**Lab: SameSite Lax bypass via method override**

PRACTITIONER

This lab's change email function is vulnerable to CSRF. To solve the lab, perform a CSRF attack that changes the victim's email address. You should use the provided exploit server to host your attack.

You can log in to your own account using the following credentials: wiener:peter

**Note**

The default SameSite restrictions differ between browsers. As the victim uses Chrome, we recommend also using Chrome (or Burp's built-in Chromium browser) to test your exploit.

**Hint**

You cannot register an email address that is already taken by another user. If you change your own email address while testing your exploit, make sure you use a different email address for the final exploit you deliver to the victim.

**Solution**

**Study the change email function**

1. In Burp's browser, log in to your own account and change your email address.
2. In Burp, go to the **Proxy > HTTP history** tab.
3. Study the POST /my-account/change-email request and notice that this doesn't contain any unpredictable tokens, so may be vulnerable to CSRF if you can bypass the SameSite cookie restrictions.
4. Look at the response to your POST /login request. Notice that the website doesn't explicitly specify any SameSite restrictions when setting session cookies. As a result, the browser will use the default Lax restriction level.
5. Recognize that this means the session cookie will be sent in cross-site GET requests, as long as they involve a top-level navigation.

**Bypass the SameSite restrictions**

1. Send the POST /my-account/change-email request to Burp Repeater.
2. In Burp Repeater, right-click on the request and select **Change request method**. Burp automatically generates an equivalent GET request.
3. Send the request. Observe that the endpoint only allows POST requests.
4. Try overriding the method by adding the \_method parameter to the query string:

GET /my-account/change-email?email=foo%40web-security-academy.net&\_method=POST HTTP/1.1

1. Send the request. Observe that this seems to have been accepted by the server.
2. In the browser, go to your account page and confirm that your email address has changed.

**Craft an exploit**

1. In the browser, go to the exploit server.
2. In the **Body** section, create an HTML/JavaScript payload that induces the viewer's browser to issue the malicious GET request. Remember that this must cause a top-level navigation in order for the session cookie to be included. The following is one possible approach:

<script>

document.location = "https://YOUR-LAB-ID.web-security-academy.net/my-account/change-email?email=pwned@web-security-academy.net&\_method=POST";

</script>

1. Store and view the exploit yourself. Confirm that this has successfully changed your email address on the target site.
2. Change the email address in your exploit so that it doesn't match your own.
3. Deliver the exploit to the victim to solve the lab.

Đây là lời giảng từ 2 video lab :

Video 1 :

