Beware of Deserialisation in .NET Methods and Classes + Code Execution via Paste!

Beware of Deserialisation in .NET Methods and Classes + Code Execution via Paste! - Soroush Dalili, NCC Group.

- Published: date not stated
- Original: <https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2018/december/beware-of-deserialisation-in-.net-methods-and-classes-code-execution-via-paste/>
- Preserved from: https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2018/december/beware-of-deserialisation-in-.net-methods-and-classes-code-execution-via-paste/ (stored) on 2026-08-09
- Capture timestamp: 20190401191940
- Licence: unknown

Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

Beware of Deserialisation in .NET Methods and Classes + Code Execution via Paste!

In light of practical exploitation for deserialisation issues in the .NET Framework [1] [2] [3], we thought it might be useful to identify .NET Framework 4.7.2 methods and classes that utilise deserialisation as it can help security researchers and developers to find and fix the potential issues. The initial result of our research can be downloaded from:

https://www.nccgroup.trust/uk/our-research/use-of-deserialisation-in-.net-framework-methods-and-classes/?research=Whitepapers

Upon sharing our findings with Microsoft in August 2018, new security notes were added to the code documentation as can be seen here: https://github.com/dotnet/dotnet-api-docs/pull/502. Our whitepaper should also contain almost all of the methods and classes in the .NET Framework that have security notes at the time of writing which might also be useful.

The following plugins were also added to the ysoserial.net project [3]:

- https://github.com/pwntester/ysoserial.net/blob/master/ysoserial/Plugins/ResxPlugin.cs
- https://github.com/pwntester/ysoserial.net/blob/master/ysoserial/Plugins/ApplicationTrustPlugin.cs

- https://github.com/pwntester/ysoserial.net/blob/master/ysoserial/Plugins/altserializationPlugin.cs
- https://github.com/pwntester/ysoserial.net/blob/master/ysoserial/Plugins/ClipboardPlugin.cs
- https://github.com/pwntester/ysoserial.net/blob/master/ysoserial/Plugins/SessionSecurityTokenHandlerPlugin.cs

Code execution via paste and deserialisation

One of the most interesting findings was the deserialisation issue exploitable upon pasting from the clipboard!

In the .NET framework, objects could be serialised and stored in a `DataObject` in Clipboard. These objects could then be deserialised automatically upon paste. Similar to any other deserialisation issues, this can be exploited when an attacker can create an arbitrary object in the clipboard. Although it was not possible to find a way to store an arbitrary object into the clipboard remotely, this might still be useful for breaking sandbox environments where executing direct commands is not possible but clipboard objects are allowed to go through. Privilege escalation might also be another avenue when an affected .NET application is running with higher privileges but there are probably other, easier techniques to do this locally when there are no restrictions.

Technical details

The `DataObject` class [4] deserialised certain clipboard objects using `BinaryFormatter`. The following formats via the `System.Windows.DataFormats` or `System.Windows.Forms.DataFormats` classes were affected:

```
CommaSeparatedValue, Dib, Dif, Locale (potentially), PenData, Riff, Serializable, StringFormat, SymbolicLink, Tiff, Wave
```

These are the same as the following format names:

```
Csv, DeviceIndependentBitmap, DataInterchangeFormat, Locale (potentially), PenData, RiffAudio, WindowsForms10Pe
```

This has been added as a plugin to the ysoserial.net project [3]: https://github.com/pwntester/ysoserial.net/blob/master/ysoserial/Plugins/ClipboardPlugin.cs It is therefore possible to store a payload into the clipboard using the following command:

```
ysoserial.exe -p Clipboard -c calc -F System.String
```

What is affected?

Apart from the applications that read special objects from the clipboard, any WPF applications [5] that utilise a `TextBox`, `PasswordBox`, or `RichTextBox` are also affected.

The following applications were found to be vulnerable that can be used as examples:

- PowerShell ISE
- Visual Studio (quick launch at the top right)
- Paint.net - patched [6] (clicking the edit menu or pasting in textbox)
- LINQPad - patched [7]

The following GIF video file shows a proof of concept:

[image: image]

Recommendation

As Microsoft has accepted this as intended behaviour, it will be up to the underlying applications to ensure they are not vulnerable by using unaffected `DataFormats` such as `Text`, `Rtf`, `Tiff`, or `Bitmap` when dealing with the data object from the clipboard.

When using a sandboxed environment, it is recommended to not allow clipboard access or to restrict it to text objects.

Risks associated with code execution by pasting from the clipboard should be assessed based on the fact that an attacker might need to execute some code in the first place to store an arbitrary object in the clipboard.

References:

[1] https://media.blackhat.com/bh-us-12/Briefings/Forshaw/BH_US_12_Forshaw_Are_You_My_Type_WP.pdf

[2] https://www.blackhat.com/docs/us-17/thursday/us-17-Munoz-Friday-The-13th-JSON-Attacks-wp.pdf

[3] https://github.com/pwntester/ysoserial.net

[4] https://referencesource.microsoft.com/#PresentationCore/Core/CSharp/system/windows/DataObject.cs,3410

[5] https://docs.microsoft.com/en-us/dotnet/framework/wpf/getting-started/walkthrough-my-first-wpf-desktop-application

[6] https://blog.getpaint.net/2018/10/22/paint-net-4-1-2-is-now-available/ (CVE-2018-18447)

[7] https://www.linqpad.net/DotNetSecurityExploit.aspx

Published date: 17 December 2018

Written by: Soroush Dalili