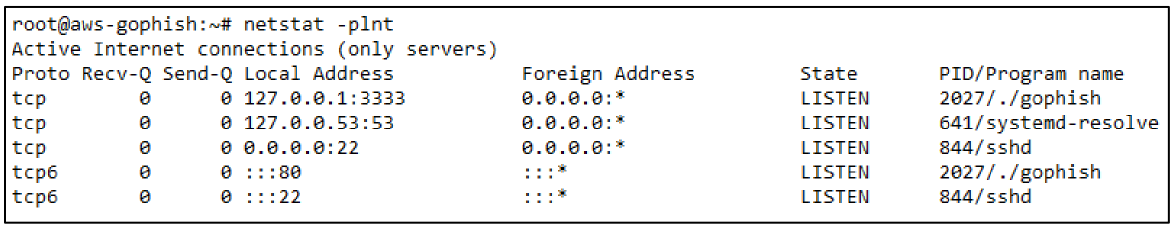
17. After the above settings, we need to test access to the Gophish console - See step 11.

18. At this point, we should be ready to start using Gophish. Before exploring the console, we need to address a Gophish limitation related to the number of SSL connections that Gophish can handle. As of today, Gophish (v.0.11.0) only supports one (1) SSL connection (defined on its config file “**config.json**”). This limitation restricts us from using multiple domains and phishing landing pages over HTTPS. To address this issue, we need to use a **reverse proxy.** The proxy will manage the SSL connections and redirect the traffic to Gophish. To achieve the goal, we use Caddy. “Caddy is an open source web server with automatic HTTPS written in Go”.[3] Caddy will also auto-deploy the Let’s Encrypt SSL certificates for the phishing domains through Certbot. The diagram below provides a visual representation for the wanted infrastructure:



Before moving on, we need to stop the Gophish service. Let's use the following command to identify the Gophish service process ID (PID):

netstat -plnt



```
root@aws-gophish:~# netstat -plnt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3333          0.0.0.0:*               LISTEN      2027/./gophish
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      641/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      844/sshd
tcp6       0      0 :::80                   :::*                    LISTEN      2027/./gophish
tcp6       0      0 :::22                   :::*                    LISTEN      844/sshd
```

Gophish PID = 2027

Then kill the Gophish process: ***kill -9 <Gophish _PID>***:

kill -9 2027

Let's now install Certbot (For our ubuntu distribution):

apt-get update

apt-get install software-properties-common

add-apt-repository universe

add-apt-repository ppa:certbot/certbot

apt-get updateb

apt-get install certbot

Installing Caddy:

curl https://getcaddy.com | bash -s personal

This will put the Caddy binary in ***/usr/local/bin/caddy***

Once Caddy is installed, let's create a caddy config file:

cd to any location where you want to keep you caddy file.

cd /root

then create a caddy config file:

nano Caddyfile

The content of the Caddyfile should have the following structure:

```
Phishing_domain_1.com {
        proxy / 127.0.0.1:8080
}
Phishing_domain_2.com {
        proxy / 127.0.0.1:8080
}
```

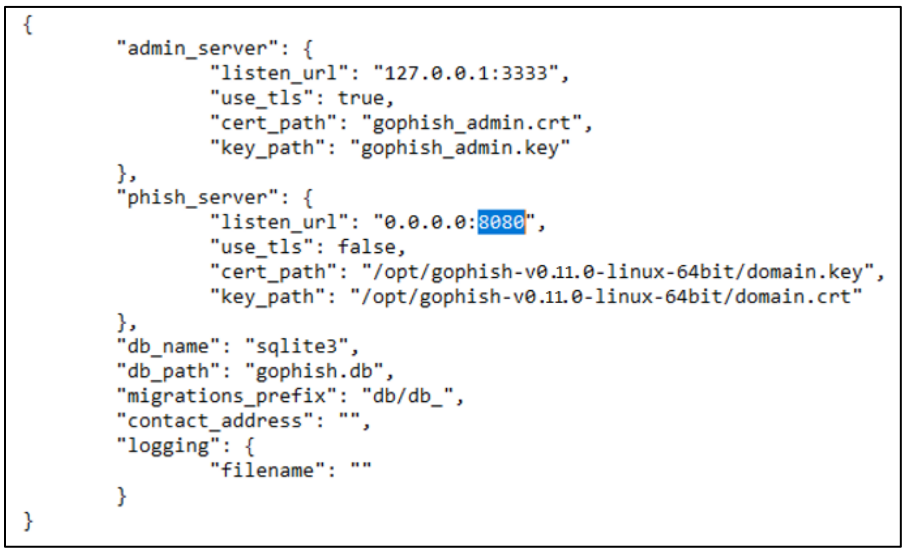
CTRL + X to exit and 'y' and 'enter' to save changes.

Note: *As you can see, we are adding to the Caddyfile all domains used for the phishing campaigns. Also, we need to make sure the domains are pointing to the AWS instance (details here DNS "A" records). Caddy will fetch the certs from Let's Encrypt and will do everything to set the reverse proxy.*

