Basic Malware Analysis

In this laboratory you will practice the knowledge acquired during the classes of em topic. Specifically, we are going to focus on the basic malware analysis.

IMPORTANT

The malware samples that have been provided bf are real. Therefore, be careful when you execute them, since they will carry out their malicious behavior in the system. You will find them available in a file compressed with the "infected " password. **It is strongly recommended to use a virtual or isolated machine to carry out this laboratory session**. The instructor is not responsible for the possible damages that can be caused by their execution.

1 Laboratory environment

In this practice, you will use the virtual machine of the Windows operating system provided in the workshop (link available for direct download at `https://webdiis.unizar.es/~ricardo/sbc-2022/advanced-malware-analysis/`).

This virtual machine contains a Windows 7 Enterprise, and it is one of the virtual machines that Microsoft provides on its official website (`https://developer.microsoft.com/es-es/microsoft-edge/tools/vms/`) to evaluate their Internet browsers. The name of the user and password of the account are:

- User: `IEUser`
- Password: `Passw0rd!`

By default, the virtual machine is distributed without having previously activated. It can be activated to avoid on the one hand the annoying messages that appear during use relative to being victims of pirate software, as well as to prevent the virtual machine from restarting unexpectedly after a certain arbitrary time.

Before performing activation, which will provide us with a valid license for 90 days, it is recommended to capture the current state of the virtual machine (that is, you can make a snapshot!). To activate it, you have to open a MS-DOS command window with administrator permission (right click at "cmd.exe ", Run as administrator; see Figure 1) and run the command:

```
slmgr /ato
```

After executing it, a new window similar to the one shown in Figure 2 will appear indicating that the activation has been successful.

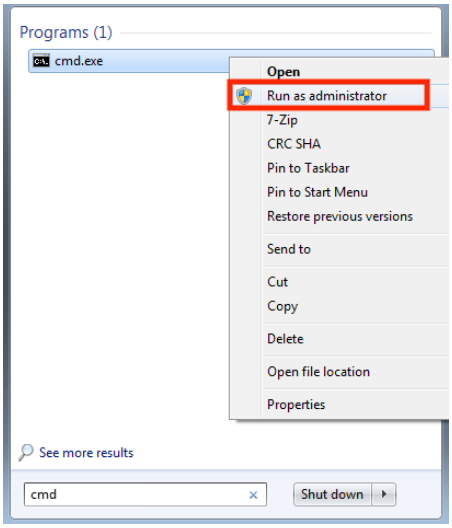

Figure 1: Execution of a MS-DOS console as an administrator user.

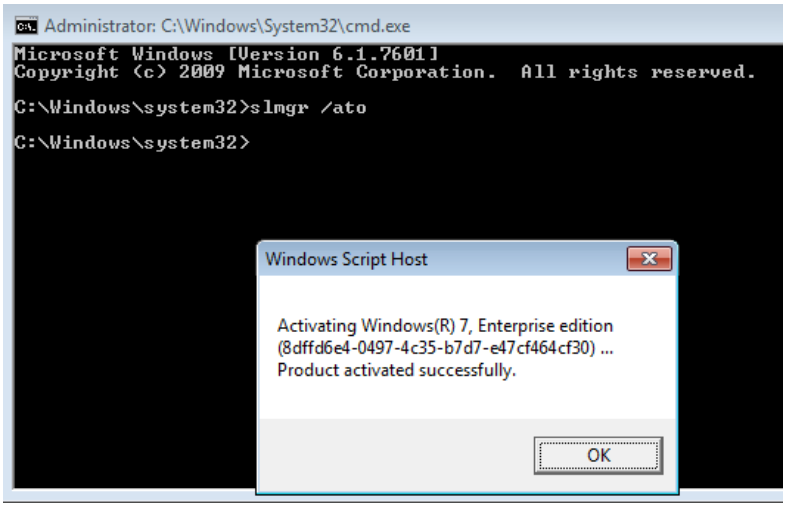

Figure 2: Windows 7 activation message (valid for 90 days).

2 Malware Analysis Methodology

Remember that the questions to answer when performing a malware analysis are the following:

- *What malicious activity is it carrying out on the system?*
- *How does this activity start?* (i.e., locate the persistence method it uses)
- *When did it infect the system, and how did it get in?*
- *How can I remove the threat and clean the infected system, and how can I prevent it from happening again?*

To answer these questions, the malware analysis methodology to follow is as follows:

1. **Static analysis phase**. This phase, also called *dead code* or *cold code analysis phase*, includes the analysis of the binary file without executing it. That is, in this phase the binary content (the bytes) of the file will be read. In this phase it will help to know in depth the assembler of the architecture where the binary is executed (this is where the reverse engineering skills that we saw in the previous section come in). This phase is divided into three basic aspects:
   - *MD5/SHA1/SHA256 signature calculation.* In this subphase, the signature of the binary will be calculated using some cryptographic algorithm such as MD5, SHA-1, or SHA-256. These algorithms allow, given an input, to calculate a number that "exclusively" identifies the binary. Once the signature has been calculated, it can be searched on the Internet in order to locate other analyzes of the malware sample and thus facilitate the analysis work itself. A good tool on Windows for this task is `HashTab` (`http://implbits.com/products/hashtab/`). On macOS or GNU/Linux systems, you can use the native commands `md5sum` or `sha1sum`, to name a few.
   - *Strings within the file.* This second subphase consists of locating all the text strings that are contained within the binary. The text strings, as variables defined within the original source code of the program and that it uses during its execution, are found within the executable file. In fact, if the executable does not have any protection mechanism (e.g., code obfuscation or encryption), these strings will be found in the file completely visible. That is, if we viewed the executable file with a word processor, the text strings would be clearly visible. The presence of certain strings is an indication of the activity that the malicious sample will be carrying out. For example, if we see an Internet IP address, or a web domain and parts of strings relating to an HTTP POST or GET request, we can infer that the malicious sample is surely connecting to that IP/web address via HTTP POST or GET requests. For this phase, you can use a tool like `strings` (native tool on UNIX-based systems, and available for Windows at `https://docs.microsoft.com/en-us/sysinternals/downloads/strings`).
   - *Import functions.* Finally, the last subphase of the static analysis consists of analyzing the Windows API functions that the executable file is using. An API function is a function provided by the operating system that makes it easier for other programs to interact with it. That is, it allows the application developer to use system resources such as files, network sockets, registry keys, create processes, etc. For example, the presence of file-related functions (for example, *FindFirstFileA*) will lead to the conclusion that the malware supposedly performs some action with files. To learn more about the functions used by the binary, it is recommended that you consult the MSDN website (`https://msdn.microsoft.com/en-us/library/ms310241`)

for a specific function. Specifically, the MSDN provides information about what a system function does, what parameters it uses, and what it returns. On Windows, tools like `CFF Explorer` (`http://www.ntcore.com/exsuite.php`) can be used to check the import functions.

All these described steps correspond to a basic static analysis of the sample. A more advanced analysis would use disassembly tools or other techniques based on flow control graphs or symbolic execution, among others. In any case, one of the biggest limitations of static analysis is that the code of the binary to be analyzed does not have to be obfuscated or protected. Furthermore, a static analysis shows us **all possible execution paths** of that program. That is, the state space of all these paths can be very large and difficult to manage.

2. **Dynamic analysis phase**. This phase, also called *live* or *hot code analysis phase*, includes the study of the program during its execution. Specifically, in this phase the interaction of the malware sample with its environment is studied. This phase can be divided into two sub-phases, depending on what is meant by environment:
   - *Interaction with the Operating System.* When we understand by environment the interaction made with the operating system, we must attend to three particular extremes: files (*what files is the program creating, modifying, or deleting?*, Registry keys (*what registry keys is the program creating, modifying, or deleting?*), and processes (*what processes is the program creating, modifying, or deleting?*). If the malware sample is observed to be creating new files, it will be necessary to analyze what type of files are created and in the case of executable files, carry out a new malware analysis with them. In the case of Registry keys, it is necessary to check which registry keys are affected (configuration changes, persistence, etc.).
   - *Interaction with the outside (Internet).* Finally, we must look at the interaction that the malware performs with the Internet. Specifically, it will be necessary to locate if the malware sample makes any connection to the Internet (*to which IP addresses or Internet domains is it trying to connect?*), and if it does make any connection, it is necessary to look at what type of information is sending, how it receives data and what type of information it receives. These servers to which you connect to send information and possibly receive orders are called *command and control servers* (*C&C servers*).

   This dynamic analysis phase that we have just described will help us to answer (almost) all the questions related to malware that were discussed at the beginning of this section. More advanced dynamic analysis would require using a debugger to analyze at runtime how CPU registers are evolving, the values returned by Windows functions called by the malware, and so on.

   Finally, it should be noted that, **unlike the static analysis, in the dynamic analysis only one of the possible execution paths is analyzed**, which may also depend on the current execution conditions.

3 Laboratory Assignments

As assignments, in this laboratory you are asked to answer the following questions for each of the malware samples selected. You can get the malware samples used in this laboratory session

as additional material in the official website of the workshop (`https://webdiis.unizar.es/~ricardo/sbc-2022/advanced-malware-analysis/`).

Malware samples of interest to this laboratory are, specifically, the following:

- *lab_malware01* (MD5 hash: `2de244d4bf9851367108fa5d80729aaf`).
- *lab_malware09* (MD5 hash: `998b668724ff88111e6edfbef7034c03`).
- *malware01* (MD5 hash: `e50e87ad5d34cf8d16d01447821d629d`).
- *malware02* (MD5 hash: `0de9765c9c40c2c2f372bf92e0ce7b68`).

Assignment 1.

Are any file in the system being created? If so, how many files and what kind are they? Detail its name and extension, as well as a brief explanation of its content.

Assignment 2.

Does the sample perform some kind of persistence? If so, detail what kind of persistence is doing. If there is any other interaction with the Windows Registry, describe it.

Assignment 3.

Does it interact with any other process? If so, describe which one (or ones) and what interaction performs.

Assignment 4.

Does it connect with an Internet address or domain? If so, detail with which domain or IP is trying to connect. In case it is a domain name, discover the IP address to which the domain name resolves. You can use this web page for this purpose: `http://whois.domantools.com/`.

Assignment 5.

Based on the analysis of the behavior you have observed, **how would you categorize this malware sample?**