1. Host Survey Quick Start

1.1 Save Credentials

Run the secret_snacktime.ps1 script, follow the prompts, and save credentials. Explanation of this script is in section 2.2.

Syntax is: .\secret_snacktime.ps1

When prompted for the file name and location enter the path to the credentials.

..\Config\creds.xml is recommended.

1.2 Run Speed Test

Run the speed_test.ps1 script. Take note of the amount of time it takes to complete about 3 iterations. Explanation of this script is in section 2.12. This script is a good opportunity to troubleshoot domain authentication issues, be sure to review the troubleshooting steps in 2.12.

Syntax is: .\speed_test.ps1 -ip <target_ip> -cred_path ..\Config\creds.txt -count 3

1.3 Begin Orchestrator Script

Schedule proc_hunter.ps1 to run at a regular interval with the orchestrator script, explained in section 2.11. Keep in mind the time from the previous step, and make sure that the next scan does not begin before the previous one finishes. This will eventually overwhelm the host's resources.

Syntax is: .\orchestrator.ps1 -cred_path ..\Config\creds.xml -out_dir ..\Data\ -host_list ..\Config\hosts.txt -sleep_seconds 60

1.4 Run Single-Use Scripts

Run heavier scripts that do not need to run at regular intervals. This includes the following:

- el_clear_finder.ps1
- psexec_finder.ps1
- service_hunter.ps1
- watson.ps1

Syntax is: .\el_clear_finder.ps1 -ip <target_ip> -cred_path ..\Config\creds.xml -out_dir ..\Data

Syntax is: .\psexec_finder.ps1 -ip <target_ip> -cred_path ..\Config\creds.xml -out_dir ..\Data

Syntax is: .\service_hunter.ps1 -ip <target_ip> -cred_path ..\Config\creds.xml -out_dir ..\Data

Syntax is: .\watson.ps1 -ComputerName <target_ip> –cred_path ..\Config\creds.xml

1.5 Upload Results to Kafka

Run the relevant Kafka upload script. Depending on the network configuration, this may require some route configuration on the Windows host sensor. Success will return a null error. More information about this step in 2.8.

To upload and archive all files in a directory, opening and leaving a terminal like this will suffice:

While(1) { Get-ChildItem -Filter *.csv | %{ .\kafka-upload-csv.ps1 -server_ip <ip_here> -kafka_topic <relevant_topic> -server_port <usually_8082> -file_path $_.FullName; Move-Item $_.FullName ..\Archives\$_}; sleep 300; }

This checks the Data directory every five minutes for CSV files, sends them to Kafka, and moves them to the archive directory.

If there are more file types, you can specify further which files are sent to which topics with a filter. All output of proc_hunter.ps1 begins with "process", so replacing

Get-ChildItem -Filter *.csv

with

Get-ChildItem -Filter process*.csv

will only upload and archive output from proc_hunter.ps1.

Syntax is: .\kafka-upload-csv.ps1 --server_ip <kafka_ip> -kafka_topic <relevant_topic> -server_port 8083 -file_path test.csv

1.6 Dump Remote Process

When an interesting process is identified, dump it remotely using the Judge Dredd script. Detailed instructions found in 2.1. For system level processes, use Bad Cop. Detailed instructions found in 2.6.

Syntax is: .\judge_dredd.ps1 -ip <target_ip> -proc_name cmd -cred_path ..\Config\creds.xml -proc_pid <pid>

1.7 Collect Interesting Files

Use the Liam Neeson script to search the remote file system for files and transfer them back to the local machine. Detailed instructions are found in 2.5.

Syntax is: .\liam_neeson.ps1 –ip <target_ip> -cred_path ..\Config\creds.xml –file_name example.txt

2. PowerShell Descriptions and Troubleshooting steps

2.1. Judge Dredd (Remote Process Dump)

What does it do?

Judge Dredd is a PowerShell script that dumps remote processes. You can specify a process name (required) to dump all processes of that name, and a PID (optional) if you want only a single process dump.

