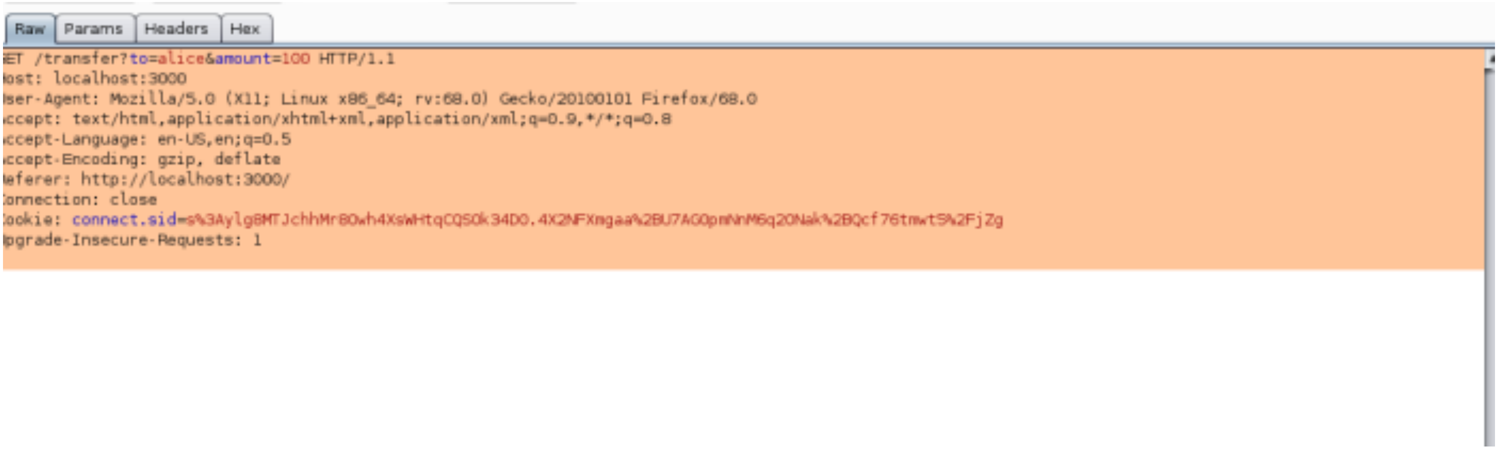
# *CSRF*

Cross Site Request Forgery, known as CSRF occurs when a user visits a page on a site, that performs an action on a different site. For instance, let's say a user clicks a link to a website created by a hacker, on the website would be an html tag such as <img src="https://vulnerable-website.com/email/change?email=pwned@evil-user.net"> which would change the account email on the vulnerable website to "pwned@evil-user.net". CSRF works because it's the victim making the request not the site, so all the site sees is a normal user making a normal request.

This opens the door, to the user's account being fully compromised through the use of a password reset for example. The severity of this cannot be overstated, as it allows an attacker to potentially gain personal information about a user, such as credit card details in an extreme case.



```
ET /transfer?to=alice&amount=100 HTTP/1.1
ost: localhost:3000
ser-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
ccept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
ccept-Language: en-US,en;q=0.5
ccept-Encoding: gzip, deflate
eferer: http://localhost:3000/
onnection: close
ookie: connect.sid=s%3Aylg8MTJchhMr80wh4XsWHtqCQS0k34D0.4X2NFXmgaa%2BU7AG0pmNnM6q20Nak%2BQcf76tmwtS%2FjZg
pgrade-Insecure-Requests: 1
```

This is looking good, parameters we can customize and a session cookie that is automatically set. Everything seems vulnerable to CSRF.

AUTOMATIC EXPLOIT

Once again, there is a nice automated scanner, which tests if a site  is vulnerable to CSRF. this tool is known as xsrfprobe and can be  install via pip using `pip3 install xsrfprobe`. This will only work using python 3(I mean come on it's 2020 you should be using python 3 anyway).
The syntax for the command isĀ `xsrfprobe -u <url>/<endpoint>`. Let's run this against our vulnerable site.