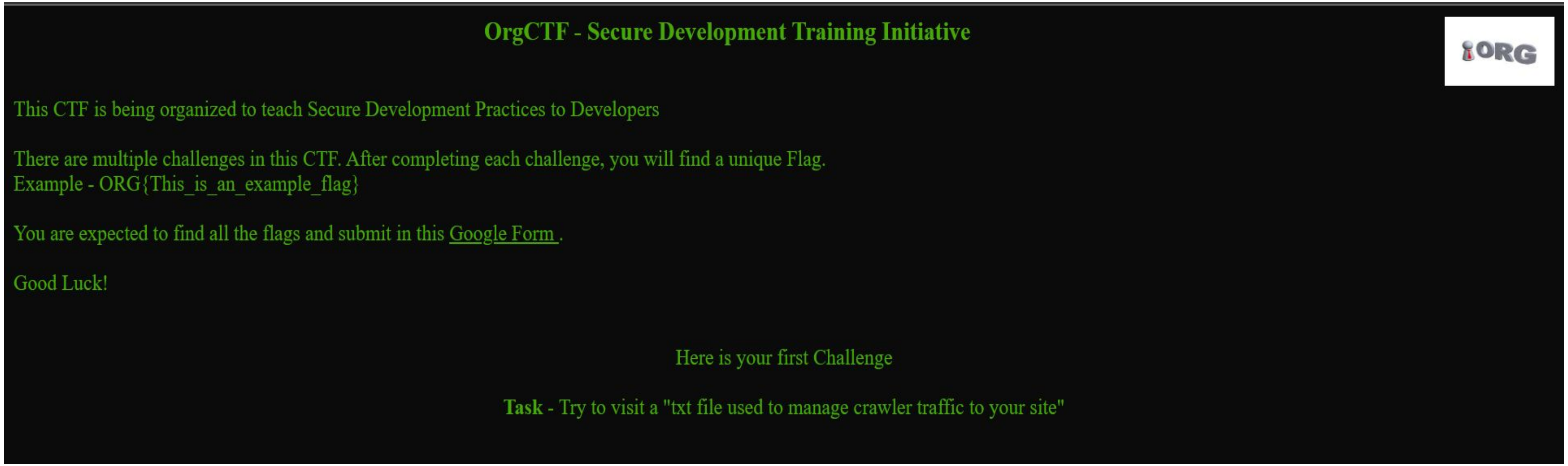
Secure Application Development CTF

Walkthrough

Overview

- The CTF is accessible at http://orgctf.onrender.com/
- There are 14 Unique flags that are to be collected based on 14 different vulnerabilities.
- The flags are to be submitted in this Google form
- The Flag begins with the word "ORG{".
    - Example - ORG{This_is_an_example_flag}
    - Note - You need to submit the complete flag - ORG{This_is_an_example_flag}

Challenge 1


OrgCTF - Secure Development Training Initiative

This CTF is being organized to teach Secure Development Practices to Developers

There are multiple challenges in this CTF. After completing each challenge, you will find a unique Flag.
Example - ORG{This_is_an_example_flag}

You are expected to find all the flags and submit in this Google Form .

Good Luck!

Here is your first Challenge

Task - Try to visit a "txt file used to manage crawler traffic to your site"

URL - http://orgctf.onrender.com/

Description - Try to visit a "txt file used to manage crawler traffic to your site"

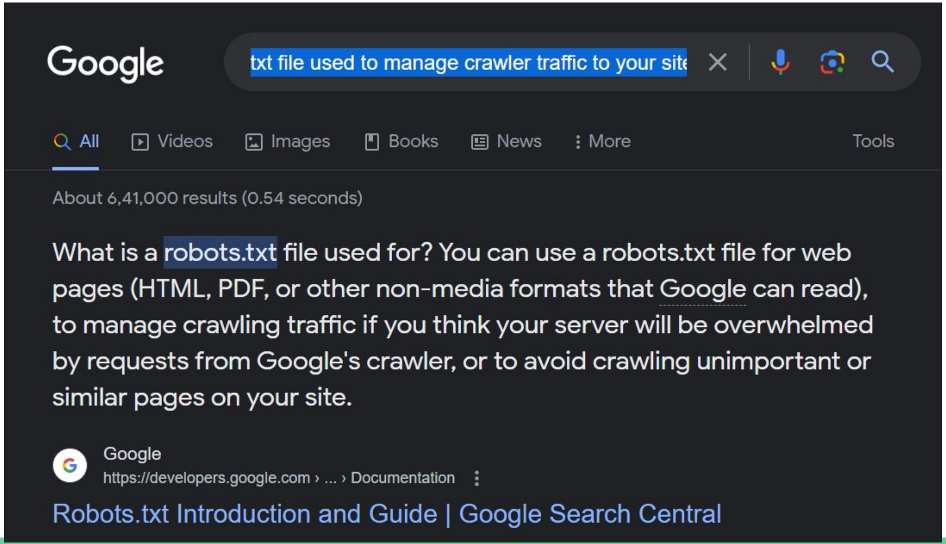
Hint - Click Here

Solution

> The challenge expects you to do a simple google search “txt file used to manage crawler traffic to your site”

> On doing this, one will get to know about a file, robots.txt

> This file is used to manage crawling traffic. Website owners can allow/disallow bots from crawling specific URLs



About 6,41,000 results (0.54 seconds)

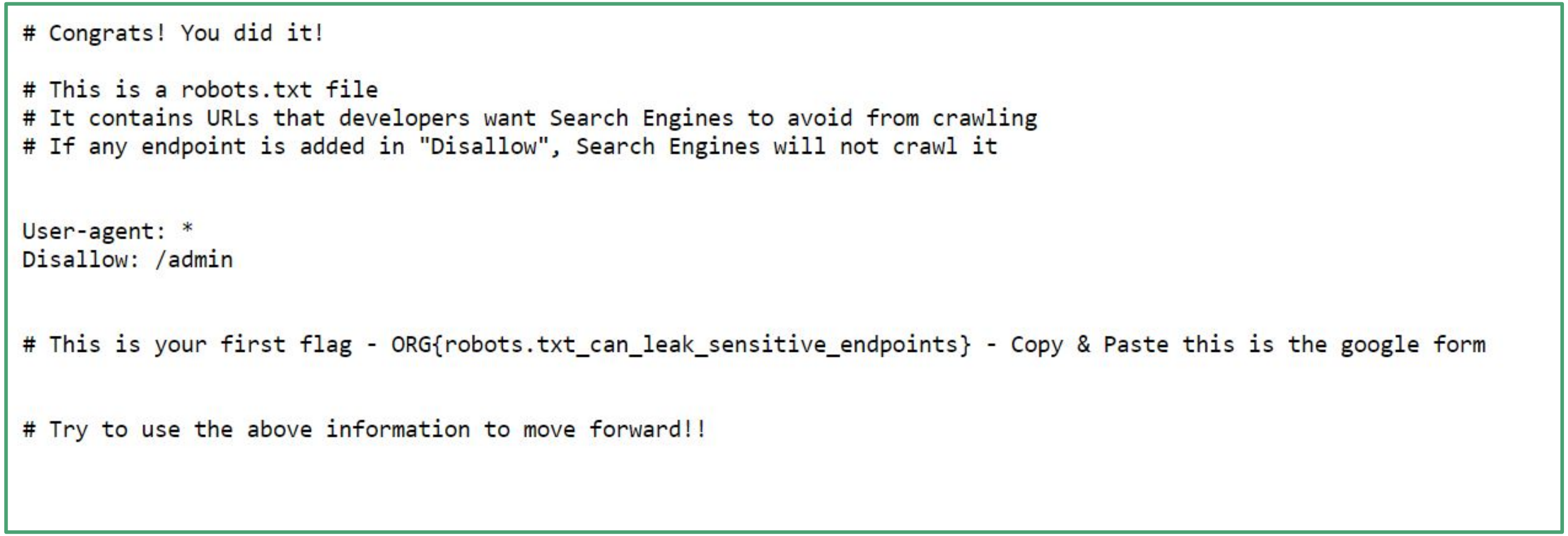
What is a robots.txt file used for? You can use a robots.txt file for web pages (HTML, PDF, or other non-media formats that Google can read), to manage crawling traffic if you think your server will be overwhelmed by requests from Google's crawler, or to avoid crawling unimportant or similar pages on your site.

Google
https://developers.google.com › ... › Documentation

Robots.txt Introduction and Guide | Google Search Central

Solution

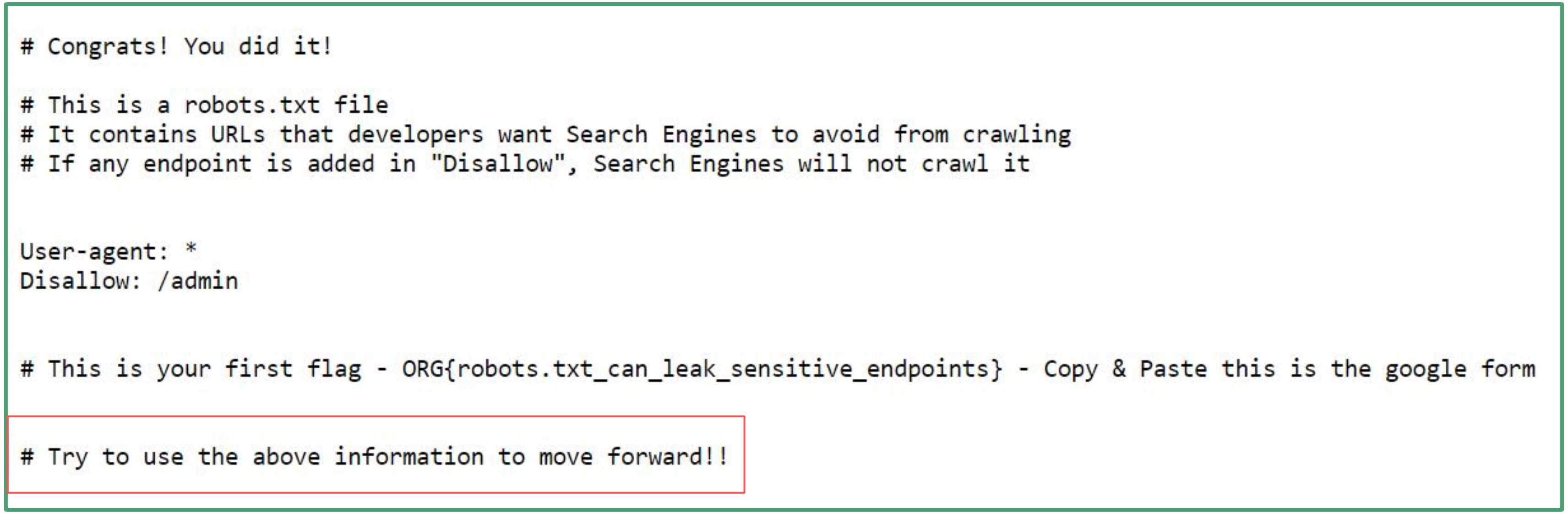
> Once, you get to know of this file, try to access the same on our CTF site - http://orgctf.onrender.com/robots.txt



```
# Congrats! You did it!

# This is a robots.txt file
# It contains URLs that developers want Search Engines to avoid from crawling
# If any endpoint is added in "Disallow", Search Engines will not crawl it


User-agent: *
Disallow: /admin


# This is your first flag - ORG{robots.txt_can_leak_sensitive_endpoints} - Copy & Paste this is the google form


# Try to use the above information to move forward!!
```

