

■ **Multiple Cross-Site Scripting vulnerabilities in Sage XRT Business Exchange**

■ **Security advisory**
07/06/2022

Mickaël Benassouli
Antoine Gicquel

Vulnerabilities description

Sage XRT Business Exchange

Sage XRT Business Exchange is a secure cloud-based platform to manage the entire banking operations of a company. It allows users to process payment chains, manage electronic signatures, and collaborate with their team anywhere, anytime.

The issues

Synacktiv has identified multiple vulnerabilities in Sage XRT Business Exchange that allows an attacker to execute JavaScript code in the context of other users' browsers. The attacker needs to be authenticated to reach the vulnerable features.

A vulnerability is present in the *Filters* and *Display model* features (*OnlineBanking > Web Monitoring > Settings > Filters / Display models*). The name of a filter or a display model is interpreted as HTML and can thus embed JavaScript code, which is executed when displayed. This is a stored XSS.

Another vulnerability is present in the *Notification* feature (*OnlineBanking > Configuration > Notifications and alerts > Alerts **). The name of an alert is interpreted as HTML when displayed, and can thus embed JavaScript code, which is executed when displayed. This is a stored XSS.

Finally, a vulnerability is present in the *File download* feature, accessible via */OnlineBanking/cgi/isapi.dll/DOWNLOADFRS*. When requesting to show the list of downloadable files, the contents of three form fields are embedded in the JavaScript code without prior sanitation. This is essentially a self-XSS.

Affected versions

The following versions are affected others version have not been tested:

- 12.4.302

Timeline

Date	Action
07/06/2022	Advisory sent to Sage.
20/06/2022	Attribution of the following CVE Number : CVE-2022-34323

Technical description and Proof-of-Concept

Vulnerabilities discovery

The application uses user-supplied data pulled from the database to build web pages that are displayed to other users as part of their normal browsing behavior. However, it does not perform an appropriate encoding before reusing that data. Therefore, an authenticated attacker can create records that will be stored and, once inserted in the final pages, will tamper with their normal content and behavior.

Stored XSS were identified on the following endpoints:

- *Filters and Display Models*

Sage XRT Business Exchange has a *Filters* feature, accessible via *OnlineBanking > Web Monitoring > Settings > Filters*. When injecting malicious HTML in the name of a filter, it gets interpreted in the *Filters* feature home page:

The screenshot shows the Sage XRT Business Exchange interface for creating a filter. The browser address bar shows a URL ending in ".em.cloud-by-sag". The page title is "sage XRT Business Exchange". The navigation menu includes "Accueil", "Paramétrage", and "Suivi des flux". The main content area contains instructions: "Pour créer un filtre, vous devez au préalable fournir les éléments qui permettront d'identifier e fois toutes les informations fournies, cliquez sur le bouton ' Enregistrer ' .". The form has two sections: "Identification" and "Définition du filtre". In the "Identification" section, the "Libellé" field contains the payload "<input/onfocus=alert(1)>". The "Type de filtre" dropdown is set to "Fichiers émis". The "Définition du filtre" section includes a list of criteria: "Application", "Entité", "Protocole", "Service", "Client", and "Montant". The "Montant" checkbox is checked, with input fields for "1" and "2". Below this, the "Application" section has a checked "Filtre par défaut" checkbox. The "Utilisation du filtre" section has two radio buttons: "Uniquement par celui qui l'a créé" (selected) and "Toute personne habilitée". A blue "Enregistrer" button is at the bottom.

Illustration 1: Malicious HTML injected in the *Libellé* (french for *Label*) field