19. To point Gophish to Caddy (reverse proxy), we need to edit the Gophish config file (**config.json**):

    nano /opt/gophish-v0.11.0-linux-64bit/config.json

    And change the "listen_url" port from **80** to **8080.**



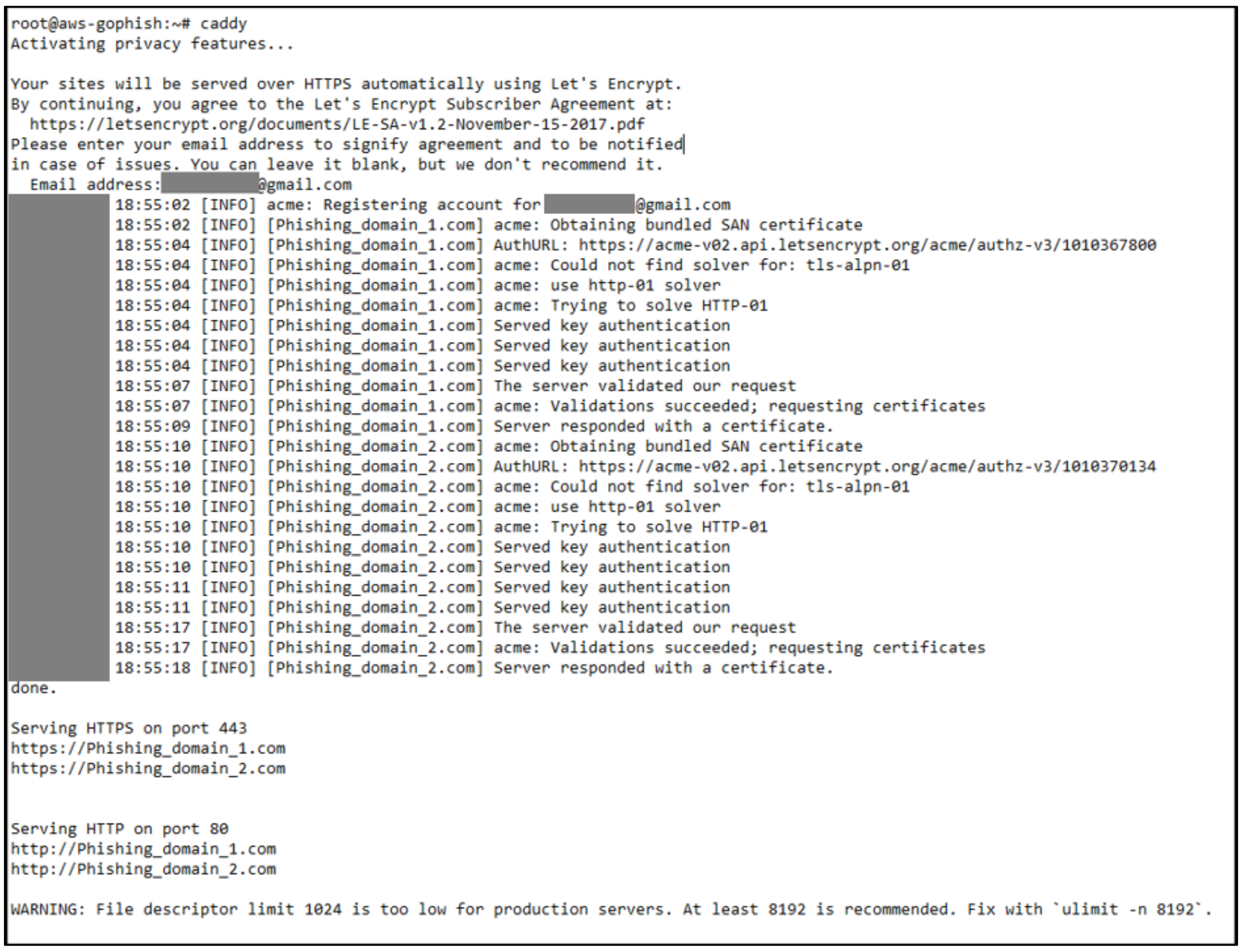
```
{
        "admin_server": {
                "listen_url": "127.0.0.1:3333",
                "use_tls": true,
                "cert_path": "gophish_admin.crt",
                "key_path": "gophish_admin.key"
        },
        "phish_server": {
                "listen_url": "0.0.0.0:8080",
                "use_tls": false,
                "cert_path": "/opt/gophish-v0.11.0-linux-64bit/domain.key",
                "key_path": "/opt/gophish-v0.11.0-linux-64bit/domain.crt"
        },
        "db_name": "sqlite3",
        "db_path": "gophish.db",
        "migrations_prefix": "db/db_",
        "contact_address": "",
        "logging": {
                "filename": ""
        }
}
```

20. Restart the Gophish service to load the changes:

    service gophish start

21. Now, we are ready to run Caddy:

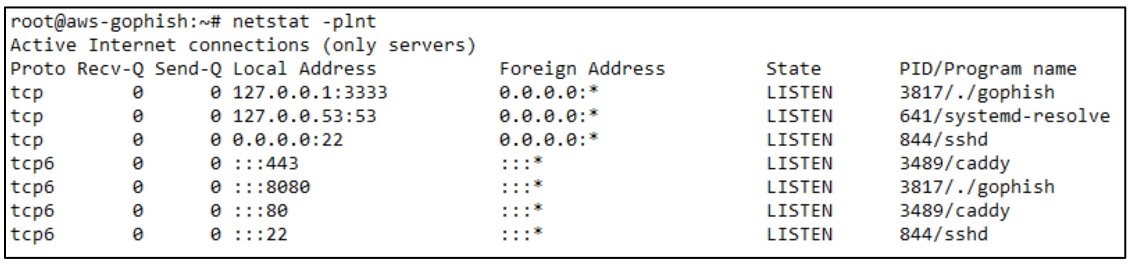
    Type ***caddy*** in the same location where you put the ***Caddyfile (e.g., /root)***



```
root@aws-gophish:~# caddy
Activating privacy features...

Your sites will be served over HTTPS automatically using Let's Encrypt.
By continuing, you agree to the Let's Encrypt Subscriber Agreement at:
  https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf
Please enter your email address to signify agreement and to be notified
in case of issues. You can leave it blank, but we don't recommend it.
  Email address: [redacted]@gmail.com
[redacted] 18:55:02 [INFO] acme: Registering account for [redacted]@gmail.com
           18:55:02 [INFO] [Phishing_domain_1.com] acme: Obtaining bundled SAN certificate
           18:55:04 [INFO] [Phishing_domain_1.com] AuthURL: https://acme-v02.api.letsencrypt.org/acme/authz-v3/1010367800
           18:55:04 [INFO] [Phishing_domain_1.com] acme: Could not find solver for: tls-alpn-01
           18:55:04 [INFO] [Phishing_domain_1.com] acme: use http-01 solver
           18:55:04 [INFO] [Phishing_domain_1.com] acme: Trying to solve HTTP-01
           18:55:04 [INFO] [Phishing_domain_1.com] Served key authentication
           18:55:04 [INFO] [Phishing_domain_1.com] Served key authentication
           18:55:04 [INFO] [Phishing_domain_1.com] Served key authentication
           18:55:07 [INFO] [Phishing_domain_1.com] The server validated our request
           18:55:07 [INFO] [Phishing_domain_1.com] acme: Validations succeeded; requesting certificates
           18:55:09 [INFO] [Phishing_domain_1.com] Server responded with a certificate.
           18:55:10 [INFO] [Phishing_domain_2.com] acme: Obtaining bundled SAN certificate
           18:55:10 [INFO] [Phishing_domain_2.com] AuthURL: https://acme-v02.api.letsencrypt.org/acme/authz-v3/1010370134
           18:55:10 [INFO] [Phishing_domain_2.com] acme: Could not find solver for: tls-alpn-01
           18:55:10 [INFO] [Phishing_domain_2.com] acme: use http-01 solver
           18:55:10 [INFO] [Phishing_domain_2.com] acme: Trying to solve HTTP-01
           18:55:10 [INFO] [Phishing_domain_2.com] Served key authentication
           18:55:10 [INFO] [Phishing_domain_2.com] Served key authentication
           18:55:11 [INFO] [Phishing_domain_2.com] Served key authentication
           18:55:11 [INFO] [Phishing_domain_2.com] Served key authentication
           18:55:17 [INFO] [Phishing_domain_2.com] The server validated our request
           18:55:17 [INFO] [Phishing_domain_2.com] acme: Validations succeeded; requesting certificates
           18:55:18 [INFO] [Phishing_domain_2.com] Server responded with a certificate.
done.

Serving HTTPS on port 443
https://Phishing_domain_1.com
https://Phishing_domain_2.com


Serving HTTP on port 80
http://Phishing_domain_1.com
http://Phishing_domain_2.com

WARNING: File descriptor limit 1024 is too low for production servers. At least 8192 is recommended. Fix with `ulimit -n 8192`.
```

