Lab 07-01

Analyze the malware found in the file Lab07-01.exe.

SHA256:
0c98769e42b364711c478226ef199bfbba90db80175eb1b8cd565aa694c09852

Let's put this file against static analysis first.

The malware seems to not be packed, there entropy is quite low (4.2), the section raw sizes against virtual sizes are looking untouched, and the imports doesn't seem hidden.

We have an import to KERNEL32.dll, ADVAPI32.dll and WININET.dll.

The interesting functions from these libraries are:

- Mutex functions (CreateMutexA, OpenMutexA)
- Avoiding activity (Sleep and SetWaitableTimer)
- Files management (GetModuleFileNameA, WriteFile)
- Process injection (VirtualAlloc)
- Services management (CreateServiceA, StartServiceCtrlDispatcherA, OpenSCManagerA)
- Thread management (CreateThread)
- Evasion (GetProcAddress, LoadLibraryA)
- Network activity (InternetOpenUrlA, InternetOpenA)

Flossing the executable revealed such strings:

- http://www.malwareanalysisbook.com (possible C2 server)
- Internet Explorer 8.0 (possible useragent)
- MalService (possible name of the service)
- Malservice (possible name of the service)
- HGL345 (possible name of mutex)

Based on just this info we can tell that the malware will try to hide itself from an analyst, preserve itself on the OS from reboot, it also will try to connect to the internet.

Let's disassemble the executable in IDA to get more insight into the logic of the malware.

At the main function the malware points to a ServiceStartTable, declares a few variables, declares a lpServiceName as "MalService" and lpServiceProc, puts an offset of sub_401040 as a lpServiceProc and then calls a StartServiceCtrlDispatcherA function to connect to a Service Manager.

Sub_401040 is intended to check for actually existing mutex called “HGL345” by using OpenMutexA, if the result is zero (there is no active such mutex), then it continues jump to loc_401064 in order to create it, otherwise if there is an active mutex then it call ExitProcess in order to avoid duplication.

Loc_401064 (in logical code execution order):

1. Creates a mutex (CreateMutexA) with a name “HGL345”,
2. Connects to a Service Control Manager (OpeningSCManagerA with a dwDesiredAccess valued 3 (SC_MANAGER_CONNECT and SC_MANAGER_CREATE_SERVICE),
3. Gets a pseudo-handle of the current process (GetCurrentProcess) but not actually using it later,
4. Calls GetModuleFileNameA with a hModule of 0 to return a filepath of currently running malware thus the previous call was not necessary.
5. Creates a service named “Malservice”, access rights 2 (create service), service type of 10 (SERVICE_WIN32_OWN_PROCESS), start type of 2 (**auto-run during system startup**).
6. Loads the effective address of FileTime, clears SystemTime.wYear, SystemTime.wHour, SystemTime.wSeconds values by entering 0.
7. Enters value of 2100 into SysteMTime.wYear and calls SystemTimeToFileTime.
8. Creates a WaitableTimer object, sets a waitable timer with a corresponding to it WaitForSingleObject call with a dwMiliseconds set to infinite, which means it will continue when the object is signaled.
9. If the WaitForSingleObject returns anything else than zero then it goes into suspension by calling sleep with an infinite parameter
10. If the return value is 0, then the Object has signaled that it’s ready to use and that leads to a loop intended for creating 20 threads.
11. Threads are intended to run immediately and the instructions for them are at StartAddress offset. They are supposed to open a connection, load up the address http://www.malwareanalysisbook.com in an infinite loop.
12. After the loop for creating threads is done, the malware moves into suspension state by calling sleep with an infinite parameter.

1. How does this program ensure that it continues running (achieves persistence) when the computer is restarted?

The malware ensures that it achieves persistence by creating a service “Malservice” with a start type 2 (auto-run during system startup).

2. Why does this program use a mutex?

The program uses mutex to ensure that there is only one instance of malware running at a time.

3. What is a good host-based signature to use for detecting this program?

A good host-based signature would be existence of a service called “Malservice” and a mutex created under a name “HGL345”.

4. What is a good network-based signature for detecting this malware?

A network-based signatures for this malware could be logs of repeatedly connections to http://www.malwareanalysisbook.com under a user agent “Internet Explorer 8.0”.

5. What is the purpose of this program?

The program intends to preserve existence at the OS with service usage and repeatedly connect to a server with 20 threads in an infinite loop.

6. When will this program finish executing?

Actually it shouldn’t happen since the services management is making sure that atleast one copy is running at a time.