```powershell
#--------------------------------------------------------------------------------------------------
#Get-IDFScanAutomation.ps1
#
#Written by: Matt Logue
#
#Last Modified by: Matt Logue
#
#Last Modified: April 9, 2020
#
#Version: 2.3
#
#Purpose: The script looks for attachements using outlook for CSV exports of ID Finder scans, creates a pivot table daily, then the most recent scan gets uploaded
#to a sharepoint document library by Get-IDFinderReports.ps1 running on the on-premise sharepoint node
#
#--------------------------------------------------------------------------------------------------

<#
.SYNOPSIS
  The script looks for attachements using outlook for CSV exports of ID Finder scans, creates a pivot table daily, then the most recent scan gets uploaded
#to a sharepoint document library by Get-IDFinderReports.ps1 running on the on-premise sharepoint node

.DESCRIPTION
  Requires
  1. Connection to the shared folder path and modify the permissions
    2. Outlook with profile setup for the idfinder@hsc.wvu.edu email account

.PARAMETER
  No required parameters

.NOTES
  Author: Matt Logue
      Last Updated by: Matt Logue
    Last Updated: April 8, 2020
#>

[CmdletBinding()]
param (
  [switch]$SessionTranscript = $true,
    [string]$LogFilePath = "C:\Users\microsoft\Documents\GitHub\HSC-PowerShell-Repository\Get-IDFScanAutomation\Logs",
  [switch]$StopOnError = $true, #keep this value true - will error if no filetype gets passed to script
  [int]$DaysToKeepLogFiles = 5, #this value used to clean old log files

  #File specific param
    [string]$scriptTest = $false,
  [string]$filepath = "\\hs.wvu-ad.wvu.edu\Public\ITS\security services\ID Finder Scan Files\"
  )

Set-Location C:\Users\microsoft\Documents\GitHub\HSC-PowerShell-Repository

#Change PowerShell window title
$Host.UI.RawUI.WindowTitle = "Get-IDFScanAutomation.ps1"
```

```

#Add references to file containing needed functions
Import-Module .\1HSC-PowerShell-Modules\HSC-CommonCodeModule.psm1

#Verify the log file directory exists
if ([string]::IsNullOrEmpty($LogFilePath))
{
  Write-ColorOutput -Message "Log file path is null" -ForegroundColor "Yellow"
  Write-ColorOutput -Message "Setting log path to C:\Users\microsoft\Documents\GitHub\HSC-PowerShell-Repository\Get-IDFScanAutomation\Get-IDFScanAutomation\Logs" -ForegroundColor "Yellow"

  $LogFilePath = "C:\Users\microsoft\Documents\GitHub\HSC-PowerShell-Repository\Get-IDFScanAutomation\Logs"

  if (!(Test-Path $LogFilePath))
  {
    New-Item $LogFilePath -Type Directory -force
  }

  if ($Error.Count -gt 0)
  {
    $Continue = Read-Host "There was an error creating the log file directory. Continue(y/n)"

    if ($Continue -ne "y")
    {
      exit
    }
  }
}

#Common variable declarations
$TranscriptLogFile = $LogFilePath + "\" + (get-date -format yyyy-MM-dd) +  "-IDF-ScanAutomation-SessionOutput.txt"
$ErrorLogFile = $LogFilePath + "\" + (get-date -format yyyy-MM-dd) +  "-IDF-ScanAutomation-ErrorLogFile.txt"

if ($SessionTranscript)
  {

    try
    {
      Stop-Transcript -ea "Stop"
    }
    catch
    {
    }

    "TranscriptLogFile: " + $TranscriptLogFile
    Start-Transcript -Path $TranscriptLogFile -Force
    "Transcript log file started"
  }

#Cleaning up old log files
Write-Verbose "Removing Old Log Files"
Remove-OldLogFile -Days $DaysToKeepLogFiles -Path $LogFilePath

$Error.clear()

```

```powershell
##########################################
#          Main Script Function          #
##########################################

#Stops any existing Outlook process
If ((Get-process Outlook -ErrorAction SilentlyContinue)) {
  Get-Process Outlook | stop-process -force
  Start-Sleep 30
    if (Get-Process Outlook -ErrorAction SilentlyContinue) {
    Get-Process Outlook | stop-process -force
    }
}

$error.Clear()

# This code block searches for the most recent report sent to idfinder@hsc.wvu.edu
Add-type -assembly “Microsoft.Office.Interop.Outlook” | out-null

$olFolders = “Microsoft.Office.Interop.Outlook.olDefaultFolders” -as [type]

if(([System.Diagnostics.Process]::GetProcesses()).name | ?{$_ -like 'outlook*'})
{
    write-host Outlook is running!