22. Check that Caddy and Gophish services are listening:

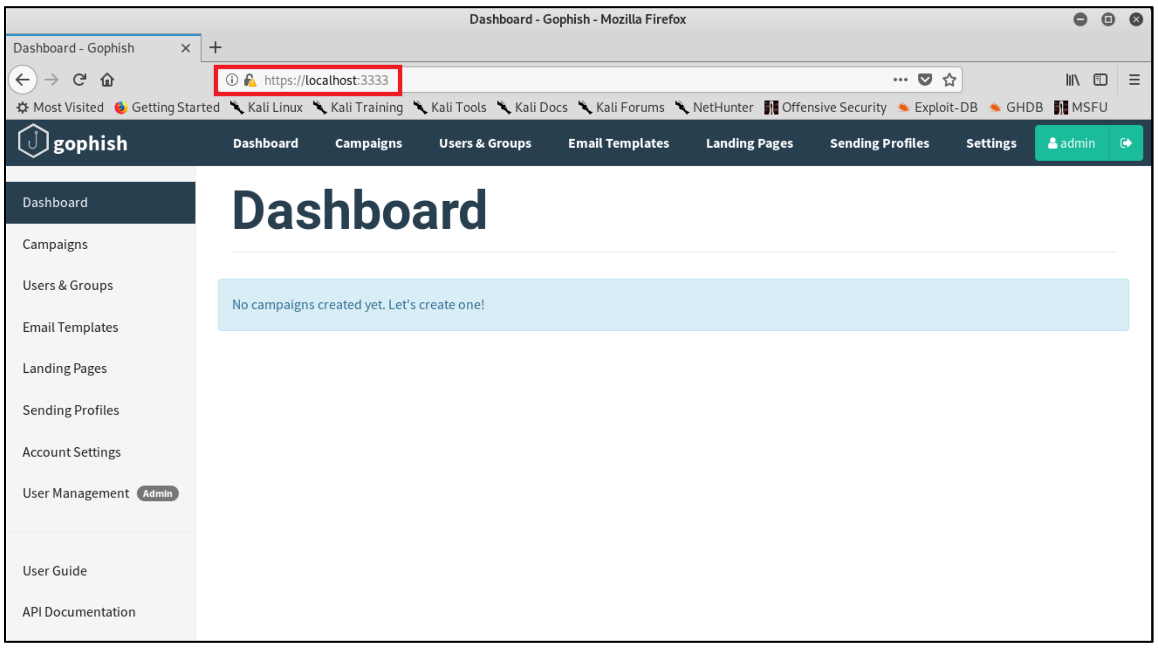
netstat -plnt



```
root@aws-gophish:~# netstat -plnt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3333          0.0.0.0:*               LISTEN      3817/./gophish
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      641/systemd-resolve
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      844/sshd
tcp6       0      0 :::443                  :::*                    LISTEN      3489/caddy
tcp6       0      0 :::8080                 :::*                    LISTEN      3817/./gophish
tcp6       0      0 :::80                   :::*                    LISTEN      3489/caddy
tcp6       0      0 :::22                   :::*                    LISTEN      844/sshd
```

23. Finally, we can access the Gophish console to run the first phishing campaign:

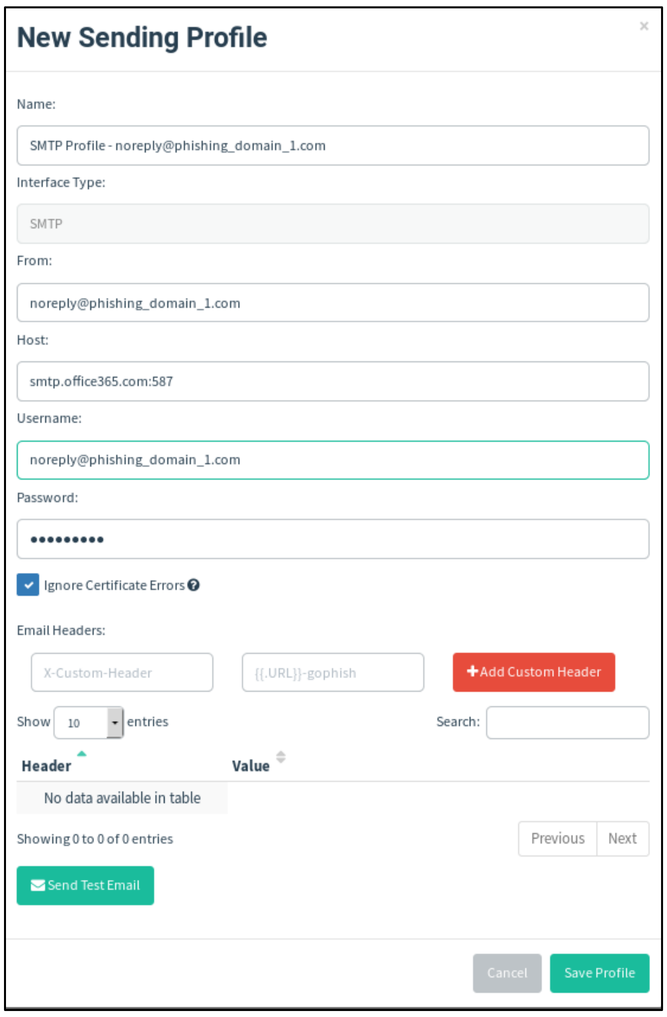
https://localhost:3333/login



Dashboard

No campaigns created yet. Let's create one!

24. Click on **Sending Profiles**, then on **+ New Profile**. Enter the SMTP information for the sender's phishing email. We could use GoDaddy or any other DNS registrar to get a custom email address.



New Sending Profile

Name: SMTP Profile - noreply@phishing_domain_1.com

Interface Type: SMTP

From: noreply@phishing_domain_1.com

Host: smtp.office365.com:587

Username: noreply@phishing_domain_1.com

Password: •••••••••

✓ Ignore Certificate Errors

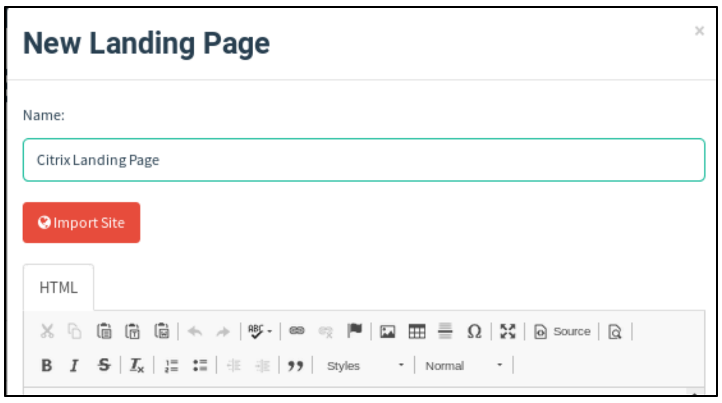
Email Headers: X-Custom-Header {{.URL}}-gophish +Add Custom Header

Show 10 entries Search:

Header	Value
No data available in table	

Showing 0 to 0 of 0 entries Previous Next

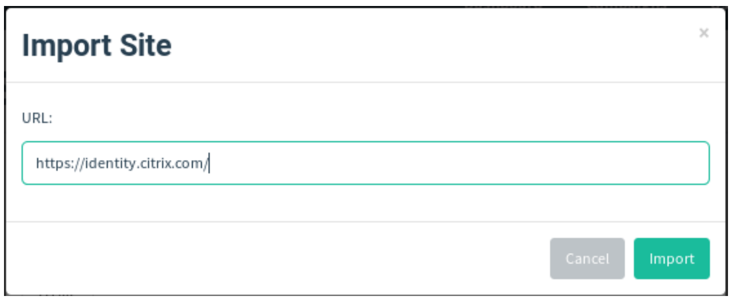
25. Next, Let's clone a landing page to use with the phishing campaign. The cloned page will help us to collect usernames and passwords. In this example, we'll be cloning a CITRIX portal (Please be aware that AWS might detect the use of intellectual property, and the phishing campaign may get flagged). Click on **Landing Pages**, then click on + **New Page,** name the landing page, and click on **Import Site**.



