



CyberJutsu

CHALLENGE WRITE UP

Operational Vulnerability



MỤC LỤC

I.Challenge - 01.....	3
1.Lab Overview	3
2.Analysis	3
II.Challenge - 02.....	7
1.Lab Overview	7
2.Analysis	7
Level 1:	7
Level 2:	15
Level 3:	19



I. CHALLENGE - 01

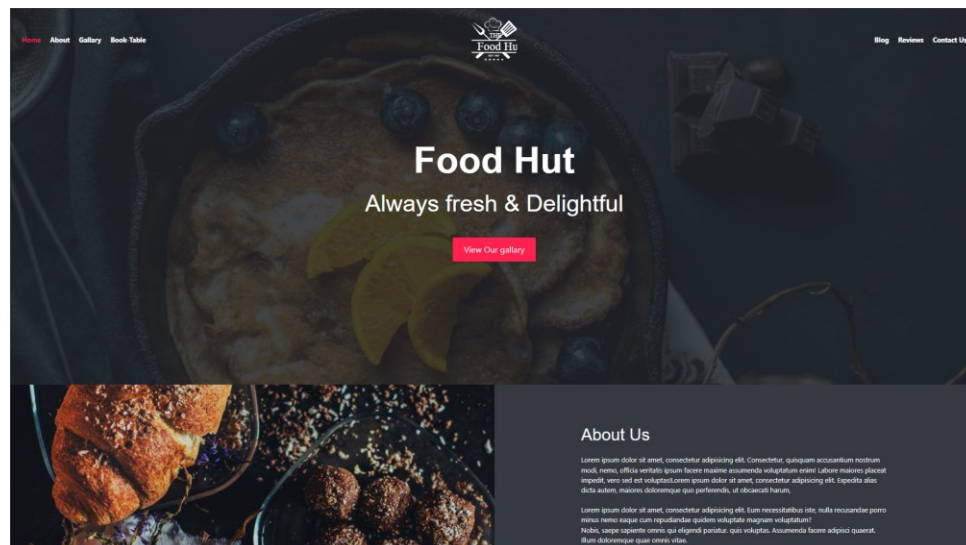
1. LAB OVERVIEW

Goal: Một thành phần của trang web đã bị **cấu hình dự permission**. Hãy tìm ra nó và lấy bí mật của nhà hàng.

**Note: Trong môi trường thí nghiệm của bài labs này, các bạn không cần thực hiện subdomains scan, directories scan, nmap.*

2. ANALYSIS

- Bước đầu tiên, sử dụng các tính năng của ứng dụng.
- Nhìn nhanh, đây là một landing page được dùng để giới thiệu về nhà hàng, nên không có nhiều tính năng để user tương tác.



- Thu hẹp phạm vi kiểm thử, thực hiện kiểm tra source của trang web.
- Một vài thứ chúng ta cần để ý:
 - Các api, resource được gọi từ trang web
 - Nội dung các file javascript



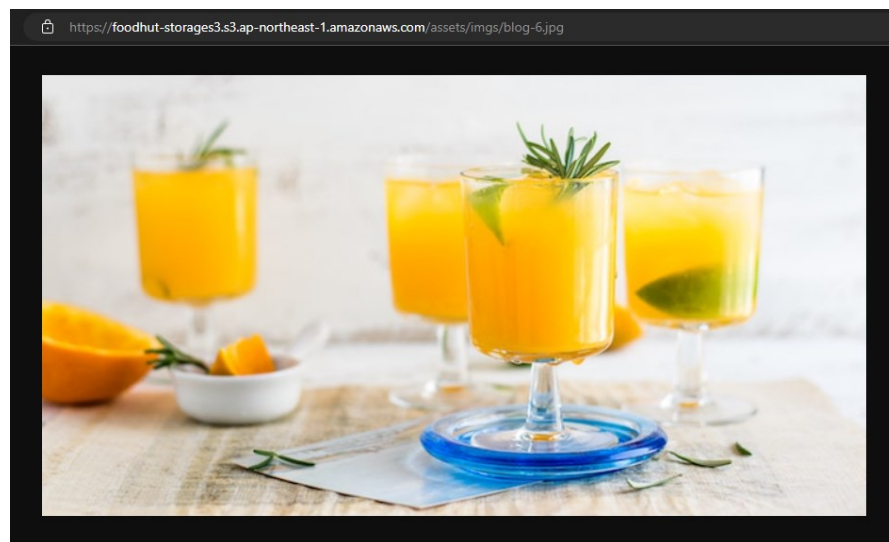
- Quan sát thấy các resource liên quan đến assets của trang web đều có điểm chung.

```
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
<meta name="description" content="Start your development with FoodHut landing page.">
<meta name="author" content="Devcrud">
<title>FoodHut</title>

<!-- font icons -->
<link rel="stylesheet" href="https://foodhut-storages3.s3.ap-northeast-1.amazonaws.com/assets/vendors/themify-icons/css/themify-icons.css">
<link rel="stylesheet" href="https://foodhut-storages3.s3.ap-northeast-1.amazonaws.com/assets/vendors/animate/animate.css">

<!-- Bootstrap + FoodHut main styles -->
<link rel="stylesheet" href="https://foodhut-storages3.s3.ap-northeast-1.amazonaws.com/assets/css/foodhut.css">
</head>
```

