**Yara Rules:**

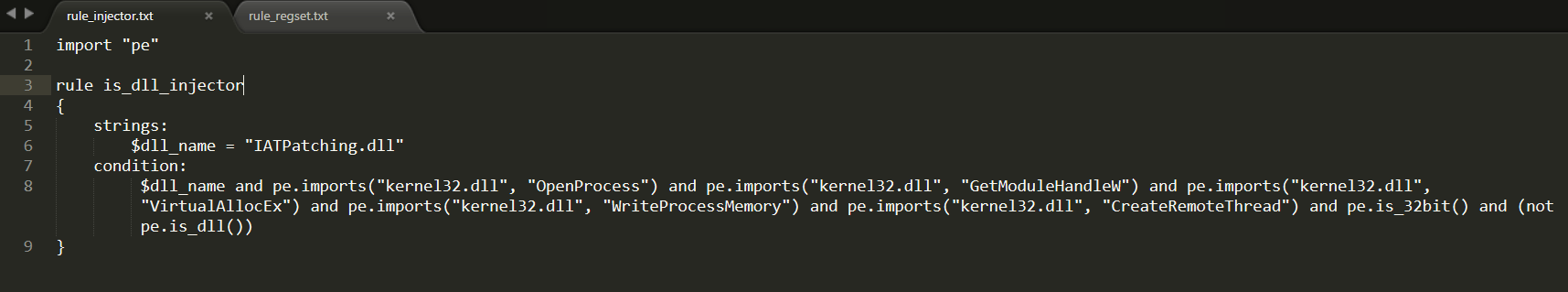
**a)**

We started with YARA rules for the **Injector.exe**:

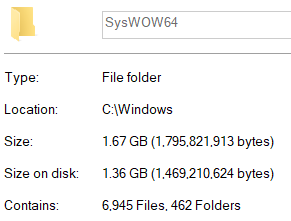
We know its inject DLL name IATPatching.dll so we will search for this string.

We know that in order to do DLL injection we need to use WINAPI: OpenProcess(to get process handle), GetModuleHandleW, VirtualAllocEx, WriteProcessMemory(to copy DLL name to the other process), and CreateRemoteThread (to run code in the other process).

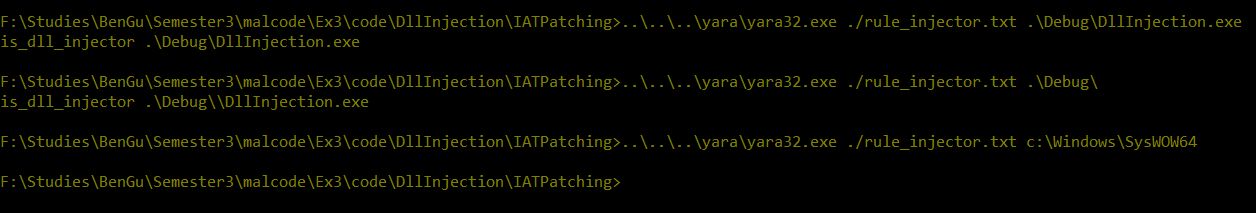
We also know that it is 32 bit executable.



We check this rules on c:\windows\sysWOW64 which contains ~7K PE files:



And our rule is 100% precise; its only detect the malicious DllInjection.exe:



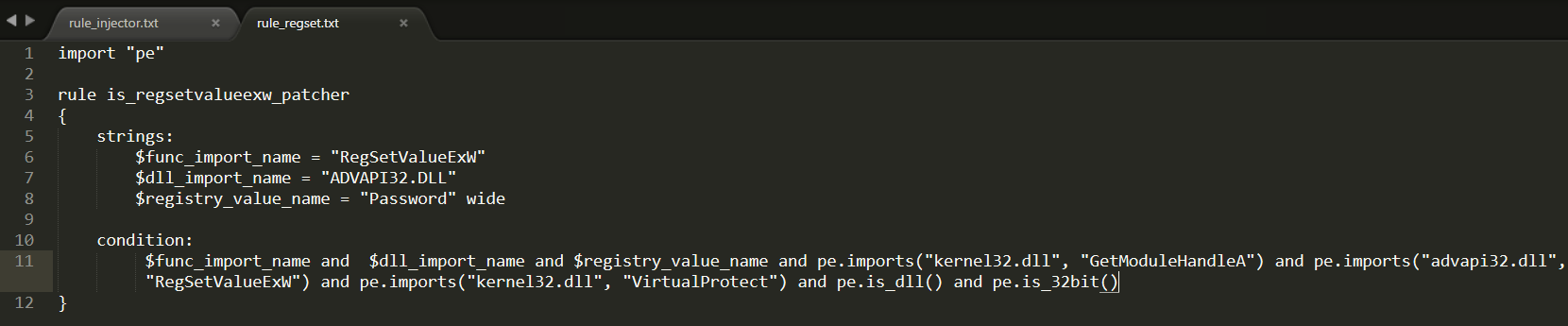
Then we wrote YARA rules for **IATPatching.dll:**

We know its patch the IAT for the WINAPI **RegSetValueExW** so we will search for this string. We also know that it search this string is the DLL **ADVAPI32.DLL** so we will search for this string also.

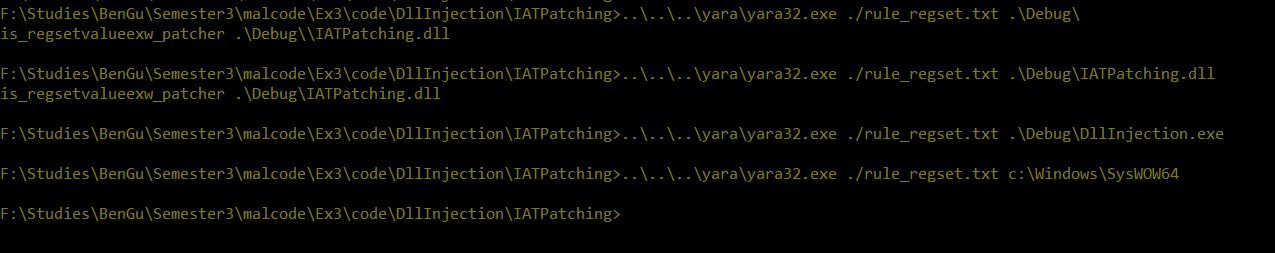
We know that in order to patch the IAT we need to use WINAPI: GetModuleHandleA(to gat base image of the Winscp process), VirtualProtect(to edit the IAT memory), and RegSetValueExW (to return the value of the original WINAPI).

We know that the value that its hook its registry writes is **Password** (in wide-string),so we will search for this string.

We also know that it is 32 bit DLL.



We check this rules on c:\windows\sysWOW64 which contains ~7K PE files, and our rule is 100% precise; its only detect the malicious IATPatching.dll:



**b)**

Both of these rules(rule\_injector.txt, rule\_regset.txt) are depends on 3 types of conditions, that all of them need to occur:

1. Dynamic API imported
2. Strings in file
3. Executable features(DLL vs Executable, 32/64 bit)

Malicious author can avoid being detected by our 1. rules by using Runtime Linking- he can use LoadLibrary and GetProcAddress and we won’t know what functions it imports.

Malicious author can avoid being detected by our 2. rules by using obfuscation - he hides his strings using symmetric encryption and the executable on disk will contains the encrypted strings. When his code needs the string, he use decryption to get the original string. This way- the PE wont contains the string and it won’t be exposed by our YARA rules.

The third type is very generic, and the malicious author won’t need to overcome this because almost all windows application is dll/exe or 32/64 (this rule doesn’t indicate of malicious software).