- How to translate `Sigma` rule into `FortiSIEM` rule
- `Sigma` rule is a very rich resource where you can make your `FortiSIEM` more powerful in terms of detection. because, it is a portal where all the experts in the field are sharing their expertise through the `Sigma` convention, which relies on the `YAML` rules and `YAML` syntax
- Note that for developing the rule, you need to have logs first in FortiSIEM and need to know how the logs are parsed, then you start building your rule
- In `Sigma` rule we will have multiple fields, but which field in FortiSIEM is this information come?, that's why we need to see the logs first in the system before trying to translate the sigma rule into FortiSIEM rule
  - for this we need to enable some auditing to achieve this

CMD & PS Auditing

```
++++++++++++++++++++++++++++++++
Part I: Command Prompt (cmd.exe)
++++++++++++++++++++++++++++++++
(1) Process Creation
═══════════════════
Logs the “A new process has been created” event natively in the Windows
Security channel which records the invoked processes and by whom

How: Group Policy > Computer Configuration > Policies > Windows Settings
> Security Settings > Advanced Audit Policy Configuration > Detailed
Tracking > Audit Process Creation > Success

(2) Command Line Arguments
═════════════════════════
Records the process creation's command line arguments and fills the
(Process Command Line) field inside the same process creation event

How: Group Policy > Computer Configuration > Policies > Administrative
Templates > System > Audit Process Creation > Include Command Line in
Process Creation Events > Enabled

Event Log path = Event Viewer > Application and Services Logs > Security
Event Log channel name = Security
Event ID: 4688

- OR -
```

Sysmon

Event Log path = Event Viewer > Applications and Services Logs > Microsoft > Windows > Sysmon > Operational
Event Log channel name = Microsoft-Windows-Sysmon/Operational
Event ID: 1

The above auditing is enough for all process creation events issued from PowerShell as well if it's used for invoking simple commands like wget but when using PS commands, things become

harder to track with just the above auditing

For example, the PS Invoke-WebRequest command is actually a script that invokes the windows native process conhost.exe, which can mislead the auditing for use case development.

For that we need to perform further auditing for PowerShell activities by following below second part

++
Part II: Powershell (powershell.exe and powershell_ise.exe)
++
Logs PowerShell operations

(1) Module Logging

Records executed PS commands
Use (*) to record all modules or filter to log only for specific PS Modules

How: Group Policy > Computer Configuration > Policies > Administrative Templates > Windows Components > Windows PowerShell > Turn on Module Logging > Enabled > Show > *

Event Log path = Event Viewer > Application and Services Logs > Windows PowerShell
Event Log channel name = Windows PowerShell
Event ID: 4103

(2) Script Block Logging

```
Records PS running inside scripts or code blocks. Beside logging the
encoded commands, it can log the de-obfuscated commands which were
passed through the -EncodedCommand argument
Carefully enable the “Log script block invocation start / stop events”
option as it can increase log volume significantly for logging the
start/stop events (detect slow attacks)

How: Group Policy > Computer Configuration > Policies > Administrative
Templates > Windows Components > Windows PowerShell > Turn on PowerShell
Script Block Logging > Enabled

Event Logs path = Event Viewer > Applications and Services Logs >
Microsoft > Windows > PowerShell > Operational
Event Log channel name = Microsoft-Windows-PowerShell/Operational
Event ID: 4104

Important Notes
═══════════════
(1) Re-enforce the new group policy settings using the command: gpupdate
/force
(2) Close and re-open the current Command Prompt or Powershell sessions
before testing the new group policy settings
(3) Sysmon records the process ID in decimal while Windows Security logs
record it in hexadecimal
(https://www.rapidtables.com/convert/number/hex-to-decimal.html)
(4) Increase maximum log sizes of both the Sysmon and PowerShell
Operational channels to keep your events for longer periods to survive
operational issues.
(5) Powershell'S Module Logging records the executed PS aliases that are
equivalent to the Windows native commands so check the actual native
windows command in the Process Create event of

the Security channel or you have to enable the Script Block Logging to
see the actual command (i.e. Scriptblock text)

Resources
═════════
https://www.rapidtables.com/convert/number/hex-to-decimal.html
https://redblueteam.wordpress.com/2020/02/08/enable-command-line-and-
powershell-audit-for-better-threat-hunting/
https://www.youtube.com/watch?v=AoAzkyrgaBY
https://logrhythm.com/blog/powershell-command-line-logging/
```