- Để ý ở các url đều trỏ đến tên miền của Amazon, cho thấy chúng sử dụng dịch vụ lưu trữ của Amazon gọi là Amazon S3 hoặc Amazon Simple Storage Service (**S3 Bucket**)
- Truy cập thử vào một url, <https://foodhut-storages3.s3.ap-northeast-1.amazonaws.com/assets/imgs/blog-6.jpg>.



- **Đặt câu hỏi:** Vì đây là một bài liên quan đến misconfigured, nên mình tự hỏi liệu s3 bucket này có bị Directory Listing như bên apache2 không ?
- Để kiểm chứng, thử xóa tên file ảnh trên url <https://foodhut-storages3.s3.ap-northeast-1.amazonaws.com/assets/imgs/>.
- Tới đây, ta được s3 trả về nội dung như sau, được định dạng bằng **XML**.



```
https://foodhut-storages3.s3.ap-northeast-1.amazonaws.com/assets/imgs/

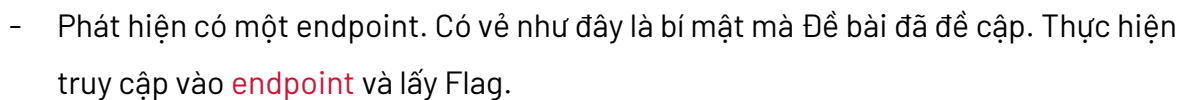
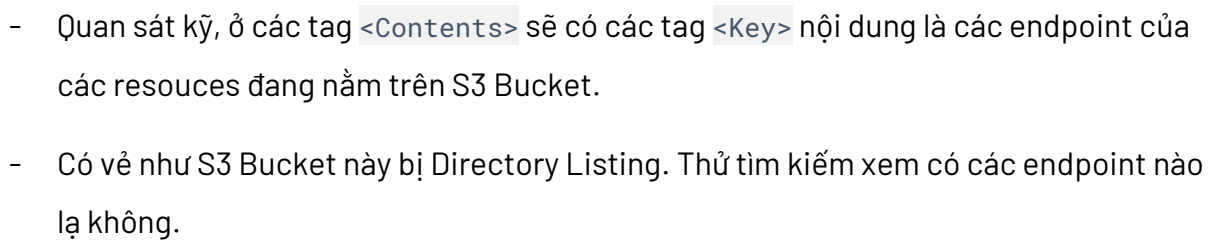
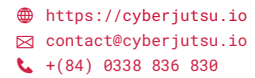
This XML file does not appear to have any style information associated with it. The document tree is shown below.

<?xml version="1.0" encoding="UTF-8" ?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The specified key does not exist.</Message>
  <Key>assets/imgs/</Key>
  <RequestId>P1BQRG96P27XQZ0H</RequestId>
  <HostId>WqBREiTGTCjWCOKwJjEyr9ILp1QHmropSh0se0XQCN47PCc1x50dkbHJg0xZhZD4L/ESCt+9jGg87hYhrzQ/cA==</HostId>
</Error>
```

- Có tag `<Message>` thông báo gì đó về việc không có key.
- Chẳng lẽ s3 bucket này không bị Directory Listing ư?
- Mình thử xóa thêm từng folder `imgs/`, `assets/`, cho đến khi về `index` của S3 Bucket.



- Xuất hiện cái gì đó là lạ. Cũng là định dạng XML nhưng mà nó lạ lẫm.





II. CHALLENGE - 02

1. LAB OVERVIEW

Ở challenge này gồm 3 level, liên quan đến cùng 1 ứng dụng.

Goal: Chiếm tài khoản có số điện thoại 0123456789

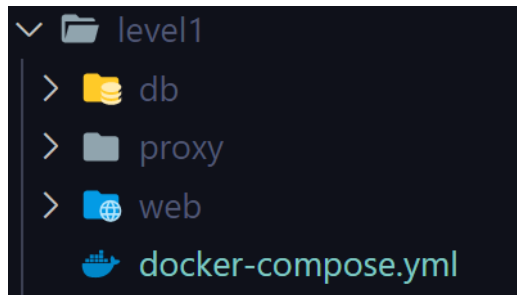
**Note: Trong môi trường thí nghiệm của bài labs này, các bạn không cần thực hiện subdomains scan, directories scan, nmap.*

2. ANALYSIS

LEVEL 1:

CÁCH HOẠT ĐỘNG CỦA ỨNG DỤNG:

- Ứng dụng gồm 2 chức năng chính:
 - > Đăng nhập theo số điện thoại và OTP thay vì mật khẩu.
 - > Lấy mã OTP thông qua số điện thoại.
- Cùng phân tích source code.