  $outlook =
[Runtime.Interopservices.Marshal]::GetActiveObject('Outlook.Application')
}
else {
$outlook = new-object -comobject outlook.application
#$outlook.OpenProfile("IDFinder")
}
$namespace = $outlook.GetNameSpace("MAPI")
Write-Output "Opening ID Finder Mailbox"
$sourceFolder = $namespace.Folders.Item("ID Finder").Folders.Item("Inbox")

$inbox = $sourceFolder.items | Select-Object -Property
senderName,SenderEmailAddress,Subject,ReceivedTime,Attachments | select -first 25
Write-Output "Processing Emails..."
$emails = $inbox | Where -Property SenderEmailAddress -eq "infosec@mail.wvu.edu" |
Sort-Object ReceivedTime -Descending | Select -first 25 -Property
Subject,Attachments,ReceivedTime


$filepath = "\\hs\public\ITS\security services\ID Finder Scan Files\"
$folderpath = $($filepath + $(Get-Date -Format yyyy-MM-dd))
if (!(Test-Path -Path $folderpath))
{
    New-Item -Path $folderpath -ItemType Directory -Force
}

$emails.attachments | `

foreach {

    Write-Host "Found File:" $_.filename
    $attr = $_.filename
    #add-content $log1 "Attachment: $attr"
    $a = $_.filename
    $fileName = "$(Get-Date -Format yyyyMMdd)" + "_" + $a
    If ( ($a.Contains("csv")) -AND ($a -notlike "*count*") -AND
($a.Contains("IDF2")) )
```

```
    {
        $filename = $filename -replace "_Locations",""
        $_.saveasfile((Join-Path $folderpath $fileName))
        Write-Output "Attachment Saved: $folderpath\$filename"
    }
}

Write-Output "Number of attachments saved: $((Get-ChildItem $folderpath | Where-
Object{!($_.PSIsContainer)}).count)"
Write-Output "****************************`n`n"


############################################################
#        Code block for creating the Pivot table           #
#        from saved attachments                            #
############################################################

#following formats the date for use in the file names and searching for the file
$date = Get-Date -Format u
$date = $date -split " "
$date = $date[0] -replace "-",""

$filetypes = @("CCs","SSNs","ePHI","TAXES")
foreach ($filetype in $filetypes)
{
    $nofilecount=0
    $filename = $date+'_IDF2_HSC_'+$filetype+'.csv'
    if (!(Test-Path "$folderpath\$filename")){
        Write-Output "No CSV Found called $filename"
        $nofilecount++
        if($nofilecount -ge 2) {
        Write-Output "Not enough CSV's Found"
            Exit 1
        }
    }
    $csvFile = (Get-ChildItem "$folderpath\$filename") #| sort LastWriteTime |
select -last 1
    $filedirectory = Split-Path -Path $csvFile -Parent
    $csvfilename = Split-Path -Path $csvFile -leaf
    Write-Output "File Found: $filedirectory\$csvfilename"
    $finalFilename = $csvfilename -replace ".csv",""

    #Create excel COM object
    $excel = New-Object -ComObject excel.application

    #Make Visible
    $excel.Visible = $false
    $excel.Caption = "IDFinder_HSC_"+$filetype
    $xlWindowState = [Microsoft.Office.Interop.Excel.XLWindowState]
    $excel.DisplayAlerts = $FALSE
    #$excel.Application.WindowState = $xlWindowState::xlMinimized

    #Add a workbook
    $workbook = $excel.Workbooks.Add()
    $ws1 = $workbook.Worksheets.Add()


