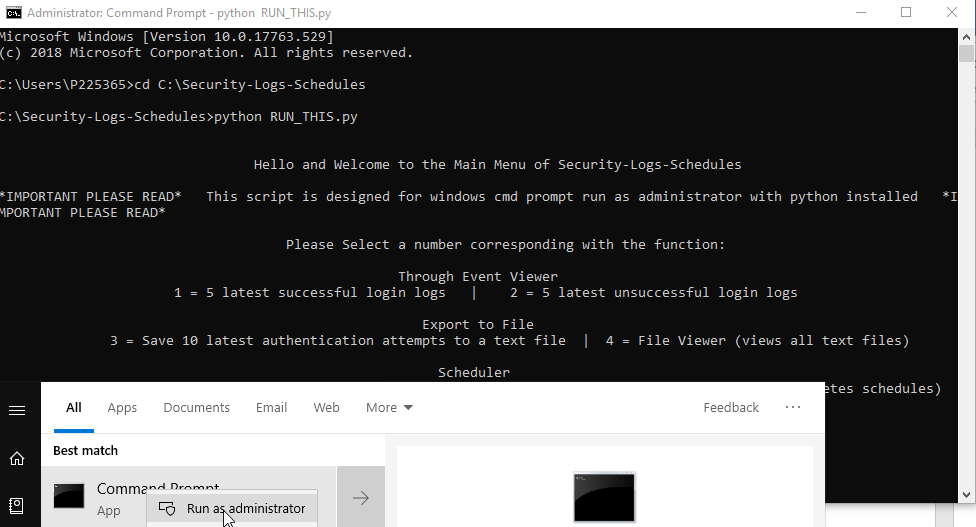
Security-Logs-Schedules



Technical and User Documentation

Author: Mario Pereira

**Contents**

[**Introduction** 3](#_Toc10901944)

[**Terminology, Abbreviations and Acronyms** 3](#_Toc10901945)

[**Executive Summary** 3](#_Toc10901946)

[**Technical Requirements** 4](#_Toc10901947)

[**Design and Code** 4](#_Toc10901948)

[**User Documentation** 10](#_Toc10901949)

[**Conclusions** 10](#_Toc10901950)

[**References** 11](#_Toc10901951)

[**Appendices** 11](#_Toc10901952)

## **Introduction**

Security-Logs-Schedules is a python script designed for the purpose of viewing security audit events 4624 and 4625 and saving the latest 10 log on audit events into txt files, it includes a file viewer to see the data written to the log files and a scheduler which can create a new scheduled task in Task Scheduler to run at a specified time triggering the script “CREATEFILEWITHLOGDATA” which will generate a txt file with the 10 latest log on audit events(ID: 4624 and 4625).

## **Terminology, Abbreviations and Acronyms**

1. Python

* Python is a scripting language and the software to write and run it is available in the References portion of this document.

1. Pseudocode

* Not real working code, this code serves the purpose of design communication between humans, by using appropriate English for replacing coding syntax.

1. IDE

* Stands for Integrate Development Environment, this is a software application that facilitates coding by providing the coder with source code editors, build automation, debuggers and other relevant coding features.

1. Docstring

* Simply put this is a multi-line string that is not assigned to any code but is specified beside coding modules or definitions to explain the process within said modules/definitions running inside the script file.

1. Readme

* This is a txt file included within the folder that holds the script. Readme files usually provide a summary of the program and its purpose, they also include special requirements, changes between versions and links to other documentation.

1. PowerShell

* This is a task framework command line shell with an integrated scripting language available natively to Windows.

## **Executive Summary**

This document’s purpose is to communicate the design, code, running requirements and author’s logic for creating and running the script “Security-Logs-Schedules.py”. The Technical Requirements entail software dependencies and how to set up the script to run successfully without any errors by detailing directories for the python script and a manner of ensuring administrative privileges are enabled during script execution. This document provides a video guide to elaborate user documentation and a Peer Review Form for community feedback in the appendices.