New Landing Page

Name: Citrix Landing Page

We'll be asked to enter the URL for the page we want to clone. Then, click on **Import** to complete the process. Next, Gophish will import the HTML code needed for the landing page. In the next step, we'll go through the HTML code to format the fields that Gophish needs for capturing the credentials.

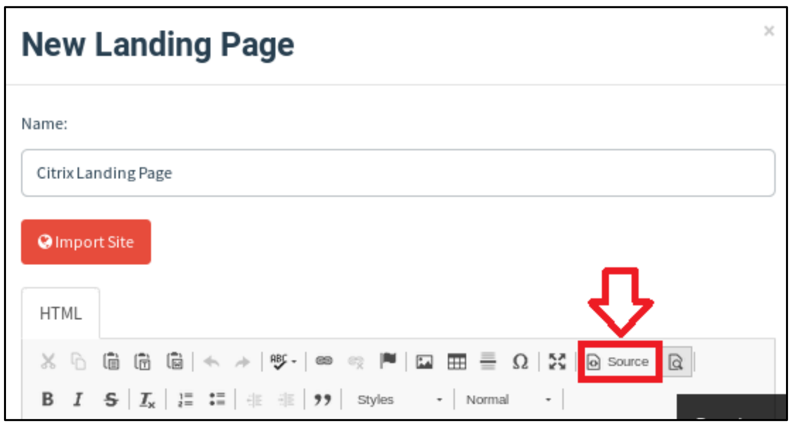

Import Site

URL:

https://identity.citrix.com/

Cancel Import

26. Click on **Source** and look for the HTML tag “<form”.


New Landing Page

Name:

Citrix Landing Page

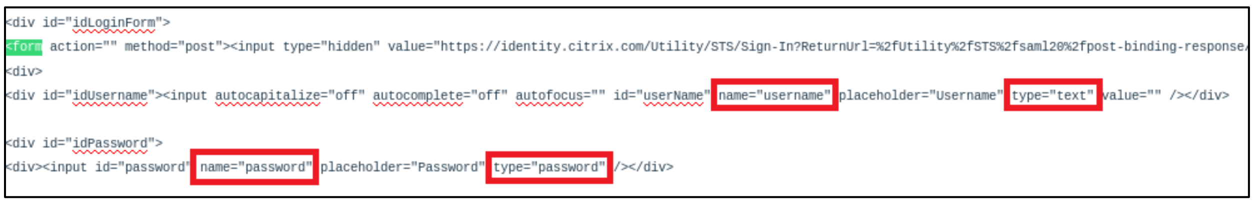
Import Site

HTML

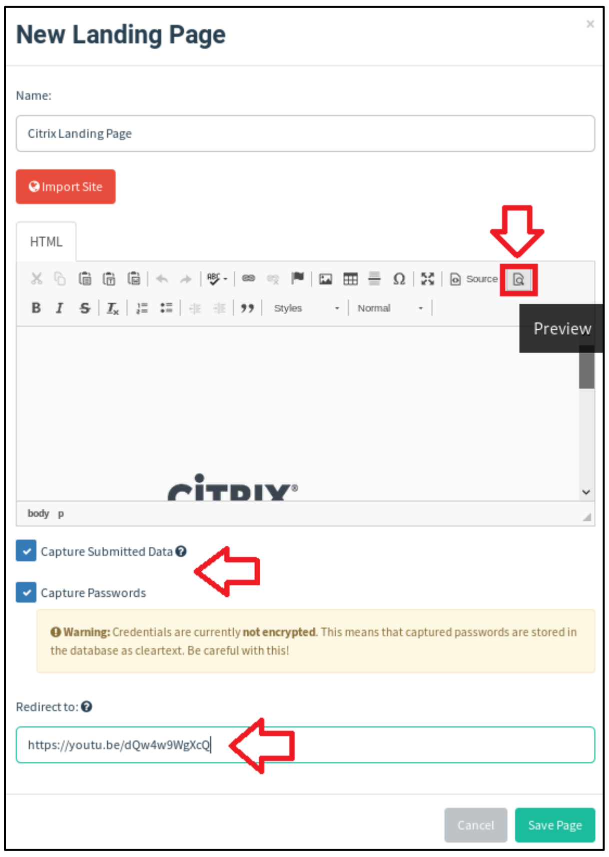
Make sure the username and password input fields are defined as follows:

```
<form action="" method="post" name="form">
<label>Username:</label>
<input name="username" type="text"/>
<label>Password:</label>
<input name="password" type="password"/>
</form>
```

Here is the HTML code for our cloned Citrix page:


```
<div id="idLoginForm">
<form action="" method="post"><input type="hidden" value="https://identity.citrix.com/Utility/STS/Sign-In?ReturnUrl=%2fUtility%2fSTS%2fsaml20%2fpost-binding-response
<div>
<div id="idUsername"><input autocapitalize="off" autocomplete="off" autofocus="" id="userName" name="username" placeholder="Username" type="text" value="" /></div>

<div id="idPassword">
<div><input id="password" name="password" placeholder="Password" type="password" /></div>
```

We might want to verify how the cloned page is rendered in the browser, so click on the **preview** icon. Before saving the page, at the bottom of the window, we select the **Capture Submitted Data** and **Capture Passwords** options. If we want to redirect users to another site after entering credentials, we could use the “**Redirect to**” field. Finally, we click on **Save Page**.



New Landing Page

Name:

Citrix Landing Page

Capture Submitted Data ✓

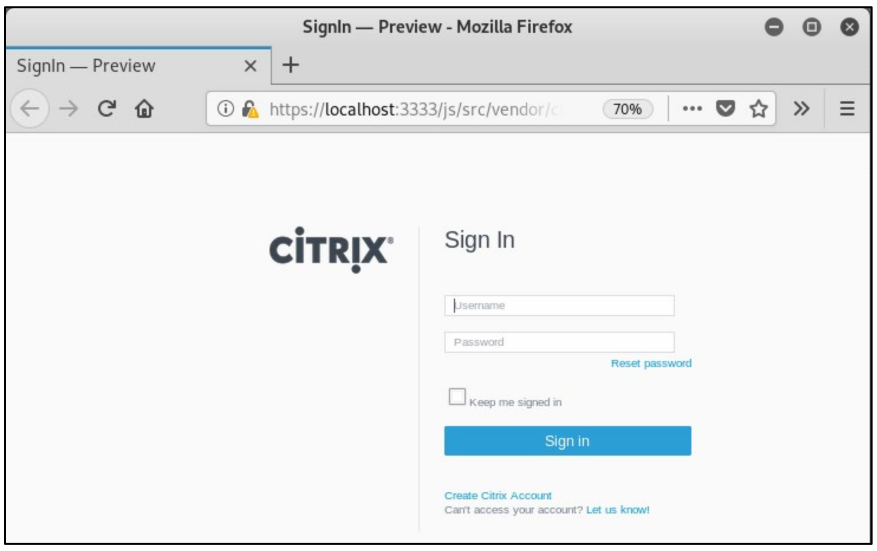
Capture Passwords ✓

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://youtu.be/dQw4w9WgXcQ

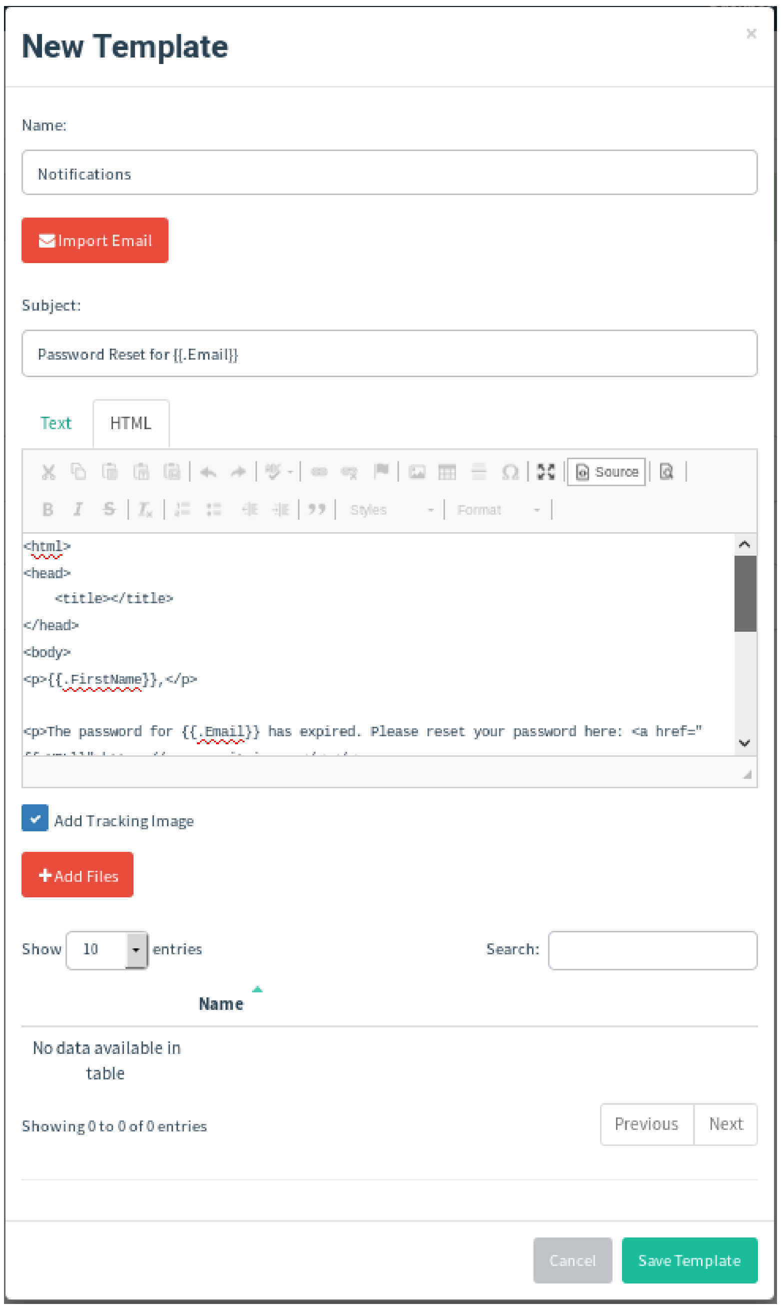
Preview – cloned Citrix portal:



SignIn — Preview - Mozilla Firefox

https://localhost:3333/js/src/vendor/c 70%

27. We are ready to create the first email template. Click on **Email Templates,** then give it a name. Fill out the Subject and the verbiage for the phishing email using the HTML tab. Click on **Save Template** when done.



New Template

Name: Notifications

Import Email

Subject: Password Reset for {{.Email}}

Text | HTML

```
<html>
<head>
    <title></title>
</head>
<body>
<p>{{.FirstName}},</p>

<p>The password for {{.Email}} has expired. Please reset your password here: <a href="
```

[✓] Add Tracking Image

Add Files

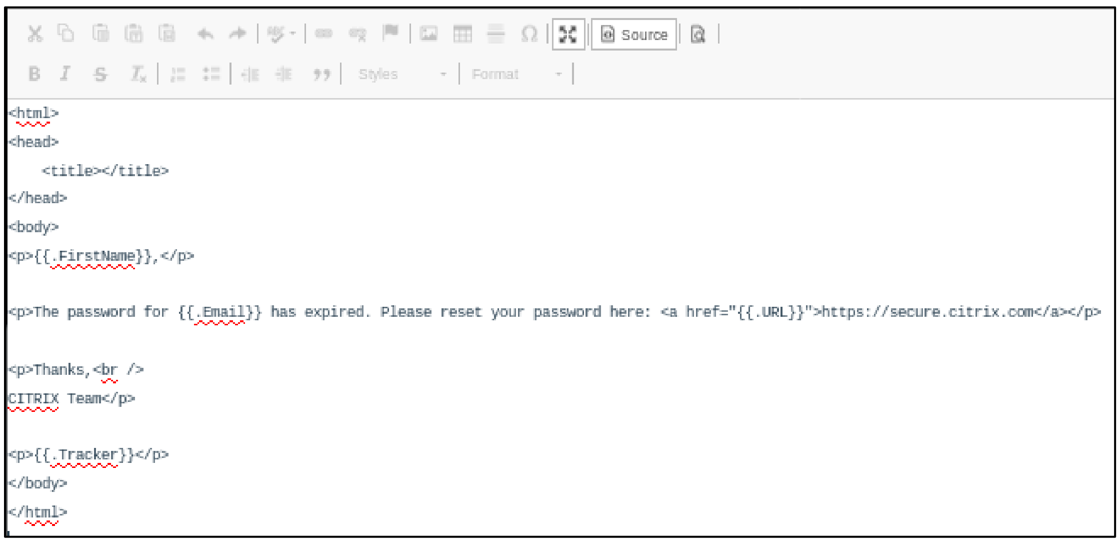
Show 10 entries   Search:

Name
No data available in table

Showing 0 to 0 of 0 entries   Previous Next

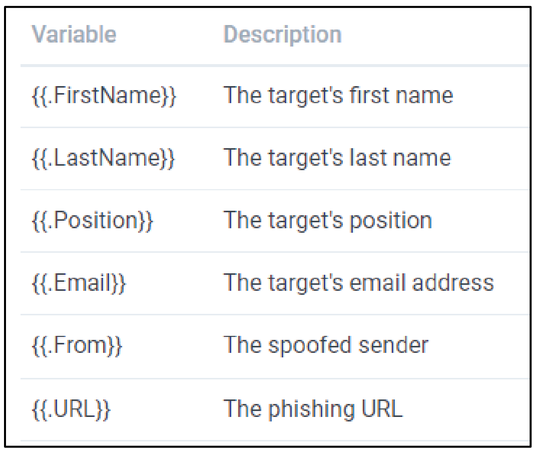
Cancel   Save Template

Here is an example of a HTML phishing email:



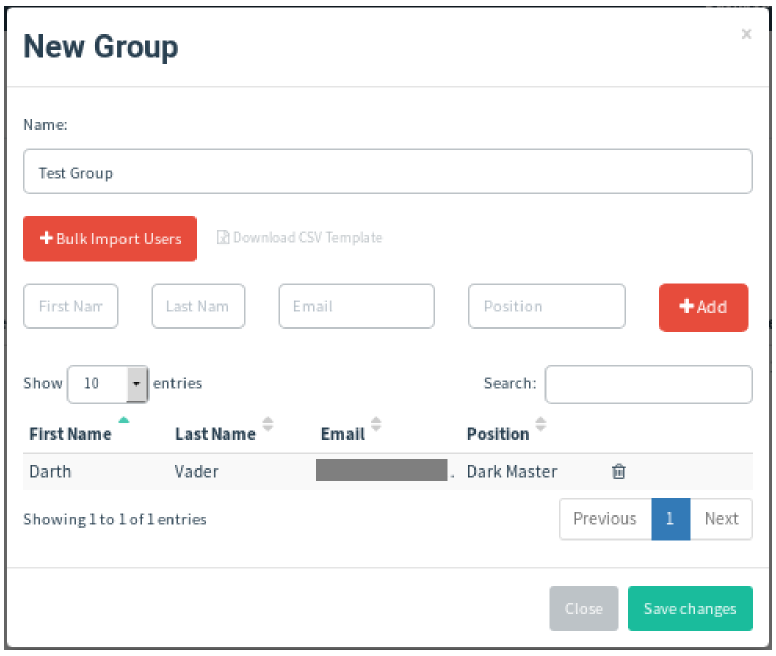
```
<html>
<head>
    <title></title>
</head>
<body>
<p>{{.FirstName}},</p>

<p>The password for {{.Email}} has expired. Please reset your password here: <a href="{{.URL}}">https://secure.citrix.com</a></p>

<p>Thanks,<br />
CITRIX Team</p>

<p>{{.Tracker}}</p>
</body>
</html>
```

We can use the following HTML variables within the phishing email:



Variable	Description
{{.FirstName}}	The target's first name
{{.LastName}}	The target's last name
{{.Position}}	The target's position
{{.Email}}	The target's email address
{{.From}}	The spoofed sender
{{.URL}}	The phishing URL

28. Before putting all the settings together, we need to create a Users' group; This is the list of users to be phished. Click on **Users & Groups**, then on + **New Group.** Name the group and add each user to the group: <First Name>, <Last Name>, <Email>, and <Position (job title)>. Finally, click on **Save Changes.**



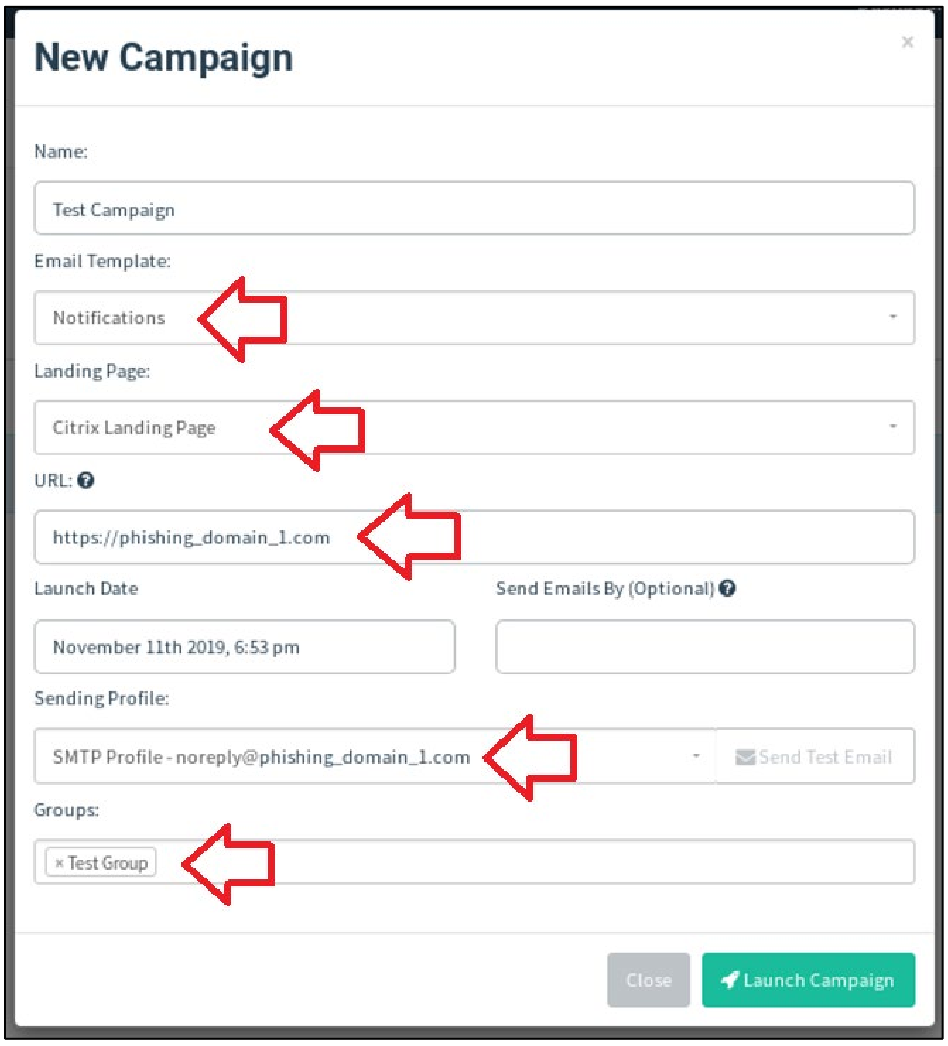
New Group

Name: Test Group

First Name	Last Name	Email	Position
Darth	Vader	[redacted]	Dark Master

Showing 1 to 1 of 1 entries

29. Lastly, Click on **Campaigns**, then on **+ New Campaign**. Name the campaign and select all the corresponding options previously set.



New Campaign

Name: Test Campaign

Email Template: Notifications

Landing Page: Citrix Landing Page

URL: https://phishing_domain_1.com

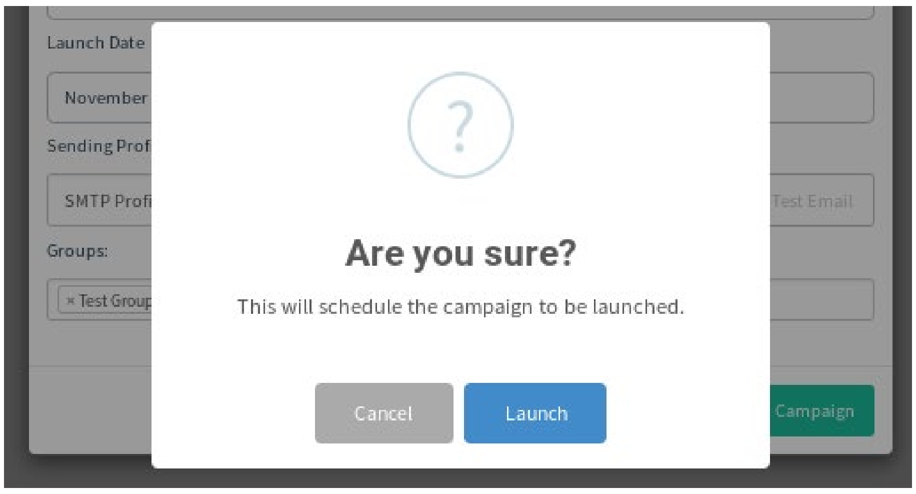
Launch Date: November 11th 2019, 6:53 pm

Send Emails By (Optional):

Sending Profile: SMTP Profile - noreply@phishing_domain_1.com

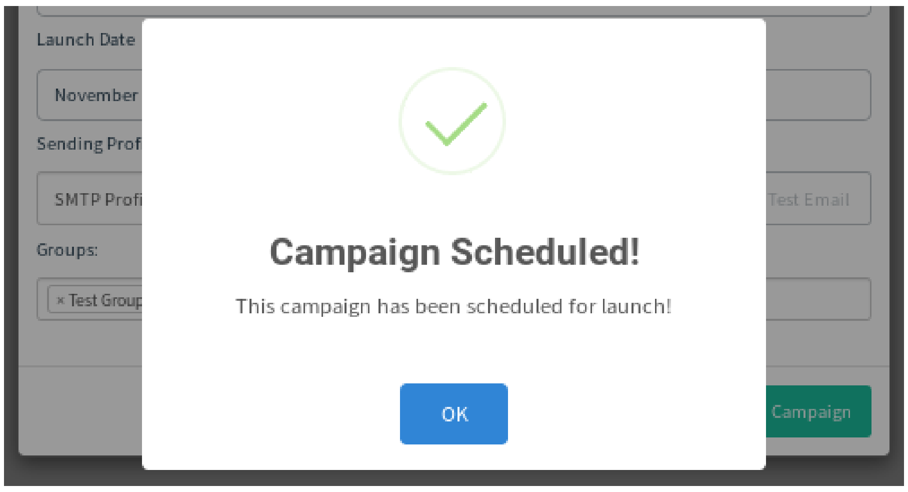
Groups: × Test Group

When ready to send out the phishing campaign, click on **Launch Campaign.** We'll be asked to confirm the action.



