Base of JavaScript for Xss

By Chaos

فهرست

BASICS of JAVACSRIPT for XSS

در این آموزش ، ما قصد داریم در مورد اصول اولیه JavaScript (JS) برای این سری از XSS آینده برای درک و بهره برداری از XSS صحبت کنیم ، بهتر از اینست که صرفا یک سری payload های آماده رو بررسی کنیم!

مقدمه

JavaScript یک زبان برنامه نویسی سمت کلاینت (Frontend) برای وب است که در HTML برای تغییر رفتار/عملکرد یک صفحه وب IE استفاده می شود. JavaScript می تواند برای تغییر سبک (CSS) و مدل شیء اسناد (DOM) یک صفحه وب استفاده شود و ممکن است یا ممکن است به تعامل/ورودی کاربر بستگی نداشته باشد.

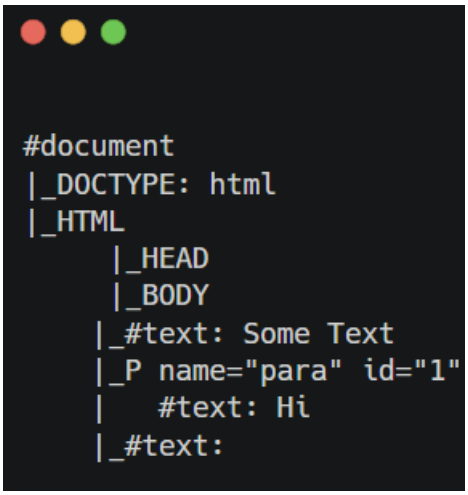
DOM چیست؟

Document Object Model به عبارت ساده تر ، هنگامی که یک صفحه وب بارگذاری می شود یک درخت DOM (ساختار داده) ایجاد می شود که شامل تمام گره های HTML از جمله عناصر/نام های برچسب ، ویژگی های برچسب ها ، شناسه برچسب ، نام برچسب ، متن/محتوای HTML در یک عنصر HTML یکی دیگر است. DOM شامل ساختار کل صفحه وب است.

یک صفحه HTML5 ساده مانند این خواهد بود:

```
<!doctype html><html><body>Some Text<p name="para" id="1">Hi</p></body></html>
```

DOM به این شکل است:



راه های مختلف برای اجرای JS:

- در داخل <script> ... </script>
- بارگیری فایل JS خارجی با ویژگی 'SRC'
- با استفاده از Event handlers
- کنسول مرورگر با استفاده از F12

JavaScript Syntax

کامنت ها

- کامنت در یک خط : //This is a Comment
- نوع دیگری از امنت در یک خط : <!--This is commented-->
- کامنت در چندخط :

```
/*Multi
line
comment*/
```

- کامنت تک خطی کمتر شناخته شده : --> This is also a Comment

اعلامیه داده ها و متغیرها در JS

(DataTypes & Variables declaration)

نحو کمی شبیه به C یعنی است. هر خط با نیمه رنگ ها به پایان می رسد (وقتی اظهارات در خطوط مختلف نوشته شده اند) و غیره. به جز این که ما نیازی به اعلام انواع متغیرها برای مثال نداریم:

```
var a=10,b=null,c="String",d=false,e={A:1,B:"String2",C:[1,2]},f=[1,2,3,"String3"],g; //Explained Below
```

a یک نوع داده عدد صحیح است

b یک شیء خالی است

c یک رشته است

d نوع داده بولی (درست/نادرست) است

e یک شی جاوا اسکریپت است

ما همچنین می توانیم در خطوط مختلف یکسان را اعلام کنیم:

```
var a=10;var b=-20;
```

نحو اساسی در JavaScript برای اعلام اشیاء ، از آنجا که اشیاء JavaScript در بریس ها اعلام می شوند زیرا {} یک شیء خالی را برمی گرداند. یک شیء مجموعه ای از چندین ویژگی است ، ارزش یک ویژگی می تواند هر نوع داده باشد. چند مورد مثال را در زیر بررسی کنید:

```
e={propertyname:"value of property"}eg: {propertyname1:"value1",propertyname2:"value2"}
```

