tinexta defence

CVE-2025-56383: When a CVE gets it wrong

Vulnerability Report

#TinextaDefenceBusiness

Malware Lab

Summary

Our Malware Lab

Tinexta Defence Malware Lab daily performs dissection of malware with the aim of timely understanding the technological evolutions of attacks, consolidating the knowledge of necessary to make more effective and faster the process of incidents responding, contributing to spreading information about emerging threats into the expert's community and among its clients.

Malware Lab analysts are continuously engaged in searching and experimenting new analysis tools, for increasing accuracy and scope of action with regard to the proliferation of new evasion and anti-analysis techniques adopted by malware.

The Malware Lab is also committed to the development of proprietary tools for malware analysis and supporting the management and response of incidents.

Besides malware analysis, Malware Lab ideated and implemented an automatic process of extraction of **Indicators of Compromise (IOC)** that is daily run on dozens of new malwares, intercepted in the wide for populating our Knowledge Base.

Corrado Aaron Visaggio

Group Chief Scientist Officer & Malware Lab Director

a.visaggio@defencetech.it

Autori del report:

- Ermes Pennucci: Cybersecurity Analyst
- Alberico Ciriello: Cybersecurity Analyst
- Matteo Concutelli: Cybersecurity Analyst

1. Executive Summary

This report analyzes CVE-2025-56383[1], a vulnerability identified in the popular open-source text editor Notepad++[2] which is currently listed with CVSS score of 8.4[3]. Furthermore, an exploit POC is already available on GitHub[4].

The balance between compliance, security and usability is delicate. What a vulnerability actually is, often boils down to the threat model of the affected users rather than the technical details of the vulnerability itself.

A bulletin for this vulnerability has also been issued by the Italian CSIRT website[5].

Normally, the disclosure of a high-severity vulnerability in widely used software should cause an immediate reaction by organizations and individuals to patch the affected systems as soon as possible. However, in this case, the situation is more nuanced and needs a deeper analysis.

The developers of Notepad++, in fact, consider this CVE out of scope for the project and closed the GitHub issue as not planned[6].

We performed an independent analysis and ultimately agree with the developers; we believe that this is not a vulnerability but rather a misunderstanding of the software's intended functionality.

[1] https://nvd.nist.gov/vuln/detail/CVE-2025-56383

[2] https://github.com/notepad-plus-plus/notepad-plus-plus

[3] https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator?name=CVE-2025-56383&vector=AV:L/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H&version=3.1&source=CISA-ADP

[4] https://github.com/zer0t0/CVE-2025-56383-Proof-of-Concept

[5] https://www.acn.gov.it/portale/w/notepad-poc-pubblico-per-lo-sfruttamento-della-cve-2025-56383

[6] https://github.com/notepad-plus-plus/notepad-plus-plus/issues/17047#issuecomment-3349864991

The report claims that this vulnerability allows “Arbitrary Code Execution”, but in reality, it is a feature that allows users to load arbitrary plugins, which according to the CVE filing, could be exploited to execute malicious code.

We do not believe that this vulnerability requires manual action, in fact typical security hygiene practices such as preventing users from writing to the Program Files directory mitigates this issue. On misconfigured systems any program is potentially vulnerable to non-administrative users tampering with installation files, making this a system configuration issue rather than a vulnerability of a specific program.

2. Vulnerability analysis

Our analysis started by looking up the official CVE description.
At the time of writing it states:

```
Notepad++ v8.8.3 has a DLL hijacking vulnerability, which can
replace the original DLL file to execute malicious code.
```

While the description is vague, it suggests a plausible DLL hijacking[7] attack vector, this is a real class of issues and often used to deploy malware.

Then, we analyzed the proof-of-concept source code from GitHub that consists of a single file with one function which displays a message box upon loading.

[7]https://attack.mitre.org/techniques/T1574/001/

```
11  BOOL WINAPI DllMain(HMODULE hModule, DWORD dwReason, PVOID pvReserved)
12  {
13      if (dwReason == DLL_PROCESS_ATTACH)
14      {
15          MessageBoxA(0, "DLL hijacking test", "DLL hijacking test", 0);
16      }
17      else if (dwReason == DLL_THREAD_ATTACH)
18      {
19
20      }
21      else if (dwReason == DLL_THREAD_DETACH)
22      {
23
24      }
25      else if (dwReason == DLL_PROCESS_DETACH)
26      {
27
28      }
29      return TRUE;
30  }
```

The code also contains additional compiler directives to export certain symbols needed to produce a valid plugin DLL as specified by the Notepad++ plugin API[8].

When compiled this produces a “canary” DLL file which will show a message to the user when it is loaded, this is a very common payload technique to test this kind of exploits.

However, the exploit procedure detailed in the repository’s README instructs the user to replace the built-in plugin NppExport.dll present in regular installations of Notepad++ within the C:\Program Files\Notepad++\plugins\NppExport\ directory with the payload dll.

Subsequently, when Notepad++ is executed, it loads the payload as a plugin causing the DLL hijacking test message to appear, indicating that the exploit succeeded.