- In above text we can see different things that we will go through to achieve the auditing

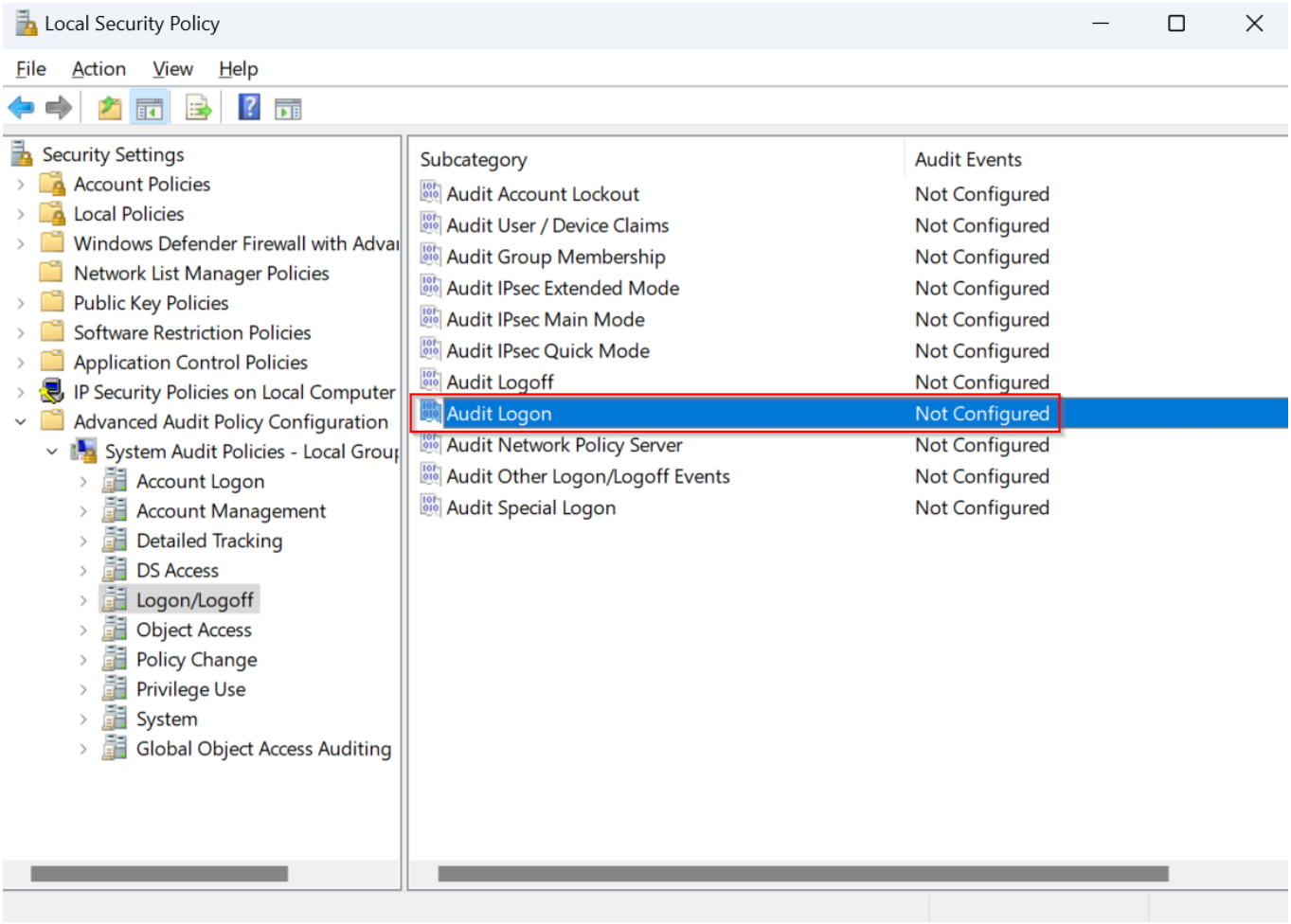
Part-1: enable auditing for command prompt