> Here, you get your first flag - ORG{robots.txt_can_leak_sensitive_endpoints}

Challenge 2



```
# Congrats! You did it!

# This is a robots.txt file
# It contains URLs that developers want Search Engines to avoid from crawling
# If any endpoint is added in "Disallow", Search Engines will not crawl it


User-agent: *
Disallow: /admin


# This is your first flag - ORG{robots.txt_can_leak_sensitive_endpoints} - Copy & Paste this is the google form

# Try to use the above information to move forward!!
```

URL - http://orgctf.onrender.com/robots.txt

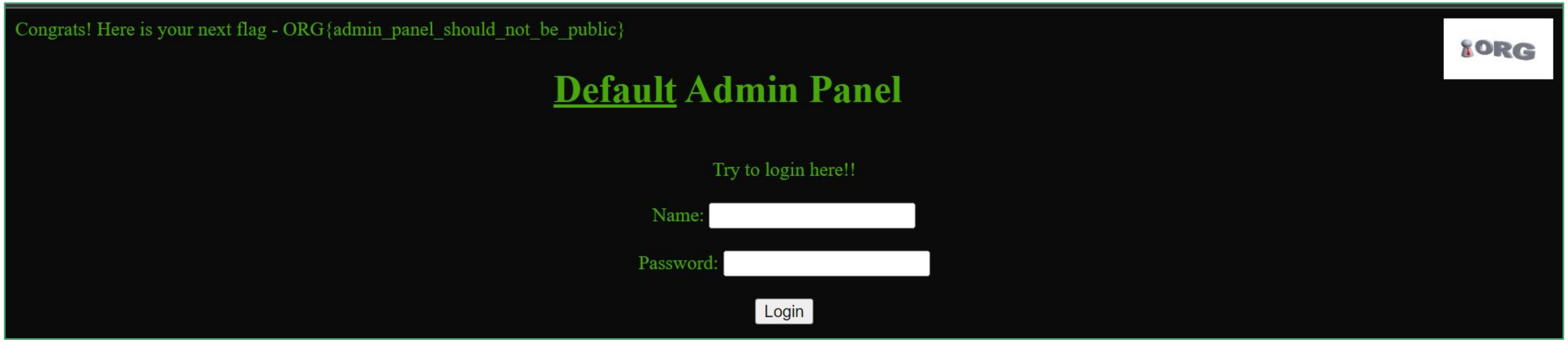
Description - Try to use the above information to move forward!!

Solution

> This challenge is an extension of the first one. It expects you to use the information revealed in challenge 1

> In the exposed robots.txt file, you get to know there is a “/admin” path on the webserver that is disallowed for all User agents.

> Once, you get to know of this path, try to access the same on our CTF site - http://orgctf.onrender.com/admin



Congrats! Here is your next flag - ORG{admin_panel_should_not_be_public}

Default Admin Panel

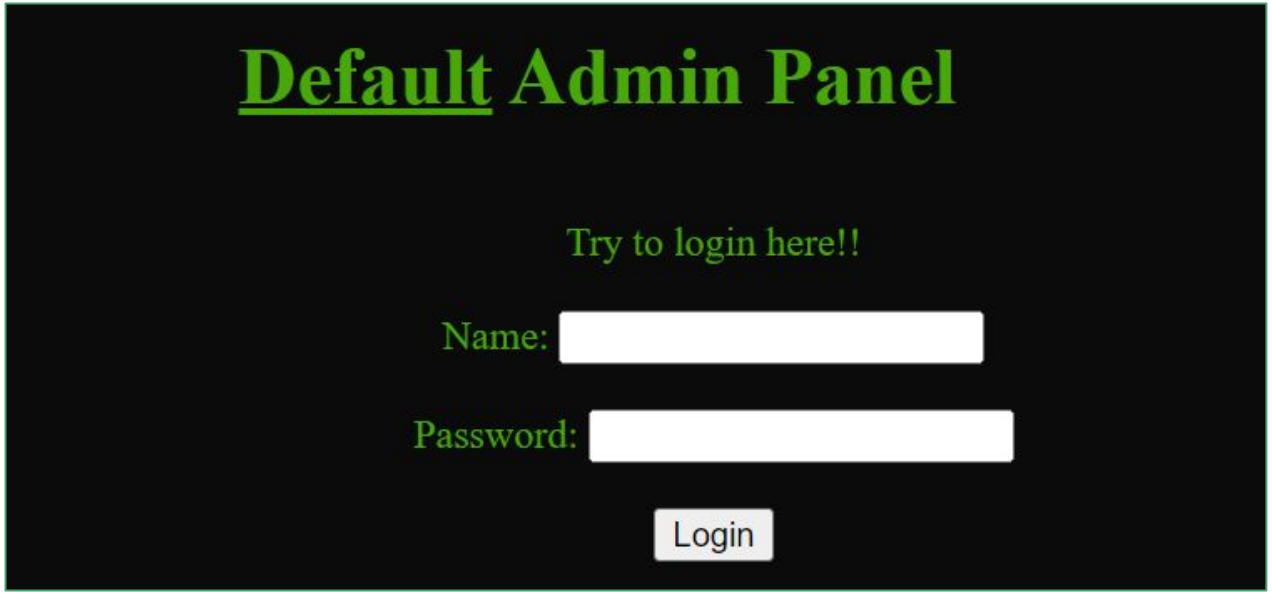
Try to login here!!

Name:

Password:

> Here, you get your second flag - ORG{admin_panel_should_not_be_public}

Challenge 3



URL - http://orgctf.onrender.com/admin

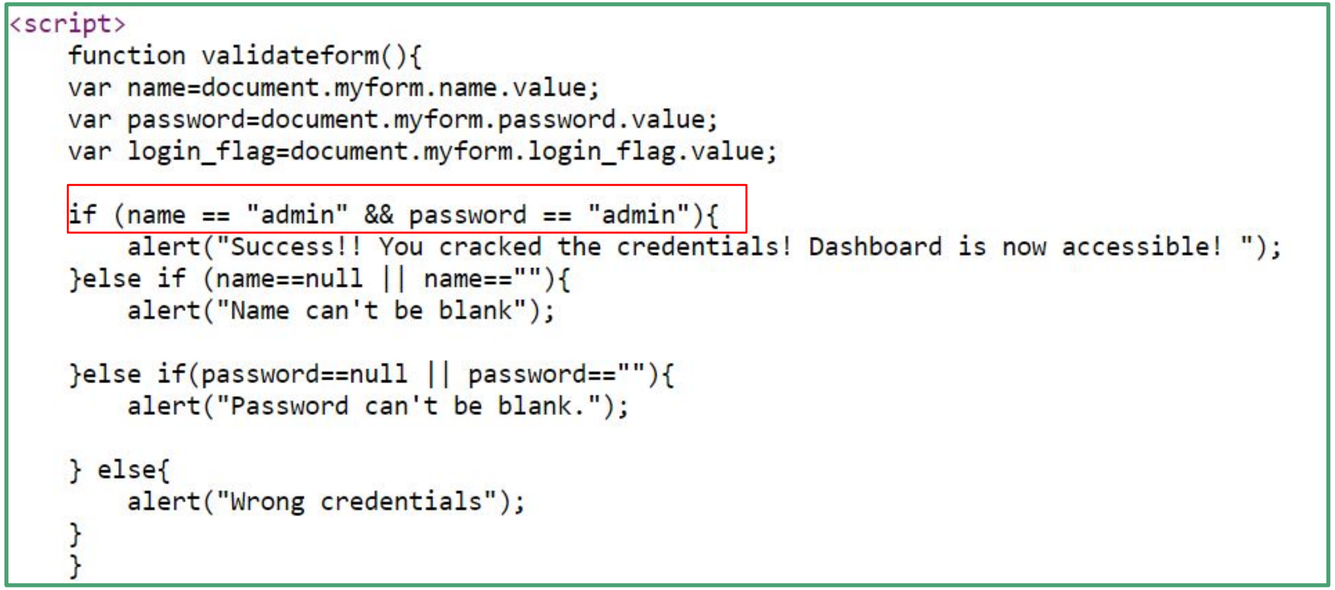
Description - Try to login here!!

Hint - A web page runs on it source code!! *Cntrl+U*

Solution

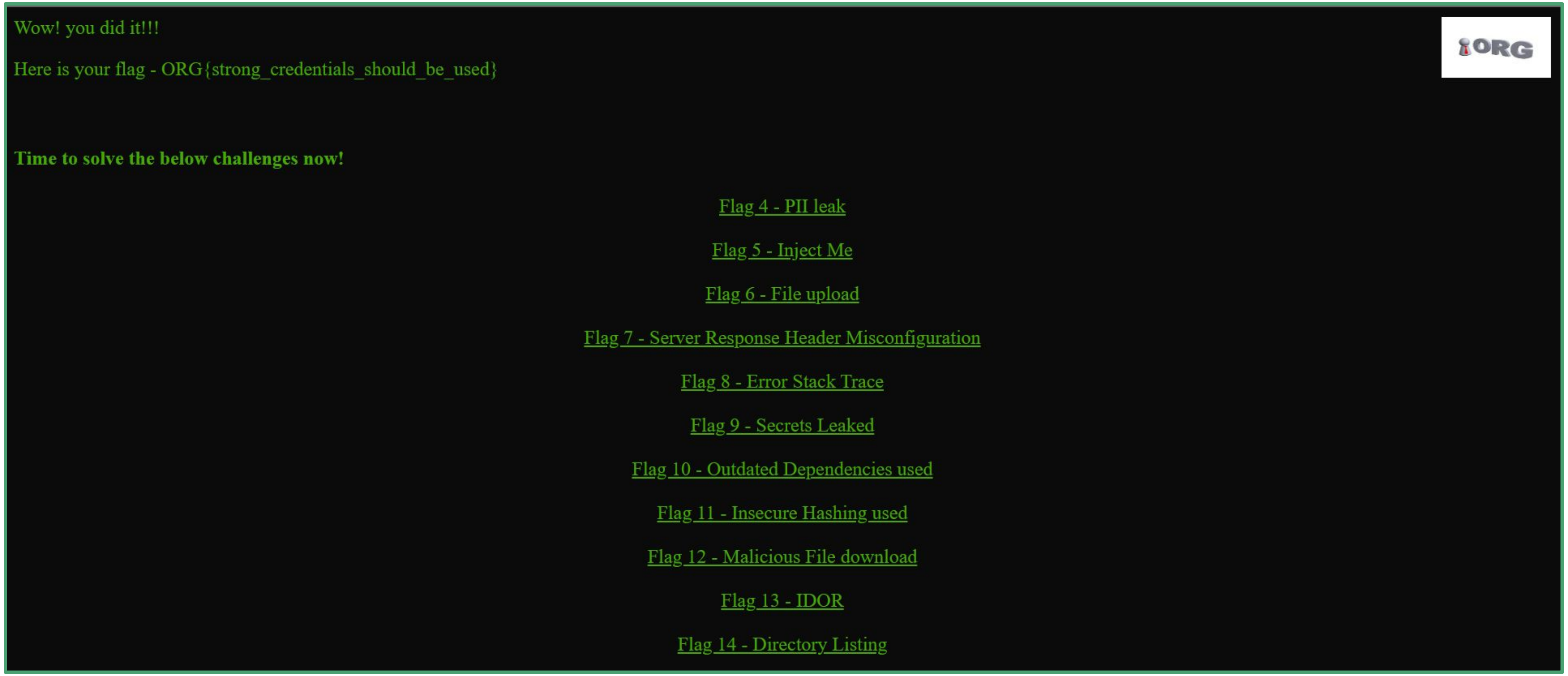
> This challenge expects you to login to the portal by guessing/revealing the credentials.