## **Technical Requirements**

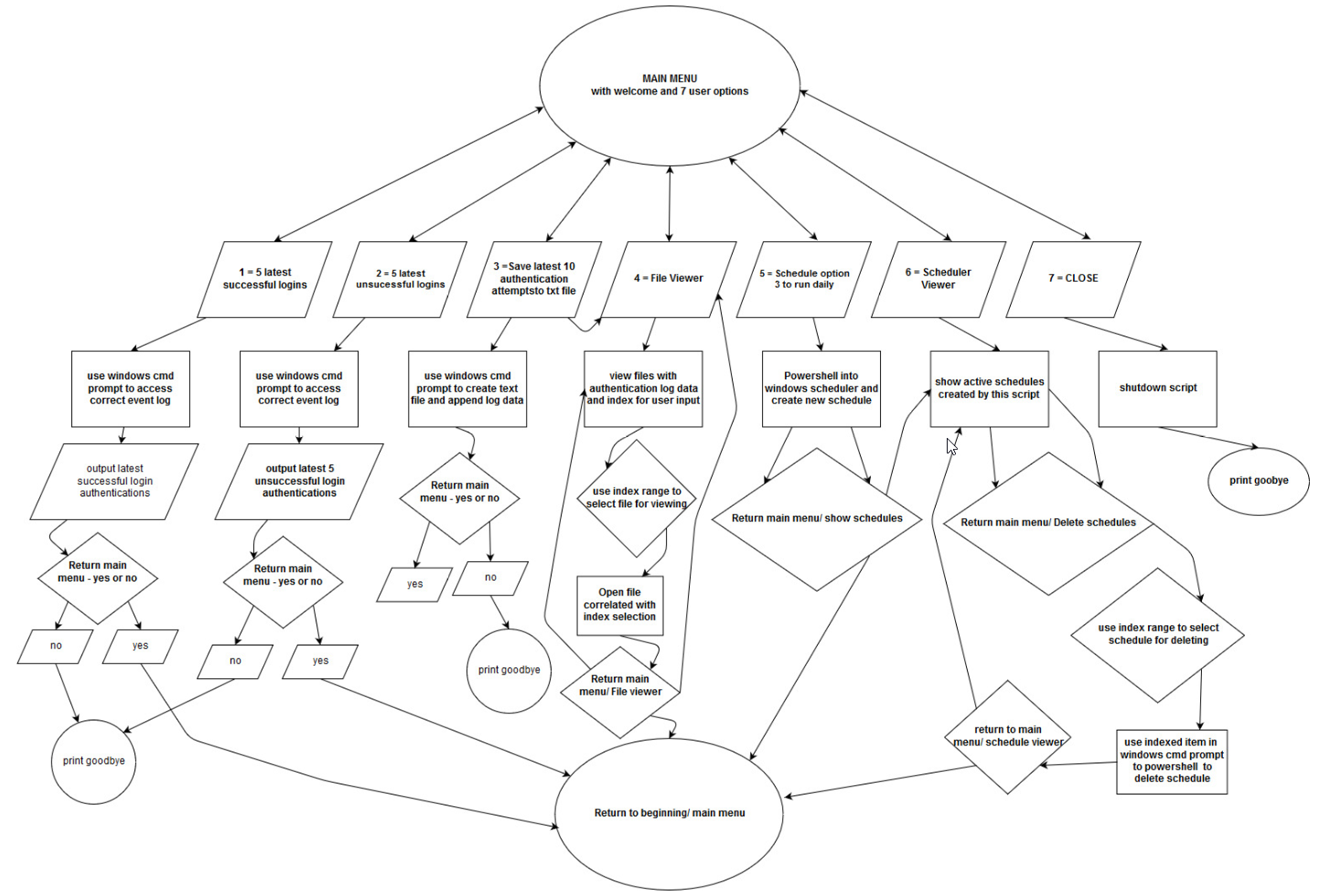
1. Administrative Privilege

* CMD Prompt must be run as Administrator
* Python must call script to run from Admin CMD Prompt
* Security Event Audits must be enabled

