

The Rising Threat of Malicious NuGet Packages Targeting .NET Developers



Nicoleta Botosan 3 minutes ago

VISMA SECURITY PROGRAM - RESEARCH AND DEVELOPMENT

8 min read

Caught in the NET 🕸️: The Rising Threat of Malicious NuGet Packages Targeting .NET Developers



Hallvard Molin Morstøl and Sverre Rynning-Tønnesen - two Master's students from the Norwegian University of Science and Technology (NTNU) want to present you some things about a recent attack targeting .NET developers through NuGet. Want to know how this affects you and maybe also how to prevent this attack? Then I recommend you check out this post which is entirely written by them! 📌

One week ago, the JFrog 🐸 Security Research team released a [blog post](#) describing a recent attack targeting .NET developers through NuGet. The attack was performed by spreading malicious NuGet packages that were downloaded more than 150K times. This specific attack used typosquatting techniques, but this is just one out of multiple possible attack vectors. Quite interesting! So let's dive in! 🤔



What does the current threat landscape look like?

Package registries like npm, PyPI, Maven Central, and NuGet can all be used for spreading malicious packages, and one should not blindly trust and download packages. The number of attackers using package registries to spread malicious packages is on the rise and while countermeasures are being worked on this problem will probably be around for a long time.

These kinds of attacks are also closely related to another form of supply chain attack where attackers are targeting weaknesses in third-party code. Here attackers can target dependencies (packages) that are used by other dependencies. Infiltrating one package, that is used by another package, which is then used by you, would compromise your program as well as all other programs having the vulnerable package in their dependency tree. As you see this means that one malicious package could make a lot of harm. To see a recent example of a big supply chain attack, just search for the 2020 SolarWinds☀️ hack.

🕶 Which packages should we look out for?

One large security vulnerability with .NET packages is that it facilitates executing code immediately upon package installation using “init.ps1”-files. Visual Studio will automatically run these “init” files without giving any warning to the user. This means that we need to make sure the malicious packages are never downloaded in the first place.

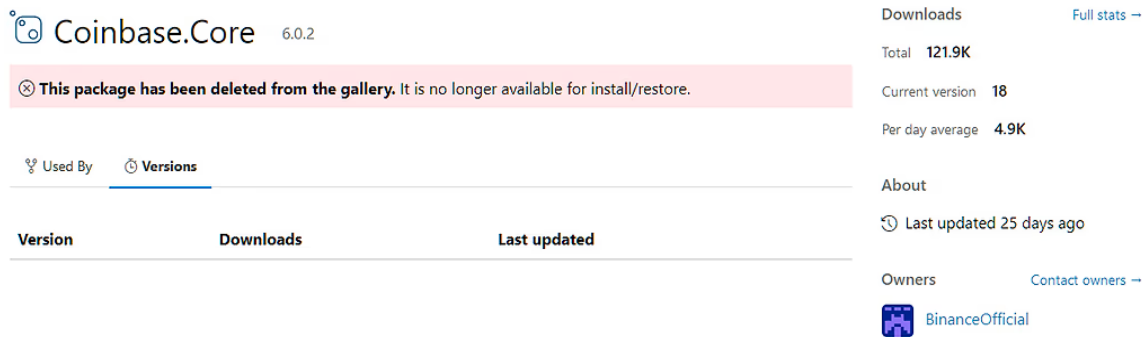
There are all kinds of tricks ✨ used to make malicious packages look legit. JFrog lists some indicators for malicious indicators in their post, but these are just a few of the many possibilities.

Some common indicators are:



Typosquatting

Typosquatting is a technique where you use a package name that is similar to a legitimate, and often popular, package. One example of such a package is "Coinbase.Core" which mimics the package "Coinbase".



Coinbase.Core 6.0.2

⊗ **This package has been deleted from the gallery.** It is no longer available for install/restore.

Used By Versions

Version	Downloads	Last updated
---------	-----------	--------------

Downloads [Full stats →](#)

Total **121.9K**


Current version **18**

Per day average **4.9K**

About

🕒 Last updated 25 days ago

Owners [Contact owners →](#)

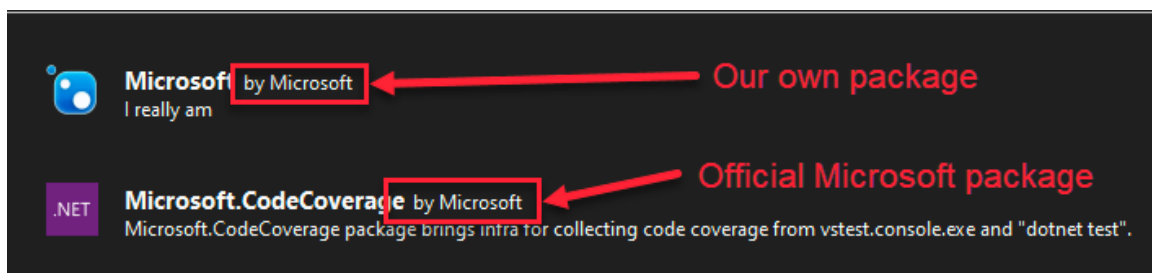
 **BinanceOfficial**


💣 Using package owner names that increase trust or are similar to known users

Another approach used is to use profile names that make the package seem more legitimate. For example "NuGetDev" or "OfficialDevelopmentTeam". One can also use names that are similar to the names of known users. Joel Verhagen is a Microsoft developer working on NuGet with the username "joelverhagen". A fake profile that tries to pretend to be Verhagen is "joeIverhagen", where the lowercase "l" is swapped with an uppercase "i".


💣 Inserting misleading metadata

The package metadata in the ".nuspec" contains various information about the package. Here the author field does not go through a verification process and the attacker could make it look like a package was released by "Microsoft" or another trusted source. This could be followed up by a well-written description (which is not the case in this example 😊) to make it look trustworthy.