(1) Process creation

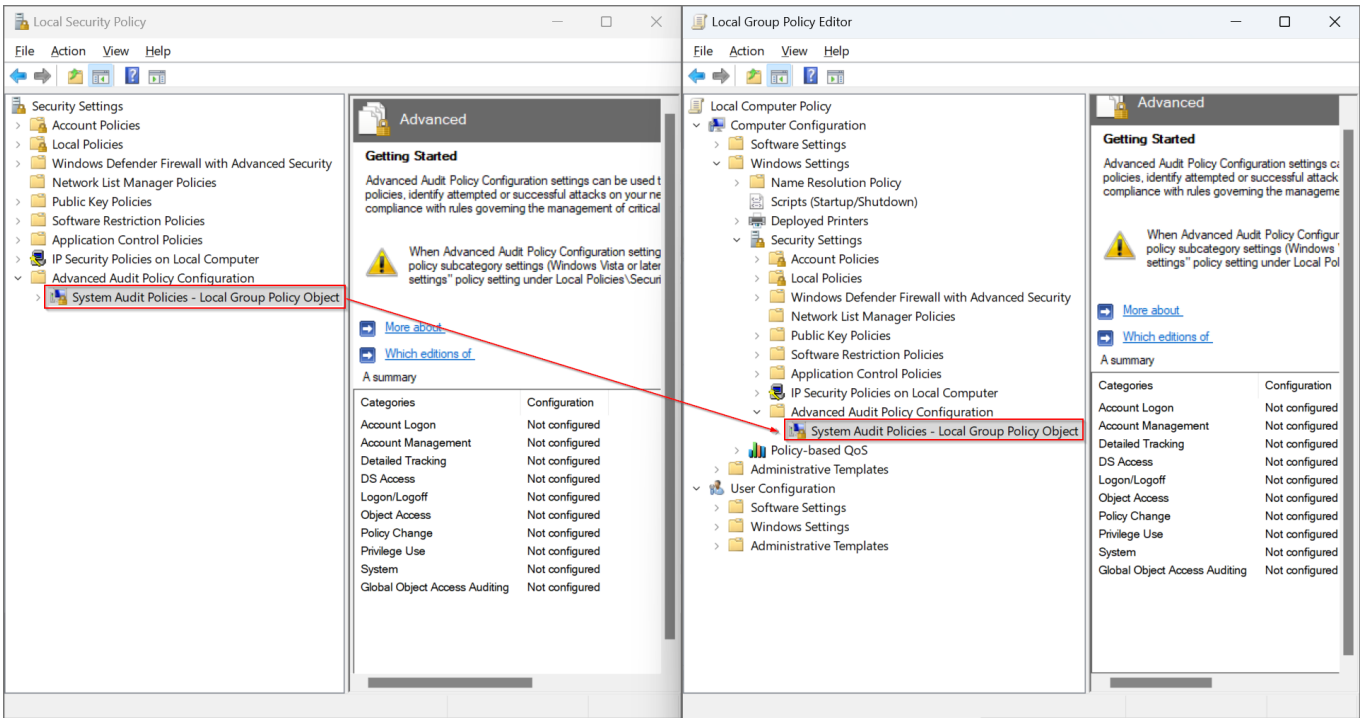
- why process creation? because, when you execute the `wget` or `curl`, it will trigger the process, this is invoking the new process
  - new process event will be triggered and logged in the windows security native channel

