1 Prepararation

1. Make sure you have Burp Suite installed, with its certificate installed as well; proxy setting in your browser is correct so that Burp Suite will serve as your proxy.

2 Background

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they’re currently authenticated. From attacker’s perspective, the key thing is to generate a valid malicious request for the victim to execute.

In this lab session, we consider a hypothetical example. Let’s suppose chase.com is designed to use GET requests to transfer money. Let’s also suppose that when you need to transfer $100 to Tim Andersen, the money transfer operation can be performed by submitting the following GET request:

GET https://www.chase.com/transfer.do?acct=Tim&amount=100 HTTP/1.1

But Jidong tells you, to transfer $100 to Tim Andersen, you should click on the following link:

https://www.chase.com/transfer.do?acct=Jidong&amount=10000

So what will happen (in the hypothetical example) if you click on this link?

3 Instructions - Part 1

Now that you know what the above link means, you probably wouldn’t click on it when Jidong asks you to do so. So Jidong needs to use some other tricks to lead you to send that same request to chase.com. There are two approaches he can try,

3.1 Approach 1

In approach #1, Jidong inserts the following code snippet into one html web page, and trick you to open that web page.

**Code Snippet #1 - Using A tag:**
Let’s figure out what code snippet #1 does. Go to w3schools: https://www.w3schools.com/html/, click 'Try it Yourself', and copy code snippet #1 into the body of that HTML file, and then click "Run". At this point, what will happen (in the hypothetical example) if you click on that "View Scarlett Johansson’s Nude Pictures"?

3.2 Approach 2

So in approach #1, the attack will be successful only if you think Scarlett Johansson is really hot and you want to see her nude pictures. What if you don’t think Scarlett Johansson is hot? Okay, so here Jidong comes up with his second approach: he inserts the following code snippet into one html web page, and trick you to open that web page.

Code Snippet #2 - Using IMG tag:
<img src="https://www.chase.com/transfer.do?acct=Jidong&amount=10000" width="0" height="0" border="0">

Let’s figure out what code snippet #2 does. Go to w3schools: https://www.w3schools.com/html/, click 'Try it Yourself', and copy code snippet #2 into the body of that HTML file, but DO NOT click "Run".

At this moment, open Burp Suite, make sure you are on the "Proxy"-"Intercept" page. Turn "Intercept" on. Now you can go back to your browser and click "Run" in the above webpage. Switch to Burp Suite, and you should see a request being sent to chase.com; if you intercept the response and click "Forward" (until you see the response), you should notice a status code "404 Not Found", this is understandable, as the request is a fake one. However, the beautiful part of this attack is, if you go back to your browser, on that w3school.com web page, you don’t see anything sent to you from chase.com. Do you????

3.3 Takeaways

Approach #2 is stealthier than approach #1. The reason is that "width='0' height='0' border='0'" part tells your browser, this is a 0x0 image. Therefore, nothing would be displayed.

4 Instructions - Part 2

What if chase.com use POST, rather than GET requests to process money transfer? In this case, it’s possible they do not take parameters in the URL, but rather, they store parameters in the POST request body. Now the request to send $100 to Tim Andersen looks like this:

POST https://www.chase.com/transfer.do HTTP/1.1
acct=Tim&amount=100

Such a request cannot be delivered using standard A tags or IMG tags, but can be delivered using a FORM tag:
Code Snippet #3 - Using FORM tag:
<form action="https://www.chase.com/transfer.do" method="POST">
<input type="hidden" name='acct' value='Jidong'/>
<input type='hidden' name='amount' value='10000'/>
<input type='submit' value='View Scarlett Johansson’s Nude Pictures'/></form>

or

Code Snippet #4 - Using FORM tag plus a Javascript
<body onload='document.forms[0].submit()'>
<form action="https://www.chase.com/transfer.do" method="POST">
<input type='hidden' name='acct' value='Jidong'/>
<input type='hidden' name='amount' value='10000'/>
</form>

Copy code snippet #3 into that w3schools page. Switch to Burp Suite, turn "Intercept" on. Switch back to that w3schools page and click "Run", click "View Scarlett Johansson’s Nude Pictures". You should then notice (in Burp Suite) the post request being sent to chase.com. Turn 'Intercept' off.

Copy code snippet #4 into that w3schools page. Switch to Burp Suite, turn "Intercept" on. Switch back to that w3schools page and click 'Run'. You should then notice (in Burp Suite) the post request being sent to chase.com.

4.1 Takeaways

Code snippet #4 is slightly better than code snippet #3, as the Javascript code will submit the form automatically, thus the victim doesn’t need to click on any other button to submit the form.

5 Reference

OWASP: Cross-Site Request Forgery (CSRF) [https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)]]