

```

1  #-----
2  #Get-IDFScanAutomation.ps1
3  #
4  #Written by: Matt Logue
5  #
6  #Last Modified by: Matt Logue
7  #
8  #Last Modified: April 9, 2020
9  #
10 #Version: 2.3
11 #
12 #Purpose: The script looks for attachments using outlook for CSV exports of ID
13 #Finder scans, creates a pivot table daily, then the most recent scan gets uploaded
14 #to a sharepoint document library by Get-IDFinderReports.ps1 running on the on-
15 #premise sharepoint node
16 #-----
17 <#
18 .SYNOPSIS
19     The script looks for attachments using outlook for CSV exports of ID Finder
20     scans, creates a pivot table daily, then the most recent scan gets uploaded
21     #to a sharepoint document library by Get-IDFinderReports.ps1 running on the on-
22     premise sharepoint node
23 .DESCRIPTION
24     Requires
25     1. Connection to the shared folder path and modify the permissions
26     2. Outlook with profile setup for the idfinder@hsc.wvu.edu email account
27 .PARAMETER
28     No required parameters
29
30 .NOTES
31     Author: Matt Logue
32     Last Updated by: Matt Logue
33     Last Updated: April 8, 2020
34 #>
35
36 [CmdletBinding()]
37 param (
38     [switch]$SessionTranscript = $true,
39     [string]$LogFilePath = "C:\Users\microsoft\Documents\GitHub\HSC-PowerShell-
40 Repository\Get-IDFScanAutomation\Logs",
41     [switch]$StopOnError = $true, #keep this value true - will error if no filetype
42 gets passed to script
43     [int]$DaysToKeepLogFiles = 5, #this value used to clean old log files
44
45     #File specific param
46     [string]$scriptTest = $false,
47     [string]$filepath = "\\hs.wvu-ad.wvu.edu\Public\ITS\security services\ID Finder
48 Scan Files\"
49 )
50
51 Set-Location C:\Users\microsoft\Documents\GitHub\HSC-PowerShell-Repository
52
53 #Change PowerShell window title
54 $Host.UI.RawUI.WindowTitle = "Get-IDFScanAutomation.ps1"

```

```
52
53 #Add references to file containing needed functions
54 Import-Module .\1HSC-PowerShell-Modules\HSC-CommonCodeModule.psm1
55
56 #Verify the log file directory exists
57 if ([string]::IsNullOrEmpty($LogFilePath))
58 {
59     Write-ColorOutput -Message "Log file path is null" -ForegroundColor "Yellow"
60     Write-ColorOutput -Message "Setting log path to
61     C:\Users\microsoft\Documents\GitHub\HSC-PowerShell-Repository\Get-
62     IDFScanAutomation\Get-IDFScanAutomation\Logs" -ForegroundColor "Yellow"
63
64     $LogFilePath = "C:\Users\microsoft\Documents\GitHub\HSC-PowerShell-
65     Repository\Get-IDFScanAutomation\Logs"
66
67     if (!(Test-Path $LogFilePath))
68     {
69         New-Item $LogFilePath -Type Directory -force
70     }
71
72     if ($Error.Count -gt 0)
73     {
74         $Continue = Read-Host "There was an error creating the log file directory.
75         Continue(y/n)"
76
77         if ($Continue -ne "y")
78         {
79             exit
80         }
81     }
82 }
83
84 #Common variable declarations
85 $TranscriptLogFile = $LogFilePath + "\" + (get-date -format yyyy-MM-dd) + "-IDF-
86 ScanAutomation-SessionOutput.txt"
87 $ErrorLogFile = $LogFilePath + "\" + (get-date -format yyyy-MM-dd) + "-IDF-
88 ScanAutomation-ErrorLogFile.txt"
89
90 if ($SessionTranscript)
91 {
92     try
93     {
94         Stop-Transcript -ea "Stop"
95     }
96     catch
97     {
98     }
99
100     "TranscriptLogFile: " + $TranscriptLogFile
101     Start-Transcript -Path $TranscriptLogFile -Force
102     "Transcript log file started"
103 }
104
105 #Cleaning up old log files
106 Write-Verbose "Removing Old Log Files"
107 Remove-OldLogFile -Days $DaysToKeepLogFiles -Path $LogFilePath
108
109 $Error.clear()
110
```

```

106 #####
107 #           Main Script Function           #
108 #####
109
110 #Stops any existing Outlook process
111 If ((Get-process Outlook -ErrorAction SilentlyContinue)) {
112     Get-Process Outlook | stop-process -force
113     Start-Sleep 30
114     if (Get-Process Outlook -ErrorAction SilentlyContinue) {
115         Get-Process Outlook | stop-process -force
116     }
117 }
118
119 $error.Clear()
120
121 # This code block searches for the most recent report sent to idfinder@hsc.wvu.edu
122 Add-type -assembly "Microsoft.Office.Interop.Outlook" | out-null
123
124 $olFolders = "Microsoft.Office.Interop.Outlook.olDefaultFolders" -as [type]
125
126 if(([System.Diagnostics.Process]::GetProcesses()).name | ?{$_ -like 'outlook*'})
127 {
128     write-host Outlook is running!