> On reading the hint, you get to know you need to view the source code of the webpage. There, it is clearly visible that the credentials are admin:admin



```
<script>
    function validateform(){
    var name=document.myform.name.value;
    var password=document.myform.password.value;
    var login_flag=document.myform.login_flag.value;

    if (name == "admin" && password == "admin"){
        alert("Success!! You cracked the credentials! Dashboard is now accessible! ");
    }else if (name==null || name==""){
        alert("Name can't be blank");

    }else if(password==null || password==""){
        alert("Password can't be blank.");

    } else{
        alert("Wrong credentials");
    }
    }
```

> Login using the found credentials to get your third flag - ORG{strong_credentials_should_be_used}



Wow! you did it!!!

Here is your flag - ORG{strong_credentials_should_be_used}

Time to solve the below challenges now!

Flag 4 - PII leak

Flag 5 - Inject Me

Flag 6 - File upload

Flag 7 - Server Response Header Misconfiguration

Flag 8 - Error Stack Trace

Flag 9 - Secrets Leaked

Flag 10 - Outdated Dependencies used

Flag 11 - Insecure Hashing used

Flag 12 - Malicious File download

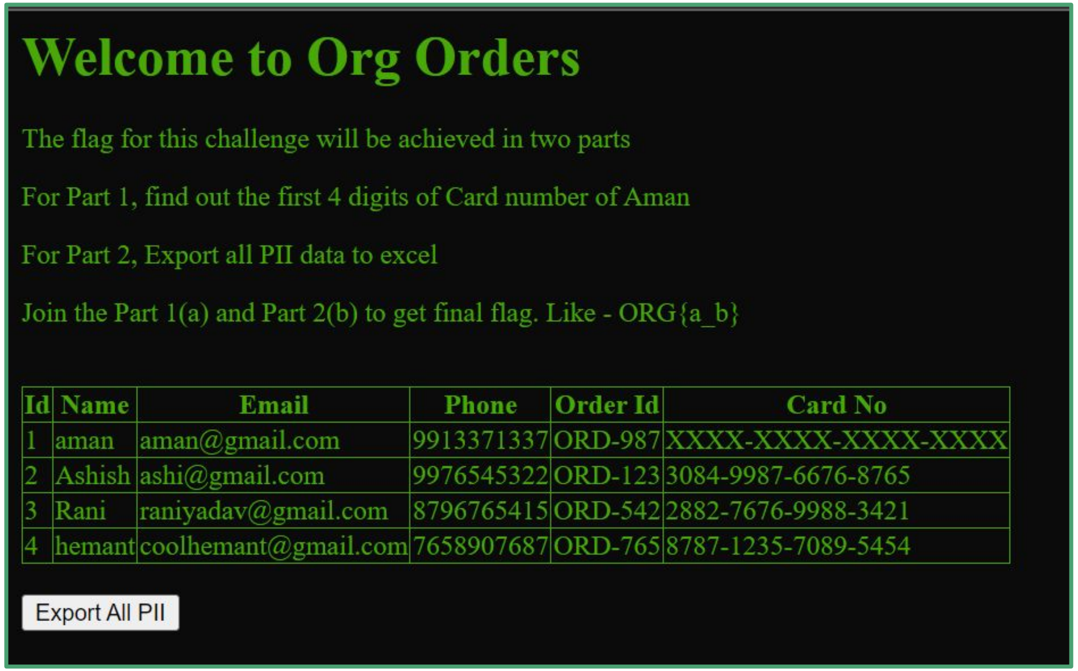
Flag 13 - IDOR

Flag 14 - Directory Listing

Now, all the challenges are individual and can be solved in any order

URL - http://orgctf.onrender.com/dashboard

Challenge 4 - PII Leak



Welcome to Org Orders

The flag for this challenge will be achieved in two parts

For Part 1, find out the first 4 digits of Card number of Aman

For Part 2, Export all PII data to excel

Join the Part 1(a) and Part 2(b) to get final flag. Like - ORG{a_b}

Id	Name	Email	Phone	Order Id	Card No
1	aman	aman@gmail.com	9913371337	ORD-987	XXXX-XXXX-XXXX-XXXX
2	Ashish	ashi@gmail.com	9976545322	ORD-123	3084-9987-6676-8765
3	Rani	raniyadav@gmail.com	8796765415	ORD-542	2882-7676-9988-3421
4	hemant	coolhemant@gmail.com	7658907687	ORD-765	8787-1235-7089-5454

URL - http://orgctf.onrender.com/orders

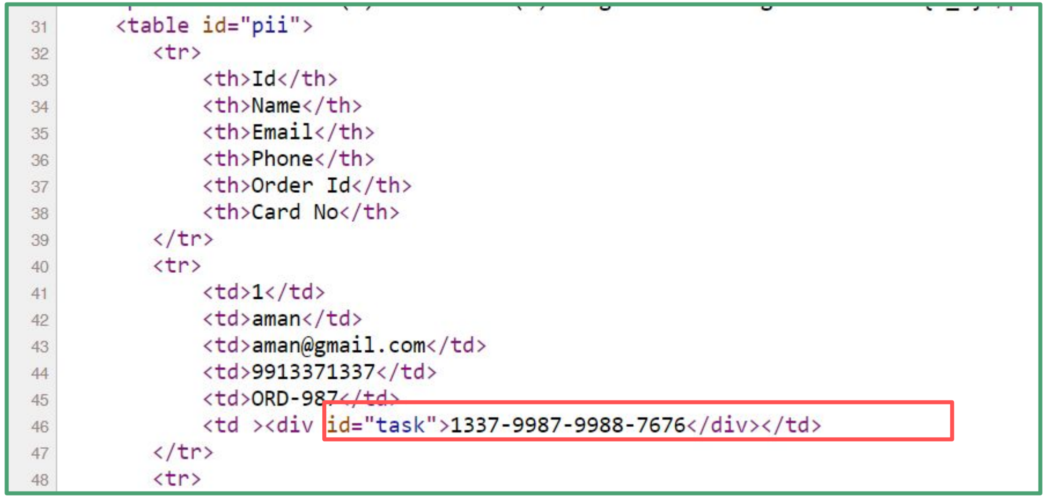
Description - The flag for this challenge will be achieved in two parts
For Part 1, find out the first 4 digits of Card number of Aman
For Part 2, Export all PII data to excel
Join the Part 1(a) and Part 2(b) to get final flag. Like - ORG{a_b}

Solution

> This challenge is to be completed in 2 parts. For Part1, we have to find out the first 4 digits of Card number of Aman.

> On viewing the table, we see card number of Aman is masked. We need to see the unmasked data.

> We view the source code of the page, to find out the card masking is being done by JS on frontend. Hence, card number can be seen in unmasked form in the source.



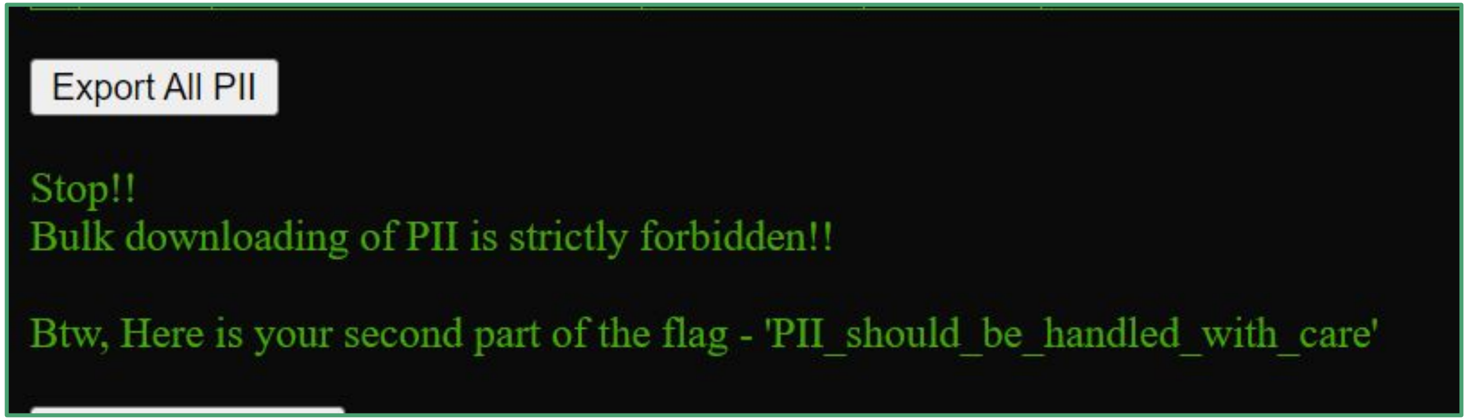
```
    <table id="pii">
       <tr>
           <th>Id</th>
           <th>Name</th>
           <th>Email</th>
           <th>Phone</th>
           <th>Order Id</th>
           <th>Card No</th>
       </tr>
       <tr>
           <td>1</td>
           <td>aman</td>
           <td>aman@gmail.com</td>
           <td>9913371337</td>
           <td>ORD-987</td>
           <td ><div id="task">1337-9987-9988-7676</div></td>
       </tr>
       <tr>
```

> We get our first part of the flag - 1337

Solution

> For Part2, we have to Export all PII data to excel.

> On clicking the Export PII button, we are asked to stop immediately and informed Bulk downloading of PII is strictly forbidden.



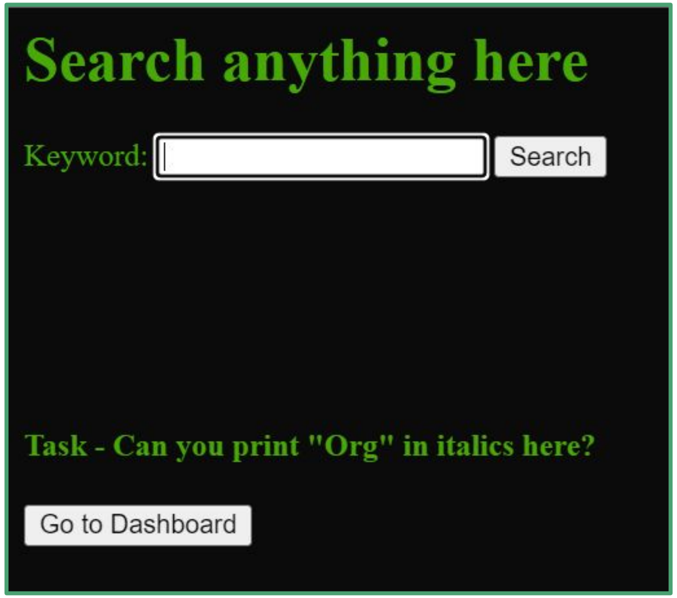
Stop!!
Bulk downloading of PII is strictly forbidden!!

Btw, Here is your second part of the flag - 'PII_should_be_handled_with_care'

> We get our second part of the flag - PII_should_be_handled_with_care

> Hence, we can for our complete flag by combining Part 1 and Part 2 - ORG{1337_PII_should_be_handled_with_care}

Challenge 5 - Inject Me


Search anything here

Keyword:

Task - Can you print "Org" in italics here?

URL - http://orgctf.onrender.com/search

Description - Can you print "Org" in italics here?