متغیرهای زیر را با استفاده از هر یک از روشهای ذکر شده در بالا امتحان کرده و تعریف کنید و سپس مقادیر آنها را بررسی کنید ، می توانید این کار را به سادگی در کنسول مرورگر اجرا کنید (F12 را روی Chrome/FF فشار دهید) ، کد زیر را paste و سپس در نام آنها تایپ کنید تا مقدار هر متغیر را یک به یک بررسی کنید.

```
var a=10,b=null,c="String",d=false,e={A:1,B:"String2",C:[1,2]},f=[1,2,3,"String3"],g;
```

یک نمونه HTML برای متغیرهای فوق:

```
<!doctype html><html><script>var a=10,b=null,c="String",d=false,
e={A:1,B:"String2",C:[1,2]},f=[1,2,3,"String3"],g;</script></html>
```


بگذارید نحوه دسترسی به اشیاء JavaScript را ببینیم ، اول از همه با استفاده از بیانیه زیر در کنسول JavaScript ، ما با استفاده از عبارت زیر ، یک شیء "OBJ" را که دارای دو ویژگی Prop1 و Prop2 است ، اعلام خواهیم کرد.

```
var obj={prop1:300,prop2:"String"};
```

به مقادیر این خصوصیات از 2 روش قابل دسترسی است

1. نماد نقطه

```
obj.prop1;
// would return the value of property prop1obj.prop2;
// would return the value of property prop2
```

2. براکت های square []

```
obj["prop1"];
// would return the value of property prop1 obj["prop2"];
// would return the value of property prop2
```


پنجره و سند در JavaScript

(Window and Document Object)

Window اصلی ترین شیء global در JavaScript است که به شیء "Window" دسترسی پیدا می کند ، شامل تمام اطلاعات مربوط به پنجره باز شده (ارتفاع ، عرض ، نام پنجره ، DOM و غیره) و همچنین پنجره بازتر آن (در صورت وجود) است. Document ویژگی "Window" فقط شیء است اما از آنجا که شیء "Window" به صورت global است ، می توان آن را با "سند" یا حتی به عنوان "Window.document" به آن دسترسی پیدا کرد.

بنابراین اساساً ، DOM (مدل Object Document) تمام اشیاء داخل "Window" را بارگیری می کند و سپس شیء "سند" در داخل شیء پنجره بارگیری می شود. برخی از خصوصیات مهم اشیاء "Window" و "Document" عبارتند از:

1. Window.Location ، که شامل تمام اطلاعات مربوط به پنجره بارگذاری شده فعلی مانند URL ، نام میزبان ، پورت ، پروتکل ، مسیر و غیره است.

```
window.location.host : "www.host.com"window.location.hostname  : "www.hostname.com"window.location.href
: "https://www.hostname.com/path/to/file.php"window.location.origin  :
"https://www.hostname.com"window.location.pathname  :"/path/to/file.php"window.location.port
:""window.location.protocol  : "https:"
```

تغییر خصوصیات شیء مکان ، صفحه را به عنوان مثال مکان هدایت می کند. "href = "https://google.com ؛ به https://google.com هدایت می شود.

2. Window.Opener ، مرجع را به پنجره والد باز می گرداند که پنجره فعلی را با استفاده از روش Windows.Open() باز کرده است به عنوان مثال:

```
window.open( URL, name); //Opens a Window of name and URL , if url is empty string it returns about:blank
page.
mywin=window.open("", "MyWindow" );//opens an about:blank window with name "MyWindow"
mywin.document.write("This is Child Window");
mywin.opener.document.write("The Parent Window");//We Could Change content of the Parent
Windowmywin.opener.document.write("The Parent Window");//We Could Change content of the Parent Window
```

3. windows.location.hash یا location.hash - این بخشی از قطعه Uri IE را برمی گرداند. هر آنچه در URL پس از "#" (نماد هش) نوشته شده است ، به عنوان مثال url زیر

https://www.securityidiots.com/#blablalabla

در این مورد تابع ما (location.hash) خروجی #blablalabla و تابع location.hash.slice خروجی BLABLALABLA IE را برمی گرداند که این کاراکتر 1 را از رشته ، که # است ، برش می دهد.

این قسمت پس از هش در URL برای استفاده از سمت client است و از این رو توسط زبان های سمت سرور قابل دسترسی نیست ، به همین دلیل در دور زدن WAF سمت سرور بسیار مفید است.