129     $outlook =
130 [Runtime.InteropServices.Marshal]::GetActiveObject('Outlook.Application')
131 }
132 else {
133 $outlook = new-object -comobject outlook.application
134 # $outlook.OpenProfile("IDFinder")
135 }
136 $namespace = $outlook.GetNameSpace("MAPI")
137 Write-Output "Opening ID Finder Mailbox"
138 $sourceFolder = $namespace.Folders.Item("ID Finder").Folders.Item("Inbox")
139
140 $inbox = $sourceFolder.items | Select-Object -Property
141 senderName,SenderEmailAddress,Subject,ReceivedTime,Attachments | select -first 25
142 Write-Output "Processing Emails..."
143
144 $emails = $inbox | Where -Property SenderEmailAddress -eq "infosec@mail.wvu.edu" |
145 Sort-Object ReceivedTime -Descending | Select -first 25 -Property
146 Subject,Attachments,ReceivedTime
147
148
149 $filepath = "\\hs\public\ITS\security services\ID Finder Scan Files\"
150 $folderpath = $($filepath + $(Get-Date -Format yyyy-MM-dd))
151 if (!(Test-Path -Path $folderpath))
152 {
153     New-Item -Path $folderpath -ItemType Directory -Force
154 }
155
156 $emails.attachments | `
157
158 foreach {
159
160     Write-Host "Found File:" $_.filename
161     $attr = $_.filename
162     #add-content $log1 "Attachment: $attr"
163     $a = $_.filename
164     $fileName = "$(Get-Date -Format yyyyMMdd)" + "_" + $a
165     If ( ($a.Contains("csv")) -AND ($a -notlike "*count*") -AND
166 ($a.Contains("IDF2")) )

```

```

161 {
162     $filename = $filename -replace "_Locations", ""
163     $_.saveasfile((Join-Path $folderpath $fileName))
164     Write-Output "Attachment Saved: $folderpath\$filename"
165 }
166 }
167
168 Write-Output "Number of attachments saved: $((Get-ChildItem $folderpath | Where-
Object{!(($_.PSIsContainer)}).count)"
169 Write-Output "*****`n`n"
170
171
172 #####
173 #         Code block for creating the Pivot table           #
174 #         from saved attachments                             #
175 #####
176
177 #following formats the date for use in the file names and searching for the file
178 $date = Get-Date -Format u
179 $date = $date -split " "
180 $date = $date[0] -replace "-", ""
181
182 $filetypes = @("CCs", "SSNs", "ePHI", "TAXES")
183 foreach ($filetype in $filetypes)
184 {
185     $nofilecount=0
186     $filename = $date+'_IDF2_HSC_'+$filetype+'.csv'
187     if (!(Test-Path "$folderpath\$filename")){
188         Write-Output "No CSV Found called $filename"
189         $nofilecount++
190         if($nofilecount -ge 2) {
191             Write-Output "Not enough CSV's Found"
192             Exit 1
193         }
194     }
195     $csvFile = (Get-ChildItem "$folderpath\$filename") #| sort LastWriteTime |
select -last 1
196     $filedirectory = Split-Path -Path $csvFile -Parent
197     $csvfilename = Split-Path -Path $csvFile -leaf
198     Write-Output "File Found: $filedirectory\$csvfilename"
199     $finalFilename = $csvfilename -replace ".csv", ""
200
201     #Create excel COM object
202     $excel = New-Object -ComObject excel.application
203
204     #Make Visible
205     $excel.Visible = $false
206     $excel.Caption = "IDFinder_HSC_" + $filetype
207     $xlWindowState = [Microsoft.Office.Interop.Excel.XLWindowState]
208     $excel.DisplayAlerts = $FALSE
209     #$excel.Application.WindowState = $xlWindowState::xlMinimized
210
211     #Add a workbook
212     $workbook = $excel.Workbooks.Add()
213     $ws1 = $workbook.Worksheets.Add()
214
215
