Lab 3: Malware Defense

Jonathan Harijanto

January 31, 2017

CS 373: Defense Against the Dark Arts - Winter 2017

Abstract

I. Blog

A. What you looked at:

In this lab assignment, I have to examine a directory named CLASS2 that contains four unidentified files. The only hint that I got was some of these files are malicious, and some of them not. Thus, the very first task that I need to do was to identify the malicious files accurately. From there, I must pick one of the malwares to be further investigated. The programs that were involved in the malware analysis are FileInsight - a sophisticated hex editor to analyze a malware, Yara - a tool to classify malware, Analyzer.py - a tool that executes a malware in an isolated environment then prints the behavior result, and FakeNet - a program that monitors network traffic.

B. How you looked at it:

The four files are named using a hash, so I couldn't obtain any clue by just reading the filename. Thus, I decided to examine these samples using FileInsight so that I could extract all the embedded strings. Based on the results, there was only one file with a hash of `4844FD851088A11E240CFE5B54096209` that's not malicious. Evidently, this file is a freeware named LADS which has a functionality to list all the alternate data streams in Windows.

The next step was to pick a random malware among the three samples to be further analyzed. And I ended up choosing a file with a hash of `068D5B62254DC582F3697847C16710B7`. At first, I didn't know what's the classification of this malware. Thus, I opened the Yara editor to write some rules and find out its classification. Below is the rule that I wrote:

```
rule Class2
{
        strings:
                $str1="KERNEL32.DLL" fullword ascii
                $str2="RegCloseKey" fullword ascii
        condition:
                all of them
}
```

Typically, malware is always aiming for Windows registry and system32 directory. However, not all of them are designed to allocate `KERNEL32.dll` and disrupt the system memory. Thus, I assumed by writing `KERNEL32.dll` and `RegCloseKey` as the rules; I could identify whether my malware sample is messing with Windows memory or not. Surprisingly, Yara found a match in my sample, which means that the sample is most likely wanted to overwrite something in Windows memory. Therefore, I ran the Analyzer.py tool to provide me with a complete result from the malware behavior.

C. What you found:

I analyzed the malware using both FileInsight and Analyzer.py to obtain an accurate conclusion. When I examined it using FileInsight, I noticed some interesting list of strings being extracted from malware `068D5B62254DC582F3697847C16710B7`. The first thing that caught my attention was some strings written in Chinese such as: `Baocun, Xianxin , Zai xian`. Next, I also found an interesting image filename called `background\_mibaoshouji.png`. When I searched for this name on Google, I discovered that it is an anime background picture used in an online game. The game background picture also explains why I saw some strings that state `POST HTTP` and `form-data name=Submit`. Based on the FileInsight analysis, I had a prediction that the malware is an executable file mimicking an online game. However, at this early stage, I might be wrong.

The next action that I did was to rename the hash malware into 'bad' and run both Analyzer.py and FakeNet program. Not long after Analyzer.py executed the malware in an isolated environment, I received

some log files that accurately displayed the malware's activities step by step. When I skimmed over the results, I observed that the malware's job was to invoke a bunch of Windows routines.

In the beginning, it called `NtCreateMutant` routine to create a mutex named hkServer. Next, the malware began to open multiple registry keys inside `HKEY_USERS` and `HKEY_LOCAL_MACHINE` directories using `NtOpenKey`. Also, interestingly enough, the malware was trying to find and open some registry keys that are related to software policies, Microsoft MUI (Multilingual User Interface), and Windows language configuration.

After the malware had enough with browsing the registry, it called a `NtCreateFile` routine to create new files in various locations. The first file was called `wm.ime` and located in a system32 directory. I believe that this IME file is important because I saw that the malware reopened, wrote (or inserted) some data, and loaded some modules (using `LdrLoadDll` routine) quite often. Furthermore, what makes this case more interesting was when the malware reinvoked a `NtOpenKey` routine again to look up for keyboard layouts in `HKEY_LOCAL_MACHINE`. As soon as I saw this, I immediately realized that the malware was trying to find a keyboard layout to enter complex characters, Chinese language (based on FileInsight's result) into the `wm.ime` file.

The final steps performed by the malware was to call `CreateProcessInternalW` routine that will instruct Windows to create a new internal process. The complete instruction sent by the malware was `C:\Windows\system32\cmd.exe /c c:\del438d2b1.bat`. Essentially, it wanted Windows to open a command prompt, go to the specified directory, and execute the batch file.

II. Conclusion

I gained new skills to identify and analyze a malware effectively. This lab taught me how to write a signature rule in Yara and read the log file yield from Cuckoo. Based on the analysis results above, I believe this malware does two things to the host (infected machine). First, it will try to access then write something on Windows system memory. Additionally, it will browse the host's system files, registry keys, and directories as much as possible.