- Nhìn vào cấu trúc thư mục file + đọc nội dung của `docker-compose.yml`, ta có thể biết được ứng dụng này gồm 3 service chính tương ứng với 3 folders.
 - > web: server đang host ứng dụng web
 - > db: là database phục vụ cho ứng dụng web
 - > proxy: làm trung gian tương tác giữa web và internet
- Tạm thời, chỉ cần quan tâm đến `web/`, gồm 3 file:
 - > `index.php`, `gen_otp.php`, `connect_db.php`



- Trong `index.php`, đây là nơi xử lý quá trình đăng nhập của ứng dụng.

```
index.php
level1 > web > src > index.php > ...
1  <?php
2  include("connect_db.php");
3  if (isset($_POST['phone']) and isset($_POST['otp'])) {
4      $time = date("Y-m-d H:i:s");
5      $phone = $conn->real_escape_string($_POST['phone']);
6      $otp = $conn->real_escape_string($_POST['otp']);
7
8      $sql = "SELECT * FROM Users WHERE phone_number='$phone' AND otp='$otp'";
9      $result = $conn->query($sql);
10
11     $info = "";
12     if ($result->num_rows > 0) {
13         while ($row = $result->fetch_assoc()) {
14             // OTP will expire in 15 minutes
15             if ((strtotime($time) - strtotime($row['opt_created_time'])) > 900) {
16                 $info = "<pre>Token is expired</pre>";
17             } else {
18                 // this a phone number of vip account
19                 if ($row['phone_number'] === '0123456789') {
20                     $info .= "<pre>Login successful</pre>";
21                     $info .= "<br>";
22                     $info .= "<pre>CBJS{FAKE_FLAG}</pre>";
23                 } else {
24                     $info = "<pre>Login successful</pre>";
25                 }
26             }
27         }
28     } else {
29         $info = "<pre>User not exist or OTP is wrong</pre>";
30     }
}
```

- > **Line 4:** `$time` lưu trữ thời điểm khi cú request đăng nhập được gửi
- > **Line 8:** `$phone` và `$otp` được nằm trong dấu nháy đơn của câu query `$sql`, kết hợp với việc được sanitize bằng hàm `real_escape_string()` (line 5,6) → SQL Injection không xảy ra ở đây.
- > **Line 15:** Kiểm tra xem mã OTP đã hết hạn hay chưa bằng cách so sánh `$time` với `opt_created_time` trong cơ sở dữ liệu (hết hạn nếu chênh lệch lớn hơn 900 giây hay 15 phút).



- Trong file `gen_otp.php`.

```
gen_otp.php X
level1 > web > src > gen_otp.php > ...
1  <?php
2  include("connect_db.php");
3
4  $info = "";
5  function generateOTP($digits = 4)
6  {
7      $i = 0;
8      $otp = "";
9      while ($i < $digits) {
10         $otp .= mt_rand(0, 9);
11         $i++;
12     }
13     return $otp;
14 }
```

- > **Line 5** → **14**, hàm `generateOTP()` sinh mã OTP ngẫu nhiên 4 chữ số bằng hàm `mt_rand()` của PHP.

```
gen_otp.php X
level1 > web > src > gen_otp.php > ...
16  if (isset($_POST['phone'])) {
17      $phone = $conn->real_escape_string($_POST['phone']);
18      $time = date("Y-m-d H:i:s");
19      $sql = "SELECT * FROM Users WHERE phone_number='$phone'";
20      $result = $conn->query($sql);
21
22      if ($result->num_rows > 0) {
23          while ($row = $result->fetch_assoc()) {
24              if ((strtotime($time) - strtotime($row['otp_created_time'])) > 900 or $row['otp'] === NULL) {
25                  $otp = generateOTP();
26                  try {
27                      $sql = "UPDATE Users SET otp='$otp', otp_created_time='$time' WHERE phone_number='$phone'";
28                      $conn->query($sql);
29                      $info .= "<pre>WE SENT OTP TO YOUR PHONE</pre>";
30                      $info .= "<a href='/index.php'>Login</a>";
31                  } catch (\Throwable $th) {
32                      throw $th;
33                  }
34              } else {
35                  $info .= "<pre>OTP is not expired</pre>";
36                  $info .= "<a href='/index.php'>Login</a>";
37              }
38          }
39      } else {
40          $info .= "<pre>Phone Number is not exists</pre>";
41      }
42 }
```

- > **Line 16** → **42**, nhiệm vụ chính là tạo ra mã OTP chỉ khi nó đã hết hạn hoặc chưa được tạo trước đó.