216     #Connect to first worksheet to rename and make active
217     $dataSheet2 = $workbook.Worksheets.Item(1)
218     $dataSheet2.Name = "PivotTable"

```

```
219 $dataSheet2.Activate() | Out-Null
220 $dataSheet = $workbook.Worksheets.Item(2)
221 $dataSheet.Name = "DataDump"
222 Write-Output "Excel Workbook created"
223
224 if ((Get-Content $csvFile).Contains("sep="))
225 {
226     (Get-Content $csvFile) | Select-Object -Skip 1 | Set-Content $csvFile
227     #removes "sep=" line from beginning of csv
228 }
229
230 $csvFileProcessed = Import-Csv -Path $csvFile
231 $processes = $csvFileProcessed
232
233 if ($filetype -eq "TAXES")
234 {
235     $dataSheet.cells.item(1,1) = "Endpoint"
236     $dataSheet.cells.item(1,2) = "Tag Name"
237     $dataSheet.cells.item(1,3) = "Location"
238     $dataSheet.cells.item(1,4) = "Unprotected Quantity"
239 }
240
241 else {
242     $dataSheet.cells.item(1,1) = "Tag Name"
243     $dataSheet.cells.item(1,2) = "Endpoint"
244     if ($filetype -eq "CCs")
245     {
246         $dataSheet.cells.item(1,3) = "Unprotected CreditCards"
247     }
248     if ($filetype -eq "SSNs")
249     {
250         $dataSheet.cells.item(1,3) = "Unprotected SSNs"
251     }
252     if ($filetype -eq "ePHI")
253     {
254         $dataSheet.cells.item(1,3) = "Unprotected Quantity"
255     }
256
257     $dataSheet.cells.item(1,4) = "Location"
258 }
259
260 Write-Output "Importing Raw data to DataDump Worksheet..."
261 $i = 2
262 if ($filetype -eq "CCs")
263 {
264     foreach($process in $processes)
265     {
266         $dataSheet.cells.item($i,1) = $process.'Tag Name'
267         $dataSheet.cells.item($i,2) = $process.Endpoint
268         $dataSheet.cells.item($i,3) = $process.'Unprotected CreditCards'
269         $dataSheet.cells.item($i,4) = $process.Location
270         $i++
271     } #end foreach process
272 } #end if
273 if ($filetype -eq "SSNs")
274 {
275     foreach($process in $processes)
276     {
277         $dataSheet.cells.item($i,1) = $process.'Tag Name'
```

```

278     $dataSheet.cells.item($i,2) = $process.Endpoint
279     $dataSheet.cells.item($i,3) = $process.'Unprotected SSNs'
280     $dataSheet.cells.item($i,4) = $process.Location
281     $i++
282 } #end foreach process
283 } #end if
284
285 if ($filetype -eq "ePHI")
286 {
287     foreach($process in $processes)
288     {
289         $dataSheet.cells.item($i,1) = $process.'Tag Name'
290         $dataSheet.cells.item($i,2) = $process.Endpoint
291         $dataSheet.cells.item($i,3) = $process.'Unprotected Quantity'
292         $dataSheet.cells.item($i,4) = $process.Location
293         $i++
294     } #end foreach process
295 } #end if
296
297 if ($filetype -eq "TAXES")
298 {
299     foreach($process in $processes)
300     {
301         $dataSheet.cells.item($i,1) = $process.'Endpoint Name'
302         $dataSheet.cells.item($i,2) = $process.'Tag Name'
303         $dataSheet.cells.item($i,3) = $process.Location
304         $dataSheet.cells.item($i,4) = $process.'Unprotected Quantity'
305         $i++
306     } #end foreach process
307 } #end if
308
309
310 # rename workbook
311 $workbook = $workbook
312
313 # looks for names of worksheets
314 $ws3 = $workbook.worksheets | where {$_.name -eq "PivotTable"} #<-----
Selects sheet 3
315 $ws1 = $workbook.worksheets | where {$_.name -eq "DataDump"}
316
317 $xlPivotTableVersion15 = 5
318 $xlPivotTableVersion14 = 4
