Content

[1 Synthesis 2](#_Toc28243717)

[1.1 Points to improve 2](#_Toc28243718)

[1.2 Synthesis table 2](#_Toc28243719)

[1.2.1 Identified vulnerabilities 2](#_Toc28243720)

[1.2.2 Action Plan 2](#_Toc28243721)

[2 Technical description 3](#_Toc28243722)

[2.1 Vulnerabilities 3](#_Toc28243723)

[2.1.1 Cross Site Request Forgery (CSRF) 3](#_Toc28243724)

[3 Annex 9](#_Toc28243725)

[3.1 Qualification of a vulnerability 9](#_Toc28243726)

[3.1.1 General risk level 9](#_Toc28243727)

[3.1.2 Perimeter 9](#_Toc28243728)

[3.1.3 Description 9](#_Toc28243729)

[3.1.4 Probability - Threat 10](#_Toc28243730)

[3.1.5 Probability – Vulnerability 11](#_Toc28243731)

[3.1.6 Probability – Exploitation likelihood 11](#_Toc28243732)

[3.1.7 Impact – Technical impact 12](#_Toc28243733)

[3.1.8 Impact – Business impact 13](#_Toc28243734)

1. Synthesis
   1. Points to improve

|  |  |
| --- | --- |
| Points for improvement | Comments |
| Cross Site Request Forgery (CSRF) | The environment is vulnerable to CSRF attacks when an attacker forces an authenticated user to perform legitimated actions but without knowing it. The user turns into the attack accomplice though ignoring it. |

* 1. Synthesis table
     1. Identified vulnerabilities

Summary of the vulnerabilities found during the audit

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ref. | Risk | Description | Vulnerability Summary | Perimeter Involved | Impacts | Exploit |
| **DEV-3** | **Moderate** | Cross Site Request Forgery (CSRF) | HTTP requests allowing to interact with the application are predictable (no random token). Thus, a malicious user could force a legitimate user to perform actions without knowing it (for example through a third-party site) | http://fr-opv.citroen.preprod.inetpsa.com/ | **Moderate** | **Possible** |

* + 1. Action Plan

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Ref. | Description | Recommendation | Perimeter Involved | Responsible | Priority | Correction Difficulty  Degree |
| **DEV-3** | Cross Site Request Forgery (CSRF) | LEXSI recommends to implement unique and unpredictable, anti-CSRF server-side tokens for verification whenever a sensitive action takes place | http://fr-opv.citroen.preprod.inetpsa.com/ | Development | **Medium term** | **Medium** |

1. Technical description
   1. Vulnerabilities

* + 1. Cross Site Request Forgery (CSRF)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Moderate** | | | | | |
| Title | Cross Site Request Forgery (CSRF) | | | | DEV- 3 |
| Perimeter | <http://fr-opv.citroen.preprod.inetpsa.com/> | | | | |
| Description | The HTTP requests allowing to interact with the application are predictable (no random token). Thus, a malicious user could force a legitimate user to perform actions without knowing it (for example via a third-party site). | | | | |
| Probability | Threat | | Vulnerability | | Exploit Probability |
| Condition | Developer | OWASP  category | A8 –Falsification of requests inter-sites (CSRF) | **Possible** |
| Technical ease of exploit |  | Discovery ease |  |
| Impact | Technical impact | | Business impact | | Global impact |
| Availability |  | Brand Image |  | **Moderate** |
| Integrity |  | Financial |  |
| Confidentiality |  | Privacy violation |  |
| Proof |  | Non-compliance |  |

|  |  |  |
| --- | --- | --- |
| Recommendations | | |
| Medium term | Responsible | Difficulty |
| LEXSI recommends to implement tokens anti-CSRF (unique and unpredictable) verified on server side at each sensitive action | Development |  |

 Description

Vulnerabilities of *Cross-Site Request Forgery*, also called CSRF ou XSRF, affect the Web applications using predictable http requests to perform operations that don’t require any secret from the user (OTP code, current password, etc.).

These attacks use the user as a trigger and he becomes accomplice without being aware of it. With this attack carried out by the user without his knowledge, a large number of authentication system are bypassed.

 State found

During the audit, it was verified that the HTTP requests submitted to the application server did not include any hazards, which therefore makes them predictable and therefore vulnerable to CSRF attacks. To illustrate this principle, let's first look at the following HTML code:

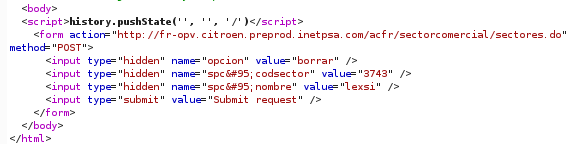


Figure 1 : HTML code allowing user deletion

This code is used to create an HTTP request whose role will be to delete the user « lexsi » within the *sectorcomercial* section. We have also taken care to create this user before in the corresponding section:



Figure 2 : Creation of the user lexsi

So, by getting a legitimate user for sending this request (via a Cross-Site Scripting vulnerability for example) we effectively manage to suppress the « lexsi » user:



Figure 3 : Post attack CSRF

 Risks

By inserting malicious HTML or JavaScript code on an external site, this code is interpreted by the user's browser and an attacker is able to force the execution of actions on the application.

