

Dynamic Analysis

When it comes to the domain of malware analysis, dynamic or behavioral analysis represents an indispensable approach in our investigative arsenal. In dynamic analysis, we observe and interpret the behavior of the malware while it is running, or in action. This is a critical contrast to static analysis, where we dissect the malware's properties and contents without executing it. The primary goal of dynamic analysis is to document and understand the real-world impact of the malware on its host environment, making it an integral part of comprehensive malware analysis.

In executing dynamic analysis, we encapsulate the malware within a tightly controlled, monitored, and usually isolated environment to prevent any unintentional spread or damage. This environment is typically a virtual machine (VM) to which the malware is oblivious. It believes it is interacting with a genuine system, while we, as researchers, have full control over its interactions and can document its behavior thoroughly.

Our dynamic analysis procedure can be broken down into the following steps:

- **Environment Setup:** We first establish a secure and controlled environment, typically a VM, isolated from the rest of the network to prevent inadvertent contamination or propagation of the malware. The VM setup should mimic a real-world system, complete with software, applications, and network configurations that an actual user might have.
- **Baseline Capture:** After the environment is set up, we capture a snapshot of the system's clean state. This includes system files, registry states, running processes, network configuration, and more. This baseline serves as a reference point to identify changes made by the malware post-execution.
- **Tool Deployment (Pre-Execution):** To capture the activities of the malware effectively, we deploy various monitoring and logging tools. Tools such as **Process Monitor (Procmon)** from Sysinternals Suite are used to log system calls, file system activity, registry operations, etc. We can also employ utilities like **Wireshark**, **tcpdump**, and **Fiddler** for capturing network traffic, and **Regshot** to take before-and-after snapshots of the system registry. Finally, tools such as **INetSim**, **FakeDNS**, and **FakeNet-NG** are used to simulate internet services.
- **Malware Execution:** With our tools running and ready, we proceed to execute the malware sample in the isolated environment. During execution, the monitoring tools capture and log all activities, including process creation, file and registry modifications, network traffic, etc.
- **Observation and Logging:** The malware sample is allowed to execute for a sufficient duration. All the while, our monitoring tools are diligently recording its every move, which will provide us with comprehensive insight into its behavior and modus operandi.
- **Analysis of Collected Data:** After the malware has run its course, we halt its execution and stop the monitoring tools. We now examine the logs and data collected, comparing the system's state to our initial baseline to identify the changes introduced by the malware.

In some cases, when the malware is particularly evasive or complex, we might employ sandboxed environments for dynamic analysis. Sandboxes, such as **Cuckoo Sandbox**, **Joe Sandbox**, or **FireEye's Dynamic Threat Intelligence cloud**, provide an automated, safe, and highly controlled environment for malware execution. They come equipped with numerous features for in-depth behavioral analysis and generate detailed reports regarding the malware's network behavior, file system interaction, memory footprint, and more.

However, it's important to remember that while sandbox environments are valuable tools, they are not foolproof. Some advanced malware can detect sandbox environments and alter their behavior accordingly, making it harder for researchers to ascertain their true nature.

Resources

Go to Questions

Table of Contents

- Introduction To Malware & Malware Analysis
- Prerequisites
 - Windows Internals
- Static Analysis
 - Static Analysis On Linux
 - Static Analysis On Windows
- Dynamic Analysis
 - Dynamic Analysis
- Code Analysis
 - Code Analysis
 - Debugging
- Creating Detection Rules
 - Creating Detection Rules
- Skills Assessment
 - Skills Assessment

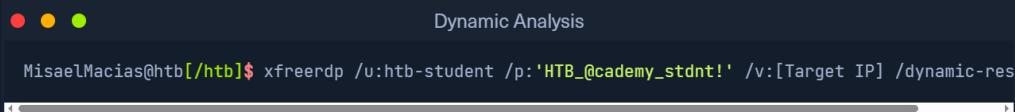
My Workstation

O F F L I N E

Start Instance

∞ / 1 spawns left

Let's now navigate to the bottom of this section and click on "Click here to spawn the target system!". Then, let's RDP into the Target IP using the provided credentials. The vast majority of the actions/commands covered from this point up to end of this section can be replicated inside the target, offering a more comprehensive grasp of the topics presented.



```
MisaelMacias@htb[/htb]$ xfreerdp /u:htb-student /p:'HTB_academy_stdnt!' /v:[Target IP] /dynamic-res
```

Dynamic Analysis With Noriben

Noriben is a powerful tool in our dynamic analysis toolkit, essentially acting as a Python wrapper for Sysinternals **ProcMon**, a comprehensive system monitoring utility. It orchestrates the operation of **ProcMon**, refines the output, and adds a layer of malware-specific intelligence to the process. Leveraging **Noriben**, we can capture malware behaviors more conveniently and understand them more precisely.