    #Connect to first worksheet to rename and make active
    $dataSheet2 = $workbook.Worksheets.Item(1)
    $dataSheet2.Name = "PivotTable"
    $dataSheet2.Activate() | Out-Null
    $dataSheet = $workbook.Worksheets.Item(2)
    $dataSheet.Name = "DataDump"
    Write-Output "Excel Workbook created"

    if ((Get-Content $csvFile).Contains("sep=,"))
    {
       (Get-Content $csvFile) | Select-Object -Skip 1 | Set-Content $csvFile
#removes "sep=," line from begining of csv
    }

    $csvFileProcessed = Import-Csv -Path $csvFile
    $processes = $csvFileProcessed


  if ($filetype -eq "TAXES")
    {
    $dataSheet.cells.item(1,1) = "Endpoint"
    $dataSheet.cells.item(1,2) = "Tag Name"
    $dataSheet.cells.item(1,3) = "Location"
    $dataSheet.cells.item(1,4) = "Unprotected Quantity"

    }
  else {
    $dataSheet.cells.item(1,1) = "Tag Name"
    $dataSheet.cells.item(1,2) = "Endpoint"
    if ($filetype -eq "CCs")
    {
      $dataSheet.cells.item(1,3) = "Unprotected CreditCards"
    }
    if ($filetype -eq "SSNs")
    {
      $dataSheet.cells.item(1,3) = "Unprotected SSNs"
    }
    if ($filetype -eq "ePHI")
    {
      $dataSheet.cells.item(1,3) = "Unprotected Quantity"
    }

    $dataSheet.cells.item(1,4) = "Location"
  }

    Write-Output "Importing Raw data to DataDump Worksheet..."
    $i = 2
    if ($filetype -eq "CCs")
    {
        foreach($process in $processes)
        {
         $dataSheet.cells.item($i,1) = $process.'Tag Name'
         $dataSheet.cells.item($i,2) = $process.Endpoint
         $dataSheet.cells.item($i,3) = $process.'Unprotected CreditCards'
         $dataSheet.cells.item($i,4) = $process.Location
         $i++
        } #end foreach process
    } #end if
    if ($filetype -eq "SSNs")
    {
        foreach($process in $processes)
        {
         $dataSheet.cells.item($i,1) = $process.'Tag Name'
         $dataSheet.cells.item($i,2) = $process.Endpoint
         $dataSheet.cells.item($i,3) = $process.'Unprotected SSNs'
         $dataSheet.cells.item($i,4) = $process.Location
         $i++
        } #end foreach process
    } #end if

    if ($filetype -eq "ePHI")
    {
        foreach($process in $processes)
        {
         $dataSheet.cells.item($i,1) = $process.'Tag Name'
         $dataSheet.cells.item($i,2) = $process.Endpoint
         $dataSheet.cells.item($i,3) = $process.'Unprotected Quantity'
         $dataSheet.cells.item($i,4) = $process.Location
         $i++
        } #end foreach process
    } #end if

  if ($filetype -eq "TAXES")
    {
        foreach($process in $processes)
        {
         $dataSheet.cells.item($i,1) = $process.'Endpoint Name'
     $dataSheet.cells.item($i,2) = $process.'Tag Name'
     $dataSheet.cells.item($i,3) = $process.Location
         $dataSheet.cells.item($i,4) = $process.'Unprotected Quantity'
         $i++
        } #end foreach process
    } #end if


    # rename workbook
    $workbook = $workbook

    # looks for names of worksheets
    $ws3 = $workbook.worksheets | where {$_.name -eq "PivotTable"} #<-------
Selects sheet 3
    $ws1 = $workbook.worksheets | where {$_.name -eq "DataDump"}

    $xlPivotTableVersion15     = 5