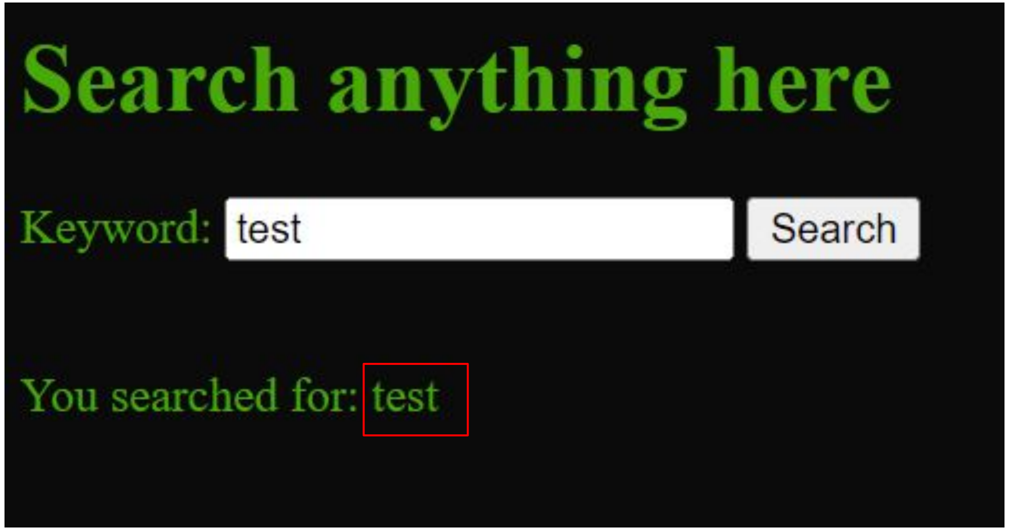
Hint - Click here

Solution

> This challenge expects you to print "Org" in italics. This can be done in HTML by using “i” tag -> <i>Org</i>

> On reading the hint, you get to confirm the same.

> On the challenge page, you find a search box. Let’s type “test” and click on Search button



Search anything here

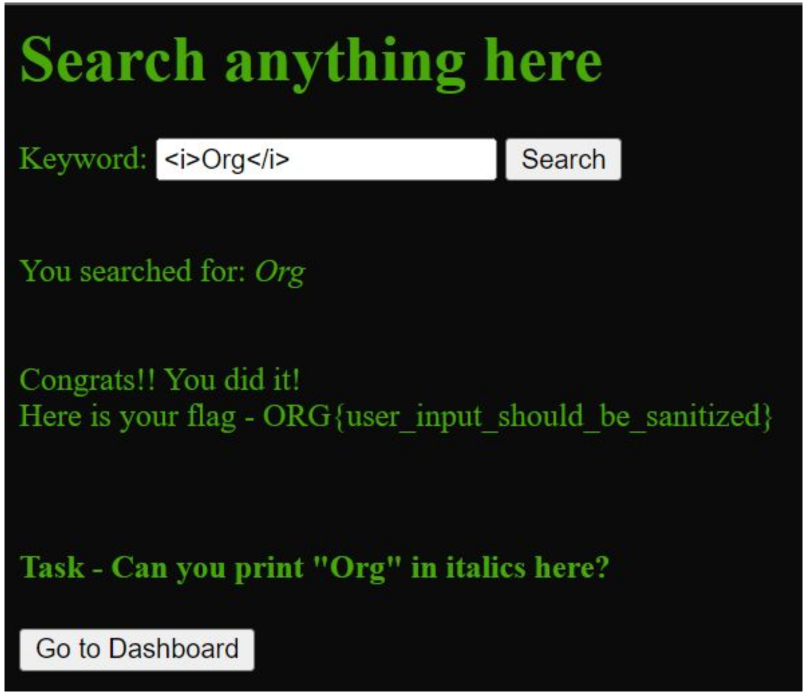
Keyword: test Search

You searched for: test

> We can see, the same keyword “test” is reflected back on the webpage

Solution

> Now, that we know whatever we type is reflected on the webpage, let's search for - <i>Org</i>



Search anything here

Keyword: <i>Org</i> Search

You searched for: *Org*

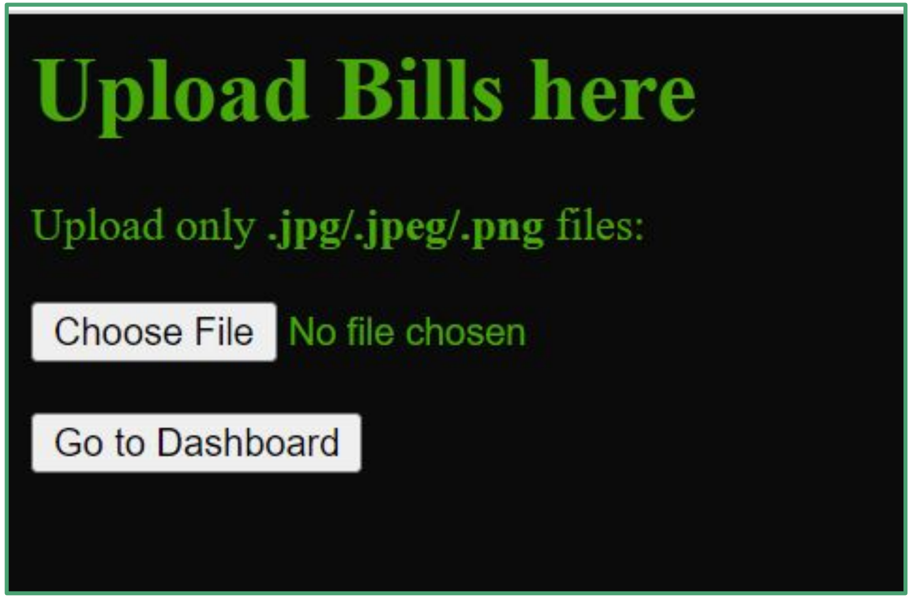
Congrats!! You did it!
Here is your flag - ORG{user_input_should_be_sanitized}

Task - Can you print "Org" in italics here?

Go to Dashboard

> Here we see Org is successfully printed in italics and we get our fifth flag - ORG{user_input_should_be_sanitized}

Challenge 6 - File Upload


Upload Bills here

Upload only **.jpg/.jpeg/.png** files:

URL - http://orgctf.onrender.com/upload

Description - Upload only .jpg/.jpeg/.png files:

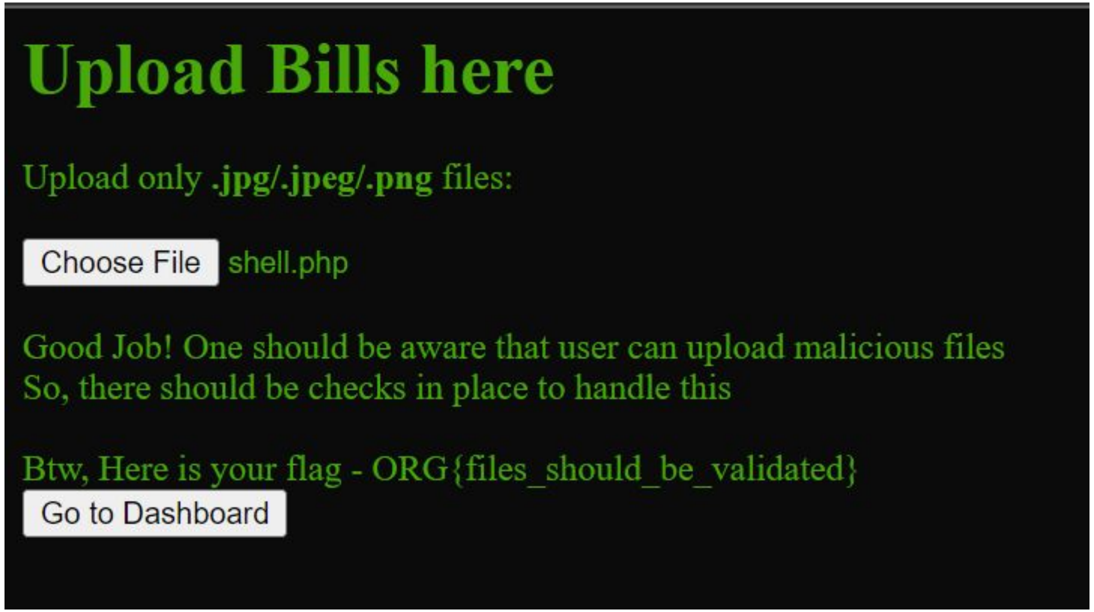
Hint - Try a disallowed filetype

Solution

> This challenge expects you to try to upload a disallowed filetype.

> Try uploading a test.php file. [create a dummy file in your system]

> On doing this we get a popup - Invalid file type! Wow, you tried something different!



Upload Bills here

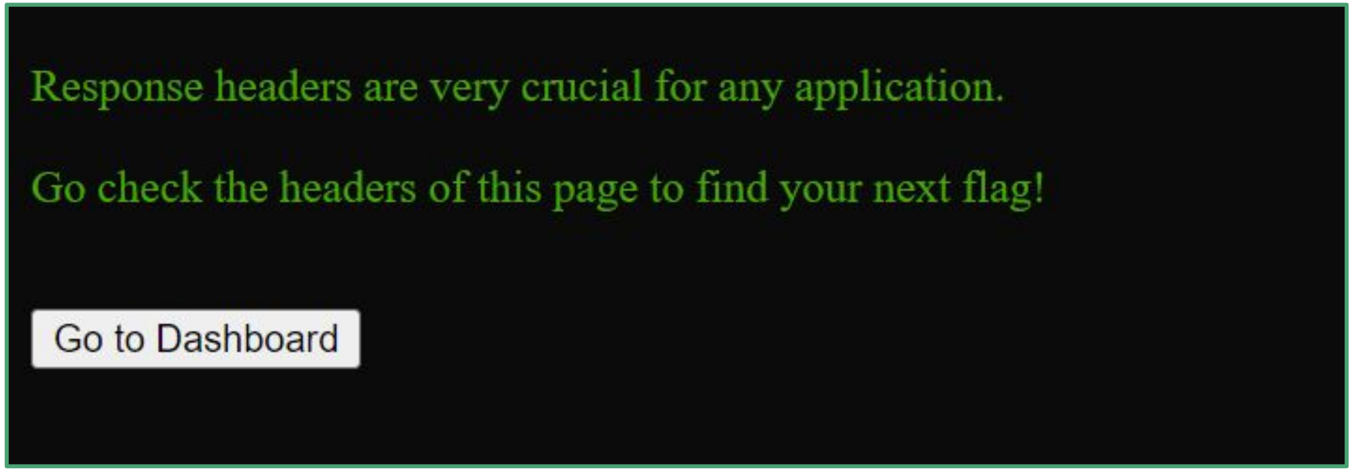
Upload only .jpg/.jpeg/.png files:

Good Job! One should be aware that user can upload malicious files
So, there should be checks in place to handle this

Btw, Here is your flag - ORG{files_should_be_validated}

> That's it, on clicking OK, we can get our sixth flag - ORG{files_should_be_validated}

Challenge 7 - Server Response Header Misconfiguration


Response headers are very crucial for any application.

Go check the headers of this page to find your next flag!