00:00 hi everyone I hope you're doing well in

00:02 this web Security Academy lab we'll

00:04 deliver a csrf attack to our victim to

00:06 change their email address

00:08 but from the lab description we know

00:10 that our victim is using a chrome-based

00:11 browser and the trick here is that

00:13 Chrome will implicitly set the same site

00:15 attribute of our victim's session cookie

00:18 to lacks if our victim has been logged

00:20 in for more than two minutes which is

00:22 the case in this lab which means that

00:24 Chrome won't be sending the session

00:26 cookie in a post request so to solve the

00:29 lab we'll have to find a way to turn

00:30 that post request into a get request to

00:33 ensure our victim's session cookie gets

00:34 sent along with the change email request

00:37 let's go to the lab and let's go to my

00:39 account we know that we can log in with

00:41 the username wiener password Peter let's

00:43 log in

00:44 we can see our current email is weiner

00:46 at normaluser.net let's try and change

00:48 this to weiner at abnormal user

00:51 .net hit enter

00:53 and now let's switch to burp and let's

00:56 go to proxy and HTTP history and let's

00:59 look at the post my account change email

01:01 request here there's no csrf cookie set

01:04 here there's no csrf token set in the

01:06 body which means this post request is

01:09 vulnerable to a csrf attack

01:11 now let's look at the cookie so let's go

01:13 to the Post login request here

01:15 and we can see that the session cookie

01:17 is being set there's an expiry date set

01:19 it's set to secure so https only and

01:22 it's only except it's not accessible

01:24 from JavaScript but what's not set is a

01:27 same-site attribute which means that for

01:29 a chrome based browser it will be set to

01:32 lacks automatically after two minutes so

01:35 let's verify that in the developer

01:37 console and then hit F12 to open the

01:40 developer console I'm using the cookie

01:42 editor extension it's quite handy to see

01:44 all the cookie attributes so for the

01:46 session cookie we can see here that the

01:48 same site attribute is not set which

01:50 means it'll be automatically set to

01:52 acts after two minutes if you don't have

01:55 this extension I'll link to it in the

01:56 description but if you don't have it

01:57 right now you can also view the cookie

01:59 attributes under application and then

02:01 under cookies here to the left

02:03 and we can see here the same site

02:05 attribute is not set back to burp let's

02:08 go to the post my account change email

02:10 request here send it to repeater

02:12 and then I'm going to generate the csrf

02:14 POC and to go to engagement tools and

02:17 generate csrfboc

02:19 if you don't have the professional

02:20 Edition don't worry I'll include a link

02:23 to this piece of code in the description

02:25 go to options and make sure that include

02:27 Auto submit script is checked and let's

02:30 copy this

02:31 Vector lab I'm going to close this

02:33 developer console and go to the exploit

02:36 server I'm going to paste our payload

02:37 here

02:38 I want to try and deliver so I'm going

02:40 to change the email first to

02:41 Wienerschnitzel and I want to try and

02:43 deliver the exploit to our victim like

02:45 this with the method set to post because

02:48 of this tidbit right here where Chrome

02:50 will make an exception for cookie set

02:53 without the same site attribute less

02:55 than two minutes ago such cookies will

02:57 also be sent with non-item potent post

02:59 top level cross-site requests so if our

03:02 victim has been logged in for less than

03:04 two minutes this should work so let's

03:05 deliver the Expo to the victim

03:07 and we don't get the message that the

03:09 lab hasn't solved successfully because

03:11 the victim has been logged in for more

03:13 than two minutes we can verify this

03:14 ourselves because if we attempt to view

03:16 the exploit we'll get the login screen

03:19 and if we log in again cleaner Peter

03:23 we see that our email hasn't been

03:25 changed to Wienerschnitzel so I'm going

03:27 to change the email again to Weiner

03:29 Schnitzel the reason I'm doing this is

03:31 because I want to re send a new post

03:33 request that we can use in repeater just

03:35 to re because we've refreshed our

03:36 session Cookie by logging in again go

03:38 back to burp close this go to proxy and

03:42 we have our latest post request here to

03:44 change our email let's send that to

03:45 repeater what I want to try now is

03:47 instead of sending it as a post request

03:49 let's see if we can set it as a send it

03:51 as a get request by changing the request

03:53 method here

03:54 and see if that works because the cookie

03:57 would be sent in a get request even with

04:00 same site set to Lacks

04:02 but what we get back is Method not

04:05 allowed

04:06 but there's another thing we can try and

04:07 that's something called method spoofing

04:09 where certain web Frameworks like the

04:11 laravel PHP framework because HTML forms

04:15 don't support certain actions like put

04:17 patch and delete these Frameworks allow

04:19 you to overwrite the request method

04:21 Behavior through a hidden input

04:23 parameter in this case underscore method

04:26 so in this case in this example we're

04:28 still sending the request as a the form

04:31 as a post request but we're telling the

04:34 web framework to through this hidden

04:36 parameter to interpret the request as a

04:38 put request and the way we could

04:40 leverage this for our exploit is we can

04:43 send our form as a get request but then

04:45 through this hidden input parameter we

04:48 can tell the web framework to interpret

04:50 it as a post request and that would do

04:52 two things for us because we send it as

04:55 a get request first that would allow us

04:57 to get around the cookie restriction

04:58 because even if the same site equals

05:02 lacks restriction is set to the victim

05:05 session cookie that doesn't count for a

05:07 get request so the session cookie of the

05:10 victim will still be sent and then

05:11 through this input parameter we tell the

05:13 underlying web framework to interpret it

05:15 as a post request and that might allow

05:17 us to get around this method not allowed

05:20 check that is happening here so let's

05:22 try that out

05:23 so I'm going to go back to the laravel

05:24 documentation here and copy this line

05:26 let's hit an input then I'm going to go

05:29 to our lab and go to the exploit server

05:32 and I'm going to change the method to

05:34 get

05:36 and let's change the email to evil

05:42 and then above that we're going to paste

05:44 our payload so input type equals hidden

05:47 and the name is underscore method and

05:51 the value will be post

05:53 and let's send this exploit to our

05:56 victim

05:58 and as you can see now we've

05:59 successfully solved the lab thank you

06:01 for watching and I hope this was helpful

06:03 to you

Video 2 :

00:00 okay so Hello friends

00:03 but have you here

00:07 uh let's let's solve this simple site

00:11 last bypass VM met override

00:14 Laboratory

00:16 and this laboratory must change the

00:18 email function which is vulnerable we

00:21 must create our exploit sending to the

00:25 victim by using the exploit server of

00:28 uh the academy and

00:32 uh overriding the method with this

00:36 exploit to the victim when the victim

00:38 will click on our button he will change

00:42 the email address without known which is

00:44 so nasty uh so okay

00:48 let's check

00:50 the content of the lab the slaps change

00:54 email function is vulnerable to

00:58 uh cross site request foraging you can

01:02 you can read this information you have a

01:05 lot of information about what is what

01:08 actually is going on and how to find the

01:10 cross

01:11 uh site we cross for a tree so

01:16 uh forgery so so let's

01:20 log into account

01:32 okay this is a default email address

01:36 let's change it to our and use

01:39 interception to get normal

01:42 requests okay here is the normal request

01:46 when we change the email sending to

01:49 repeater

01:52 sending again C

01:54 we have the same email with without

01:56 changing it but we

01:58 sending the normal request let's turn

02:02 100 to found

02:03 so

02:05 let's prepare our

02:09 let's generate our exploit by using the

02:13 include Auto submit button

02:16 generated and

02:19 using

02:22 using this on our exploit server going

02:26 there

02:31 here we must make little changes before

02:35 we continue

02:37 let's change the action form with cat

02:39 because

02:41 we need to cut

02:43 the head from the victim and

02:46 we already have a

02:48 P tablet and that prevent

02:52 email so let's change this to prevent

02:54 one because we don't have it and this

02:58 email is not register

03:00 so going

03:03 to

03:04 the next line from the action form and

03:08 change here

03:12 yes I'm already

03:15 change here the height of a right post

03:18 metal because we need we need to hide

03:21 this post

03:22 this overwrite method when we want to

03:25 send to the victims of store

03:28 the script and

03:32 how you see

03:34 the email is prevented if we check and

03:38 view the exploit we will change this

03:41 look

03:43 okay the victim click now the email is

03:46 pivoted one so

03:48 going back to the exploit server this is

03:52 a little testing and change to the

03:54 kubernet3

03:57 okay store

04:00 and

04:01 let's try to sending to the victim if

04:04 everything is okay

04:06 the email will be permanently and it

04:09 will solve the lab

04:13 who yeah voila it's normal to solve a

04:16 lot so that's it it's not so hard and

04:19 actually so nasty

04:22 um

04:23 okay friends bye have a nice day