In our case, this vulnerability allows:

* **To usurp the user identity** on the site ;
* **To remove the privileges of a user** on the application ;
* To damage **the brand image** of the service modifying the users’ data content

Recommendations

In order to be protected from the CSRF attacks, it is necessary that the operation cannot be predictable.

LEXSI recommends implementing a system of synchronized tokens that have to be generated in an unique and unpredictable way, and then verified on server side as described in the below diagram:

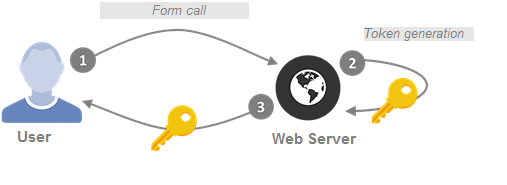


Figure 4: anti-CSRF token generation

The token generated must:

* Be **stored in the session**
* Have **a short expiration date** (10 minutes for example)
* Be **linked to an specific action**
* Be **unpredictable** : generated by means of a safe and long enough random ( 30 characters for example)

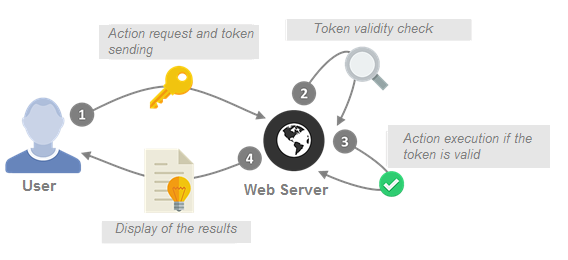


Figure 5: Verification of the anti-CSRF token

The provided token must :

* Be **identical** to the one stored in the session
* Not be already **expired**
* **Correspond to the action** for which it has been generated
* Be **removed** from the session in all cases (whether valid or not)

In addition, before changing a password, the old password must be provided.

Some additional considerations must be taken into account in order to secure

the resilience of the system:

* **HTTP X-FRAME-OPTIONS** **headers**: These headers are to be placed in each HTTP response sent by the server to prevent inclusion
* **Referer verification** : although it is possible for a user to modify his own HTTP Referer, it is not possible to do so during a CSRF attack

It is also possible to guard against these attacks by demanding the resolution of a CAPTCHA or by asking for the user's password before executing a request. However, unlike the synchronized token system, this operation will not be transparent for the user (he will have to inform the CAPTCHA or provide his password) and will reduce the ergonomics of the application.

 Exploit scenario

The diagram below illustrates the principle of the attack:

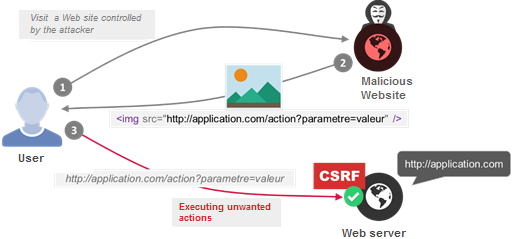


Figure 6: Operating exploit diagram of a CSRF vulnerability

1. A user is authenticated on the audited web application and in parallel visits another website which is controlled by an attacker
2. The attacker’s website returns an image (or more generally an HTML / JavaScript code) pointing to pages of the audited web application so as to force the sending of HTTP requests on the application targeted by the user’s browser
3. The user’s Internet browser attempts to load the resources (in this example an image) and automatically performs an HTTP request to the audited application, in a transparent way and with the privileges of the user’s account since the browser sends these requests with the user’s session cookie. Actions are therefore carried out without user’s knowledge, by usurping his identity and privileges.

If the audited application uses the POST method to perform an action, the attacker only has to replace the image with a form containing the required parameters. In order to be transparent to the user, this form can be present in an invisible *iframe* and automatically submit the form via JavaScript code.

For example, using the following code:

<iframe name=**"ifr"** id=**"ifr"** width=**"0"** height=**"0"** border=**"0"** frameborder=**"0"**></iframe>

<form action=**"http://application.com/action"** method=**"post"** id=**"formAction"** target=**"ifr"**>

<!— **Paramètres à envoyer** -->

<input type=**"hidden"** name=**"parametre1"** value=**"valeur1"** />

<input type=**"hidden"** name=**"parametre1"** value=**"valeur1"** />

<input type=**"submit"** name=**"Submit"** />

</form>

<script>

document.getElementById('formAction').submit();

</script>

 References

[www.owasp.org/index.php/Cross-Site\_Request\_Forgery\_(CSRF)](http://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF))

[www.owasp.org/index.php/Cross-Site\_Request\_Forgery\_(CSRF)\_Prevention\_Cheat\_Sheet](http://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet)

1. Annex
   1. Qualification of a vulnerability

Each identified vulnerability is qualified by our teams. The qualification is enriched with several criteria. We indicate here the correspondence of each of this criteria.