Are you sure?

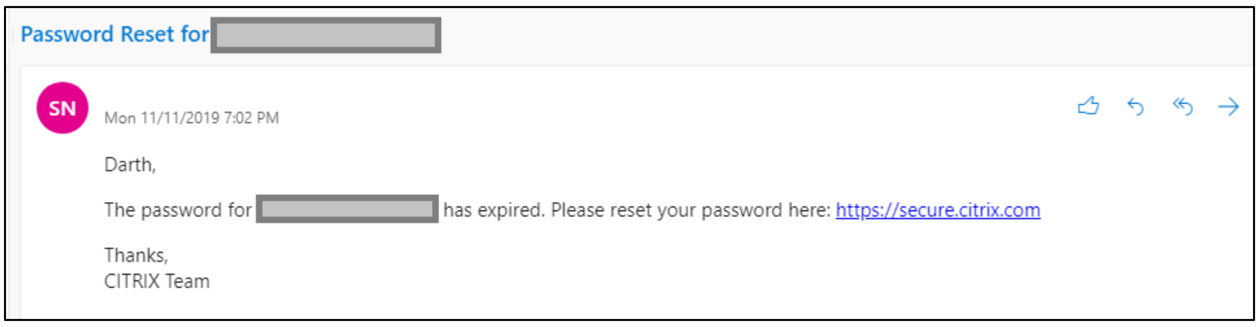
This will schedule the campaign to be launched.



Campaign Scheduled!

This campaign has been scheduled for launch!

30. Once the campaign is sent, the targeted users will receive the email below.



Password Reset for [redacted]

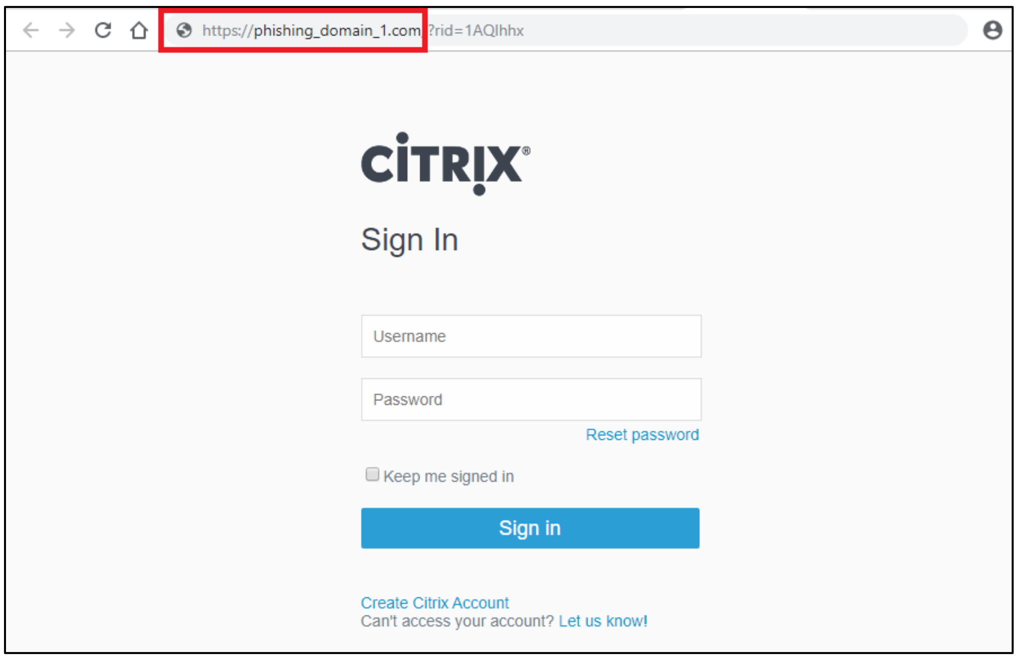
SN Mon 11/11/2019 7:02 PM

Darth,

The password for [redacted] has expired. Please reset your password here: https://secure.citrix.com

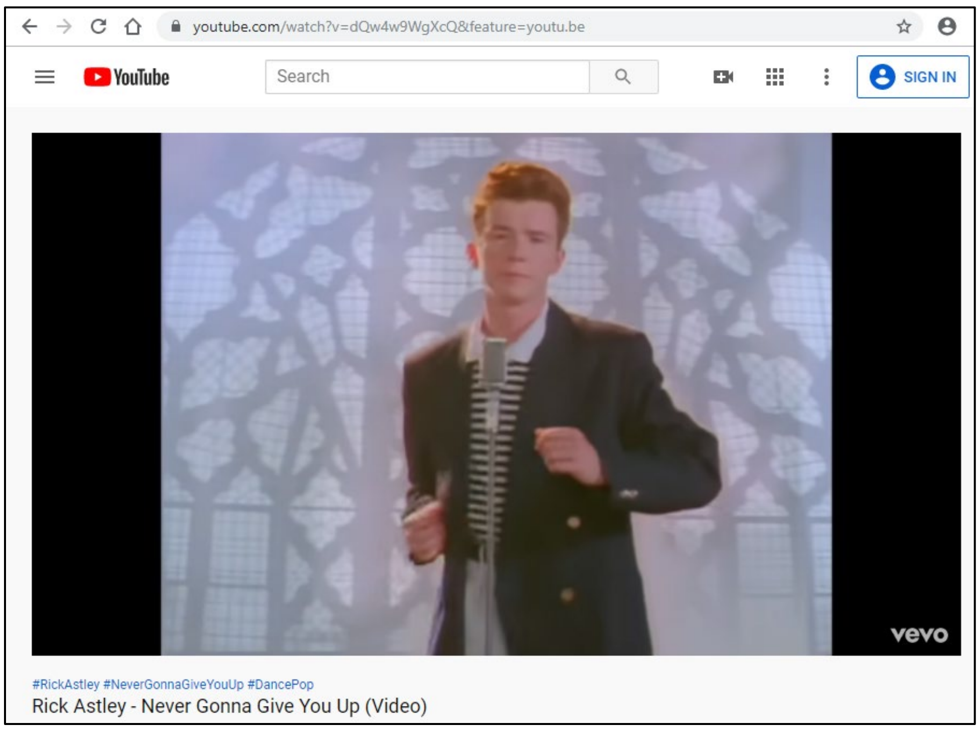
Thanks,
CITRIX Team

Upon clicking on the “https://secure.citrix.com” link, the users will be pointed to the landing page.