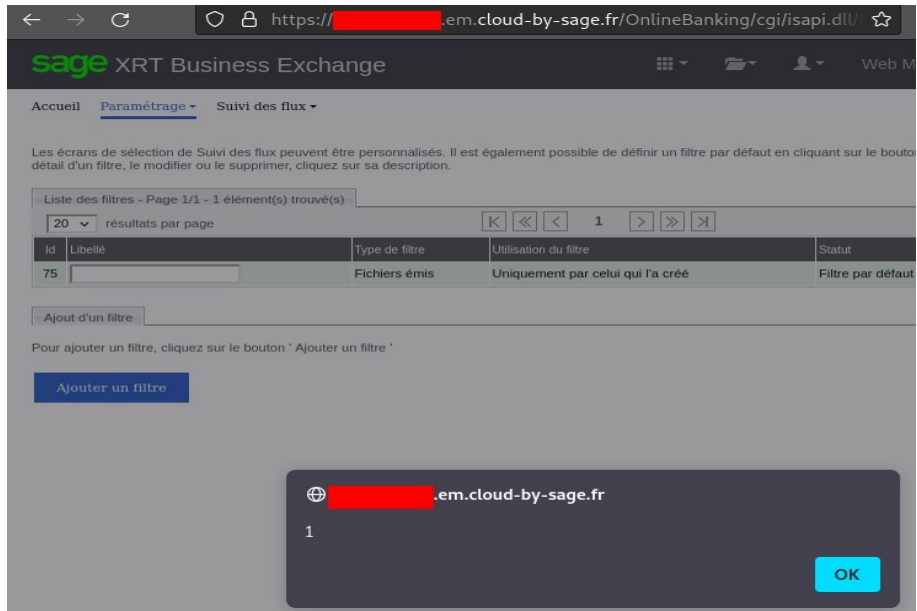


Illustration 2: Interpretation of the HTML in the *Filters* home page

The same issue is present in the *Display models* feature (*OnlineBanking > Web Monitoring > Settings > Display models*):

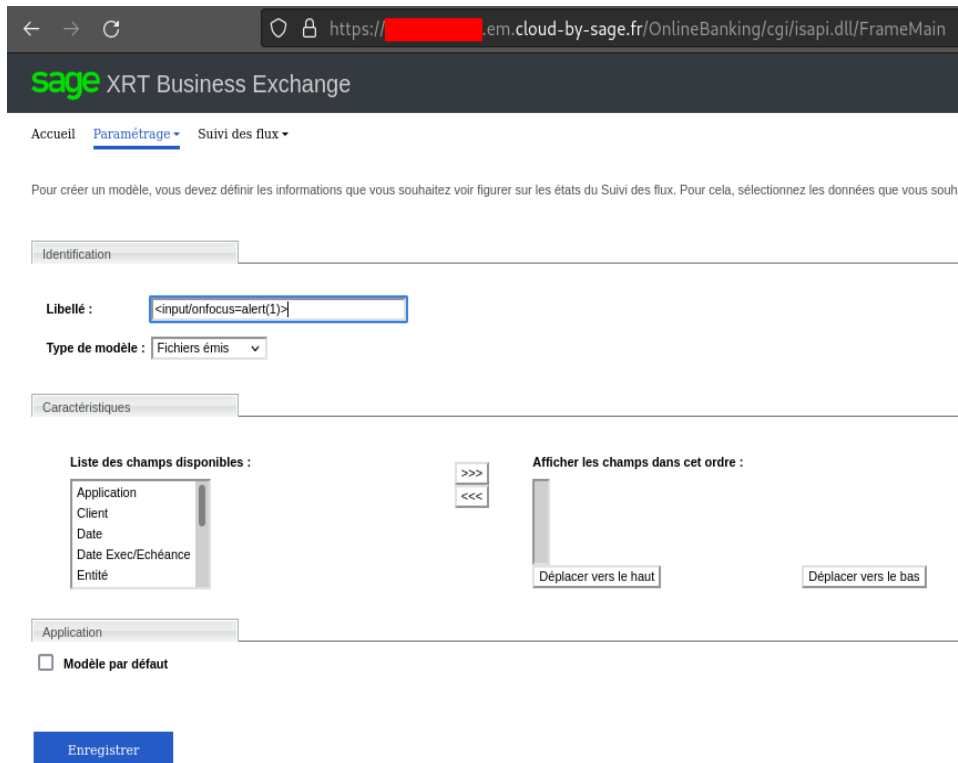


Illustration 3: Malicious HTML injected in the *Libellé* (french for *Label*) field