Note:

- for local PC - we need to enable security auditing in `Local Security Policy`



- in our case its `Local Group Policy`. But, in production you will have Domain and it's related policy
  - this is same as `Local Security Policy`, but it's just a pointer to local security policy



Note:

- all the script mentioned in the text is for `Computer Configuration`, not for `User Configuration`
- Other than `Logon/Logoff` auditing, we need to enable `Auditi Process Creation` in `Detailed Tracking`
  - here `Success` is enough, because we don't care if the process failed to create
- this is important, when you enable logging for your SIEM is to tune from the source, not to wait until your log come to FortiSIEM or any SIEM solution and then filter from SIEM itself, because this will consume your EPS (Events Per Second).
  - so this is important to filter from the source, no need to enable the failure
  - our use case, our objective here is to monitor for any successful process creation, doesn't matter to me if the process creation fails

Note:

- enabling the `Detailed Tracking` is not enough because, you will get the name of the process, but you will not get the line arguments
  - to verify this, go to `Event Viewer > Windows Logs > Security` and see the logs
  - now open the CMD and execute any executable (e.g., calc)
  - now refresh the logs, you will see the windows calculator process creation, but if you check `Process Command Line:` is not written, so the `calc` itself is not written
  - for this we need to enable some other option to record the command line argument

(2) Command Line Arguments

- go to `Local Group Policy` > `Administrative Templates > System > Audit Process Creation` . select this configuration and enable it
  - this policy is to `Include command line in process creation events`
- now you can apply computer configuration policy and your user configuration policy using

```
gpupdate /force
```

- this is more relevant if you are in domain and you are doing this on domain controller
  - this is to make sure that our policy is applied as well
- now if you try the same process again by executing any executable in command prompt. this time it will show the `Process Command Line`
- instead of doing all this, we can also do this in `Sysmon`
  - you can see it in `Event Viewer` > `Application and Services > Micorosoft > Windows > Sysmon > Operational`

Note:

- windows event ID `1` (sysmon) or `4688` (standard windows channel), will provide you same information

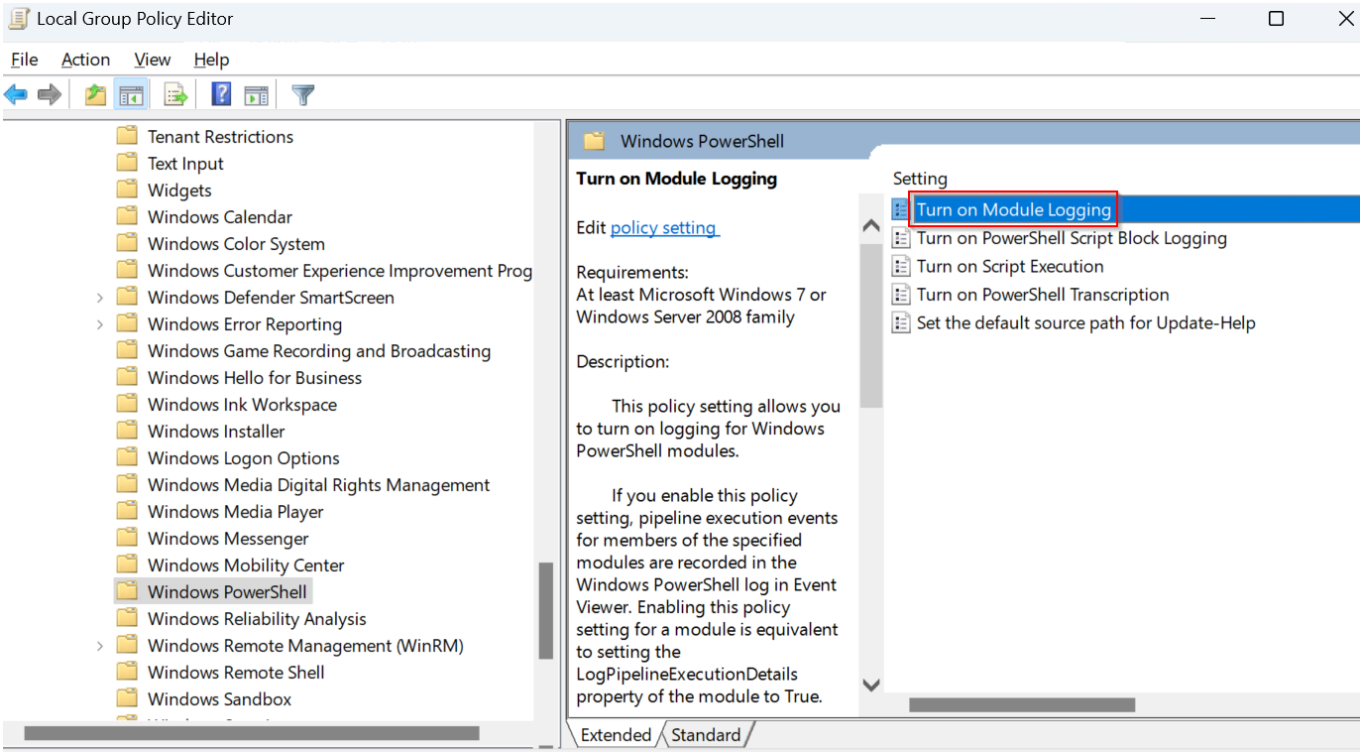
Note:

- note that, this is enough for any command that is invoked from the command prompt or the powershell.
- However, in many of the cases, the Powershell is not just issuing the native windows command prompt, the powershell itself has its own commands that are not realized by the standard logging in windows
- actually these commands are scripts that are being translated into different commands
- sometimes, some of the commands, the windows commands itself, the native commands are translated into aliases in powershell.
- so the previous auditing, what we did is not enough or not covering all the powershell cases. that's we need to do further auditing of the powershell.

Part-2: enable auding for powershell (powershell.exe nad powershell_ise.exe)

(1) Module Logging

- this is the prerequisite
- in powershell, there are multiple modules like management modules, network modules, etc., that you can invoke and start using the commands under it.



- enable this and also `powershell script block logging` to make sure that we see the actual command that is written in the powershell prompt, not the translated commands or aliases.
  - e.g., if you write `dir` in the powershell, this will not be logged in the module logging, it will be seen like `Get-ChildItem` or whatever the powershell syntax.
  - if we want to see the `dir` itself, we need to enable the `powershell script block logging`

Note:

- you don't necessarily need to enable `Log script block invocation start/stop events` all the time, you need to enable it for some cases for short periods. for examples, if you are expecting some attack to happen, if you are making hunting through the Dark web and you get to know about some people, or some bad guys are talking about your company in the dark web.
  - this will increase the logs volume, just enable it for specific use cases

(2) Script Block Logging

- this is very important, because most of the commands issued through the powershell is scripts that are broken into different commands

- the first method (Module logging) is just records the actual commands that are executed, but it will not care about the command executed by the script

Note:

- these events will be written in the powershell logs: `Event Viewer > Applications and Services Logs > Microsoft > Windows > PowerShell > Operational`
  - this is similar to sysmon

- look for `4103` (Module logging) and `4104` (Script block logging)
  - in 4103 you will not see the actual commands which you have executed in the powershell. so look in 4104

- now based on the result, you can write the rules based on the logs which are available in `Analytics` tab
  - this is the beauty of getting the logs into FortiSIEM, checking the fields and then start write your rule

- Sysmon records the process ID in decimal while Windows Security logs record it in hexadecimal (https://www.rapidtables.com/convert/number/hex-to-decimal.html)

  - this is very important, because if you are expecting something and you are getting something else, so this is important to know exactly what you are getting before designing your rule

  - finally, one of the prerequisite is, you have to edit your template for your agent and make sure that you select all the channels that we just mentioned
    - in `ADMIN > Windows Agent > Event`