CHECKPOINT:

- Mã OTP có 4 chữ số tức giá trị của mã OTP sẽ nằm trong khoảng 0000→9999 (10000 giá trị)
- Không có cơ chế rate-limit khi đăng nhập sai OTP

Ý TƯỞNG / GIẢI THUYẾT:

- Cho trung bình mất 0,5 giây để gửi một cú request đăng nhập.
- Trong vòng 15 phút, ta sẽ gửi được 1800 request → thử được 1800 mã OTP.
- Với xác suất thành công 1800/10000 (~ 20%) do không có rate-limite, ta hoàn toàn có thể đăng nhập với đúng mã OTP trước khi nó hết hạn.

TIẾN HÀNH KHAI THÁC:

- Viết scripts đơn giản (1) để thực hiện ý tưởng hoặc có thể sử dụng các công cụ có sẵn, ở đây là một extension có trong Burpsuite (2).
- Vấn đề của cách khai thác này, nếu ta mất càng ít thời gian để gửi request → tỉ lệ thành công cao hơn.

(1) Script python đơn giản: @0aa9a3



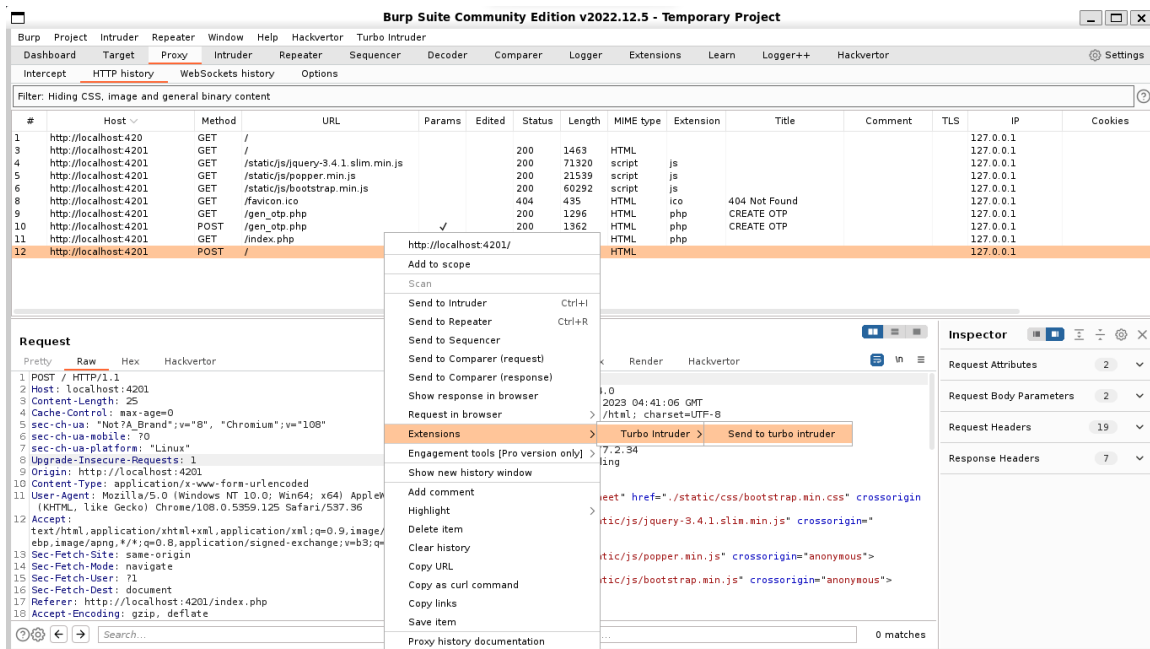
(2) Sử dụng Burpsuite:

Cài đặt extension Turbo Intruder: Extensions → BApp Store → Turbo Intruder → Install

