Dynamic analysis

It involves analyzing a sample by executing it in an isolated environment and monitoring its activities, interaction, and effect on the system.

1. Lab Environment Overview

Configure a safe production environment, to prevent virus from infecting your machine.

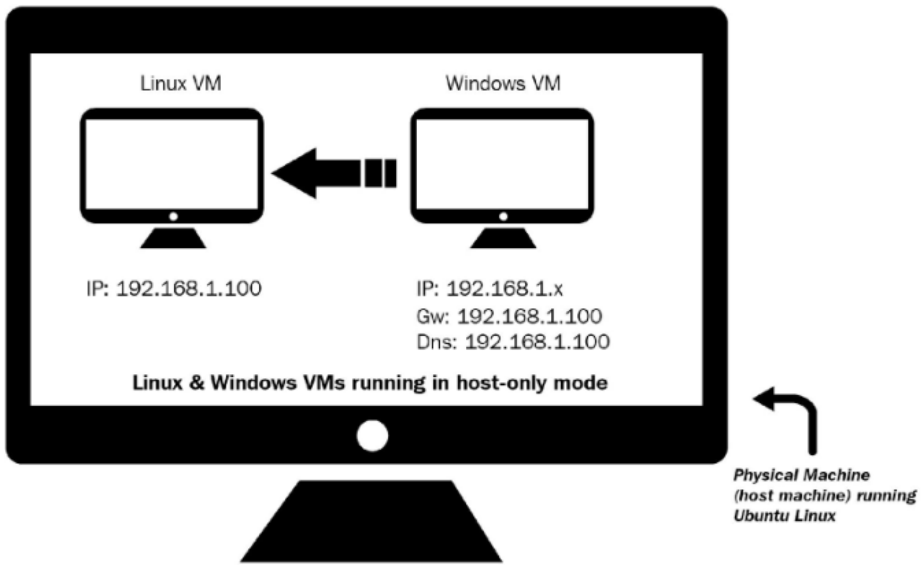
Linux VM
Windows VM
IP: 192.168.1.100
IP: 192.168.1.x
Gw: 192.168.1.100
Dns: 192.168.1.100
Linux & Windows VMs running in host-only mode
Physical Machine (host machine) running Ubuntu Linux

Figure 1: Lab environment

2. System and Network Monitoring

The objective is to gather real-time data related to malware behavior and its impact on the system. There are different types of monitoring:

- **Process monitoring:** Monitoring the process activity and examining the properties of the result process
- **File system monitoring:** Monitoring the real-time filesystem activity during malware execution.
- **Registry monitoring:** Involves monitoring the registry keys accessed/modified and the registry data that is being read/written by the malicious binary
- **Network monitoring:** Involves monitoring the live traffic to and from the system during malware execution.

3. Dynamic Analysis (Monitoring) Tools

3.1 Process Inspection with Process Hacker

> Open Source, multi-purpose tool that helps in monitoring system resources.

Once the malware has executed, this tool can help identify the newly created malware process.

3.2 Determining System Interaction With Process Monitor

> Advanced monitoring tool that shows the real time interaction of the processes with the filesystem, registry, and process/thread activity.

When executed as Admin, this tool shows all the system's processes, you can filter (`Ctrl + L`), pause(`Ctrl + E`) and clear (`Ctrl + X`) events.

3.3 Logging System Activities Using Noriben

> Noriben is a Python script that works in conjunction with Process Monitor and helps in collecting, analyzing and reporting runtime indicators of the malware.

To use Noriben, download it and copy the `Procmon.exe` file into de Noriben folder. Open Noriben, and this will automatically open the Process Monitor. Once you are finished, exit the process monitor. Results will be logged to a text and a CSV file.

3.4 Capturing Network Traffic with Wireshark

Use Wireshark to capture network traffic. To start capturing: `Capture > Options (or Ctrl + k) > select network interface > Start`

3.5 Simulating Network Services with INetSim

Most malware connect to their Command and Control servers when executed. To prevent this and analyze the malware, we need to mock the server.

Apart from simulating services, INetSim can log communications, and it can also be configured to respond to HTTP/HTTPS requests and return any file based on the extension.

If for example, a malware wants to communicate with it's CC server and INetSim isn't available, the request will produce a DNS error, because it doesn't know how to translate the address. With INetSim, the traffic is translated into the computer's own IP address. This way, we can see what the malware intends to do, and respond to it.

4. Dynamic Analysis Steps

Steps to determine the functionality of the malware.