4. Window.Location.Search - این ویژگی رشته پرس و جو یا پارامتر "دریافت" را برمی گرداند. به عنوان مثال:

```
https://www.securityidiots.com/?xyz=1&abc=zen
"location.search;" would return "?xyz=1&abc=zen"
```

Document.Domain - از این ویژگی برای بازگشت hostname محل اجرای JavaScript استفاده می شود. مشابه مکان. hostname این امر عمدتاً برای تأیید اجرای XSS در حوزه سمت راست استفاده می شود.

Document.Cookie - این ویژگی برای به دست آوردن همه کوکی ها به عنوان یک رشته استفاده می شود ، اما اگر یک پرچم "httponly" در کوکی ها وجود داشته باشد ، از طریق JavaScript قابل دسترسی نیست.

document.getElementsById ('123') - این یک روش (یک تابع) است که برای به دست آوردن همه عناصر (node ها) با 123 مقدار "شناسه" ویژگی ارائه شده در آرگومان استفاده می شود. ما حتی می توانیم به عنوان مثال node های انتخاب شده را اصلاح و حذف کنیم:

```
<!doctype html>
<html>
<p id="someID">Select Me</p>
<script>
var selected=document.getElementById("someID");
/*would store the <p id="someID">Select Me</p> node in selected variables*/
</script>
</html>
```

بسیاری از روشهای مشابه دیگر نیز برای دستکاری DOM مانند:

```
document.getElementsByName('Name');
document.getElementsByTagName('TagName');
document.getElementsByClassName('ClassName');
```

با این وجود می توان چندین برچسب با همان مقدار نام برچسب ها ، ویژگی های نام و کلاس وجود داشت ، در این حالت ، مجموعه ای از گره ها بازگردانده می شوند و از طریق شاخص های آرایه مانند selected[0] و selected[1] و غیره قابل دسترسی هستند.

5. document.innerhtml - از این روش برای نوشتن محتوای HTML در یک گره انتخاب شده استفاده می شود. به عنوان مثال:

```
<!doctype html>
<html>
<p id="someID">Select Me</p>
<script>
var selected=document.getElementById("someID");
/*would store the <p id="someID">Select Me</p> node in selected variable*/
selected.innerHTML="<p>Text blabla</p>";
/*you would see the content inside <p id="someID"> has now changed to <p>Text blabla</p>*/
</script>
</html>
</html>
```

هر ورودی کاربر به این موضوع بسیار خطرناک است حتی اگر فیلتر شود مقادیر < , > , " , ' فقط با استفاده از backslash (\) ، اعداد (0-9) ، الفبای کاربر مخرب می تواند JavaScript را اجرا کند.

6. document.write/document.writeln("HTML Content") - این برای بازنویسی (حذف HTML موجود) کل ساختار صفحه وب با آرگومان ارائه شده آن (محتوای HTML) استفاده می شود. به عنوان مثال:

```
<!doctype html>
<html>
<script>
document.write("Contents of The Page");
</script>
</html>
```

هر ورودی کاربر به این موضوع بسیار خطرناک است حتی اگر فقط با استفاده از < , > , " , ' و backslash (\) ، شماره ها (0-9) فیلتر شود ، الفبای کاربر مخرب می تواند JavaScript را اجرا کند

7. document.createElement('Element Name') - از روش CreateElement برای ایجاد یک عنصر HTML جدید با JavaScript استفاده می شود. به عنوان مثال اگر می خواهیم یک عنصر IMG با SRC = http://securityidiots.com/post_images/backxss.png ایجاد کنیم

```
var imgtag = document.createElement("img");//creates img element
imgtag.src="http://securityidiots.com/post_images/backxss.png";//adds attribute src to img element
document.body.appendChild(imgtag);//appends the created element to the body tag of the DOM
```

توابع در JavaScript

مانند زبان‌های دیگر، در JavaScript نیز توابعی وجود دارد که برای اجرای مجموعه‌ای از دستورات/فرمان‌ها استفاده می‌شود تا از تکرار کد جلوگیری شود.