To understand how **Noriben** empowers our dynamic analysis efforts, let's first quickly review **ProcMon**. This tool, from **Sysinternals Suite**, monitors real-time file system, Registry, and process/thread activity. It combines the features of utilities like **Filemon**, **Regmon**, and advanced features like filtering, advanced highlighting, and extensive event properties, making it a powerful system monitoring tool for malware analysis.

However, the volume and breadth of information that **ProcMon** collects can be overwhelming. Without proper filtering and contextual analysis, sifting through this raw data becomes a considerable challenge. This is where **Noriben** steps in. It uses **ProcMon** to capture system events but then filters and analyzes this data to extract meaningful information and pinpoint malicious activities.

In our dynamic malware analysis process, here's how we employ **Noriben**:

- **Setting Up Noriben:** We initiate **Noriben** by launching it from the command line. The tool supports numerous command-line arguments to customize its operation. For instance, we can define the duration of data collection, specify a custom malware sample for execution, or select a personalized **ProcMon** configuration file.
- **Launching ProcMon:** Upon initiation, **Noriben** starts **ProcMon** with a predefined configuration. This configuration contains a set of filters designed to exclude normal system activity and focus on potential indicators of malicious actions.
- **Executing the Malware Sample:** With **ProcMon** running, **Noriben** executes the selected malware sample. During this phase, **ProcMon** captures all system activities, including process operations, file system changes, and registry modifications.
- **Monitoring and Logging:** **Noriben** controls the duration of monitoring, and once it concludes, it commands **ProcMon** to save the collected data to a CSV file and then terminates **ProcMon**.
- **Data Analysis and Reporting:** This is where **Noriben** shines. It processes the CSV file generated by **ProcMon**, applying additional filters and performing contextual analysis. **Noriben** identifies potentially suspicious activities and organizes them into different categories, such as file system activity, process operations, and network connections. This process results in a clear, readable report in HTML or TXT format, highlighting the behavioral traits of the analyzed malware.

Noriben's integration with YARA rules is another notable feature. We can leverage YARA rules to enhance our data filtering capabilities, allowing us to identify patterns of interest more efficiently.

For demonstration purposes, we'll conduct dynamic analysis on a malware specimen named **shell.exe**, found in the **C:\Samples\MalwareAnalysis** directory on this section's target machine. Follow these steps:

- Launch a new Command Line interface and make your way to the **C:\Tools\Noriben-master** directory.

- Initiate Noriben as indicated.

```
C:\Tools\Noriben-master> python .\Noriben.py
[*] Using filter file: ProcmonConfiguration.PMC
[*] Using procmon EXE: C:\ProgramData\chocolatey\bin\procmon.exe
[*] Procmon session saved to: Noriben_27_Jul_23__23_40_319983.pml
[*] Launching Procmon ...
[*] Procmon is running. Run your executable now.
[*] When runtime is complete, press CTRL+C to stop logging.
```

- Upon seeing the **User Account Control** prompt, select **Yes**.
- Proceed to **C:\Samples\MalwareAnalysis** and activate **shell.exe** by double-clicking.
- **shell.exe** will identify it is running within a sandbox. Close the window it created.
- Terminate **ProcMon**.
- In the Command Prompt running Noriben, use the **Ctrl+C** command to cease its operation.

```
C:\Tools\Noriben-master> python .\Noriben.py
[*] Using filter file: ProcmonConfiguration.PMC
[*] Using procmon EXE: C:\ProgramData\chocolatey\bin\procmon.exe
[*] Procmon session saved to: Noriben_27_Jul_23__23_40_319983.pml
[*] Launching Procmon ...
[*] Procmon is running. Run your executable now.
[*] When runtime is complete, press CTRL+C to stop logging.

[*] Termination of Procmon commencing... please wait
[*] Procmon terminated
[*] Saving report to: Noriben_27_Jul_23__23_42_335666.txt
[*] Saving timeline to: Noriben_27_Jul_23__23_42_335666_timeline.csv
[*] Exiting with error code: 0: Normal exit
```