319 $xlPivotTableVersion12 = 3
320 $xlPivotTableVersion10 = 1
321 $xlCount = -4112
322 $xlDescending = 2
323 $xlDatabase = 1
324 $xlHidden = 0
325 $xlRowField = 1
326 $xlColumnField = 2
327 $xlPageField = 3
328 $xlDataField = 4
329 $xlDirection = [Microsoft.Office.Interop.Excel.XLDirection]
330
331 #looks for final non-empty row
332 $finalrow=$ws1.Cells.End($xlDirection::xlDown).Row
333
334 #creates pivot datasource on $ws1 then creates pivot table on $ws3 at position
R1C1

```

```
335     $PivotTable = $workbook.PivotCaches().Create($xlDatabase, $ws1.name +  
"!R1C1:R"+ $finalrow +"C4", $xlPivotTableVersion15)  
336     $PivotTable.CreatePivotTable($ws3.name + "!R1C1", "Tables1") | Out-Null  
337     [void]$ws3.Select();  
338     $ws3.Cells.Item(1,1).Select() | Out-Null  
339     #hide Pivot Table field list  
340     $workbook.ShowPivotTableFieldList = $false  
341  
342     #creates pivot table fields  
343     $PivotFields1 = $ws3.PivotTables("Tables1").PivotFields("Tag Name")  
344     $PivotFields1.Orientation = $xlRowField  
345     $PivotFields1.Caption = "Units"  
346  
347     $PivotFields2 = $ws3.PivotTables("Tables1").PivotFields("Endpoint")  
348     $PivotFields2.Orientation = $xlRowField  
349  
350     $PivotFields5 = $ws3.PivotTables("Tables1").PivotFields("Location")  
351     $PivotFields5.Orientation = $xlRowField  
352  
353     $PivotFields3 = $ws3.PivotTables("Tables1").PivotFields("Location")  
354     $PivotFields3.Orientation = $xlDataField  
355     #looks at $filetype for naming of 2nd column of pivot table  
356     if ($filetype -eq "CCs")  
357     {  
358         $PivotFields3.Name = "Files with Unprotected CreditCards"  
359     }  
360     if ($filetype -eq "SSNs")  
361     {  
362         $PivotFields3.Name = "Files with Unprotected SSNs"  
363     }  
364     if ($filetype -eq "ePHI")  
365     {  
366         $PivotFields3.Name = "Files with Unprotected Quantities"  
367     }  
368     if ($filetype -eq "TAXES")  
369     {  
370         $PivotFields3.Name = "Files with Unprotected Quantities"  
371     }  
372  
373     #looks at $filetype for creating 3rd column of pivot table  
374     if ($filetype -eq "CCs")  
375     {  
376         $PivotFields4 = $ws3.PivotTables("Tables1").PivotFields("Unprotected  
CreditCards")  
377         $PivotFields4.Orientation = $xlDataField  
378         $PivotFields4.Function = $xlSum  
379         $PivotFields4.Caption = "Sum of Unprotected CreditCards"  
380     }  
381  
382     if ($filetype -eq "SSNs")  
383     {  
384         $PivotFields4 = $ws3.PivotTables("Tables1").PivotFields("Unprotected SSNs")  
385         $PivotFields4.Orientation = $xlDataField  
386         $PivotFields4.Function = $xlSum  
387         $PivotFields4.Caption = "Sum of Unprotected SSNs"  
388     }  
389     if ($filetype -eq "ePHI")  
390     {  
391         $PivotFields4 = $ws3.PivotTables("Tables1").PivotFields("Unprotected  
Quantity")
```

```

392     $PivotFields4.Orientation = $xlDataField
393     $PivotFields4.Function = $xlSum
394     $PivotFields4.Caption = "Sum of Unprotected Quantities"
395 }
396 if ($filetype -eq "TAXES")
397 {
398     $PivotFields4 = $ws3.PivotTables("Tables1").PivotFields("Unprotected
Quantity")
399     $PivotFields4.Orientation = $xlDataField
400     $PivotFields4.Function = $xlSum
401     $PivotFields4.Caption = "Sum of Unprotected Quantities"
402 }
403
