Task 1

**STATIC TOOLS TO BE USED:**

**BINTEXT**

* **BinText** displays **all** the **strings** **within** **executable**.
* A program contains strings if it **prints a message**, **connects to a URL**, or c**opies a file to a specific location**.
* These strings can give us an **idea** **of working of executable**.
* If the executable is **packed** or **obfuscated**, **no useful strings can be seen**.
* In that case **dynamic analysis is the option.**
* ANSI strings will show a **green** "**A**"
* Unicode strings (double byte ANSI) will have a **red** "**U**"
* Resource strings have a **blue** "**R**"
* Look for IPs, calls, processes etc

**PEiD**

* You can use PEiD to detect the **type of packer** or **compiler** **employed** to **build an application**.
* If not packed, we can see the executables / DLLs that are called.
* Look at **Virtual Size** vs **Raw Data**.

**PEVIEW**

* PEview tool allows us to browse through header information.
* The **left pane** displays the **main parts of the PE header**.
* IMAGE\_DOS\_HEADER and MS-DOS Stub Program are historical and offer no information of interest to us.
* IMAGE\_NT\_HEADERS shows the NT headers. The signature is always the same and can be ignored.
* The IMAGE\_FILE\_HEADER entry, highlighted and displayed in the **right panel**, contains **basic information about the file**. The **Time Date Stamp description** tells us **when this executable was compiled**, which can be very useful.
* Old compile time suggests that this is an older attack, and antivirus programs might contain signatures for the malware.
* A new compile time suggests the reverse.
* All Delphi programs use a compile time **of June 19, 1992**. If you see that compile time, you’re probably looking at a Delphi program.
* Malware writer can easily fake the compile time.
* The IMAGE\_OPTIONAL\_HEADER section includes several important pieces of information. The Subsystem description indicates whether this is a **console** or **GUI** **program**.
* Console programs have the value IMAGE\_SUBSYSTEM\_WINDOWS\_CUI and **run inside a command window**.
* GUI programs have the value IMAGE SUBSYSTEM\_WINDOWS\_GUI and **run within the Windows system**.
* Less common subsystems such as **Native** or **Xbox** also are used.
* The most interesting information comes from the section headers, which are in **IMAGE\_SECTION\_HEADER**.
* These headers are used to **describe each section of a PE file**.
* The **compiler generally creates** and **names** the **sections of an executable**, and the **user has little control over these names**.
* As a result, the **sections are usually consistent from executable to executable** and any **deviations may be suspicious**.
* **Virtual Size** tells us **how much space is allocated for a section during the loading process**.
* The **Size of Raw Data** at shows **how big the section is on disk**.
* These two values should usually be equal, because data should take up just as much space on the disk as it does in memory.
* The section sizes can be useful in detecting packed executables. If the **Virtual Size** is **much** **larger** than the **Size of Raw Data**, you know that the **section takes up more space in memory than it does on disk**.
* NOT PACKED:

|  |  |  |
| --- | --- | --- |
| **SECTION** | **VIRTUAL SIZE** | **SIZE OF RAW DATA** |
| **.text** | 7AF5 | 7C00 |
| **.data** | 17A0 | 0200 |
| **.rdata** | 1AF5 | 1C00 |
| **.rsrc** | 72B8 | 7400 |

* PACKED AND SECTION NAMES:

|  |  |  |
| --- | --- | --- |
| **SECTION** | **VIRTUAL SIZE** | **SIZE OF RAW DATA** |
| **.text** | A000 | 0000 |
| **.data** | 3000 | 0000 |
| **.rdata** | 4000 | 0000 |
| **.rsrc** | 19000 | 3400 |
| **Dijfpds** | 20000 | 0000 |
| **.sdfuok** | 34000 | 3313F |
| **Kijijl** | 1000 | 0200 |

**DEPENDENCY WALKER**

* lists only dynamically linked functions in an executable.
* Top right panel shows us the **DLLs imported functions**.
* DLLs:
  + Kernel32 = core functionality, access and manipulation of memory, files, hardware,
  + Advapi32 = advanced windows components, service manager and registry,
  + User32 = user-interface components, buttons, scroll bars, components for controlling and responding to user action,
  + Gdi32 displaying and manipulating graphics,
  + Ntdll = interface to windows kernel,
  + WSock32= networking dlls,
  + Ws2\_32 = network related tasks,
  + Wininet higher-level networking functions FTP, HTTP, NTP.

**RESOURCE HACKER**

* When you click through the items in Resource Hacker, you’ll see the strings, icons, and menus.
* The menus displayed are **identical to what the program uses**.
* The panel on the left shows all resources included in this executable.
* Malware often stores an embedded program or driver in .rsrc and, before the program runs, they extract the embedded executable or driver.
* Resource Hacker lets you extract these files for individual analysis.

**DYNAMIC** **TOOLS TO BE USED:**

**PROCESS MONITOR**

**PROCESS EXPLORER**

**REGSHOT**

**FAKENET**