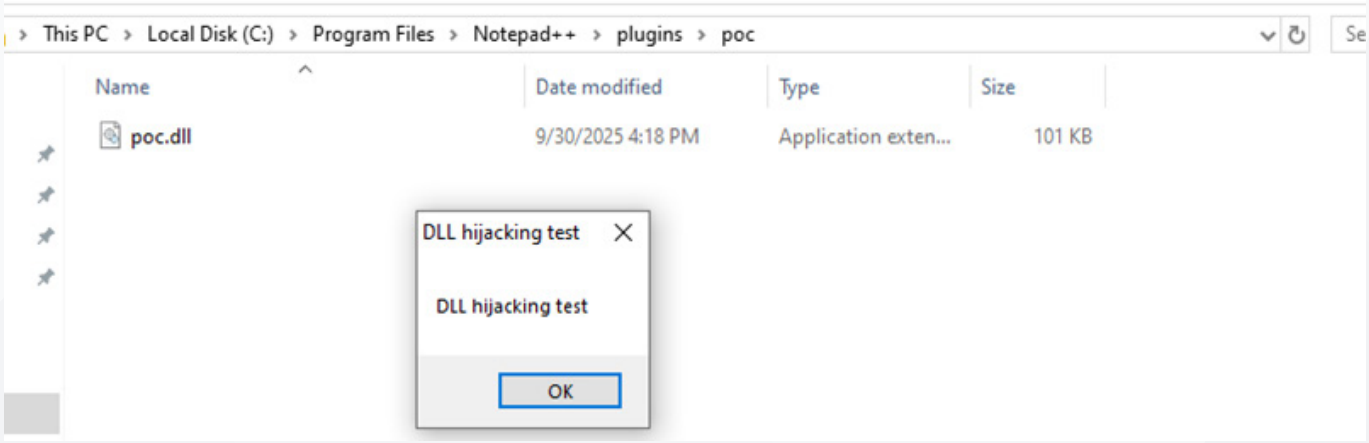
[8] https://github.com/npp-plugins/plugintemplate

We believe that this is a major misunderstanding of how native binary plugins work which is common to many applications beyond Notepad++, and as well as a misinterpretation of the threat model concerning software execution within the context of a user.

2.1 Reproducing the exploit

Firstly, to reproduce this issue, there is no need to replace an existing plugin, just copying the payload dll with any name in the plugins folder is sufficient to “exploit” it, therefore, the DLL hijacking part of the report is completely redundant.

This can be seen in our example configuration:



However, this is not a vulnerability, rather it is an intended behavior. Notepad++ is extensible by design and it is loading all the extensions present in the plugins folder.

Certain outlets claim that the root cause for this alleged vulnerability is the lack of integrity validation of loaded libraries, to this we argue that it isn’t applicable to a plugin system. White-listing only built-in plugins would mean essentially dropping plugin support, similarly, requiring digital signatures on all plugins would restrict the extensible nature of the software.

For an IT security team digital signatures would be a valid mitigation, If the company needs strict control over what software runs on a system. Applocker and App Control for Business[9] allow deploying policies to enforce digital signatures on all software and dll files of the system. We can't stress enough that this is a mitigation for the threat model of "users executing arbitrary code" rather than this specific alleged vulnerability.

A second overlooked aspect of this CVE report is the operation needed to install the payload, that is, writing in the installation directory of Notepad++.

Normally, regular users are not allowed to write in the Program files directory, since doing this would allow for an endless amount of privilege escalation pathways.

The POC explicitly requires the would-be attacker to copy the malicious plugin to the Notepad++ directory, this is only possible in the following three cases:

1. The user has already local administrative access. In this case there is no defense from this, the user already has complete control over the system regardless of Notepad++.
2. Notepad++ is installed in the user's AppData directory which is writeable but private to the user. In this case, the user might install malicious plugins, but the impact would be equivalent to manually executing arbitrary executable files downloaded from the internet.
3. Notepad++ is installed in a global writeable location and a non-administrative malicious user is able to install a malicious plugin. In this case the impact could cross user boundaries, but this would be a system misconfiguration rather than a Notepad++ specific issue since most software would also be exploitable under this condition.

[9]https://learn.microsoft.com/en-us/windows/security/application-security/application-control/app-control-for-business/appcontrol-and-applocker-overview

For us, none of this demonstrates a vulnerability in Notepad++, rather, it only demonstrates that an attacker with enough privileges can modify the application's files. In this scenario it would be possible to replace any exe or dll file of any installed software to achieve the same result.

As a final note, while this is not a vulnerability, it is definitely a potential stealth persistence technique to mask a malicious implant as a Notepad++ extension. It's not the first time we've seen attacks weaponizing regular software installed in the AppData directory, some examples are VS Code[10] and even Microsoft Teams[11].

[10]https://www.broadcom.com/support/security-center/protection-bulletin/malicious-vscode-extensions-infecing-users-with-cryptominer

[11]https://taggart-tech.com/quasar-electron/

3 Conclusions

We have thoroughly analyzed the provided PoC and the publicly available information about CVE-2025-56383 and our conclusion is that it is not a valid issue, and the CVE should be rejected.

As of now the National Vulnerability Database claims that CVE-2025-56383 is undergoing analysis and we'll know the final verdict in the coming weeks.

This is not the first time a security report causes a CVE to be issued for a non-existing vulnerability[12]. Mistakes happen, this is why it is critical for organizations to have security teams able to analyze reports and evaluate the actual impact of vulnerabilities.

On correctly configured systems where regular users cannot write to the Program Files directory this issue is not exploitable, and any risk caused by tampering with a per-user installation of Notepad++ is equivalent to the risk of running any other arbitrary executable.

We strongly recommend to employ EDR software and security policies to mitigate the actions of potential users fallen victim to phishing attacks or active attackers who already gained a foothold on an endpoint.

[12]https://daniel.haxx.se/blog/2023/08/26/cve-2020-19909-is-everything-that-is-wrong-with-cves/

tinexta
defence

Next | Donexit | Foramil | Innodesi

Via Giacomo Peroni, 452 – 00131 Roma
tel. 06.45752720 – info@defencetech.it
www.tinextadefence.it

#TinextaDefenceBusiness