- **Reverting to the clean snapshot:** Reverting VM to the clean snapshot
- **Running the monitoring/dynamic analysis tools:** Run the tools before the specimen
- **Executing the malware specimen:** Run malware sample with administrator privileges
- **Stopping the monitoring tools:** Stop the tools once the malware has executed for a given time
- **Analyzing the results:** Collecting the data/reports from the monitoring tools and analyzing them to determine the malware's behavior and functionality.

5. Putting it all Together: Analyzing a Malware Executable

Example use of static and dynamic analysis on sample `sales.exe`.

5.1 Static Analysis of the sample

1. Determine the file type.
   - `file sales.exe`
2. Determine the cryptographic hash.
   - `md5sum sales.exe`
3. Extract the ASCII strings from the binary
   - `strings sales.exe > strings.txt`
   - In this specific example, the strings command references the `RUN` registry key. This means it has the ability of starting itself again, once the computer restarts.
   - It also shows `WinExec` and `ShellExecuteA` API calls, this means it can invoke other programs or create new processes.
4. Query the hash value with antivirus detection sites, like *VirusTotal*.
   - In this specific instance, ours matches a malware called *poisonIvy*

Dynamic Analysis of the sample

1. Revert both server and infected machines to clean snapshots
2. Start *Noriben* with *ProcessHacker* on the Windows (infected) machine
3. Start *INetSim* on the server (Linux) machine
4. Start *Wireshark* on the server (Linux) machine
5. Execute the malware as superuser on Windows, for a given period of time
6. Stop *Noriben* after stopping the executable on Windows
7. Stop *INetSim* after stopping the executable on Linux
8. Stop *Wireshark* after stopping the executable on Linux

Now, we have all the logs and registry we need to further analyse the malware.

1. Check which new processes have been created after the execution
2. Check the network packets sent and received by the malicious program

6. Dynamic-Link Library (DLL) Analysis

A *Dynamic-Link Library (DLL)* is a module that contains functions (Export functions or exports) that can be used by another program (Such as an executable or a DLL)

6.1 Why attackers use DLL's

- Cannot be executed by double-clicking
- DLL's need a host process to run, and the malware author can attach to a legitimate process
- Injecting into a already running process provides the attacker the capability yo persist in the system
- A DLL injected to a process, has access to the process's memory space, therefore, it can access it's variables, such as user's passwords.
- Easier to analyse a `.exe` than a DLL.

6.2 Analyzing the DLL Using rundll32.exe

`rundll.exe` can be used in Windows to launch a malware.

```
rundll32.exe <full path to dll>, <export function>
<optional arguments>
```

- **Full path to DLL:** Full path to DLL. No spaces or special characters
- **Export function:** Function in the DLL that will be called after the DLL is loaded
- **Optional arguments:** Passed to the export function when it is called
- **The comma:** Put between the full path to the DLL and the export function

6.2.1 Working of rundll32.exe

Steps `rundll32.exe` takes once executed:

1. Validation of command-line arguments
2. Loads supplied DLL (`DLLMain` function (DLL entry point))
3. Obtains address of exported function and calls the function.
4. Optional arguments supplied to the loaded DLL if provided.

If at any point one of the previous steps cannot be executed, `rundll32.exe` terminates.

6.2.2 Launching the DLL Using `rundll32.exe`

Example 1 - Analyzing a DLL With No Exports

Whenever a DLL is loaded, its entry point function gets called, which in turn calls the `DLLMain` function. An attacker can implement malicious functionality in this function without exporting any functions. DLL's like the ones described above still have to be executed with the same notation:

```
rundll32.exe <full path to dll>, <export function>
<optional arguments>
```

So, in order to do this, we can write whatever we want in the `<export function>` parameter. After this command, an error will occur, because there was no export function to be found, but the malicious code has already been executed.

Example 3 - Analyzing a DLL Accepting Export Arguments

The DLL `SearchCache.dll` consists of an export function (`_flushfile@16`) whose functionality is to delete a file. This function accepts an argument, which is the file to delete.

To test this DLL, the following command has to be entered:

```
rundll32.exe c:\samples\SearchCache.dll,_flushfile@16
C:\samples\file_to_delete.txt
```

After this execution, the file will be deleted from disk

6.3 Analyzing a DLL with Process Checks

Some DLL's check the process they are running under. This means they can change their behavior based on this.

To analyse this type of DLL's, there is a tool named *RemoteDLL*. This tool allows you to inject a DLL into any process running on the system.