*Note dedicated to application penetration testing*

The metrics presented reflect the founding principles of the OWASP methodology for risk weighting. For more information it is possible to refer to this address: [*https://www.owasp.org/index.php/OWASP\_Risk\_Rating\_Methodology*](https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology).

* + 1. General risk level

The level of risk is calculated based on the probability and the impact.

Depending on the risk level calculated, five categories are defined by increasing criticality:

|  |  |
| --- | --- |
| Indicator | Risk level |
|  | Insignificant risk. Information indicated because opposite to the state of the art |
|  | Minor risk |
|  | Moderate risk |
|  | Major risk |
|  | Critical risk |

* + 1. Perimeter

The perimeter defines the scope or exploit zone of the reported vulnerability. It can correspond to a specific instance of the audit perimeter (e.g. an URL, an IP address) or to a set of components (e.g. Windows servers, Front Office).

* + 1. Description

The description briefly presents the vulnerability and the identified risks. For more details on the vulnerability, it is necessary to refer to the description of the identified vulnerability section.

* + 1. Probability - Threat

|  |  |  |
| --- | --- | --- |
| Field | Details | |
| **Condition** | Anonymous  user | The exploit is possible without requiring a user account |
| Authenticated user | The exploit requires a user account |
| Internal user | Exploit requires internal access to the IS and potentially a user account. |
| Service account | Exploit requires a service account |
| Developer | The exploit requires project-related knowledge (typically a developer who participated in the project or has access to the source code). |
| Administrator | Operation requires an administrative (technical or functional) account. |
| **Technical ease of exploit** |  | Theoretical vulnerability, not exploitable in the state at the time of the audit |
|  | Advanced technical skills required for the exploit (creation of exploit code) or requiring significant investment (calculation servers for example) |
|  | Advanced technical skills required for the exploit (mastery of the main tools and attack techniques). |
|  | Network, system, or development skills required for the exploit. |
|  | The exploit is achievable automatically using tools widely distributed on the Internet or with only a few required skills |

* + 1. Probability – Vulnerability

|  |  |  |
| --- | --- | --- |
| Field | Details | |
| **OWASP**  **Category** | Corresponds to the main category defined in the [*OWASP Top Ten*](https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)  to which the vulnerability is attached  Note : The vulnerabilities are not all correlated with the elements in this repository (e.g. logical flaw) | |
| **Ease of discovery** |  | Impossible or practically impossible |
|  | Difficult |
|  | Moderate |
|  | Easy |
|  | Automatic tools available to discover the vulnerability |

* + 1. Probability – Exploitation likelihood

The exploit probability is a weighting assigned by the auditor and depends on the vulnerability, its ease of exploit, its prevalence, and its accessibility on the audited perimeter.

|  |  |
| --- | --- |
| Weighting | Description |
| **Unlikely** | Almost zero probability of vulnerability exploitation |
| **​Improbable** | Likelihood of exploit : Improbable |
| **​Possible** | Likelihood of exploit : Possible |
| **​Probable** | Likelihood of exploit : Probable |
| **​Almost certain** | Likelihood of exploit : Almost certain |

* + 1. Impact – Technical impact

The technical impacts are divided into 4 categories:

* + **Availability** means reaching the service level of the application
  + **Integrity**  concerns the modification of the data handled by the application
  + **Confidentiality** comes into play when technical and or sensitive information is disclosed
  + **Proof** (or **Traceability**) is the ability to monitor the attacker, and in particular the ability to identify him precisely

|  |  |  |
| --- | --- | --- |
| Field | Details | |
| **Availability** |  | No impairment |
|  | Low disturbance (e.g. slight slowdown) |
|  | Moderate disturbance |
|  | Important disturbance |
|  | Total unavailability (e.g. server shutdown) |
| **Integrity** |  | No damage to the data handled by the application |
|  | Data not important or relating to a user that can be altered |
|  | Moderate portion of editable data, or limited number of users affected by the changes |
|  | A significant part of the data can be altered by the exploit |
|  | All data can be permanently altered during the exploit |
| **Confidentiality** |  | No impairment |
|  | Very low information leakage (information that is not very sensitive or associated with few users) |
|  | Moderate information leak |
|  | Leak of important information (sensitive documents, information about users, etc.) |
|  | All data can be recovered |
| **Proof / Traceability** |  | Not impacted |
|  | Potentially traceable attacker |
|  | Partially traceable attacker |
|  | Significant loss of attacker tracking |
|  | Complete loss of attacker tracking |

* + 1. Impact – Business impact

The business impacts are divided into 4 categories:

* + The **brand image** corresponds to the company’s reputation damage
  + The **financial impact** estimates the pecuniary damage suffered
  + The **violation of privacy** occurs when personal data of user is returned to unauthorized persons
  + **Non-compliance** is the default of the company following a vulnerability related to:
    - A standard (e.g. PCI-DSS or RGSv2)
    - An internal security policy (e.g. PSSI)
    - A legal obligation (e.g. CNIL)

(e.g.: personal data disclosed when a contract obligates the company to protect it)