1. Software Dependencies

* Designed for Windows 10
* Python installed to PATH
* PowerShell

## **Design and Code**

1. Flow Chart
2. Pseudocode

defining view\_and\_delete\_schedules function

command = cmd prompt('schtasks /query | findstr /i "Created"')

result = subprocess(command)

choice = (enumerate lines(result))

print (result)

askuser = input "delete schedule? y/n - "

min = 0

if askuser == y:

print(choice)

nowask = "type the number beside the schedule u want to delete"

if nowask <= min

deletecmd = result(nowask)

cmd prompt ( 'schtasks /delete /tn "' + deletecmd + '" /f'

else

Print 'invalid input'

elif askuser == n:

print "returning to main menu"

main\_menu

defining log\_scheduler

askuser = input "What time would you like the script to run everyday?"

time = timestamp

uniquefilename = "Created" + time

longstring\_for\_ps1\_lines = poweshell script e.g. "Start-Process powershell -Verb runAs \n"

"$action = New-ScheduledTaskAction -Execute 'Python.exe' -Argument 'C:\Security-Logs-Schedules\CREATEFILEWITHLOGDATA.py'\n",

"$trigger = New-ScheduledTaskTrigger -Daily -At " + time + "\n",

"Register-ScheduledTask -Action $action -Trigger $trigger -TaskName '" + uniquefilename +

"' -Description 'This creates logs of the latest authentications recorded at the set time in the Python Assessment Folder'\n"

create file called uniquefilename to write into

write longstring\_for\_ps1\_lines

save and exit file

cmd prompt ( powershell 'locationOfTheScript\uniquefilename.ps1'

defining da\_file\_eyes

files = list current working directory but only .txt files

enumerate files

askuser = "which line would you like to read?"

min = 0

if askuser <= min

setchoice = (files[askuser])

open file in working directory called setchoice to read

elif askuser ==

print 'u forgot to type'

else

print invalid input

defining da\_file\_genie

time = timestamp

logname = AuthenticationLog + time + .txt

create file called logname to write into

cmd prompt (audit event log and export data to file - "powershell Get-EventLog -logname security -instanceid 4625,4624 -newest 10 > + logname")

defining ReRun

askuser = input "Return to the main menu? (y/n):"

if askuser == y

main\_menu

elif askuser == n

print 'ok, goodbye then.'

break/finish script

else

print invalid input

ReRun

defining the main\_menu

print print(" Please Select a number corresponding with the function:")

print(" ")

print(" Through Event Viewer")

print(" 1 = 5 latest successful login logs | 2 = 5 latest unsuccessful login logs")

print(" Export to File")

print(" 3 = Save 10 latest authentication attempts to a text file | 4 = File Viewer (views all text files)")

print(" Scheduler")

print(" 5 = Schedule option 3 to run at a particular time | 6 = Schedule Viewer (views and deletes schedules)")

print("Exit:")

print("7 = Close")

askuser = input "input number and press enter - "

if askuser == 1

output = cmd prompt ( powershell get windows event ID 4624)

print output

ReRun

elif askuser == 2

output = cmd prompt ( powershell get windows event ID 4625)

print output

ReRun

elif askuser == 3

da\_file\_genie

ReRun

elif askuser == 4

da\_file\_eyes

ReRun

elif askuser == 5

log\_scheduler

ReRun

elif askuser == 6

view\_and\_delete\_schedules

ReRun

elif askuser == 7

print "goodbye"

break/shutdown script

else

print 'Invalid Input'

1. IDE

* Jetbrain’s Pycharm Community Edition 2018.3.4

1. Docstrings

* def view\_and\_delete\_schedules():

"""The purpose of function view\_and\_delete\_schedules is to view the schedules created by this script only, there are

function within this one which serve for the purpose of input validation as well allowing for more functionality like

deleting the viewed schedules from the console in the running script"""

* def deletor():

"""The deletor function serves the view\_and\_delete\_schedules function in deleting the schedules"""

* def out(command):

"""This out(command) function is simply a subprocess for querying task scheduler for schedules with the word 'created'"""

* def log\_scheduler():