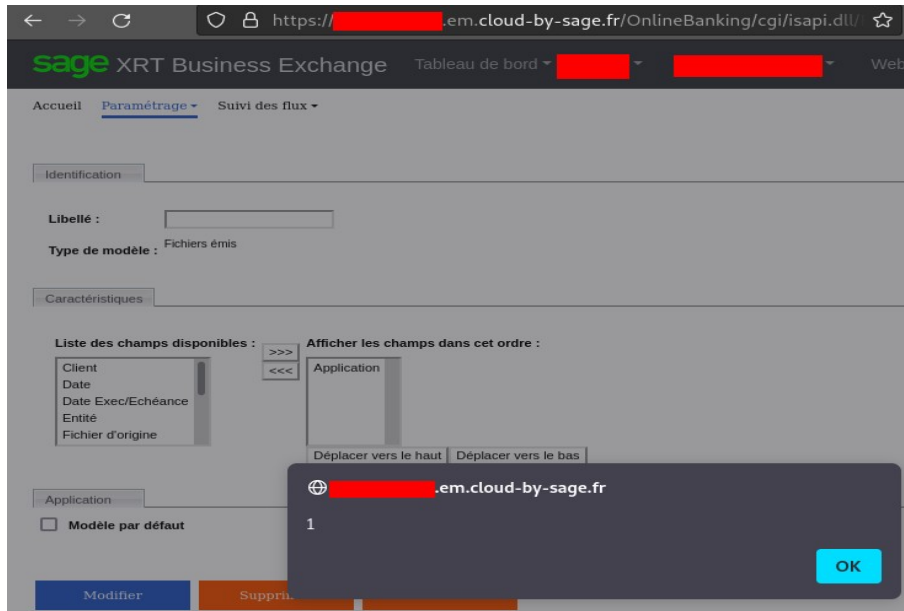


Illustration 4: Interpretation of the HTML in the *Display models* edit page

- *Notifications and alerts:*

This feature is available via the following path in the UI: *OnlineBanking > Configuration > Notifications and alerts*. Its purpose is to notify an administrator, for instance when a transaction is made. Once again, the name of an alert is interpreted as plain HTML when displayed back:

Accueil Paramètres utilisateur ▾ Notifications et alertes ▾

Un certain nombre de tâches étant automatisées, vous pouvez définir les personnes à informer via l'envoi d'alertes en fonction de l'événement devant déclencher une alerte, définissez pour chaque alerte, les groupes de destinataires. Vous pouvez affiner la définition de l'alerte en cliquant sur 'Plus d'options'. Cliquez ensuite sur le bouton 'Enregistrer'. La fréquence d'envoi des alertes dépend d'une tâche planifiée.

Alerte

Libellé :

Application :

Catégorie de transaction :

Alertes sur statut de transaction

1er et 2nd groupes de destinataires

<input type="checkbox"/> Statut 'A compléter'	<input type="text" value="▾"/>	<input type="text" value="▾"/>
<input type="checkbox"/> Statut 'En attente'	<input type="text" value="▾"/>	<input type="text" value="▾"/>
<input type="checkbox"/> Statut 'Validé'	<input type="text" value="▾"/>	<input type="text" value="▾"/>
<input type="checkbox"/> Statut 'Autorisé'	<input type="text" value="▾"/>	<input type="text" value="▾"/>

Illustration 5: Malicious HTML injected in the *Libellé* (french for *Label*) field

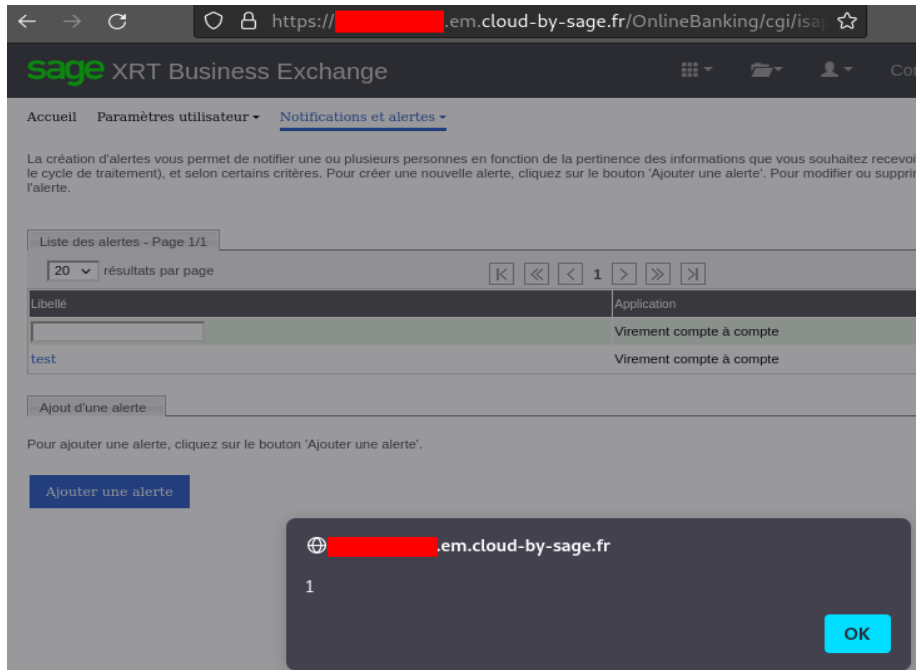


Illustration 6: Interpretation of the HTML in the Alerts home page

- File download:

This feature is available at the following URI: `/OnlineBanking/cgi/isapi.dll/DOWNLOADFRS`. When clicking the "Display the list" button, a POST request is sent to the server. The values of the `OVPNAME`, `JDAY` and `FREQUENCY` parameters of the request are then embedded directly in the JavaScript of the page returned by the server, as demonstrated here:

- The request:

```
POST /OnlineBanking/cgi/isapi.dll/DOWNLOADFRS HTTP/2
Host: *****.em.cloud-by-sage.fr
Cookie: SESSIONID=UR****[...]****==; _ga=GA1.2****[...]****98; _gid=GA1.2****[...]****62;
i18next=fr-FR
Content-Type: application/x-www-form-urlencoded
Content-Length: 157
[...]

STATUS=&FUNCNAME=&TITLE=&PRINT_ID=&PRINT_NAME=&INFO=YES&LIST=YES&SERVICE=UPLOAD&OVPNAME=.
%27%3Balert%281%29%3B%27&DATE=. &JDAY=. %27%3Balert%282%29%3B%27&FREQUENCY=. %27%3Balert
%283%29%3B%27
```

- Extract from the server's response:

```
<!--
[...]
// -----
function InitForm() {
document.FormDOWNLOAD.OVPNAME.value=
'.';alert(1);';document.FormDOWNLOAD.JDAY.value=
'.';alert(2);';document.FormDOWNLOAD.FREQUENCY.value =
'.';alert(3);';ChangeDate();
}
[...]
// -->
```

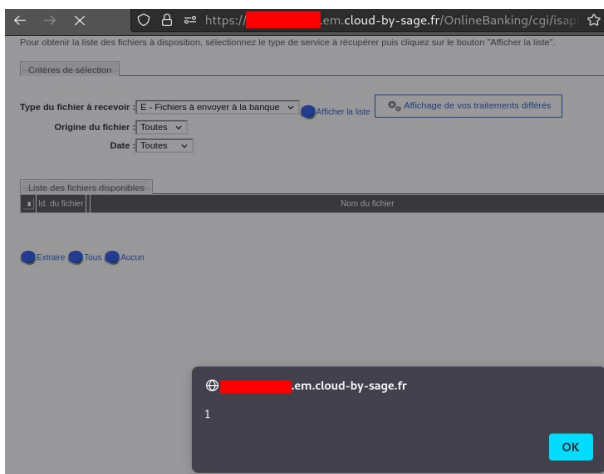


Illustration 7: First alert (parameter `OVPNAME`)

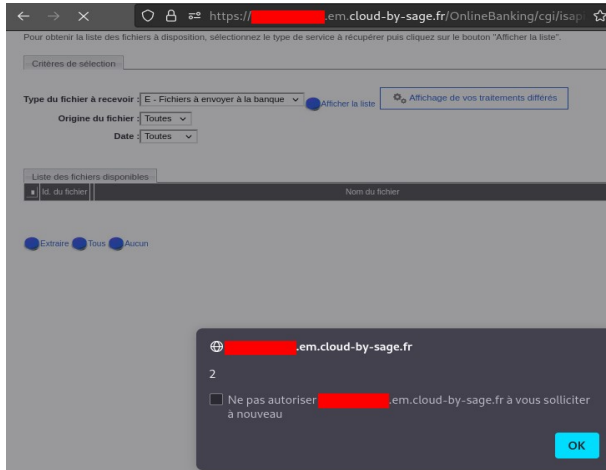


Illustration 8: Second alert (parameter *JDAY*)

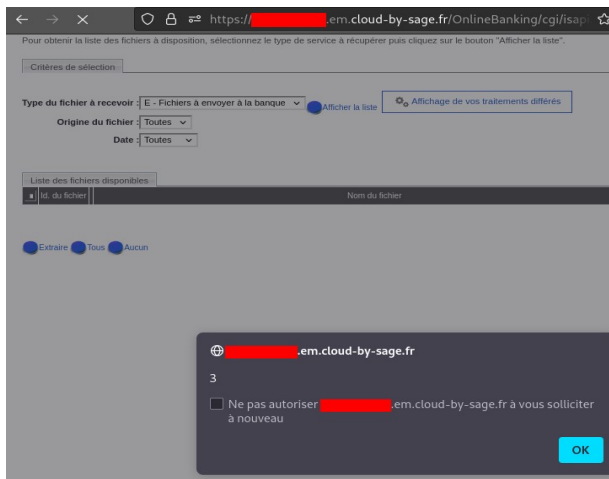


Illustration 9: Third alert (parameter *FREQUENCY*)

Impact

Depending on the cookies attributes, an authenticated attacker could either steal an administrator cookie or perform actions on the website on behalf of an administrator, such as adding a new user and granting them administrator privileges. With their newly obtained administrator privileges, an attacker would be able to gain remote command execution as detailed in the advisory on multiple authenticated blind SQL Injections in XRT.