URL - http://orgctf.onrender.com/response

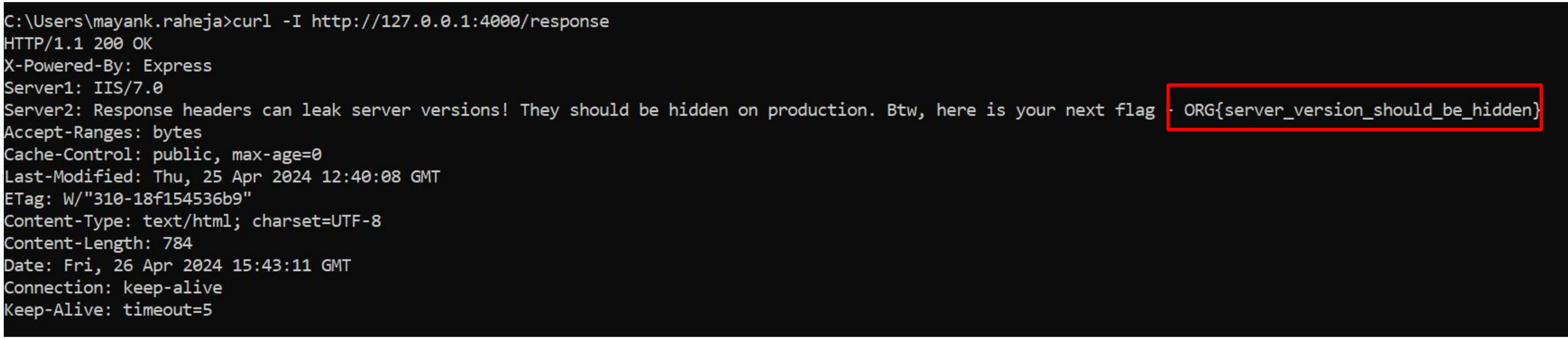
Description - Go check the headers of this page to find your next flag!

Hint - Click Here

Solution

> This challenge expects you to view the headers of the webpage.

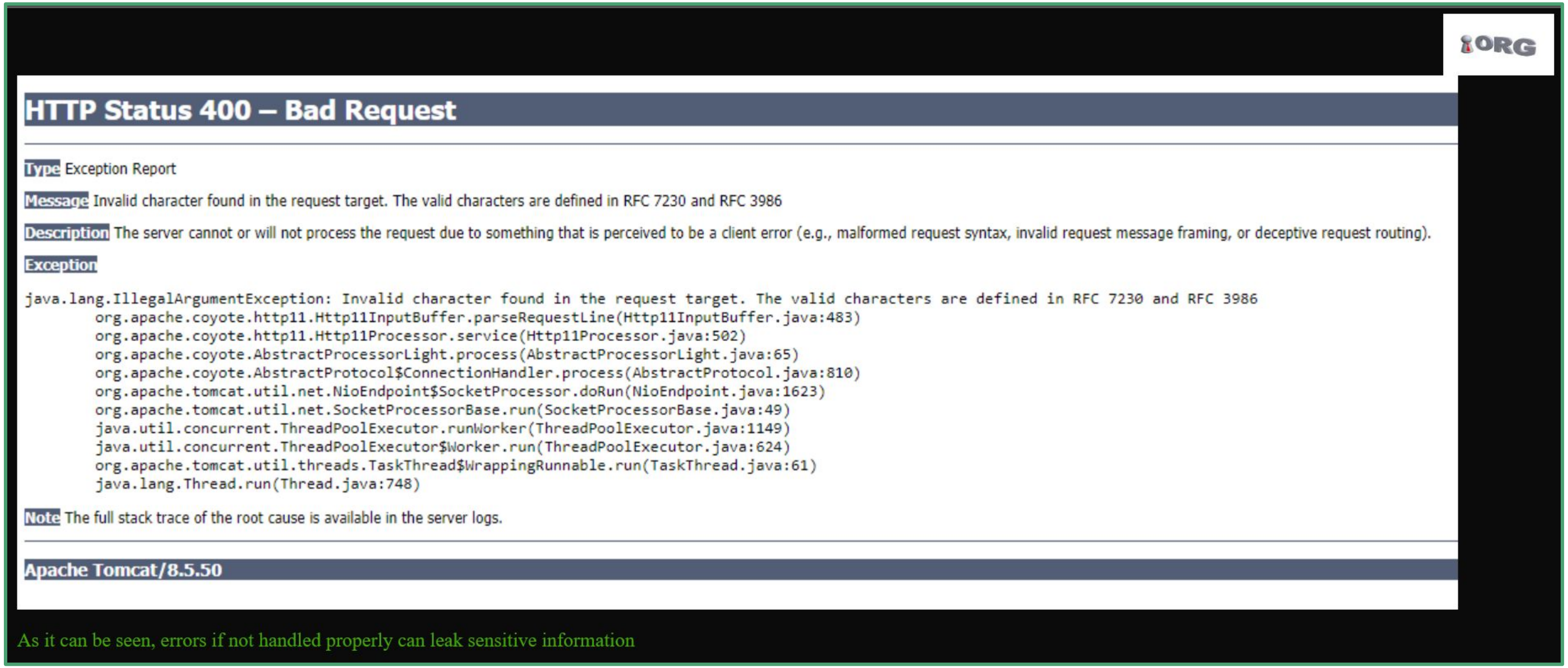
> You can do this by going to the Networks tab in your browser or by hitting a curl - “curl -I http://orgctf.onrender.com/response”



```
C:\Users\mayank.raheja>curl -I http://127.0.0.1:4000/response
HTTP/1.1 200 OK
X-Powered-By: Express
Server1: IIS/7.0
Server2: Response headers can leak server versions! They should be hidden on production. Btw, here is your next flag - ORG{server_version_should_be_hidden}
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Thu, 25 Apr 2024 12:40:08 GMT
ETag: W/"310-18f154536b9"
Content-Type: text/html; charset=UTF-8
Content-Length: 784
Date: Fri, 26 Apr 2024 15:43:11 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```

> Here we get our seventh flag - ORG{server_version_should_be_hidden}

Challenge 8 - Error Stack Trace


iORG

HTTP Status 400 – Bad Request

Type Exception Report

Message Invalid character found in the request target. The valid characters are defined in RFC 7230 and RFC 3986

Description The server cannot or will not process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).

Exception

```
java.lang.IllegalArgumentException: Invalid character found in the request target. The valid characters are defined in RFC 7230 and RFC 3986
	org.apache.coyote.http11.Http11InputBuffer.parseRequestLine(Http11InputBuffer.java:483)
	org.apache.coyote.http11.Http11Processor.service(Http11Processor.java:502)
	org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.java:65)
	org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:810)
	org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1623)
	org.apache.tomcat.util.net.SocketProcessorBase.run(SocketProcessorBase.java:49)
	java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
	java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
	org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
	java.lang.Thread.run(Thread.java:748)
```

Note The full stack trace of the root cause is available in the server logs.

Apache Tomcat/8.5.50

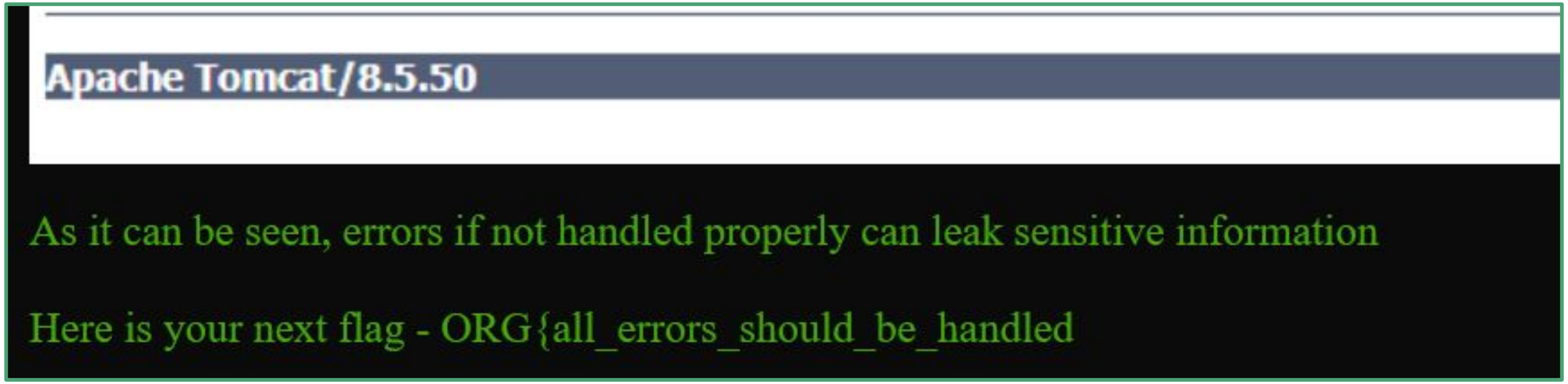
As it can be seen, errors if not handled properly can leak sensitive information

URL - http://orgctf.onrender.com/error

Solution

> This challenge is created to make you aware about error stack trace vulnerability leaking sensitive information.

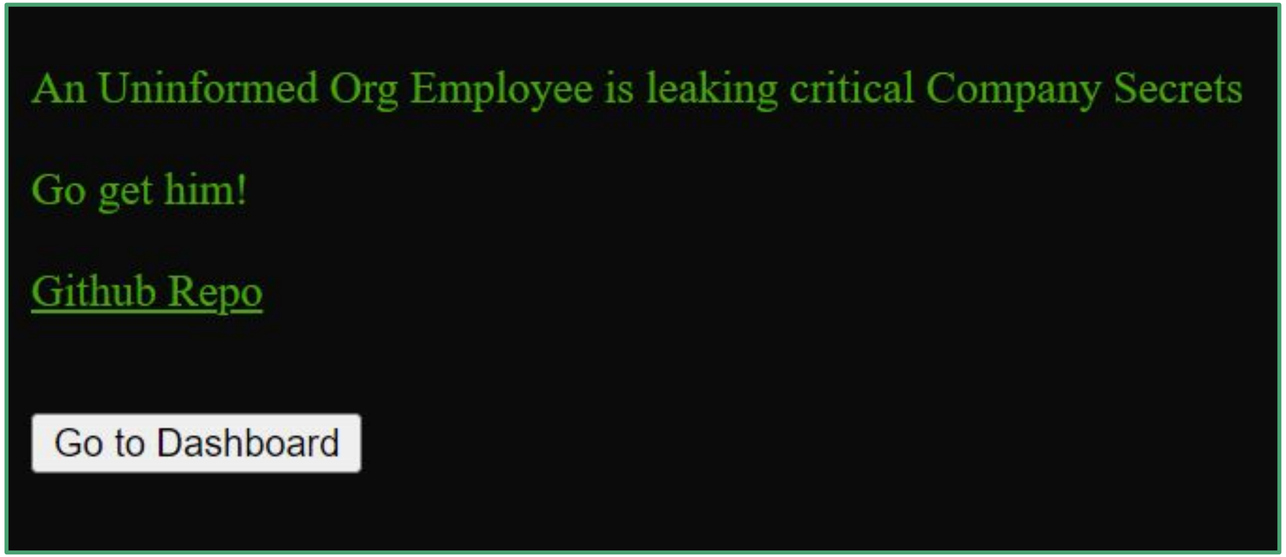
> No task is to be performed here.