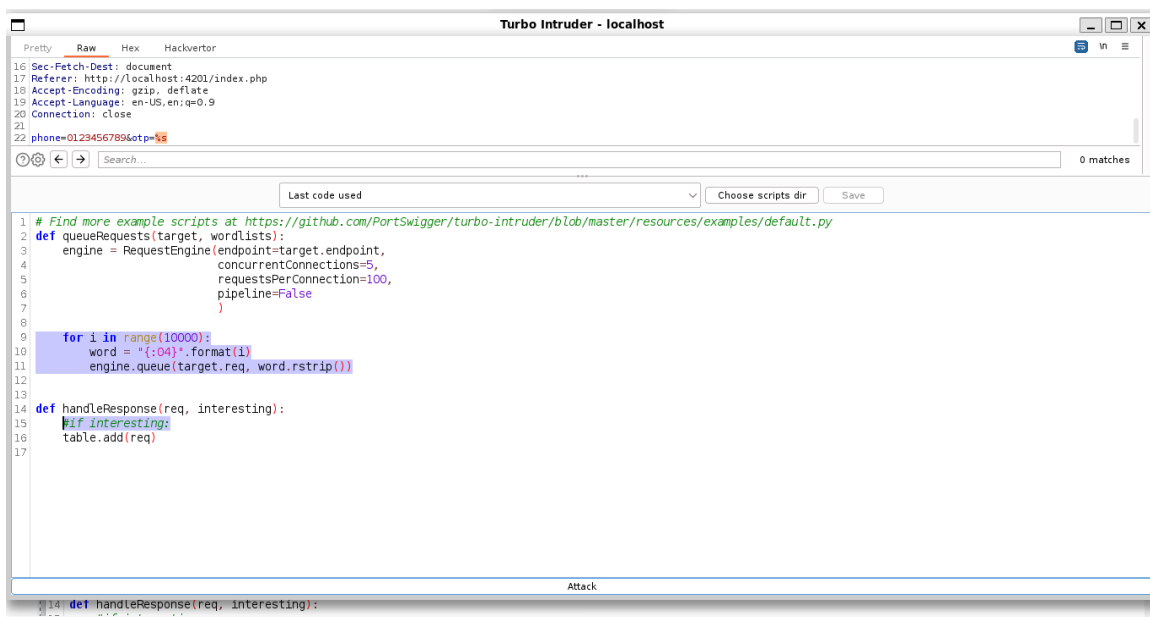
The screenshot shows the Burp Suite interface with the BApp Store open. The 'Turbo Intruder' extension is highlighted in the list. The interface includes a menu bar at the top with options like Dashboard, Target, Proxy, Intruder, Repeater, Sequencer, Decoder, Comparer, Log, Extensions, and Learn. The BApp Store section displays a table of available extensions with columns for Name, Installed, Rating, Popularity, Last updated, System impact, and Detail. The 'Turbo Intruder' extension is highlighted with a red box and a red '3' in the 'Installed' column. To the right of the table, there is a 'Scanner' section with a 'Basic use' section and an 'Estimated system impact' section. The 'Estimated system impact' section shows an overall rating of 'Low' and a table with columns for Memory, CPU, and Time, all showing 'Low' values. At the bottom right, there is an 'Install' button with a red '4' next to it.

Name	Installed	Rating	Popularity	Last updated	System im...	Detail
Stepper		☆☆☆☆☆	+	16 Jul 2020	Low	
Subdomain Extractor		☆☆☆☆☆	+	02 Dec 2019	Low	
Taborator		☆☆☆☆☆	+	20 Dec 2022	Low	Requires Bur...
Target Redirector		☆☆☆☆☆	+	04 Apr 2018	Low	Requires Bur...
ThreadFix		☆☆☆☆☆	+	25 Jan 2017	Low	Requires Bur...
Timeinator, Time Bas...		☆☆☆☆☆	+	25 Feb 2022	Low	
Timestamp Editor		☆☆☆☆☆	+	18 Mar 2021	Low	
Token Extractor		☆☆☆☆☆	+	10 Feb 2022	Low	
Token Incrementor		☆☆☆☆☆	+	27 Nov 2020	Low	
TokenJar		☆☆☆☆☆	+	09 Jun 2022	Low	
Turbo Data Mine		☆☆☆☆☆	+	20 Apr 2022	Low	
Turbo Intruder	3	☆☆☆☆☆	+	06 Jun 2023	Low	
Upload-Scanner		☆☆☆☆☆	+	21 Feb 2022	Low	Requires Bur...
UPnP Hunter		☆☆☆☆☆	+	06 Dec 2021	Low	
UUID Detector		☆☆☆☆☆	+	23 Feb 2017	Low	
ViewState Editor		☆☆☆☆☆	+	10 Mar 2021	Low	
WAF Cookie Fetcher		☆☆☆☆☆	+	16 Jan 2018	Low	
WAFDetect		☆☆☆☆☆	+	25 Aug 2021	Low	Requires Bur...
Wayback Machine		☆☆☆☆☆	+	18 Jun 2018	Low	
WCF Deserializer		☆☆☆☆☆	+	15 Jun 2017	Low	
Web Cache Deception ...		☆☆☆☆☆	+	23 Nov 2017	Low	Requires Bur...
WebAuthn CBOR Deco...		☆☆☆☆☆	+	09 Dec 2022	Low	
Webinspect Connector		☆☆☆☆☆	+	10 Aug 2016	Low	Requires Bur...
WebSphere Portlet St...		☆☆☆☆☆	+	17 Feb 2015	Low	
Wordlist Extractor		☆☆☆☆☆	+	20 Apr 2017	Low	
WordPress Scanner		☆☆☆☆☆	+	25 Feb 2022	Low	
WS Security		☆☆☆☆☆	+	10 Feb 2022	Medium	
WSDL Wizard		☆☆☆☆☆	+	01 Jul 2014	Low	
Wsdler		☆☆☆☆☆	+	01 Nov 2016	Low	
XChromeLogger Deco...		☆☆☆☆☆	+	15 Dec 2021	Low	

- **Bước 1:** Thực hiện tạo mã OTP mới.
- **Bước 2:** Bất cứ request đăng nhập bằng Burpsuite, chuột phải vào forward sang Turbo Intruder.