How does it work?

It uses PS-Remoting and Invoke-Command from PowerShell to dump a process on the remote system. It transfers procdump.exe (from SysInternals Suite) and runs it on the user-specified processes. Finally, the files are transferred one by one to the local host and the working directory is cleaned up and removed.

How do you use it?

Syntax is: .\judge_dredd.ps1 -ip <target_ip> -proc_name cmd –cred_path ..\Config\creds.xml –proc_pid 1863

It is important to note that the credentials must be saved in XML format with the secret_snacktime.ps1 script.

Troubleshooting

1. Verify that the procdump.exe file is located in the same directory as the script.
2. Verify that the credentials are saved in the correct format, and that they work on the remote machine.
3. Verify that the host is up.
4. Verify that the credentials have appropriate permission to dump processes.
5. Verify that a directory called “Dumps” exists in the current working directory’s parent

2.2 Secret Snacktime (Secure Credential Saving)

What does it do?

This script saves the credentials of the user to the local machine as a secure string. The credentials can only be used from this machine, and this script only works on a machine with a functioning trusted platform module (TPM).

How does it work?

The script prompts the user for credentials, converts them to a secure string, and saves them as an XML file. After running the script, open the XML file with Notepad to better understand the format.

How do you use it?

Syntax is: .\secret_snacktime.ps1

Troubleshooting

1. Verify that the machine has a TPM.
2. Verify that the machine is running a recent version of PowerShell (Version 3 or greater)

3. Proc Hunter (Observe Remote Processes)

What does it do?

Proc Hunter is a script that takes output from three commands (Get-Process, Get-WmiObject win32_process, Get-NetTCPConnection) and performs a kind of "in place enrichment" by matching connections with processes if applicable. This data is saved as a CSV on the local machine.

How does it work?

This uses PS-Remoting to run the three commands above and save them as a CSV locally. There is a verbose help menu (try running Get-Help .\proc_hunter.ps1 for explanation and examples).

How do you use it?

Syntax is: .\proc_hunter.ps1 -ip <target_ip> -cred_path ..\Config\creds.xml –out_dir ..\Data

It is important to note that the credentials must be saved in XML format with the secret_snacktime.ps1 script. The out_dir switch usually is followed by "..\Data\" which is recommended, unless there is a specific reason to save to another directory.

Troubleshooting

1. Verify that the credentials are saved in the correct format, and that they work on the remote machine.
2. Verify that the host is up.
3. Verify that the credentials have appropriate permission to view running processes. If this is not the case, the script will still work, there will just be blank data fields in the output.

4. Dr. Watson (Retrieve Existing Crash Dumps)

What does it do?

Dr. Watson is a short PowerShell script that is intended to be run once on the domain. It checks the specified host for existing crash dumps and if they exist, transfers them to the local host.

How does it work?

It uses PS-Remoting to run a recursive search of the machine for any .dmp files. It then copies the files from the session to the local machine and closes the session.

How do you use it?

Syntax is: .\watson.ps1 -ComputerName <target_ip> –cred_path ..\Config\creds.xml

It is important to note that the credentials must be saved in XML format with the secret_snacktime.ps1 script.

Troubleshooting

1. Verify that the credentials are saved in the correct format, and that they work on the remote machine.
2. Verify that the host is up.
3. Verify that the credentials have appropriate permission to read locations of dump files.
4. Verify that the "Dumps" directory exists one directory up from where you are running the scripts.

5. Liam Neeson (Retrieve remote file)

What does it do?

Given a complete file name (or a filename with the * wildcard) this script will recursively search the remote host for all matches and transfer them back to the local machines "Files" directory.

How does it work?

This uses PS-Remoting to establish a session, Invoke-Command to search the remote machine, and copies the file(s) found from the session before tearing it down. The files absolute path is maintained in the file name, so there is no user option to specify the output directory. The transferred files will always appear in the "Files" directory, and the script will fail if this directory does not exist.