    $xlPivotTableVersion14     = 4
    $xlPivotTableVersion12     = 3
    $xlPivotTableVersion10     = 1
    $xlCount                 = -4112
    $xlDescending             = 2
    $xlDatabase                = 1
    $xlHidden                  = 0
    $xlRowField                = 1
    $xlColumnField             = 2
    $xlPageField               = 3
    $xlDataField               = 4
    $xlDirection        = [Microsoft.Office.Interop.Excel.XLDirection]

    #looks for final non-empty row
    $finalrow=$ws1.Cells.End($xlDirection::xlDown).Row

    #creates pivot datasource on $ws1 then creates pivot table on $ws3 at position
R1C1
```

```
    $PivotTable = $workbook.PivotCaches().Create($xlDatabase, $ws1.name + "!R1C1:R"+ $finalrow +"C4", $xlPivotTableVersion15)
    $PivotTable.CreatePivotTable($ws3.name + "!R1C1","Tables1") | Out-Null
    [void]$ws3.Select();
    $ws3.Cells.Item(1,1).Select() | Out-Null
    #hide Pivot Table field list
    $workbook.ShowPivotTableFieldList = $false

    #creates pivot table fields
    $PivotFields1 = $ws3.PivotTables("Tables1").PivotFields("Tag Name")
    $PivotFields1.Orientation = $xlRowField
    $PivotFields1.Caption = "Units"

    $PivotFields2 = $ws3.PivotTables("Tables1").PivotFields("Endpoint")
    $PivotFields2.Orientation = $xlRowField

    $PivotFields5 = $ws3.PivotTables("Tables1").PivotFields("Location")
    $PivotFields5.Orientation = $xlRowField

    $PivotFields3 = $ws3.PivotTables("Tables1").PivotFields("Location")
    $PivotFields3.Orientation = $xlDataField
    #looks at $filetype for naming of 2nd column of pivot table
    if ($filetype -eq "CCs")
    {
      $PivotFields3.Name = "Files with Unprotected CreditCards"
    }
    if ($filetype -eq "SSNs")
    {
      $PivotFields3.Name = "Files with Unprotected SSNs"
    }
    if ($filetype -eq "ePHI")
    {
      $PivotFields3.Name = "Files with Unprotected Quantities"
    }
  if ($filetype -eq "TAXES")
    {
      $PivotFields3.Name = "Files with Unprotected Quantities"
    }

    #looks at $filetype for creating 3rd column of pivot table
    if ($filetype -eq "CCs")
    {
      $PivotFields4 = $ws3.PivotTables("Tables1").PivotFields("Unprotected CreditCards")
      $PivotFields4.Orientation = $xlDataField
      $PivotFields4.Function = $xlSum
      $PivotFields4.Caption = "Sum of Unprotected CreditCards"
    }

    if ($filetype -eq "SSNs")
    {
      $PivotFields4 = $ws3.PivotTables("Tables1").PivotFields("Unprotected SSNs")
      $PivotFields4.Orientation = $xlDataField
      $PivotFields4.Function = $xlSum
      $PivotFields4.Caption = "Sum of Unprotected SSNs"
    }
    if ($filetype -eq "ePHI")
    {
      $PivotFields4 = $ws3.PivotTables("Tables1").PivotFields("Unprotected Quantity")
      $PivotFields4.Orientation = $xlDataField
      $PivotFields4.Function = $xlSum
      $PivotFields4.Caption = "Sum of Unprotected Quantities"
    }
  if ($filetype -eq "TAXES")
    {
      $PivotFields4 = $ws3.PivotTables("Tables1").PivotFields("Unprotected
Quantity")
      $PivotFields4.Orientation = $xlDataField
      $PivotFields4.Function = $xlSum
      $PivotFields4.Caption = "Sum of Unprotected Quantities"