https://phishing_domain_1.com?rid=1AQlhhx

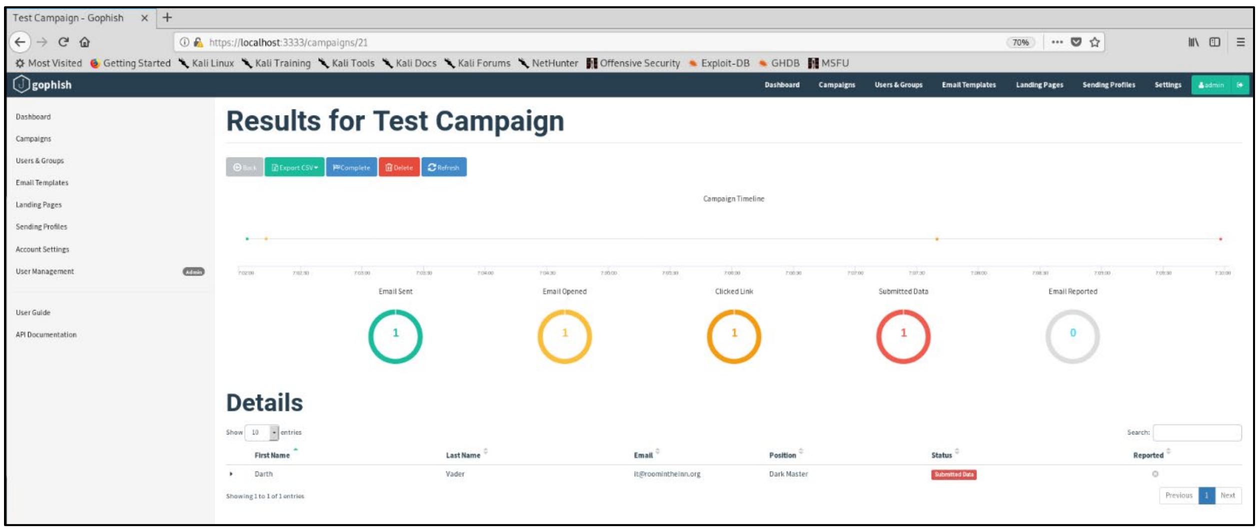
After entering credentials, users are then redirected to the URL set on step 26.



youtube.com/watch?v=dQw4w9WgXcQ&feature=youtu.be

YouTube
Search
SIGN IN
vevo
#RickAstley #NeverGonnaGiveYouUp #DancePop
Rick Astley - Never Gonna Give You Up (Video)

All Testing results will be displayed on the Dashboard:



Test Campaign - Gophish

https://localhost:3333/campaigns/21

gophish

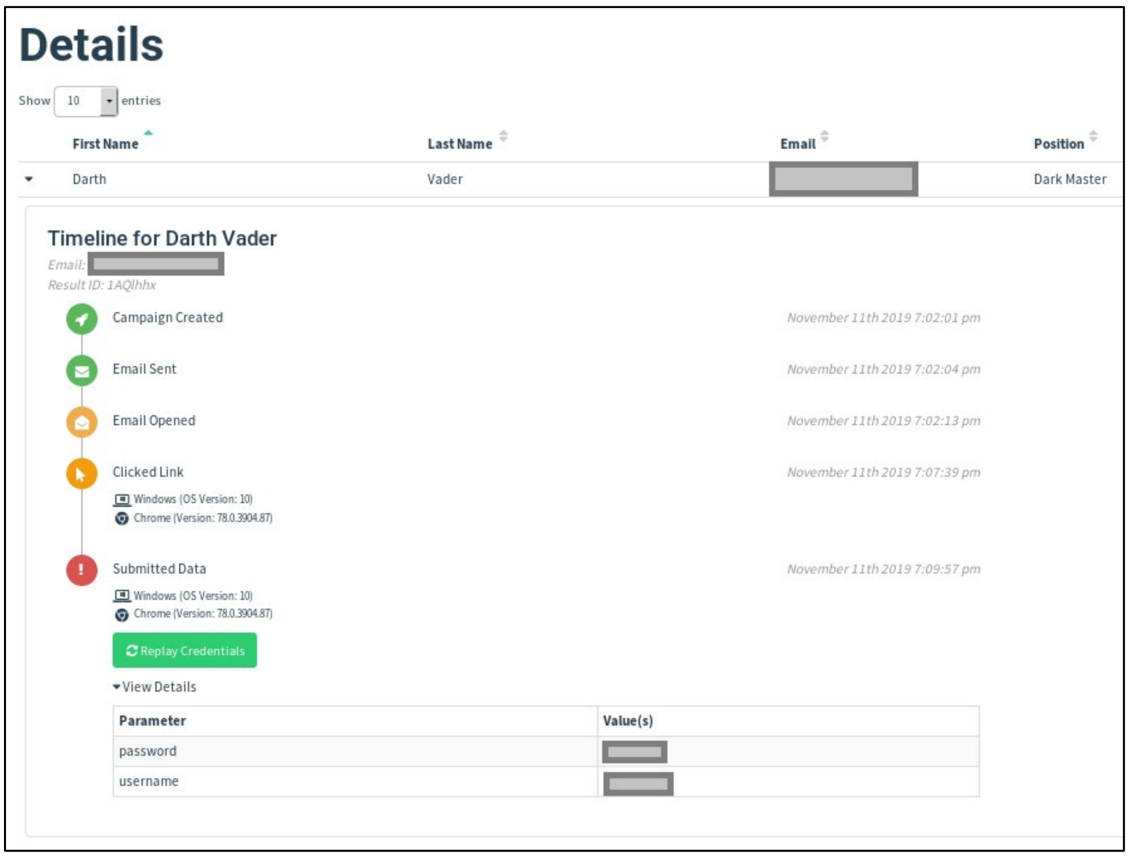
Results for Test Campaign

Details

Show 10 entries

First Name	Last Name	Email	Position	Status	Reported
Darth	Vader	it@roomintheinn.org	Dark Master	Submitted Data	

Showing 1 to 1 of 1 entries



Details

Show 10 entries

First Name	Last Name	Email	Position
Darth	Vader	[redacted]	Dark Master

Timeline for Darth Vader

Email: [redacted]
Result ID: 1AQlhhx

- Campaign Created — *November 11th 2019 7:02:01 pm*
- Email Sent — *November 11th 2019 7:02:04 pm*
- Email Opened — *November 11th 2019 7:02:13 pm*
- Clicked Link — *November 11th 2019 7:07:39 pm*
  - Windows (OS Version: 10)
  - Chrome (Version: 78.0.3904.87)
- Submitted Data — *November 11th 2019 7:09:57 pm*
  - Windows (OS Version: 10)
  - Chrome (Version: 78.0.3904.87)

Parameter	Value(s)
password	[redacted]
username	[redacted]

In this example, we have set up an affordable phishing platform capable of managing multiple phishing domains. We also demonstrated how simple it is to clone portals and set email templates for our phishing campaigns. For administrators who want to expand on other Gophish capabilities, such as API calls for phishing campaigns automatization, we encourage you to visit the official documentation and GitHub repositories.

References:

1. *https://getgophish.com/documentation/*
2. *https://github.com/gophish/gophish/releases/tag/v0.11.0*
3. *https://caddyserver.com/*

Jorge Jaque

Lead Security Consultant

jjaque@lbmc.com

LBMC Information Security