You'll observe that Noriben generates a **.txt** report inside it's directory, compiling all the behavioral information it managed to gather.

```
--=] Sandbox Analysis Report generated by Noriben v1.8.2
--=] Developed by Brian Baskin : brian @@ thebaskins.com @bbaskin
--=] The latest release can be found at https://github.com/Rurik/Noriben

--=] Execution time : 18.68 seconds
--=] Processing time : 4.41 seconds
--=] Analysis time : 7.58 seconds

Processes Created :
=====
[CreateProcess] powershell.exe : 2052 > "C:\Samples\shell.exe"[Child PID : 928]
[CreateProcess] shell.exe:928 > "%WinDir%\System32\cmd.exe /k ping [REDACTED] -n 5"[Child PID : 1636]
[CreateProcess] cmd.exe:1636 > "\??%\WinDir%\System32\conhost.exe 0xffffffff -ForceV1"[Child PID : 5376]
[CreateProcess] cmd.exe:1636 > "ping [REDACTED] -n 5"[Child PID : 9284]

File Activity :
=====
[CreateFile] svchost.exe : 2320 > % AllUsersProfile % \Microsoft\Windows\AppRepository\StateRepository -
[CreateFile] svchost.exe:2320 > % AllUsersProfile % \Microsoft\Windows\AppRepository\StateRepository - Ma
```

As discussed, **Noriben** uses **ProcMon** to capture system events but then filters and analyzes this data to extract meaningful information and pinpoint malicious activities.

Noriben might filter out some potentially valuable information. For instance, we don't receive any insightful data from Noriben's report about how **shell.exe** recognized that it was functioning within a sandbox or virtual machine.

Let's take a different approach and manually launch **ProcMon** (available at **C:\Tools\sysinternals**) using its default, more inclusive, configuration. Following this, let's re-run **shell.exe**. This might give us insights into how **shell.exe** detects the presence of a sandbox or virtual machine.

Then, let's configure the filter (**Ctrl+L**) as follows and press **Apply**.



Display entries matching these conditions:

Process Name is shell.exe then Include

Reset Add Remove

Column	Relation	Value	Action
<input checked="" type="checkbox"/> Process Name	is	shell.exe	Include
<input checked="" type="checkbox"/> Operation	is	RegQueryValue	Include
<input checked="" type="checkbox"/> Result	is	SUCCESS	Include
<input checked="" type="checkbox"/> Process Name	is	System	Exclude
<input checked="" type="checkbox"/> Process Name	is	Procmon64.exe	Exclude
<input checked="" type="checkbox"/> Process Name	is	Procmon.exe	Exclude
<input checked="" type="checkbox"/> Process Name	is	Procexp64.exe	Exclude
<input checked="" type="checkbox"/> Process Name	is	Procexp.exe	Exclude
<input checked="" type="checkbox"/> Process Name	is	Autoruns.exe	Exclude

OK Cancel Apply

Finally, let's navigate to the end of the results. There can observe that **shell.exe** conducts sandbox or virtual machine detection by querying the registry for the presence of **VMware Tools**.

1:05:47...		shell.exe	4360		RegQueryValue	HKCR\CLSID\{9ac9fbef-e0a2-4ad6-b4ee-e212013ea917}\InProcServer32(Default)	SUCCESS
1:05:47...		shell.exe	4360		RegQueryValue	HKCR\CLSID\{9ac9fbef-e0a2-4ad6-b4ee-e212013ea917}\InProcServer32\ThreadingModel	SUCCESS
1:05:47...		shell.exe	4360		RegQueryValue	HKLMSOFTWARE\VMware, Inc.\VMware Tools\InstallPath	SUCCESS
1:05:47...		shell.exe	4360		RegQueryValue	HKLMSOFTWARE\VMware, Inc.\VMware Tools\InstallPath	SUCCESS
1:05:47...		shell.exe	4360		RegQueryValue	HKLMSOFTWARE\Microsoft\Windows NT\CurrentVersion\LanguagePackDataStore_V1.0\DataFilePath	SUCCESS

VPN Servers

⚠ Warning: Each time you "Switch", your connection keys are regenerated and you must re-download your VPN connection file.

All VM instances associated with the old VPN Server will be terminated when switching to a new VPN server.

Existing PwnBox instances will automatically switch to the new VPN server.

US Academy 3 Medium Load

PROTOCOL ● UDP 1337 ○ TCP 443

[DOWNLOAD VPN CONNECTION FILE](#)

Connect to Pwnbox
Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location UK 160ms

! Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left



Waiting to start...

Enable step-by-step solutions for all questions  



Questions

Answer the question(s) below to complete this Section and earn cubes!

 Download VPN Connection File



Target(s): [Click here to spawn the target system!](#)

 RDP to with user "htb-student" and password "HTB_academy_stdnt!"

+ 1  Use Noriben to perform dynamic analysis on shell.exe. Enter the IP address shell.exe pings as your answer.

127.0.0.1

 Submit



◀ Previous

Next ➔

 [Mark Complete & Next](#)



Powered by 