- Set market cho Turbo Intruder bằng cách thay đổi value của params otp thành %s. (bôi đỏ)
- Modify lại script handler của Turbo Intruder như sau (bôi xanh)





- Bấm Attack và đợi kết quả.

The screenshot shows the Turbo Intruder interface with a table of requests and two panels for request details.

Row	Payload	Status	Words	Length	Time	Label	Queue ID
6481	6484	200	562	1571	6		6485
0	0003	200	558	1532	5		4
1	0002	200	558	1532	5		3
2	0001	200	558	1532	5		2
3	0000	200	558	1532	6		1
4	0004	200	558	1532	5		5
5	0006	200	558	1532	6		7
6	0008	200	558	1532	5		9
7	0009	200	558	1532	5		10
8	0005	200	558	1532	5		6
9	0007	200	558	1532	6		8
10	0013	200	558	1532	5		14
11	0012	200	558	1532	6		13
12	0010	200	558	1532	7		11
13	0014	200	558	1532	7		15
14	0011	200	558	1532	10		12
15	0015	200	558	1532	8		16
16	0016	200	558	1532	11		17
17	0017	200	558	1532	10		18

The bottom panels show the raw data for two requests. The left panel shows a POST request to / HTTP/1.1 with various headers and a body. The right panel shows an HTTP/1.1 200 OK response with headers and an HTML body containing CSS and JavaScript links.

FYI: nếu để ý kĩ, cuộc tấn công vừa rồi Turbo Intruder đã gửi khoảng 500 requests/giây và hoàn thành gửi 10000 requests trong vòng 20 giây.

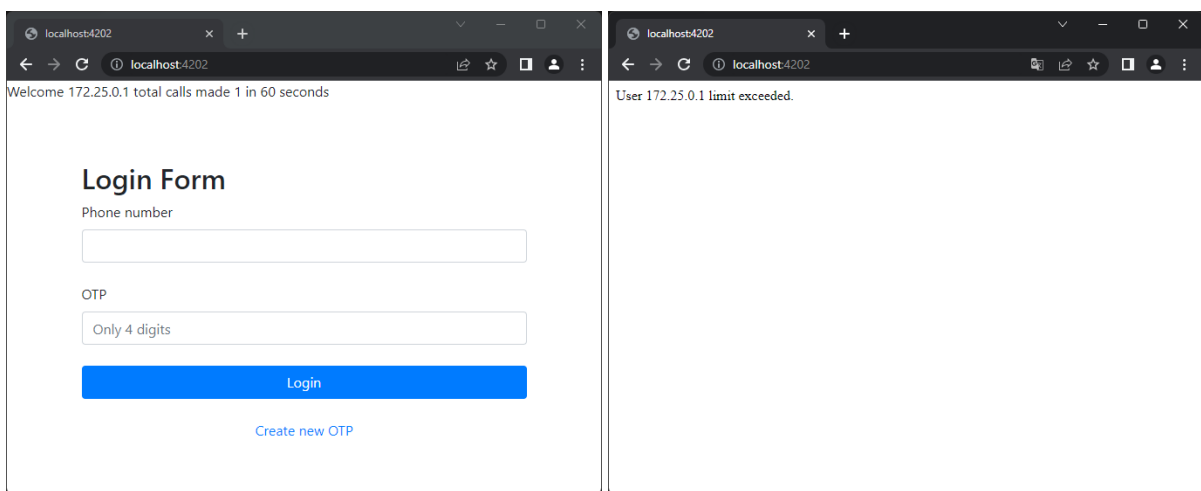
- **Bước 3:** Tới đây, chỉ cần đăng nhập với OTP vừa tìm được là lấy được flag.



LEVEL 2:

CÁCH HOẠT ĐỘNG CỦA ỨNG DỤNG:

- Về cơ bản, ứng dụng cũng có các chức năng giống level 1.
- Tuy nhiên, có thêm một tính năng mới là rate-limit, nôm na là chỉ cho phép 1 IP trong một thời gian nhất định được gửi một lượng requests, nếu vượt sẽ bị ban như hình.