تعریف توابع در JavaScript

یک تابع در JavaScript با استفاده از کلمه‌ی کلیدی function تعریف می‌شود، سپس نام تابع آمده و بعد از آن پرانتز () قرار می‌گیرد. درون پرانتز می‌توان نام آرگومان‌ها/پارامترها را که با کاما جدا شده‌اند، نوشت. توابع در JavaScript مشابه سایر زبان‌ها فراخوانی می‌شوند:

```
functionname([arg1, arg2]); // [] نشان‌دهنده اختیاری بودن آرگومان‌ها/پارامترها است
```

روش‌های تعریف و فراخوانی توابع:

```
//Defining
var x=function myFunction(a, b) {
    return a * b;
};

//Calling
z=x(30,40);// would return 1200 in variable z
```

ما همچنین می‌توانیم تابعی را مستقیماً بدون ذخیره کردن آن در متغیر فراخوانی کنیم:

```
//Defining
function myFunction(a, b) {
    return a * b;
};

//Calling
z=myFunction(30,40);
```

همچنین می‌توانیم توابعی را بدون نام تعریف کنیم که به آن‌ها "توابع بی‌نام" یا "Closures" گفته می‌شود:

```
//Defining
var x=function(arg){
  return arg*100;
};

//Calling
x(10);//would return 1000
```

و نیز می‌توان تعریف و فراخوانی را همزمان انجام داد:

```
var x=function(arg){
  return arg*100;
}(10);
// function would be called & would return 1000 in x variable
```

جمع کردن رشته‌ها درJavaScript

از آنجا که JavaScript زبان **loosely typed** است، می‌توان رشته‌ها را با رشته‌های دیگر و همچنین با انواع داده‌ای دیگر مانند عدد (صحیح یا اعشاری) یا بولی با استفاده از عملگر + با مضمون افزودن استفاده کرد.

```
var x="aaaaaa"+"bbbb"; //console.log(x) would return aaaaaabbbb
var x="aaa"+1234;//console.log(x) would return aaa1234
```

توابع مهم JavaScript مرتبط با XSS

console.log – برای نوشتن خروجی در کنسول توسعه‌دهنده مرورگر استفاده می‌شود که برای اهداف اشکال‌زدایی مفید است. ما می‌توانیم مقادیر متغیرها یا توابع را در کنسول ببینیم، تغییر دهیم یا بازنویسی کنیم.

```
<!doctype html>
<html>
<script>
console.log("Test!!");
</script>
</html>
```

مرورگر را باز کنید، کلید F12 یا Ctrl+Shift+J را فشار دهید و صفحه را بازخوانی کنید. پیغام "Test!!" را در کنسول خواهید دید.

دریافت مقادیر متغیرها در کنسول:

```
<!doctype html>
<html>
<script>
var a = 10, b = null, c = "String", d = false, e = {A:1, B:"String2", C:[1,2]}, f = [1,2,3,"String3"], g;
console.log("Value of a:");
console.log(a);
console.log("Value of b:");
console.log(b);
console.log("Value of c:");
console.log(c);
console.log("Value of d:");
console.log(d);
console.log("Value of e:");
console.log(e);
console.log("Value of f:");
console.log(f);
console.log("Value of g:");
console.log(g);
</script>
</html>
```

صفحه را باز کرده و کنسول مرورگر را بررسی کنید تا مقادیر متغیرها نمایش داده شوند.

alert(1); prompt(1); confirm(1)

- alert("") پنجره‌ای از نوع هشدار با متنی که به آن داده شده نمایش می‌دهد. هیچ مقداری بازنمی‌گرداند.
- prompt("") پنجره‌ای باز می‌کند تا ورودی از کاربر گرفته شود. مقدار وارد شده توسط کاربر را بازمی‌گرداند
- confirm("") یک پنجره تأیید باز می‌کند. اگر کاربر روی OK کلیک کند مقدار trueو در صورت Cancel مقدار false بازمی‌گرداند

```
var x = prompt("Enter Value:");
var conf = confirm("Are You Sure To blablabla?");
```

از دیدگاهXSS ، این توابع برای تست و اطمینان از اجرای JavaScript در هدف استفاده می‌شوند.

setTimeout() این تابع یک تابع دیگر را بعد از تعداد مشخصی میلی‌ثانیه اجرا می‌کند

setInterval() این تابع یک تابع دیگر را در بازه‌های زمانی مشخص، به صورت

به همین دلیل، توابع setTimeout و setInterval می‌توانند برای اجرای JavaScript مخرب) مانند (XSS استفاده شوند؛ مخصوصاً اگر ورودی کاربر به عنوان پارامتر اول داده شود.

```
setTimeout(function(){ alert("Hello"); }, 3000);
// will call anonymous function after 3 seconds
setInterval(alert(1),3000);
// will call alert(1) after every 3 seconds
```

var z = new Function('arg1','arg2','body') این روش یک شی Functionجدید ایجاد می‌کند که مشابه تعریف تابع با function است