404 #sorts pivot table by the largest numbers in the 4th pivot field
405 $PivotFields1.AutoSort($xlDescending, $PivotFields4.Name)
406
407 #collapses all row fields
408 Foreach ($PivotField in $ws3.PivotTables("Tables1").PivotFields()) {
409     try {
410         $PivotField.ShowDetail = $False
411     }
412     catch {
413         #this throws and unhandled exception if it's not in a try/catch block to
ignore the error
414     }
415 }
416
417 Write-Output "Created PivotTable"
418 #Changes formatting of table also based on file type
419 $row = 1
420 $Column = 1
421 $ws3.Cells.Item($row,$column)= 'Units'
422 if ($filetype -eq "CCs")
423 {
424     $ws3.PivotTables("Tables1").TableStyle2 = "PivotStyleMedium14"
425 }
426 if ($filetype -eq "SSNs")
427 {
428     $ws3.PivotTables("Tables1").TableStyle2 = "PivotStyleMedium12"
429 }
430 if ($filetype -eq "ePHI")
431 {
432     $ws3.PivotTables("Tables1").TableStyle2 = "PivotStyleMedium13"
433 }
434 if ($filetype -eq "TAXES")
435 {
436     $ws3.PivotTables("Tables1").TableStyle2 = "PivotStyleMedium10"
437 }
438 $ws3.columns.item(1).columnWidth = 60
439 $ws3.PivotTables("Tables1").HasAutoFormat = $false
440 $ws3.PivotTables("Tables1").ShowTableStyleRowStripes = $true
441 Write-Output "PivotTable Formatted"
442
443 #script block for testing to make excel table visible
444 if ($scriptTest -eq $true)
445 {
446     $excel.Visible = $true
447     $excel.Application.WindowState = $xlWindowState::xlMaximized
448 }
449

```



```

450     if (!(Test-Path "$filedirectory\Processed"))
451     {
452         New-Item -Path "$filedirectory\Processed" -ItemType Directory
453     }
454     #save files
455     $workbook.activate()
456     $workbook.SaveAs($filedirectory+"\Processed\"+$finalfilename+".xlsx")
457     #if $script test is false then it will close the workbook and release the
variables
458     if ($scriptTest -eq $false)
459     {
460         $Excel.Workbooks.Close()
461         $Excel.Quit()
462     }
463     $fileCreated = Get-ChildItem
($filedirectory+"\Processed\"+$finalfilename+".xlsx")
464     Write-Output "Report created: $fileCreated"
465
466     [System.Runtime.InteropServices.Marshal]::ReleaseComObject($ws1) | Out-Null
467     [System.Runtime.InteropServices.Marshal]::ReleaseComObject($ws3) | Out-Null
468     [System.Runtime.InteropServices.Marshal]::ReleaseComObject($workbook) | Out-
Null
469     [System.Runtime.InteropServices.Marshal]::ReleaseComObject($excel) | Out-Null
470
471
472     Write-Output "`nReport created for the $filetype file`n"
473 }
474
475 Get-Process EXCEL | Stop-Process
476
477 #Writes Errors to log file
478 if ($Error.Count -gt 0)
479 {
480     Write-Verbose $("Error Count: " + $Error.count)
481
482     for ($i=0 ; $i -lt $Error.count ; $i++)
483     {
484         Add-Content -path $ErrorLogFile -value $Error[$i].InvocationInfo.Line
485         Add-Content -path $ErrorLogFile -value
$Error[$i].InvocationInfo.PositionMessage
486         Add-Content -path $ErrorLogFile -value "*****"
487     }
488
489     if ($StopOnError)
490     {
491         Write-Verbose "Stopping due to error"
492
493         Return
494     }
495 }
496
497 Get-Process Outlook | Stop-Process -Force
498
499 #ends session transcript
500 if ($SessionTranscript)
501 {
502     "TranscriptLogFile: " + $TranscriptLogFile
503
504     Stop-Transcript
505 }

```

```
506 "Transcript log file stopped"  
507 }
```