```
Apache Tomcat/8.5.50

As it can be seen, errors if not handled properly can leak sensitive information

Here is your next flag - ORG{all_errors_should_be_handled
```

> Here we get our eighth flag - ORG{all_errors_should_be_handled}

Challenge 9 - Secrets leaked


An Uninformed Org Employee is leaking critical Company Secrets

Go get him!

Github Repo

URL - http://orgctf.onrender.com/github

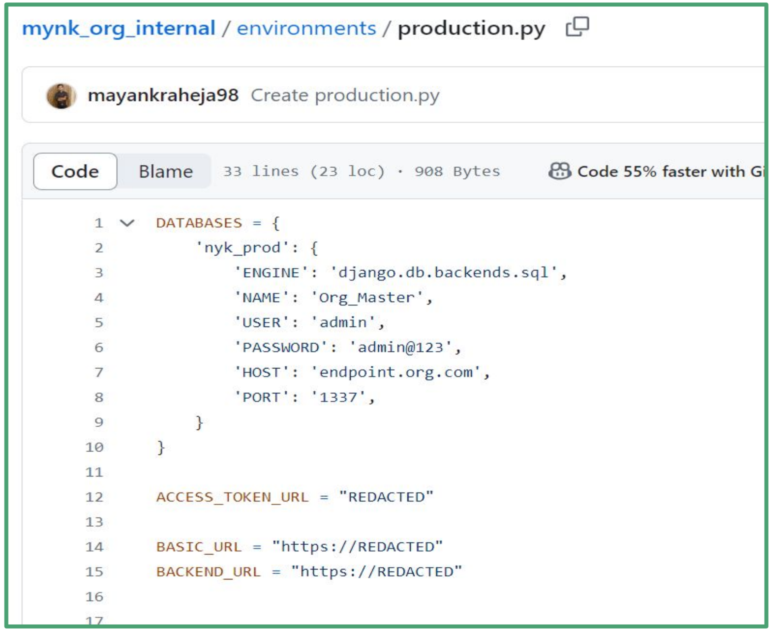
Description - An Uninformed ORG Employee is leaking critical Company Secrets. Go get him!

Solution

> This challenge is created to make you aware about developers leaking sensitive information in code.

> Open the github repo link provided -> Open the production.py file in environments directory.

> There it can be seen the user has hardcoded DB username, password, host, S3 bucket and other details.



mynk_org_internal / environments / production.py

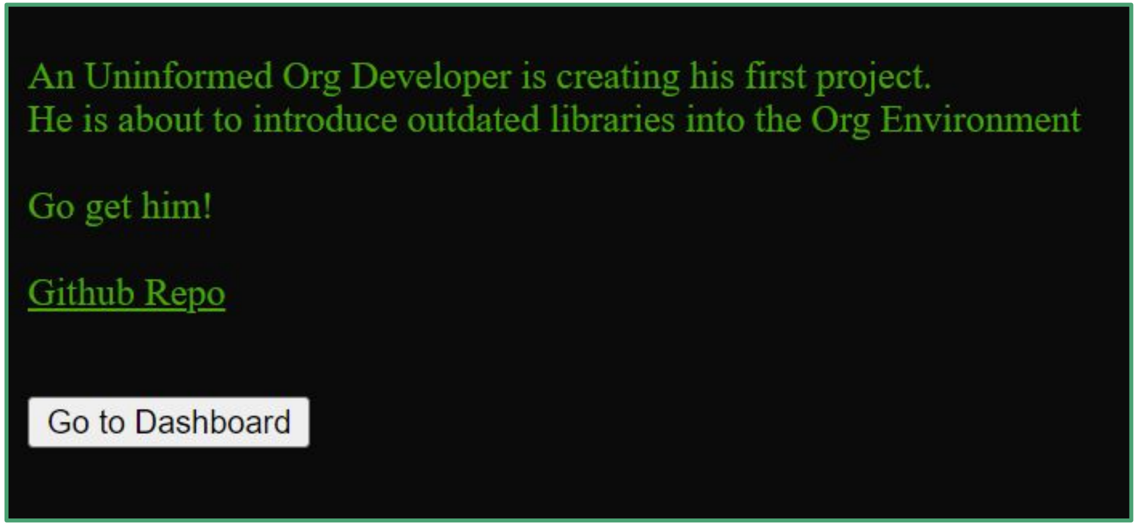
mayankraheja98 Create production.py

Code Blame 33 lines (23 loc) · 908 Bytes Code 55% faster with Gi

```
DATABASES = {
    'nyk_prod': {
        'ENGINE': 'django.db.backends.sql',
        'NAME': 'Org_Master',
        'USER': 'admin',
        'PASSWORD': 'admin@123',
        'HOST': 'endpoint.org.com',
        'PORT': '1337',
    }
}

ACCESS_TOKEN_URL = "REDACTED"

BASIC_URL = "https://REDACTED"
BACKEND_URL = "https://REDACTED"

```

> On reading the last line, go to http://orgctf.onrender.com//flag9 and we get our ninth flag - ORG{do_not_hardcode_secrets}

Challenge 10 - Outdated Dependencies used



An Uninformed Org Developer is creating his first project.
He is about to introduce outdated libraries into the Org Environment

Go get him!

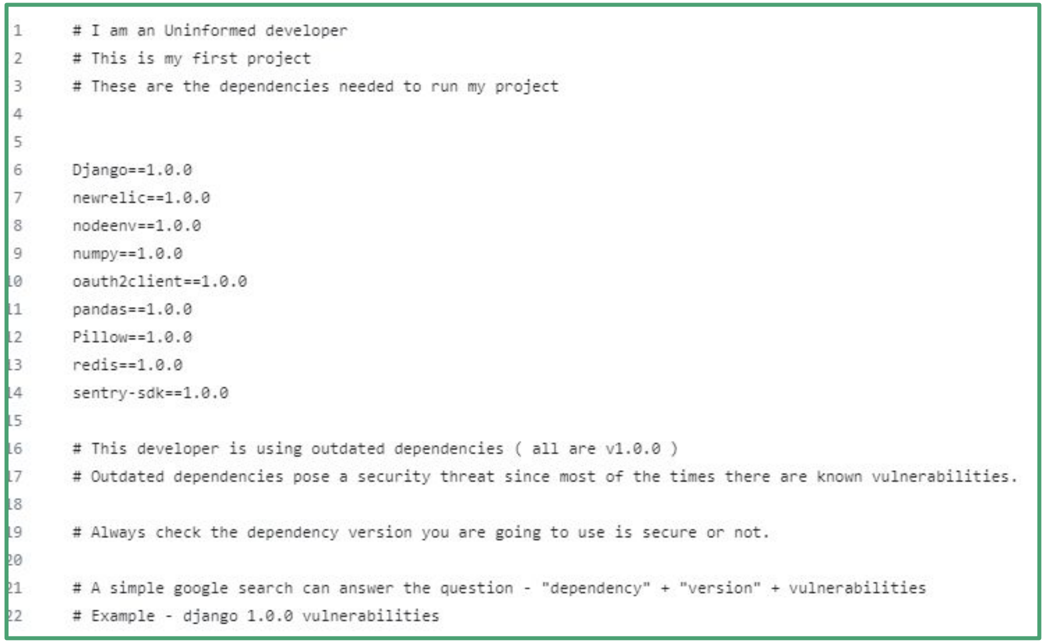
Github Repo

URL - http://orgctf.onrender.com/dependency

Description - An Uninformed Org Developer is creating his first project. He is about to introduce outdated libraries into the Org Environment. Go get him!

Solution

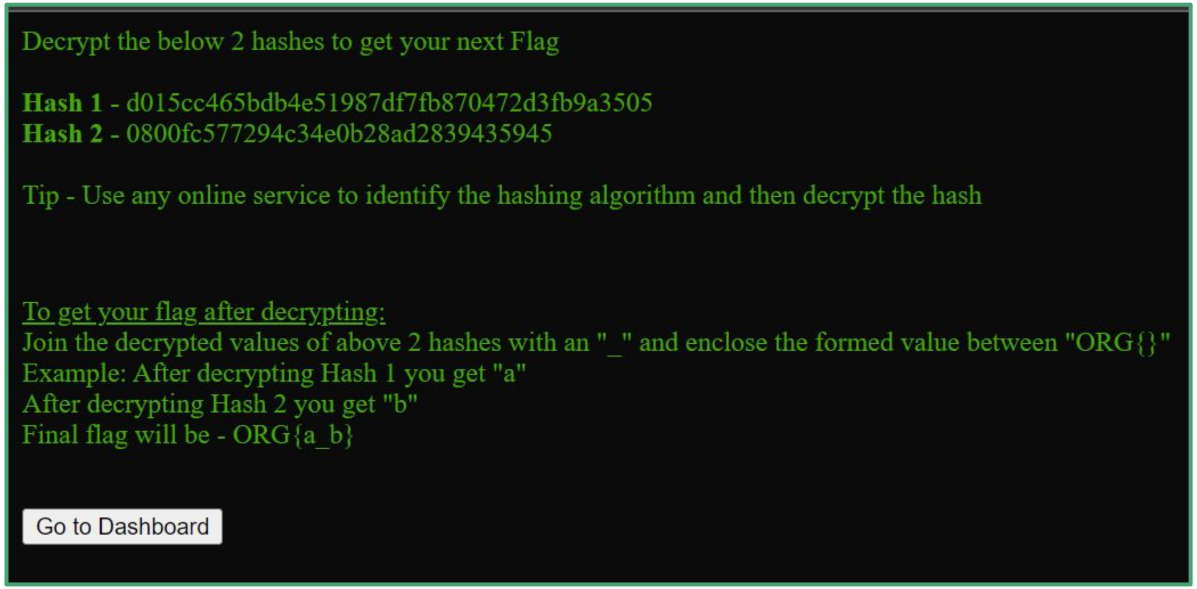
> This challenge is created to make you aware about vulnerabilities introduced by using outdated and vulnerable libraries
> Open the github repo link provided -> Open the requirements.txt file.
> There it can be seen 1.0.0 version of various libraries being used. Older versions of libraries usually have known vulnerabilities.



```
# I am an Uninformed developer
# This is my first project
# These are the dependencies needed to run my project


Django==1.0.0
newrelic==1.0.0
nodeenv==1.0.0
numpy==1.0.0
oauth2client==1.0.0
pandas==1.0.0
Pillow==1.0.0
redis==1.0.0
sentry-sdk==1.0.0

# This developer is using outdated dependencies ( all are v1.0.0 )
# Outdated dependencies pose a security threat since most of the times there are known vulnerabilities.

# Always check the dependency version you are going to use is secure or not.

# A simple google search can answer the question - "dependency" + "version" + vulnerabilities
# Example - django 1.0.0 vulnerabilities
```

> On reading the last line, go to http://orgctf.onrender.com/flag10 and we get our tenth flag - ORG{do_not_use_outdated_dependencies}

Challenge 11 - Insecure Hashing used



Decrypt the below 2 hashes to get your next Flag

Hash 1 - d015cc465bdb4e51987df7fb870472d3fb9a3505
Hash 2 - 0800fc577294c34e0b28ad2839435945

Tip - Use any online service to identify the hashing algorithm and then decrypt the hash

To get your flag after decrypting:
Join the decrypted values of above 2 hashes with an "_" and enclose the formed value between "ORG{}"
Example: After decrypting Hash 1 you get "a"
After decrypting Hash 2 you get "b"
Final flag will be - ORG{a_b}

URL - http://orgctf.onrender.com/hash

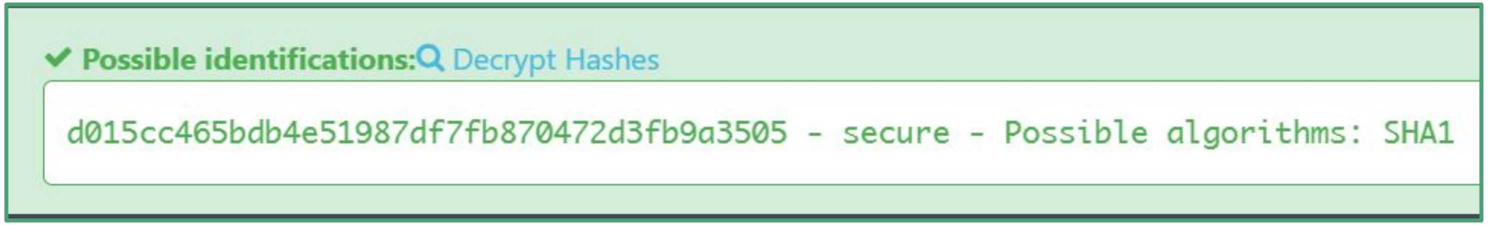
Description - Decrypt the below 2 hashes to get your next Flag
Hash 1 - d015cc465bdb4e51987df7fb870472d3fb9a3505
Hash 2 - 0800fc577294c34e0b28ad2839435945

Solution

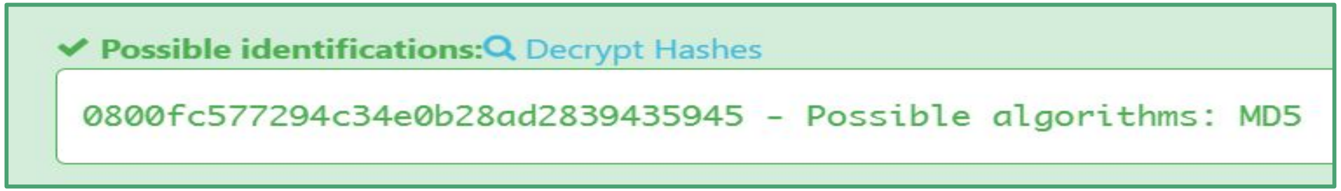
> To complete this challenge we need to decrypt 2 hashes provided.

> To decrypt the hash, we first need to know what type of hash it is. We can search for hash identifiers on google or we can use the Reference 1 provided

> Doing this, we find out that Hash 1 is SHA 1 and Hash2 is MD5


Possible identifications: Decrypt Hashes

