**Malware Analysis using Deep Exploit , REMNux & Cuckoobox**

Malware analysis is the process of learning how malware functions and any potential repercussions of a given malware. Malware code can differ radically, and it's essential to know that malware can have many functionalities. These may come in the form of viruses, worms, spyware, and Trojan horses. Each type of malware gathers information about the infected device without the knowledge, or authorization of the user. It is used in the following cases:

* **Computer security incident management** [If an organization believes that malware may have entered into its system].
* **Malware Research** [Academic or industry forum where malware researchers perform malware analysis].
* **Indicator of compromise extraction** [Sellers of software solutions and products may conduct bulk malware analysis in order to determine potential new indicators of compromise].

There are four stages form a pyramid that grows in intricacy.

* **Fully-automated analysis**: One of the simplest ways to assess a suspicious program is to scan it with fully-automated tools.
* **Static properties analysis**: In order to get a more in depth look at malware, it is imperative to look at its static properties.
* **Interactive behavior analysis**: To observe a malicious file, it might oftentimes be put in an isolated laboratory to see if it directly infects the laboratory. Analysts will frequently monitor these laboratories to see if the malicious file tries to attach to any hosts.
* **Manual code reversing**: Reversing the code of the malicious file can decode encrypted data that was stored by the sample, determine the logic of the file’s domain, and see other capabilities of the file that did not show up during the behavioral analysis.

**Malware Analysis using Deep Exploit**

DeepExploit is a fully automated penetration test tool linked with Metasploit. It is capable of identifying the status of all opened ports on the target server and executes the exploit at pinpoint using Machine Learning. Key features of DeepExploit are efficient execution of exploits, deep penetration, self-learning (using reinforcement learning), learning time is very fast (it uses distributed learning by multi-agents), powerful intelligence gathering (Port scanning, Analyzes HTTP responses, Contents exploration). DeepExploit consists of the machine learning model (A3C) and Metasploit. The A3C executes exploits to the target servers via RPC API. The A3C is developed by Keras and Tensorflow, a famous ML framework based on Python. The A3C consists of multiple neural networks.

First install & set-up the DeepExploit and Metasploit in your system. Then select the target and run the DeepExploit in the selected target. Four step process will be followed by the tool which are as follows:

1. Intelligence Gathering: It gathers the target server information such as OS type, Opened port, Product name, Product version using Nmap. It examines the Web Ports and analyses the HTTP responses using Scrapy.
2. Exploitation: It executes the exploit to the first target server using trained data and identified product information. It can execute exploits at pinpoint (minimum 1 attempt).
3. Post-Exploitation: It executes the pivoting using the opened session in the Exploitation step. If the DeepExploit does not have a direct connection to the internal server then it executes exploits through the compromised server.
4. Reporting: It then generates a scan report that summarizes vulnerabilities.

**Malware analysis using YARA**

YARA is a tool that identifies malware by creating descriptions that look for certain characteristics. Each description can be either a text or a binary pattern. These descriptions are called “rules”. And by using rules that specify regex patterns, YARA enables the detection of specific patterns in files that might indicate that the file is malicious. By using hex patterns, plain text patterns, wild-cards, case-insensitive strings, and special operators, YARA rules can be incredibly diverse and effective at detecting a wide range of malware signatures. We can write our own rules, also there are plenty of well-defined YARA rules files available for download on Github. Besides analyzing malware, YARA can also be used to analyze the nature of files and classify file contents. YARA rules consist of sets of strings and a boolean expression, hex patterns, plain text patterns, wild-cards, case-insensitive strings, and special operators.

Running YARA from the command-line

To run YARA two things will be required: YARA-Rules file and the target to be scanned. The target can be a file, a folder, or a process.

* yara [OPTIONS] RULES\_FILE TARGET
* yara [OPTIONS] -C RULES\_FILE TARGET *(for compiled rules file)*
* yara [OPTIONS] RULES\_FILE\_1 RULES\_FILE\_2 RULES\_FILE\_3 TARGET *(passing multiple source files)*

YARA rules can be incredibly diverse and effective at detecting a wide range of malware signatures. Any file that contains the string, expression or pattern mentioned in the YARA-Rule will be flagged as malware.

**Malware analysis using Cuckoo Sandbox**

Cuckoo Sandbox is the leading open source automated malware analysis system. We can pass any suspicious file through it and Cuckoo will provide a detailed report outlining the behavior of the file when executed inside a realistic but isolated environment. This tool is able to analyze many different malicious files as well as malicious websites, trace API calls & general behaviour of the file, dump and analyze network traffic (even when encrypted with SSL/TLS) and perform advanced memory analysis of the infected virtualized system.

After installation & set-up. Start the Cuckoo daemon and the processing utilities.

* supervisorctl start cuckoo:

# You’ll need the trailing colon (i.e., cuckoo:) so to denote the Cuckoo supervisor group, containing the Cuckoo daemon process as well as the various processing utilities.

The easiest way to submit an analysis is to use the “*cuckoo submit”* utility.

* cuckoo submit /path/to/binary *(local binary)*
* cuckoo submit --url <http://www.example.com> *(URL)*
* cuckoo submit --priority 5 /path/to/binary *(local binary & higher priority)*
* cuckoo submit --machine cuckoo1 /path/to/binary *(local binary to be run on virtual machine cuckoo1)*
* cuckoo submit --memory /path/to/binary *(local binary & taking a full memory dump of the analysis machine)*

For more details: https://cuckoo.sh/docs/usage/index.html