Detection of Malware Incidents

Objectives:

At the end of this episode, I will be able to:

Understand what are the Malware detection techniques that Incident Responders should be aware of.

Explain what common live system Malware Analysis Techniques that IH&R team members should be aware of are.

Identify what common static system Malware Analysis Techniques that IH&R team members should be aware of are.

Define what the steps to analyze a memory dump of the compromised system using the volatility framework are.

External Resources:

Detection of Malware

What are the Malware Detection Techniques that Incident Responders should be aware of? -

Malware programs exhibit specific properties that can help responders to identify or distinguish them from safe software programs.

```
▪ Live System/Dynamic Analysis (behavioral analysis) - detects changes made
to the live system, and mainly involves activities that include monitoring
ports, processes, and registries for any abnormal or malicious activities.

Live System Malware Analysis Techniques include:

    ▪ Port monitoring
    ▪ Process monitoring
    ▪ Registry monitoring
    ▪ Windows services monitoring
    ▪ Startup programs monitoring
    ▪ Event logs monitoring
    ▪ Installation monitoring
    ▪ Files and folder monitoring
    ▪ Device drivers monitoring
    ▪ Network traffic monitoring
    ▪ DNS monitoring/resolution
    ▪ API calls monitoring
    ▪ Scheduled task monitoring
    ▪ Browser activity monitoring
```

Registry monitoring --> Windows automatically executes instructions in the following sections of the registry:

```
▪ Run
▪ RunServices
▪ RunOnce
▪ RunServicesOnce
▪ HKEY_CLASSES_ROOT\exefile\shell\open\command "%1" %*
```

Windows Services monitoring --> Malware may also employ rootkit techniques to manipulate the following registry keys to hide their processes and services.

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services
```

These malicious services run as the SYSTEM account or other privileged accounts.

Startup Programs monitoring -->

o Windows Startup Setting

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce

o Explorer Startup Setting

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders, Common Startup

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders, Common Startup

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders, Startup
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders, Startup

o IE Startup Setting

HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\URLSearchHooks

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\Toolbar

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\Extensions

HKEY_CURRENT_USER\SOFTWARE\Microsoft\Internet Explorer\MenuExt

```
▪ Memory Dump/Static Analysis (code analysis) - going through the executable
binary code without actually executing it to have a better understanding of
the malware and its purpose; the process of investigating an executable file
without running or installing it.

Some of the static malware analysis techniques are:

    ▪ File fingerprinting
    ▪ Local and online malware scanning
    ▪ Performing strings search
    ▪ Identifying packing/obfuscation methods
    ▪ Finding the portable executables (PE) information
    ▪ Identifying file dependencies
    ▪ Malware disassembly
```

Memory Dump Analysis: Finding the Portable Executables (PE) Information - The
Portable Executables (PE) format is the executable file format used on
Windows OSs, which stores the information a Windows system requires to manage
the executable code.

The PE format contains header and sections, which store metadata about the file
and code mapping in an OS.

The PE of a file contains the following sections:

```
▪ .text: Contains instructions and program codes that the CPU executes
▪ .rdata: Contains the import and export information as well as other
read-only data used by the program
▪ .data: Contains the program’s global data, which the system can access
from anywhere
▪ .rsrc: Comprises the resources employed by the executables, such as icons,
images, menus, and strings, as this section offers multilingual support
```

Memory Dump Analysis: Identifying File Dependencies -

```
. Kernel32.dll - Core functionality, such as access and manipulation of memory,
files, and hardware

. Advapi32.dll - Provides access to advanced core Windows components such as
the Service Manager and Registry

. User32.dll - User interface components, such as buttons, scrollbars, and
components for controlling and responding to user actions

. Gdi32.dll - Functions for displaying and manipulating graphics

. Ntdll.dll - Interface to the Windows kernel

. WSock32.dll and Ws2_32.dll - Networking DLLs that help to connect to a
network or perform network-related tasks

. Wininet.dll - Supports higher-level networking functions
```

Memory Dump Analysis Using the Volatility Framework - Volatility is a
python-based memory analysis tool that is capable of performing various
forensic operations.

What are the steps to analyze a memory dump of the compromised system using
volatility framework? :

```
▪ Create a memory dump of the system and store as a .dd image file
or .mem file on the analysis system.

▪ Use a sandbox environment, preferably a Linux based virtual machine, to
analyze the memory dump of a compromised system for detection and analysis of
malware.

▪ Install the Volatility memory forensics tool on the analysis system by using
 the Linux command apt-get install volatility.
```

To analyze the image using the volatility forensics tool from the command line
interface of the Linux system, navigate to the volatility folder by using the
cd /usr/share/volatility command and then use the following syntax:

```
python vol.py [plugin] –f [image] –profile=[profile name]
```

Use the following command to analyze the basic information about the images like
operating system and image date and time in the volatility tool:

```
python vol.py imageinfo -f /root/Desktop/memdump.mem
```

Use the following command to analyze the running processes of the image file in
the volatility tool:

```
python vol.py pslist --profile=WIN2019 –f /root/Desktop/memdump.mem
```

Use the following command to analyze the services of the image file in the
volatility tool:
python vol.py svcscan --profile=WIN2019 –f /root/Desktop/memdump.mem | more

Use the following command to analyze the registry hives of the image file in
the volatility tool:

```
python vol.py hivelist --profile=WIN2019 –f /root/Desktop/memdump.mem



▪ Intrusion Analysis - deals with analyzing intrusion detection systems and
other perimeter security systems for malware as well as detecting the malware
by its abnormal behavior after intrusion.
```

Detecting the malware by intrusion analysis based on the behavior of malware
include:

```
▪ Detecting malware by its covert storage/hiding techniques
▪ Detecting malware by its covert communication techniques
```

Intrusion Analysis: Detecting Malware by Its Covert Storage/Hiding Techniques -

```
▪ SSDT Patching/Hooking - SSDT stands for System Service Descriptor
Table (SSDT), which is a table present in the Windows operating system kernel,
which stores the entry level addresses.
```

Each SSDT table entry indicates a set of functions that are to be performed
based on the user requirement.

Every function or operation executed by the kernel comes via the SSDT, malware
such as kernel-mode rootkits attempt to alter and modify the data structure of
the kernel and attempt to hook the SSDT table, enabling the rootkit to affect
all the user operations.

The rootkit overwrites instructions in the SSDT, making them point toward
rootkit functions rather than legitimate functions, enabling them to perform
malicious operations when called.