```
d015cc465bdb4e51987df7fb870472d3fb9a3505 - secure - Possible algorithms: SHA1
```

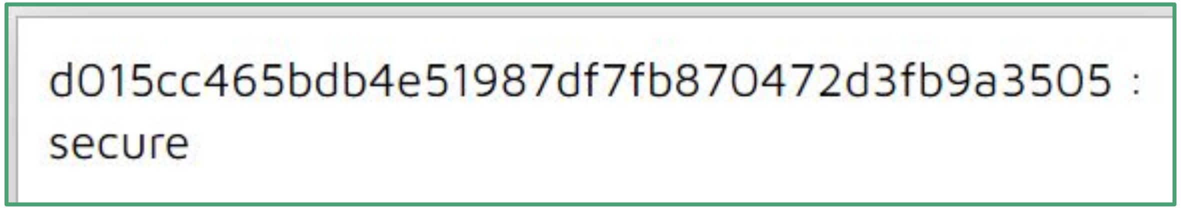

Possible identifications: Decrypt Hashes

```
0800fc577294c34e0b28ad2839435945 - Possible algorithms: MD5
```

> Next, we need to decrypt these SHA1 and MD5 hashes. We can use the Reference 2 and Reference 3 for that.

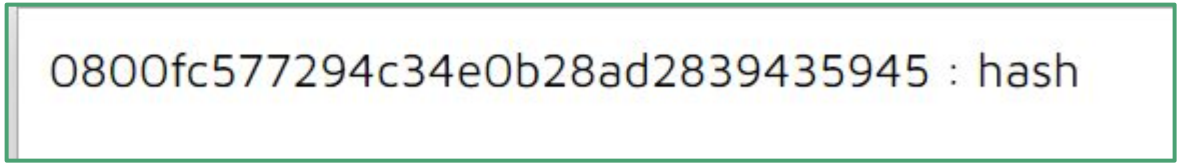
Solution

> On decrypting Hash1(SHA1) we get - “secure”



```
d015cc465bdb4e51987df7fb870472d3fb9a3505 :
secure
```

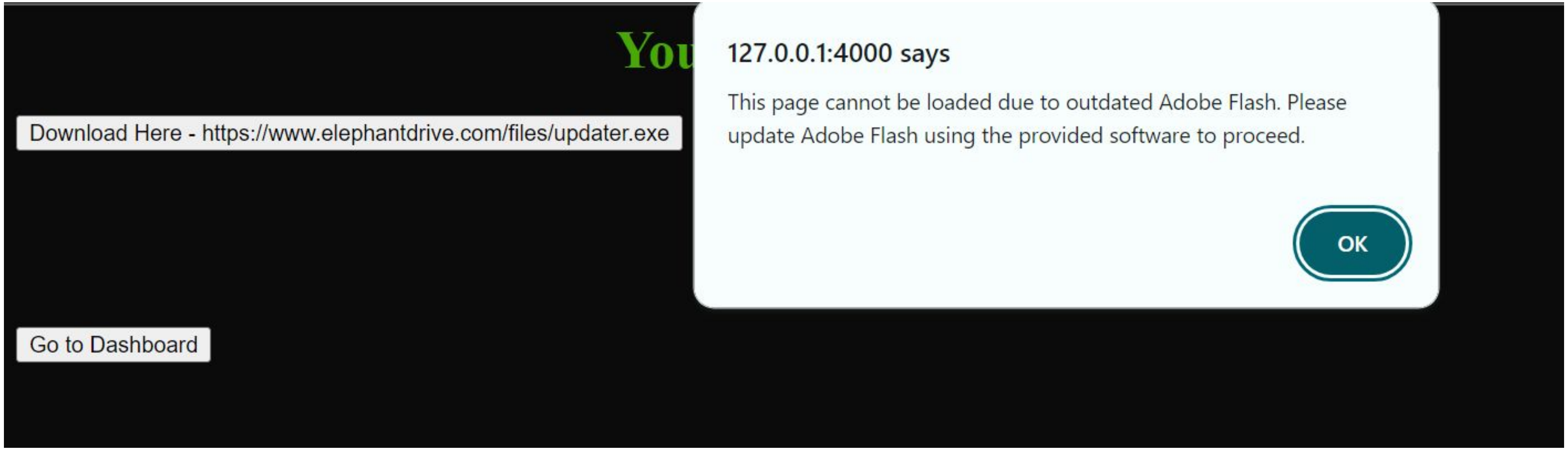
> On decrypting Hash2(MD5) we get - “hash”