- Trong folder `web/`, ngoài 3 file ở level1, đã xuất hiện thêm file mới: `rate_limit.php`
- File này được dùng để triển khai một rate-limit từ PHP và Redis. (Nguồn: [@Digital Ocean - ow To Implement PHP Rate Limiting with Redis on Ubuntu 20.04](#))
- Cùng phân tích source code của `rate_limit.php`.

```
7 $max_calls_limit = 100;  
8 $time_period     = 60;  
9 $total_user_calls = 0;
```

- **Line 7 → 8**, thiết lập giới hạn số lượt truy cập tối đa `$max_calls_limit` (100 cú requests) đến web-server trong khoảng thời gian `$time_period` (60 giây)
- **Line 9**, khởi tạo giá trị ban đầu `$total_user_calls` – đây là biến dùng để lưu trữ và theo dõi số lượt truy cập vào web-server.



```
11 if (!empty($_SERVER['HTTP_CLIENT_IP'])) {  
12     $user_ip_address = $_SERVER['HTTP_CLIENT_IP'];  
13 } elseif (!empty($_SERVER['HTTP_X_FORWARDED_FOR'])) {  
14     $user_ip_address = $_SERVER['HTTP_X_FORWARDED_FOR'];  
15 } else {  
16     $user_ip_address = $_SERVER['REMOTE_ADDR'];  
17 }
```

- **Line 11 → 17**, xác định IP của user hiện tại từ các trường của HTTP theo thứ tự sau:
 - > `$_SERVER['HTTP_CLIENT_IP']`, nếu có request tồn tại Header Client-IP, thì web-server sẽ lấy IP từ giá trị của Header đó.
 - > `$_SERVER['HTTP_X_FORWARDED_FOR']`, nếu có request tồn tại Header X-Forwarded-For, thì web-server sẽ lấy IP từ giá trị của Header đó.
 - > Nếu không có 2 Header trên, thì web-server sẽ lấy IP từ Source IP nằm trong TCP packet.
- Tại sao việc lấy IP của user lại phức tạp như vậy?
- Hiện tại (hay trong đa số thực tiễn) ta không trực tiếp truy cập vào web-server mà phải thông qua một con Proxy. Các Proxy này sẽ được cấu hình để tự động thêm các Header Client-IP và X-Forwarded-For để lưu chuyển IP của người dùng cho web-server xử lý.





- Quan sát cấu hình của Proxy trong file `proxy/nginx.conf`.

```
nginx.conf X
level2 > proxy > nginx.conf
1  server {
2      # Server name and port
3      listen 80;
4      server_name _;
5
6      location / {
7          proxy_pass http://web;
8          proxy_set_header Host $host;
9          proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
10     }
11 }
```

- **Line 8 và 9**, cấu hình cho nginx tự động thêm Header X-Forwarded-For với giá trị là ip của Client.

```
19  if (!$redis->exists($user_ip_address)) {
20
21      $redis->set($user_ip_address, 1);
22      $redis->expire($user_ip_address, $time_period);
23      $total_user_calls = 1;
24  } else {
25      $redis->INCR($user_ip_address);
26      $total_user_calls = $redis->get($user_ip_address);
27      if ($total_user_calls > $max_calls_limit) {
28          echo "User $user_ip_address limit exceeded.";
29          exit();
30      }
31  }
```

- **Line 19 → 30**, đây là flow xác định và theo dõi số lượt truy cập vào web-server của user, bao gồm tạo mới hoặc tăng giá trị đếm số lượt truy cập trong Redis và kiểm tra đã vượt giới hạn hay chưa.

CHECKPOINT:

- Các tính năng cũ vẫn được giữ nguyên.
- Implement rate-limit trên web-server.
- Rate-limit dựa trên IP, giới hạn 100 cú requests trong 60s cho mỗi IP.
- IP được lấy từ header 2 Header, Client-IP và X-Forwarded-For.



Ý TƯỞNG / GIẢ THUYẾT:

- Vẫn có thể bruteforce được nhưng tỉ lệ là 1500/10000 (15%).
- Các Header là một untrusted data cộng với việc rate-limit dựa trên IP → mà IP dựa vào Header → IP là một Untrusted Data (không check IP nhận từ các Headers có hợp lệ hay không)
- Ý tưởng: cải tiến script ở Level 1, cứ gửi 1 requests sẽ thay đổi IP một lần dựa vào việc gán giá trị cho Header Client-IP.

TIẾN HÀNH KHAI THÁC:

- Script python được cải tiến: @e3c655
- Ngoài ra mọi người cũng có thể sử dụng Turbo Intruder, bằng cách modify lại handler để xử lý thêm phần Header.



LEVEL 3:

CÁCH HOẠT ĐỘNG CỦA ỨNG DỤNG:

- Ứng dụng không có gì mới, các tính năng vẫn như cũ.
- Triển khai rate-limit trên Proxy thay vì Web-Server như level 2.
- Đọc `proxy/nginx.conf` để xem anh lập trình viên đã cấu hình rate-limit như thế nào.