How do you use it?

Syntax is: .\liam_neeson.ps1 –ip <target_ip> -cred_path ..\Config\creds.xml –file_name example.txt

The above code finds file with the exact name "example.txt". The following command transfers all text files **(do not do this)**:

.\liam_neeson.ps1 –ip <target_ip> -cred_path ..\Config\creds.xml –file_name *.txt

It is important to note that the credentials must be saved in XML format with the secret_snacktime.ps1 script. The out_dir switch usually is followed by "..\Data\" which is recommended, unless there is a specific reason to save to another directory.

Troubleshooting

1. Verify that the credentials are saved in the correct format, and that they work on the remote machine.
2. Verify that the host is up.
3. Verify that the credentials have appropriate permission to read the files requested.
4. Verify that the "Files" directory exists one directory level up from where the script is being run.

6. Bad Cop (Dump Remote System Level Process)

What does it do?

This script acts exactly as Judge Dredd, except for system level processes.

How does it work?

This script also works exactly like Judge Dredd, except it also transfers PsExec.exe to execute procdump.exe with system level privileges. This script should be used sparingly.

How do you use it?

Syntax is: .\bad_cop.ps1 -ip <target_ip> -proc_name cmd -cred_path ..\Config\creds1.xml

It is important to note that the credentials must be saved in XML format with the secret_snacktime.ps1 script.

Troubleshooting

1. Depending on network configuration, running executable from user directories may be disabled. This will be indicated by an "Access Denied" error while running Bad Cop. An undesirable workaround is to access the PsExec.exe syntax directly with the process PID you wish to dump. This may look something like this:
   .\PsExec.exe \\<target_ip> -s -u <username> -p <password> .\procdump.exe -ma <process_pid>
   This will copy the executable to the System32 directory and execute it from there. This will leave the executable in the directory, so cleanup will need to be done manually.
2. Verify that the credentials are saved in the correct format, and that they work on the remote machine.
3. Verify that the host is up.
4. Verify that the credentials have appropriate permission to read locations of dump files.
5. Verify that the "Dumps" directory exists one directory up from where you are running the scripts.

7. PsExec Finder (Check event logs for PSEXESVC service)

What does it do?

Search host event logs for PSEXESVC, save results as text file with verbose information including date that event was created.

How does it work?

It uses PS-Remoting to access event logs and search for events indicative of PsExec use. This is a simple string match looking for the name of the service created to use PsExec.

How do you use it?

Syntax is: .\psexec_finder.ps1 -ip <target_ip> -cred_path ..\Config\creds.xml -out_dir ..\Data

It is important to note that the credentials must be saved in XML format with the secret_snacktime.ps1 script.

Troubleshooting

1. Verify that the credentials are saved in the correct format, and that they work on the remote machine.
2. Verify that the host is up.
3. Verify that the credentials have appropriate permission to access the event logs.

8. Kafka Upload CSV/JSON (Upload output of scripts to SvS infrastructure)

What does it do?

These two scripts upload either CSV or JSON output from the other host sensor scripts. They interact with Kafka directly and allow the user to specify a topic.

How does it work?

This script uses the Invoke-RestMethod cmdlet to make a web request to send the data from the CSV/JSON file to the infrastructure.

How do you use it?

Syntax is: .\kafka-upload-csv.ps1 --server_ip <kafka_ip> -kafka_topic <relevant_topic> -server_port 8083 -file_path test.csv

If this upload succeeds, the error message will be null. The syntax is the same for JSON scripts.

Troubleshooting

1. If there are errors, make sure the host is up.
2. Verify that there are no Kafka firewall errors.

9. Cleared Event Log Finder (Check event logs for clearing events)

What does it do?

Search host event logs for Event ID indicating that the event logs for Security were cleared.

How does it work?

It uses PS-Remoting to access event logs and search for events with Event ID 1102.

How do you use it?