```
0800fc577294c34e0b28ad2839435945 : hash
```

> Using these two, we can form our final flag - ORG{secure_hash}

Challenge 12 - Malicious File download


You

Download Here - https://www.elephantdrive.com/files/updater.exe

127.0.0.1:4000 says

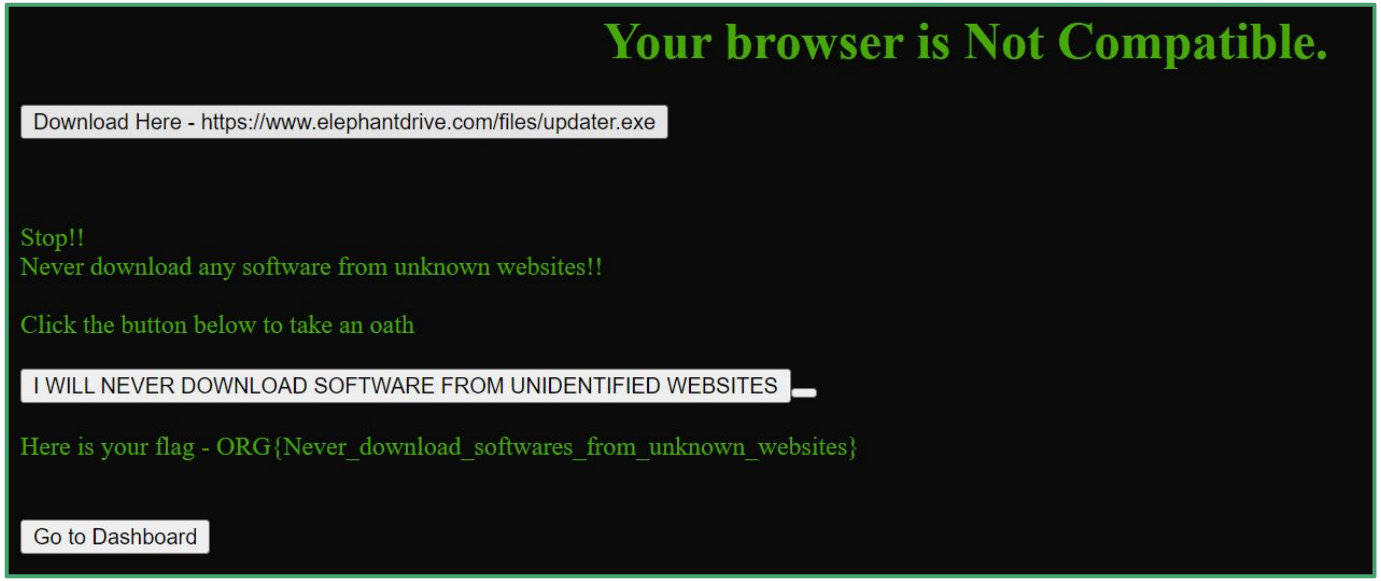
This page cannot be loaded due to outdated Adobe Flash. Please update Adobe Flash using the provided software to proceed.

URL - http://orgctf.onrender.com/download

Description - This page cannot be loaded due to outdated Adobe Flash. Please update Adobe Flash using the provided software to proceed.

Solution

> This challenge is created to make you aware about vulnerabilities introduced by downloading softwares/installers from untrusted websites
> It can be seen that the website is asking us to download Adobe Flash from an unknown source - “elephantdrive.com” . As soon as you click on the Download button, you are asked to Stop!



Your browser is Not Compatible.

Download Here - https://www.elephantdrive.com/files/updater.exe

Stop!!
Never download any software from unknown websites!!

Click the button below to take an oath

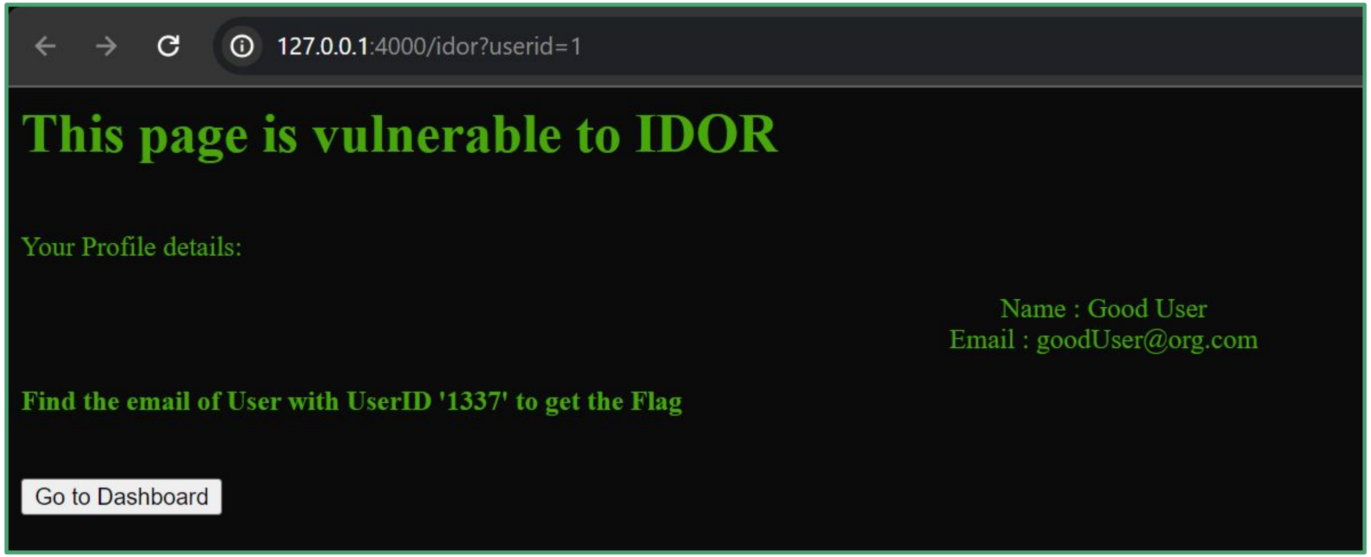
I WILL NEVER DOWNLOAD SOFTWARE FROM UNIDENTIFIED WEBSITES

Here is your flag - ORG{Never_download_softwares_from_unknown_websites}

Go to Dashboard

> On clicking on the button to take oath, we get our twelfth flag - ORG{Never_download_softwares_from_unknown_websites}

Challenge 13 - IDOR


127.0.0.1:4000/idor?userid=1

This page is vulnerable to IDOR

Your Profile details:

Name : Good User
Email : goodUser@org.com

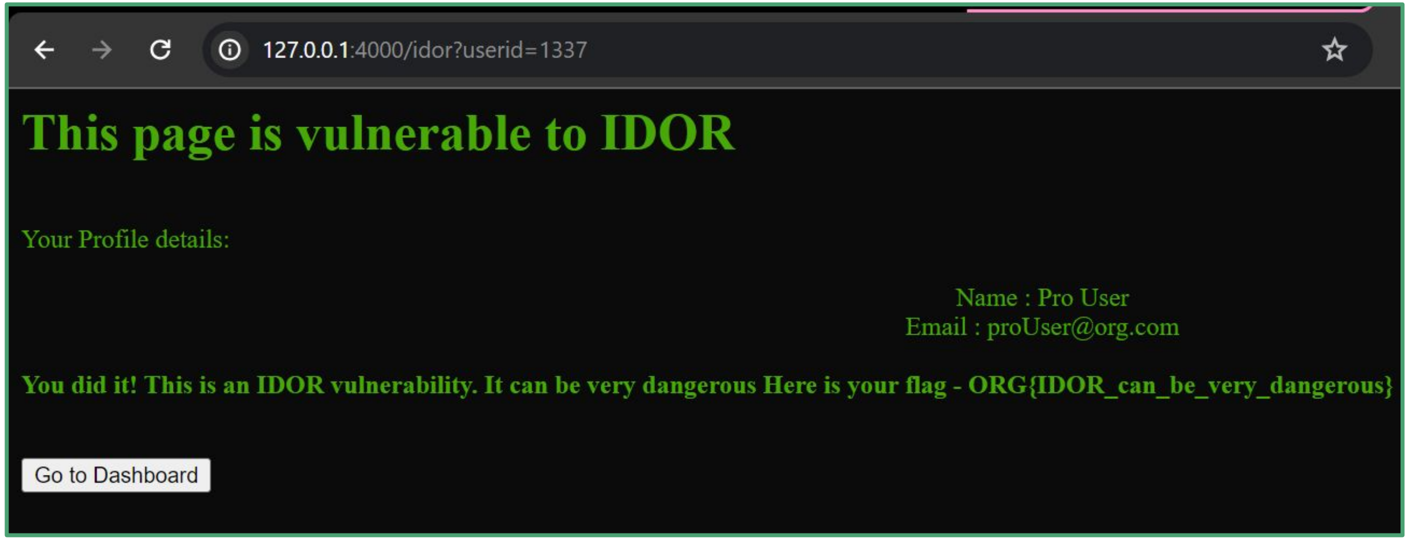
Find the email of User with UserID '1337' to get the Flag

URL - http://orgctf.onrender.com/idor?userid=1

Description - Find the email of User with UserID '1337' to get the Flag

Solution

> This challenge is created to make you aware about IDOR vulnerabilities
> The task is to Find the email of UserID 1337
> In the URL we can see there is a userid parameter that is currently set to 1, which is supposed to be our current user id.



127.0.0.1:4000/idor?userid=1337

This page is vulnerable to IDOR

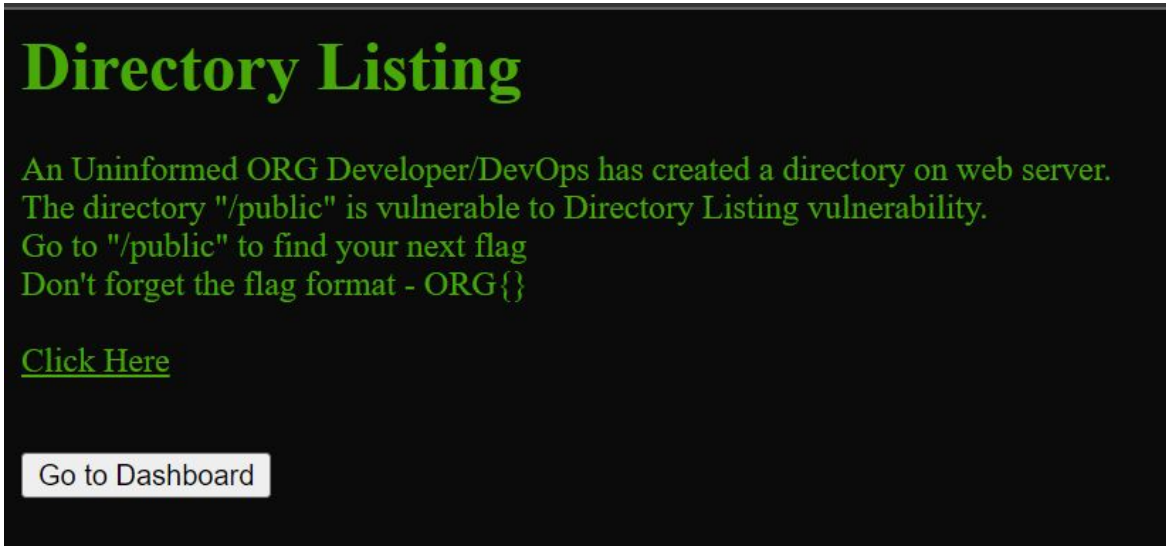
Your Profile details:

Name : Pro User
Email : proUser@org.com

You did it! This is an IDOR vulnerability. It can be very dangerous Here is your flag - ORG{IDOR_can_be_very_dangerous}

> On changing the value of the userid parameter to 1337, we get our thirteenth flag - ORG{IDOR_can_be_very_dangerous}

Challenge 14 - Directory Listing



Directory Listing

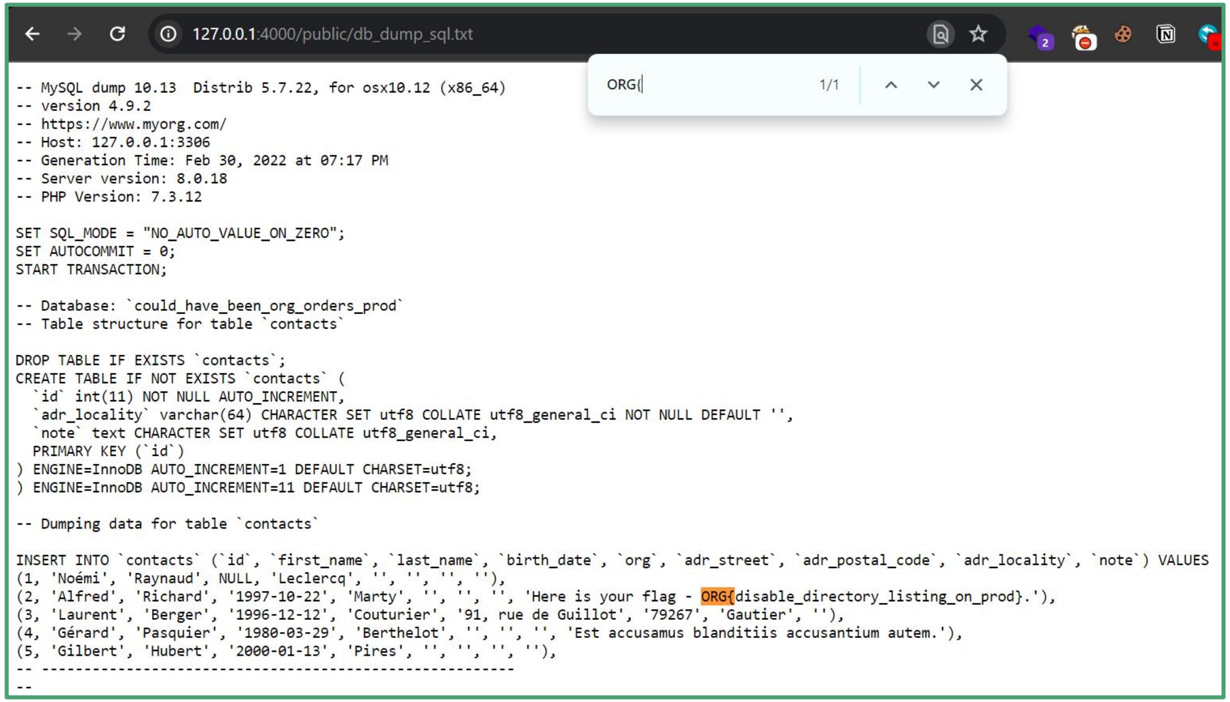
An Uninformed ORG Developer/DevOps has created a directory on web server.
The directory "/public" is vulnerable to Directory Listing vulnerability.
Go to "/public" to find your next flag
Don't forget the flag format - ORG{}

URL - http://orgctf.onrender.com/dirlist

Description - An Uninformed Org Developer/DevOps has created a directory on web server. The directory "/public" is vulnerable to Directory Listing vulnerability. Go to "/public" to find your next flag

Solution

> This challenge is created to make you aware about directory listing
> On visiting http://orgctf.onrender.com/public and reading the readMe file, we get to know 2 sensitive files are exposed.
> We need to check both the files to find our flag.


127.0.0.1:4000/public/db_dump_sql.txt

ORG{ 1/1

```
-- MySQL dump 10.13  Distrib 5.7.22, for osx10.12 (x86_64)
-- version 4.9.2
-- https://www.myorg.com/
-- Host: 127.0.0.1:3306
-- Generation Time: Feb 30, 2022 at 07:17 PM
-- Server version: 8.0.18
-- PHP Version: 7.3.12

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;

-- Database: `could_have_been_org_orders_prod`
-- Table structure for table `contacts`

DROP TABLE IF EXISTS `contacts`;
CREATE TABLE IF NOT EXISTS `contacts` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `adr_locality` varchar(64) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL DEFAULT '',
  `note` text CHARACTER SET utf8 COLLATE utf8_general_ci,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=utf8;

-- Dumping data for table `contacts`

INSERT INTO `contacts` (`id`, `first_name`, `last_name`, `birth_date`, `org`, `adr_street`, `adr_postal_code`, `adr_locality`, `note`) VALUES
(1, 'Noémi', 'Raynaud', NULL, 'Leclercq', '', '', '', ''),
(2, 'Alfred', 'Richard', '1997-10-22', 'Marty', '', '', '', 'Here is your flag - ORG{disable_directory_listing_on_prod}.'),
(3, 'Laurent', 'Berger', '1996-12-12', 'Couturier', '91, rue de Guillot', '79267', 'Gautier', ''),
(4, 'Gérard', 'Pasquier', '1980-03-29', 'Berthelot', '', '', '', 'Est accusamus blanditiis accusantium autem.'),
(5, 'Gilbert', 'Hubert', '2000-01-13', 'Pires', '', '', '', ''),
-- --------------------------------------------------------
--
```

> On opening db_dump_sql file and searching for "ORG{", we get our fourteenth flag - ORG{disable_directory_listing_on_prod}