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.

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

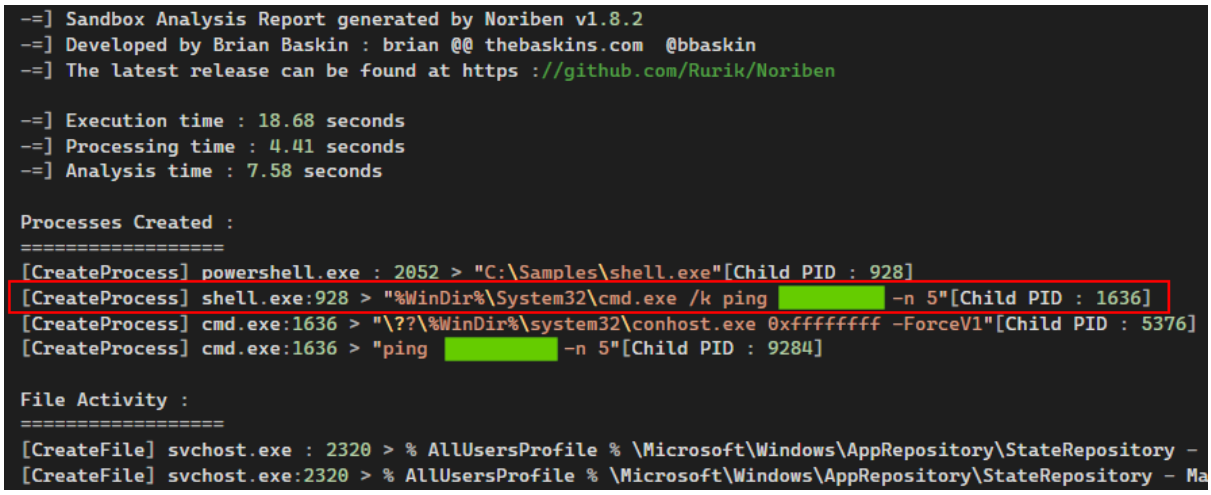
[*] Procmon terminated

[*] 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.

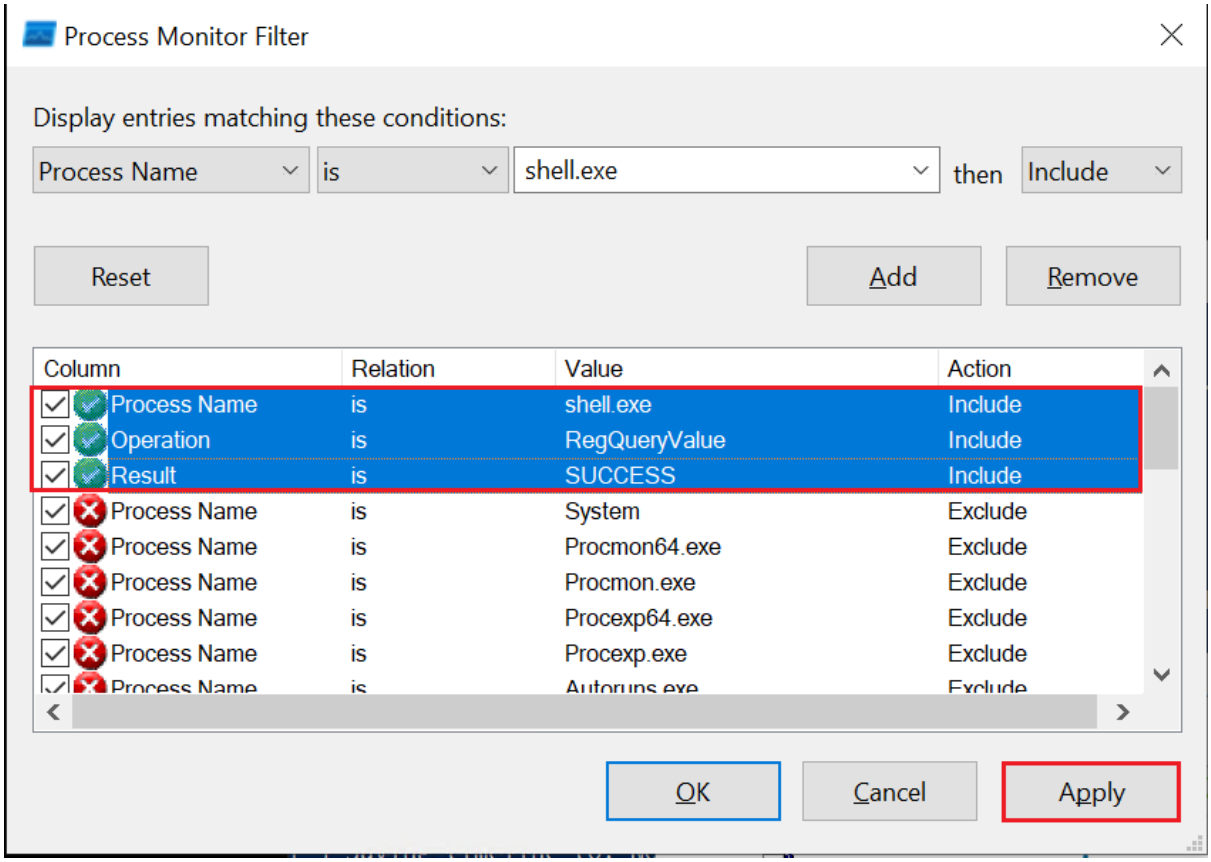


As discussed, Noriben uses ProcMon to capture system events but then filters and analyzes this data to extract meaningful information and pinpoint malicious activities.

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.



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.