 **Microsoft** by Microsoft ← Our own package

Microsoft.I really am

 **Microsoft.CodeCoverage** by Microsoft ← Official Microsoft package

Microsoft.CodeCoverage package brings intra for collecting code coverage from vstest.console.exe and "dotnet test".

💣 Bumping the total number of downloads

As already mentioned the malicious "Coinbase.Core" package found by JFrog was downloaded 150K times. This is more than the actual "Coinbase" package which is downloaded 118K times. The attackers have most likely artificially bumped the download count using bots to boost the package's trustworthiness.

💣 Using malicious dependencies

Another thing to look out for are the dependencies included. As mentioned, the packages can be used in a supply chain attack where the code in the package you download is fine but the dependency used can be malicious.

```

1 <?xml version="1.0"?>
2 <package xmlns="http://schemas.microsoft.com/packaging/2011/08/nuspec.xsd">
3   <metadata>
4     <id>DiscordRichPresence.API</id>
5     <version>1.1.3.18</version>
6     <title>DiscordRichPresence</title>
7     <authors>Lachee</authors>
8     <projectUrl>https://github.com/Lachee/discord-rpc-csharp</projectUrl>
9     <requireLicenseAcceptance>false</requireLicenseAcceptance>
10    <description>A .NET implementation of Discord's Rich Presence functionality. This library
11      supports all features of the Rich Presence that the official C++ library supports, plus a few
12      extra.
13      "Players love to show off what they are playing with Discord's status feature. With Rich Presence
14      you can add beautiful art and detailed information to show off your game even more. This lets
15      players know what their friends are doing, so they can decide to ask to join in and play together."
16      - Discord
17      This is an unofficial, non-discord supported library.</description>
18    <copyright>2022</copyright>
19    <tags>Discord, Rich, Presence, RPC, Discord-RPC</tags>
20    <dependencies>
21      <dependency id="DiscordRichPresence" version="1.1.3.18" />
22      <dependency id="Manage.Carasel.Net" version="1.0.3" />
23    </dependencies>
24  </metadata>
25  <files>
26    <file src="tools\install.ps1" target="tools\install.ps1" />
27  </files>
28 </package>
  
```

Annotations in the image:

- Legitimate Package:** Points to the `<dependency id="DiscordRichPresence" version="1.1.3.18" />` line.
- Malicious Package:** Points to the `<dependency id="Manage.Carasel.Net" version="1.0.3" />` line.
- No Payload:** Points to the `<file src="tools\install.ps1" target="tools\install.ps1" />` line.



Sooo, how should we protect ourselves?

All developers should do an assessment of packages before adding them to their projects. If it is possible, it is good to have a second pair of eyes to help during the evaluation of the package. Since many programming languages allow the packages to execute code on installation, such as NuGet and npm, the developer should make sure they trust the code before downloading it. To prevent being a target of a typosquatting attack you should copy the download command from the official package registry site.

```

.NET CLI  Package Manager  PackageReference  Paket CLI  Script & Interactive  Cake
> dotnet add package Newtonsoft.Json --version 13.0.3
  
```

When adding a dependency you should make it clear in the pull request why you have added the dependency. It can also be valuable to have guidelines in the

group regarding what information should be included in a pull request when adding new dependencies. After dependencies are added to a project you should use some tool to scan the project regularly for vulnerabilities using dependency scanners such as Github Dependabot, Snyk, etc.

If you want to take some extra steps for increasing security, it is possible to scan the packages before adding them using [OSS-gadgets](#). OSS-Gadgets is a set of Microsoft-developed tools that can help find obfuscated strings, identify potential backdoors, calculate a metric for the risk of using a package, etc. They claim it is still in public preview and that it is not ready for production use, so one should still be critical when adding packages. After the packages are assessed you can choose to self-host the package with a set version in a private package registry. Then you can have a policy to only use packages from the private package registry. When updating packages to newer versions it is then important to rescan the package before adding it to the private package registry. But note that these tools are no guarantee for making sure packages are safe to use.

 Luckily we're working on fixing it! But we need your help!!!

We are currently writing a master's thesis about software supply chain security, where we focus on NuGet. We plan to create a tool, to wrap the "dotnet add package"-command which first fetches relevant information about the package and then automatically assesses the package using a set of trust criteria and thresholds. The information will be displayed to the developers and they are prompted if they would still like to add the package.

We are currently working on figuring out which trust criteria are valuable and the thresholds that make sense to use. Finding good trust criteria that differentiate benign and malicious packages using the available data from NuGet and other sites is hard. But by combining different trust criteria it will be harder for attackers to make the malicious packages seem trustworthy.

If you would like to help with assessing trust criteria, we would appreciate it if you answered this survey regarding trust criteria <https://forms.gle/WnTEJxSqze2gReot7>.

Soon we will also reach out to some of you to run user tests on this tool we are really appreciating all your help! So thank you!

About us:

The students:

- Hallvard Molin Morstøl (hallvarm@stud.ntnu.no) [LinkedIn profile](#)

- Sverre Rynning-Tønnesen (sverrery@stud.ntnu.no) [LinkedIn profile](#)

Supervisors:

- Daniela Soares Cruzes (daniela.soares.cruzes@visma.com)

- Monica Iovan (monica.iovan@visma.com)

We really hope that you found this interesting. Curious to find more about what they are planning to do? Then don't hesitate to invite [@Daniela Cruzes](#) and/or [@Monica Iovan](#) to a virtual ☕ - and they will give you more details if you want to onboard and give a helping hand to this promising master thesis!



Happy Spring!

#nuget

#security

#securityawareness

#securityresearch