    }

    #sorts pivot table by the largest numbers in the 4th pivot field
    $PivotFields1.AutoSort($xlDescending, $PivotFields4.Name)

    #collapses all row fields
    Foreach ($PivotField in $ws3.PivotTables("Tables1").PivotFields()) {
        try {
        $PivotField.ShowDetail = $False
        }
        catch {
        #this throws and unhanded exception if it's not in a try/catch block to
ignore the error
        }
    }

    Write-Output "Created PivotTable"
    #Changes formating of table also based on file type
    $row = 1
    $Column = 1
    $ws3.Cells.Item($row,$column)= 'Units'
    if ($filetype -eq "CCs")
    {
      $ws3.PivotTables("Tables1").TableStyle2 = "PivotStyleMedium14"
    }
    if ($filetype -eq "SSNs")
    {
      $ws3.PivotTables("Tables1").TableStyle2 = "PivotStyleMedium12"
    }
    if ($filetype -eq "ePHI")
    {
      $ws3.PivotTables("Tables1").TableStyle2 = "PivotStyleMedium13"
    }
  if ($filetype -eq "TAXES")
    {
      $ws3.PivotTables("Tables1").TableStyle2 = "PivotStyleMedium10"
    }
    $ws3.columns.item(1).columnWidth = 60
    $ws3.PivotTables("Tables1").HasAutoFormat = $false
  $ws3.PivotTables("Tables1").ShowTableStyleRowStripes = $true
    Write-Output "PivotTable Formatted"

    #script block for testing to make excel table visible
    if ($scriptTest -eq $true)
    {
     $excel.Visible = $true
     $excel.Application.WindowState = $xlWindowState::xlMaximized
    }

```

```powershell
    if (!(Test-Path "$filedirectory\Processed"))
    {
        New-Item -Path "$filedirectory\Processed" -ItemType Directory
    }
    #save files
    $workbook.activate()
    $workbook.SaveAs($filedirectory+"\Processed\"+$finalfilename+".xlsx")
    #if $script test is false then it will close the workbook and release the variables
    if ($scriptTest -eq $false)
    {
      $Excel.Workbooks.Close()
      $Excel.Quit()
    }
    $fileCreated = Get-ChildItem ($filedirectory+"\Processed\"+$finalfilename+".xlsx")
    Write-Output "Report created: $fileCreated"

    [System.Runtime.Interopservices.Marshal]::ReleaseComObject($ws1) | Out-Null
    [System.Runtime.Interopservices.Marshal]::ReleaseComObject($ws3) | Out-Null
    [System.Runtime.Interopservices.Marshal]::ReleaseComObject($workbook) | Out-Null
    [System.Runtime.Interopservices.Marshal]::ReleaseComObject($excel) | Out-Null


    Write-Output "`nReport created for the $filetype file`n"
}

Get-Process EXCEL | Stop-Process

#Writes Errors to log file
if ($Error.Count -gt 0)
{
  Write-Verbose $("Error Count: " + $Error.count)

  for ($i=0 ; $i -lt $Error.count ; $i++)
  {
    Add-Content -path $ErrorLogFile -value $Error[$i].InvocationInfo.Line
    Add-Content -path $ErrorLogFile -value $Error[$i].InvocationInfo.PositionMessage
    Add-Content -path $ErrorLogFile -value "******************************"
  }

  if ($StopOnError)
  {
    Write-Verbose "Stopping due to error"

    Return
  }
}

Get-Process Outlook | Stop-Process -Force

#ends session transcript
if ($SessionTranscript)
  {
    "TranscriptLogFile: " + $TranscriptLogFile

    Stop-Transcript

    "Transcript log file stopped"
}
```