ورودی کاربر در پارامتر سوم بسیار خطرناک است و می‌تواند به XSS منجر شود، اگر این تابع در صفحه فراخوانی شود.

eval("JavaScript code") این تابع رشته‌ای که به آن داده شده را به عنوان JavaScriptاجرا می‌کند.

ورودی کاربر به eval() بسیار خطرناک است.

```
var z=new Function('arg1','arg2','alert(1)')
z(1,2);//would cause alert(1)
eval('var x=1; alert(x);');
```

Event Handlers

به زبان ساده، Event Handlersویژگی‌های خاصی هستند که همراه با تگ‌های خاصی (و گاهی با همه تگ‌ها) استفاده می‌شوند تا هنگام وقوع یک رویداد خاص، یک قطعه کد اجرا شود. از آن‌ها برای گنجاندن جاوا اسکریپت به‌صورت درون‌خطی بدون استفاده از تگ <script>استفاده می‌شود، ولی فقط زمانی اجرا می‌شوند که آن رویداد خاص اتفاق بیفتد.

آن رویداد می‌تواند هر چیزی باشد مثل کلیک کردن، رفتن موس روی یک تگ، دوبار کلیک کردن، فشردن یک کلید، انتخاب یا عدم انتخاب یک عنصر، بارگذاری صفحه و غیره.

ساختار پایه:

```
<tagname someattribute1=value onSomeEvent="var x=10;alert(x);//javascript code here">
```

مثال ساده‌ای از رویداد هنگام کلیک:

ویژگی مربوط به کلیک onclickاست. مثال:

Click Me

زمانی که کسی روی لینک "Click Me" کلیک کند، رویداد onclickفعال شده و تابع جاوااسکریپت اجرا می‌شود نمایش پنجره alert و سپس هدایت صورت می‌گیرد.

برخی ازEvent Handler های رایج:

1. **onmouseover / onclick**
   - onmouseoverوقتی فعال می‌شود که موس روی یک عنصر قرار گیرد
   - onclickزمانی که روی عنصر کلیک شود

2. **onmousedown / onmouseup**

- onmousedownزمانی که کلیک انجام شود
- onmouseupزمانی که کلیک رها شود

3. **onfocus**
   - زمانی که تمرکز (فوکوس) روی یک فیلد ورودی قرار گیرد
4. **onblur**
   - وقتی فوکوس از یک فیلد برداشته شود
5. **onkeypress / onkeyup / onkeydown**
   - onkeypressوقتی کلید فشرده می‌شود
   - onkeydownوقتی کلید فشرده می‌شود و هنوز رها نشده
   - onkeyupوقتی کلید رها می‌شود
6. **onload**
   - زمانی که صفحه بارگذاری شود
7. **onunload**
   - وقتی صفحه ترک شود
8. **onerror**
   - وقتی در هنگام بارگذاری یک فایل، خطا رخ دهد

لیست بسیار بزرگی از Event Handler ها وجود دارد که در مرورگرها قابل استفاده است و نوشتن تمام آن‌ها ممکن نیست. ولی راهی وجود دارد برای دیدن همه‌ی آن‌ها: کافی‌ست کنسول توسعه‌دهنده مرورگر را باز کنید (Ctrl+Shift+J) و تایپ کنید on، سپس در صورت نیاز با زدن Ctrl+Space لیستی از Event Handler ها نمایش داده می‌شود. حالا می‌توانید هرکدام را انتخاب کرده و در گوگل جست‌وجو کنید تا بفهمید برای چه کاری و با کدام تگ‌ها قابل استفاده است.

همچنین لیستی از Event Handler ها توسط Dr.Mario (@0x6d6172696f) در این لینک ارائه شده:

http://pastebin.com/raw/WwcBmz5J

که در هنگام دور زدن فیلترهای WAF بسیار کاربردی است.