Syntax is: .\el_clear_finder.ps1 -ip <target_ip> -cred_path ..\Config\creds.xml -out_dir ..\Data

It is important to note that the credentials must be saved in XML format with the secret_snacktime.ps1 script.

Troubleshooting

1. Verify that the credentials are saved in the correct format, and that they work on the remote machine.
2. Verify that the host is up.
3. Verify that the credentials have appropriate permission to access the event logs.

10. Service Hunter (Collect services of host)

What does it do?

Query host and collect information about services.

How does it work?

It uses PS-Remoting to collect information about services and their attributes. Exports a local CSV.

How do you use it?

Syntax is: .\service_hunter.ps1 -ip <target_ip> -cred_path ..\Config\creds.xml -out_dir ..\Data

It is important to note that the credentials must be saved in XML format with the secret_snacktime.ps1 script.

Troubleshooting

1. Verify that the credentials are saved in the correct format, and that they work on the remote machine.
2. Verify that the host is up.
3. Verify that the credentials have appropriate permission to access service listings.

11. Orchestrator (Schedule and manage resources for proc_hunter.ps1)

What does it do?

Manages regular enterprise scans with proc_hunter.ps1, reads host list from specified file and scans at specified interval.

How does it work?

It uses PS-Jobs and PS-Remoting to manage resources and scan the network, writing the files to the specified directory. It reads from a user-specified host list, so this will need to be updated.

How do you use it?

Syntax is: .\orchestrator.ps1 -cred_path ..\Config\creds.xml -out_dir ..\Data\ -host_list ..\Config\hosts.txt -sleep_seconds 60

It is important to note that the credentials must be saved in XML format with the secret_snacktime.ps1 script.

The host list is configured with one host per line, as a simple text file. This script will scan 30 hosts in parallel, and wait for more resources to become available. Once all hosts have been surveyed, the script will sleep for 60 seconds and start again from the beginning. This will continue until the script is manually stopped.

Problems may not be apparent with this script at first, be sure to initially monitor the "Data" directory and watch the file sizes. If there are a lot of empty files, it may be necessary to check credentials. The same is true if there are no files being generated at all.

Note: This script schedules jobs, so if you kill it with ctrl+C, there still may be jobs executing. A quick way to view the jobs is with the Get-Job command. Pipe this to remove job like the following to clear all remaining jobs: Get-Job | %{ Remove-Job $_ }

Troubleshooting

1. Verify that the credentials are saved in the correct format, and that they work on the remote machine.
2. Verify that the host is up.
3. Verify that the credentials have appropriate permission to access service listings.
4. Verify that there are hosts in the hosts.txt file, and that the hosts are valid.

12. Speed Test (Get survey time)

What does it do?

Runs the Proc Hunter survey script several times and sees how long it takes.

How does it work?

Times several iterations with the Get-Date cmdlet, and prints stats to terminal.

How do you use it?

Syntax is: .\speed_test.ps1 -ip <target_ip> -cred_path ..\Config\creds.txt -count 3

It is important to note that the credentials must be saved in XML format with the secret_snacktime.ps1 script. The above command runs the Proc Hunter script three times and displays results.

Troubleshooting

This script is a good opportunity to troubleshoot issues with domain authentication.

1. Verify that the credentials are saved in the correct format, and that they work on the remote machine. This can be accomplished with PsExec (which is in the bin directory) if permitted.

   PsExec.exe \\192.168.2.2 -u <username> -p <password> cmd

   There are domains that require the credentials to be in the format **domain\username**.
2. Verify that the remote host is up.
3. Make sure that PSRemoting is enabled.

   Enable-PSRemoting

   If this command throws an error, you may need the **-SkipNetworkProfileCheck**
4. Make sure the trusted hosts file is correctly configured.

   Set-Item -Path WSMan:\localhost\Client\TrustedHosts -Value * -Force
5. If you have reached this step and nothing still works, asking an admin to clone a working account and remove it from all unnecessary security groups will likely tease out more specific issues, specifically domain trust issues.