"""The function log\_scheduler's purpose is to create a task schedule for a specific time that will create a log file

of the l0 latest security audit event IDs 4624 and 4625, it does this by creating a powershell .ps1 file with CMD prompt

and then opening it with powershell as a local admin, this kind of makes the script location specific - if the script

is run from any directory other than 'C:\Security-Logs-Schedules' it wont be able to call the powershell .ps1 file"""

* def da\_file\_eyes():

"""This function - da\_file\_eyes was created to view the script directory files, but only show whats relevant - the log

authentication .txt files"""

* def da\_file\_genie():

"""This function - da\_file\_genie creates the latest 10 login security audit logs"""

* def menu():

"""This is my main menu, to give it a basic menu feel I have been a bit silly and added some fat spacing, manually attempting

to allign text to center. The function serves as a menu with if and elifs as options for the above functions."""

* def ReRun(); AltReRun2(); alt\_invalid(); AltReRun():

"""The functions ReRun, AltReRun2, alt\_invalid and AltReRun all serve as input validation and will prevent invalid user

input from breaking the scripts functionality."""

1. Readme File

Security-Logs-Schedules

Security-Logs-Schedules is an Admin tool developed for reviewing event IDs 4625, 4624 (successful and failed login attempts) it allows one to create logs of unsuccessful/successful logins and view them in the running script, the script also offers a Scheduler, which will create task schedules for auditing the event log for the last 10 login authentications.

Purpose

Security-Logs-Schedules is a python script designed for the purpose of viewing security audit events 4624 and 4625 and saving the latest 10 log on audit events into txt files, it includes a file viewer to see the data written to the log files and a scheduler which can create a new scheduled task in Task Scheduler to run at a specified time triggering the script “CREATEFILEWITHLOGDATA” which will generate a txt file with the 10 latest log on audit events(ID: 4624 and 4625).

Requirements

1. Designed and Tested for Windows 10

2. Must run Python script through CMD prompt with Administrative Privileges

3. Python must be installed to path

4. PowerShell is installed and accessible

5. Security Event Auditing enabled.

Guide

This script will only run properly from path "C:/Security-Logs-Schedules" <--- This cannot be renamed.

CMD PROMPT TO SCRIPT <cd C:/Security-Logs-Schedules>

<"C:/Security-Logs-Schedules\RUN THIS.py"> --------- dont type <>

Everything should work perfectly fine if user is administrator and has run cmd prompt as admininstrator and brought executed file through python.

Was designed with Pycharm so should run perfectly fine to run on there if started Pycharm application is started as admin.

User video guide available at -

## **User Documentation**

1. A brief guide is in the readme file, which is available just above.
2. The official User Documentation for this script is available in the form of a video on YouTube - <https://youtu.be/lq-BrCckxwU>

## **Conclusions**

The script which has been documented here can be considered as an admin utility, it provides a semi-automated way of auditing for security ID 4625 and 4624 in the Windows Security Event Log, create files and can schedule the local PC to create those files. This requires minimum setup/installation processes but does require guidance in the form of Readme instructions as there are environment parameters such as script location and admin privilege to consider when running the script, failure to follow these minimal steps will lead to at least 3 of the functions in the script crashing or giving PowerShell errors. The documentation and script are distributed on GitHub under the author name “Monsid”, and a video user guide is available on YouTube (see User Documentation section above).

## **References**

1. Software used to Design and Produce Script

* Creating Flowcharts - www.draw.io
* Writing Pseudocode – Notepad
* IDE for writing Code – Jetbrain’s Pycharm Community Edition 2018.3.4

1. Software used to test script.

* Python 3.7 from console.
* Jetbrain’s Pycharm in debug mode.
* Google forms – peer review testing.
* VMware Workstation – changing between desktop environment variables

1. Software used to produce Documentation.

* Microsoft Word
* Notepad
* Radeon ReLive – video capture
* DaVinci Resolve by Black Magic Design - <https://www.blackmagicdesign.com/products/davinciresolve/>

## **Appendices**

1. Testing - Peer Review

* https://forms.gle/xrkuX1w5k4nud4Gp6 currently ongoing, feel free to include a response.

1. Distribution – Github

* <https://github.com/Monsid/Security-Logs-Schedules>