Sysinternals

Check for exeutables that are unsigned and open virus total report

```
sigcheck -e -u -vr [path]
```

Checking DLLs of specific process

```
C:\Windows\system32>tasklist /m /fi "pid eq 1304"
```

Analyzing processes

```
1- Process running with wrong parent process. Example would be the authentication manager lsass.exe
running as a child process of explorer.exe

2- Process executable running from suspicious locations such as c:\temp or c:\users\

3- Misspelled processes.

4- Process with long command line containing weird or encoded characters and URLs.
```

Analyzing events

When analyzing events in Windows, we five special consideration to the below events:

```
Event ID 4624: successful authentication
Event ID 4648: login attempt using alternate credentials (run as, for instance)
Event ID 4672: super-user account login
```

We can investigate these event IDs with the below powershell command

```
PS> Get-WinEvent -FilterHashtable @{path='.\security.evtx';id=4624,4648,4672}
```

If we want to extract specific fields such as logged in username, domain or remote workstation we need to specify the field numbers in the command. The below command does this for event ID [4624] and export it to a [CSV] file

```
Get-WinEvent -FilterHashtable @{path='.\security.evtx';id=4624} ` | Select-Object -Property timecreated,
id, @{label='username';expression={$_.properties[5].value}}, @{label='domain';expression=
{$_.properties[6].value}}, @{label='Source';expression={$_.properties[18].value}} ` | export-csv
output_events_4624.csv
```

For every even ID, field numbers differ.
The below command extracts username, domain and remote workstation for event id [4684]

```
Get-WinEvent -FilterHashtable @{path='.\security.evtx';id=4648} ` | Select-Object -Property timecreated, id, @{label='username';expression={$_.properties[5].value}}, @{label='domain';expression={$_.properties[6].value}}, @{label='Source';expression={$_.properties[12].value}} ` | export-csv OUTPUT_events_4648.csv
```

The last one extracts username and domain fields for event ID [4672]

```
Get-WinEvent -FilterHashtable @{path='.\security.evtx';id=4672} ` | Select-Object -Property timecreated, id, @{label='username';expression={$_.properties[1].value}}, @{label='domain';expression={$_.properties[2].value}} ` | export-csv output_events_4672.csv
```

Hunting Malwares with Yara

Yara is a pattern matching tool that can identify malwares based on patterns such as binary, hexadecimal or textual strings. Malwares like any other programs stores textual data which can be inspected with [strings] tool.

Installing Yara

Through package manager

```
sudo apt install yara
```

Manually

```
wget https://github.com/VirusTotal/yara/archive/v4.0.2.tar.gz
cd yara-4.0.2
chmod +x configure
./configure
chmod +x bootstrap.sh
./bootstrap.sh
make
sudo make install
cd yara-4.0.2
```

Creating rules

When creating Yara rules, we store them in files called [Yara rule files]. Every Yara command takes two arguments; the [yara rule file] and the [name of the file against which the yara rule file will run].

Yara rule files have the extension of [.yar].
All yara rule files follow the example syntax below.

```
rule examplerule {
 condition: true
}
```

[examplerule] is the name of the rule and [condition] is the condition.
Example conditions are below

```
Desc : Is used to summarise the objective of the yara rule. It doesn't influence the rule itself and is
considered much like comments in coding.

Meta: Reserved for descriptive information about the author of the yara rule

Strings: Used to check for text or hexadecimal strings with files or programs

Conditions

Weight
```

Example Yara Rule

```
rule strings-checker {
		meta:
				author = "motasem"
				description = "test rule"
				created = "11/10/2021 00:00"
		strings:
				$hello-world = "Hello World"
				$ext = ".txt"
		condition:
				$hello-world and $ext
}
```

The above rule looks for the string [Hello World] and the if the file extension is [txt]. You can try this rule against a directory

```
yara -s strings-checker.yar /home/
```

Automated yara rules creation

Automated creation of yara rules relies on providing malicious files and malwares as arguments to the tool so that the tool will scan the file for malicious strings and add them to a yara rule

YarGen

YarGen is a yara rule Generator.

```
https://github.com/Neo23x0/yarGen
```

You can generate Yara rules by providing the malicious file as an argument

```
python3 yarGen.py --update
python3 yarGen.py -m /path-to-malware-file.rar --excludegood -o /path-to-output-yara-rule-file
```

Valhalla

```
https://www.nextron-systems.com/valhalla/
```

Yara scanners

Yara scanners are tools that scan for Indicators of compromise based on ready made yara rules.

LOKI Yara Scanner

```
https://github.com/Neo23x0/Loki
```

This scanner hunts for malwares based on [name], [yara rule], [hash] and [C2 back connect]

```
python loki.py -p [path to be scanned]
```

Useful modules for malware analysis

The below modules can be used to generate yara rules that hunt for specific characterstics related to malwares

Python pefile

```
pip install pefile
```

Cuckoo sandbox

```
https://cuckoosandbox.org/
```

Resources for Yara rules

```
https://github.com/InQuest/awesome-yara
```