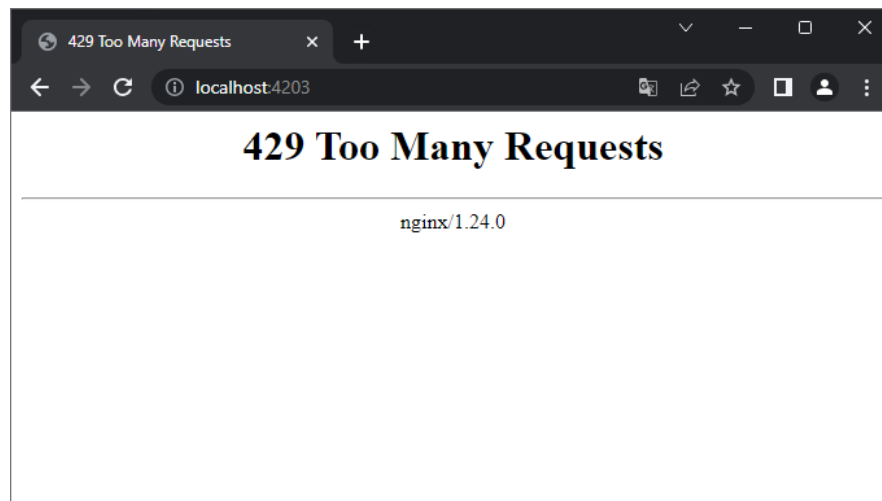
```
nginx.conf X
level3 > proxy > nginx.conf
1  # Define the rate limit zone
2  limit_req_zone $binary_remote_addr zone=one:10m rate=15r/m;
3
4  server {
5      # Server name and port
6      listen 80;
7      server_name _;
8
9      # Rate Limit configuration
10     limit_req zone=one burst=10 nodelay;
11     limit_req_status 429;
12
13     location / {
14         proxy_pass http://web;
15         proxy_set_header Host $host;
16         proxy_set_header X-Real-IP $remote_addr;
17         proxy_set_header X-Forward-For $proxy_add_x_forwarded_for;
18     }
19 }
```

- > **Line 1:** Định nghĩa zone `one` để theo dõi số lượng request từ mỗi địa chỉ IP.
 - o Kích thước của zone là 10 megabytes: lưu trữ thông tin về các requests mà IP đó gửi.
 - o Giới hạn là 15 requests mỗi phút
- > **Line 10 - 11:** Cấu hình thêm về giới hạn zone `one`
 - o Trước khi áp dụng giới hạn: Cho phép gửi tối đa 10 requests.
 - o Vượt quá giới hạn: sẽ không bị timeout (nodelay)
 - o Vượt quá giới hạn: Server trả về status code 429.

CHECKPOINT:



- Về cơ bản, khi truy cập vào Web-Server, các traffic phải đi qua Proxy.
- Và khi thực hiện gửi nhiều request (vượt quá giới hạn) sẽ bị Proxy trả về status code 429.



Ý TƯỞNG / GIẢ THUYẾT:

Mình có 2 giả thuyết chính:

- Với rate-limit này liệu ta vẫn có thể thực hiện bruteforce? (1)
- Còn có con đường nào khác để vào Web-Server mà không đi qua Proxy hay không? (2)

TIẾN HÀNH KHAI THÁC:

(1) Câu trả lời là có, nhưng tỉ lệ khá thấp.

(2) Làm sao để biết được con Web-Server này có mở một cổng khác ra ngoài internet không ?

- Nếu là môi trường thực tiễn, mọi người có thể thực hiện scan ports để tìm kiếm.



- Nhưng đây là môi trường lab và ta được cung cấp source code, nên ta hoàn toàn có thể check được.
- Nhìn vào `docker-compose.yml`

```
19 web:
20   container_name: 2FA_level3_web
21   depends_on:
22     - database
23   ports:
24     - "4204:80"
25   build: ./web
26   environment:
27     - MYSQL_HOSTNAME=database
28     - MYSQL_DATABASE=myDB
29     - MYSQL_USER=db_user
30     - MYSQL_PASSWORD=db_password
31
32 proxy:
33   container_name: 2FA_level3_proxy
34   ports:
35     - "4203:80"
36   image: nginx:stable
37   volumes:
38     - ./proxy/nginx.conf:/etc/nginx/conf.d/default.conf
39   restart: always
```

- > **Line 24:** đây là cấu hình để mapping port 80 của container web-server với port 4204 của máy host.
- > Điều này có nghĩa, là ta hoàn toàn có thể truy cập vào web-server thông qua port 4204.

A screenshot of a web browser window. The address bar shows "localhost:4204". The page content is a login form with the title "Login Form". It has two input fields: "Phone number" and "OTP". The "OTP" field has a placeholder text "Only 4 digits". Below the input fields is a blue "Login" button. At the bottom of the form, there is a link "Create new OTP" in blue text.

localhost:4204

Login Form

Phone number

OTP

Login

[Create new OTP](#)

- Tới đây, có thể sử dụng lại **script** và **Burpsuite** ở level 1.