AJAX (Asynchronous Javascript And XML)

AJAXروشی است برای:

- به‌روزرسانی بخشی از یک صفحه وب بدون بارگذاری مجدد کل صفحه
- ارسال درخواست به سرور پس از بارگذاری صفحه
- دریافت اطلاعات از سرور پس از بارگذاری
- ارسال داده به سرور در پس‌زمینه

مراحل ساخت یک درخواست پایهAJAX

1. ایجاد یک شیء از نوع XMLHttpRequest
2. استفاده از متد onloadبرای تعریف تابعی که بعد از دریافت پاسخ اجرا شود
3. استفاده از متد open برای مشخص کردن نوع درخواست(GET/POST/...) ، و URL مورد نظر
4. استفاده از متد sendبرای ارسال درخواست

ساختار پایهAJAX

```
<script>
var xhrobj = new XMLHttpRequest(); // ایجاد شی XMLHttpRequest
xhrobj.onload = function() {
  if (this.status === 200) {
    // کاری که بعد از دریافت موفق پاسخ انجام می‌دهیم
  }
};
xhrobj.open("GET", "URL موردنظر یا فایل");
xhrobj.send();
</script>
```

SAME ORIGIN POLICY (SOP)

شما نمی‌توانید به راحتی پاسخ هر صفحه‌ای را بخوانید. اگر این امکان وجود داشت، هرکسی می‌توانست با AJAX به بانک یا ایمیل شما متصل شود و اطلاعات حساس را بخواند. بنابراین، **SOP** اجرا می‌شود و فقط اجازه خواندن پاسخ‌هایی را می‌دهد که از همان origin (مبدأ) آمده باشند.

دو صفحه زمانی "هم‌مبدأ (Same Origin)" در نظر گرفته می‌شوند که:

- **پروتکل** آن‌ها یکی باشد http ، https و...
- **پورت** مشخص شده یکی باشد
- **هاست** یکی باشد

مشکل SOP و معرفی CORS

گاهی نیاز داریم از origin متفاوت، پاسخ دریافت کنیم. در این مواقع، از **CORS** (Cross Origin Resource Sharing) استفاده می‌شود تا اجازه خواندن پاسخ‌ها از مبدأهایی که مجاز هستند، داده شود.

در درخواست CORS ، یک هدر Origin همراه با درخواست فرستاده می‌شود که نشان می‌دهد مبدأ درخواست AJAX کدام است. در پاسخ نیز هدر زیر دریافت می‌شود:

Access-Control-Allow-Origin: http://host

این یعنی فقط همان host می‌تواند پاسخ را بخواند.

نکته امنیتی :اگر مقدار این هدر روی * باشد (یعنی هر مبدأی مجاز است)، خطر وجود دارد. هرچند به‌صورت پیش‌فرض **کوکی‌ها ارسال نمی‌شوند**، اما اگر این هدر در پاسخ نیز باشد:

Access-Control-Allow-Credentials: true;

یعنی اجازه ارسال کوکی همراه درخواست نیز داده شده است.

اما مرورگرها اجازه نمی‌دهند که این دو هدر **با هم** به شکل زیر ارسال شوند:

Access-Control-Allow-Origin: *

Access-Control-Allow-Credentials: true

پس دوباره امن هستیم... البته اگر مقدار Origin به‌درستی اعتبارسنجی شود. در غیر این صورت، منجر به افشای اطلاعات حساس خواهد شد.

برای اطلاعات بیشتر درباره این موضوع:

- https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy
- http://www.geekboy.ninja/blog/exploiting-misconfigured-cors-cross-origin-resource-sharing/
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS

خواندن پاسخ سرور

در ادامه مثالی برای خواندن محتوای یک فایل readme.html از طریق AJAX آورده شده است:

```
<script>
var xhrobj = new XMLHttpRequest();
xhrobj.onload = function() {
  if (this.status === 200) {
    var resp = this.responseText; // محتوای پاسخ در responseText ذخیره می‌شود
    alert("Server Response: " + resp);
  }
};
xhrobj.open("GET", "/readme.html");
xhrobj.send();
</script>
```

در آینده، استفاده از AJAX زمانی مورد نیاز خواهد بود که بخواهیم مقادیری مانند nonce یا توکن‌های CSRF را از طریق XSS بدست آورده و حملات CSRF را از طرف دیگر کاربران انجام دهیم.