

BLUE TEAM LABS ONLINE

REPORT ON CHALLENGE

“Reverse Engineering - A Classic Injection”

The analysis was conducted on a Kali Linux environment, making network adapter configuration irrelevant, as the malware relied exclusively on Windows DLL functions for its operations.

Ghidra file analysis:

| | |
|------------------------------------|---|
| Project File Name: | analyseme.exe |
| Last Modified: | Tue Dec 16 11:44:49 EST 2025 |
| Readonly: | false |
| Program Name: | analyseme.exe |
| Language ID: | x86:LE:32:default (4.6) |
| Compiler ID: | windows |
| Processor: | x86 |
| Endian: | Little |
| Address Size: | 32 |
| Minimum Address: | 00400000 |
| Maximum Address: | 004073ff |
| # of Bytes: | 17456 |
| # of Memory Blocks: | 6 |
| # of Instructions: | 0 |
| # of Defined Data: | 754 |
| # of Functions: | 1 |
| # of Symbols: | 86 |
| # of Data Types: | 39 |
| # of Data Type Categories: | 4 |
| Compiler: | visualstudio:unknown |
| Created With Ghidra Version: | 11.4.3 |
| Date Created: | Tue Dec 16 11:44:49 EST 2025 |
| Executable Format: | Portable Executable (PE) |
| Executable Location: | /home/kali/Desktop/BTLO Reverse Engineering - A Classic Injection/analyse |
| Executable MD5: | 66bd29e885429f3e371e745ca32896b1 |
| Executable SHA256: | ff362a3f7078f8b5793e8d2cac35de29aeldab6608cfcl545c24c9e2372c892a |
| FSRL: | file:///home/kali/Desktop/BTLO Reverse Engineering - A Classic Injection/ |
| PDB Age: | 1 |
| PDB File: | btlo.pdb |
| PDB GUID: | 9bd3cc3b-709f-43b7-9081-a659986971a9 |
| PDB Version: | RSDS |
| Preferred Root Namespace Category: | |
| Relocatable: | true |
| SectionAlignment: | 4096 |

The strings utility was employed to extract and examine any readable ASCII characters present within the executable

```
L$ strings analyseme.exe
!This program cannot be run in DOS mode.
Rich
.text
.rdata
@.data
.rsrc
@.reloc
SVWP
5$T@
Y_^[
h%+@
SVW
@t=f
Y_^[
hP+@
hp+@
0SVW
HHuT
Y_^[
h0<@
hl1@
h`1@
h\1@
hP1@
Y_^[
hP<@
```

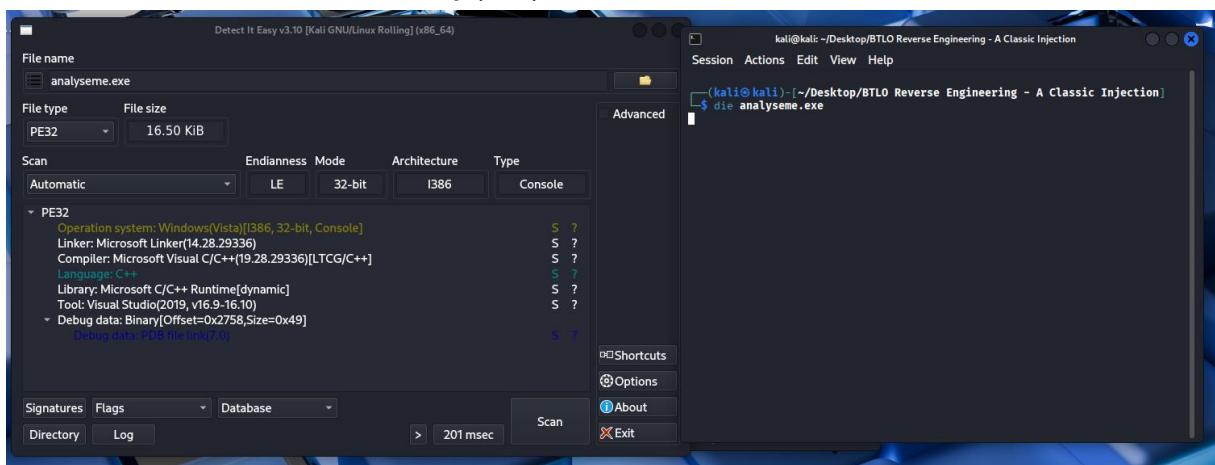
Several characteristic strings associated with shellcode injection techniques were identified.

```
.rsrc$02
WriteProcessMemory
WaitForSingleObject
Sleep
VirtualAllocEx
CreateProcessW
CreateRemoteThread
KERNEL32.dll
?? 1_Lockit@std@@QAE@XZ
api-ms-win-crt-locale-l1-0.dll
UnhandledExceptionFilter
SetUnhandledExceptionFilter
GetCurrentProcess
TerminateProcess
IsProcessorFeaturePresent
QueryPerformanceCounter
GetCurrentProcessId
GetCurrentThreadId
GetSystemTimeAsFileTime
InitializeSListHead
IsDebuggerPresent
GetModuleHandleW
memcpy
.?AVtype_info@?
.?AVbad_alloc@std@@
.?AVbad_alloc@std@@
```

QUESTION 1

What is the name of the compiler used to generate the EXE?

The objective of the first question was to identify the compiler used to generate the executable file. The Detect It Easy (DIE) tool was utilized to obtain this information.



The analysis revealed that the executable was compiled using Microsoft Visual Studio, and the programming language used was C/C++.

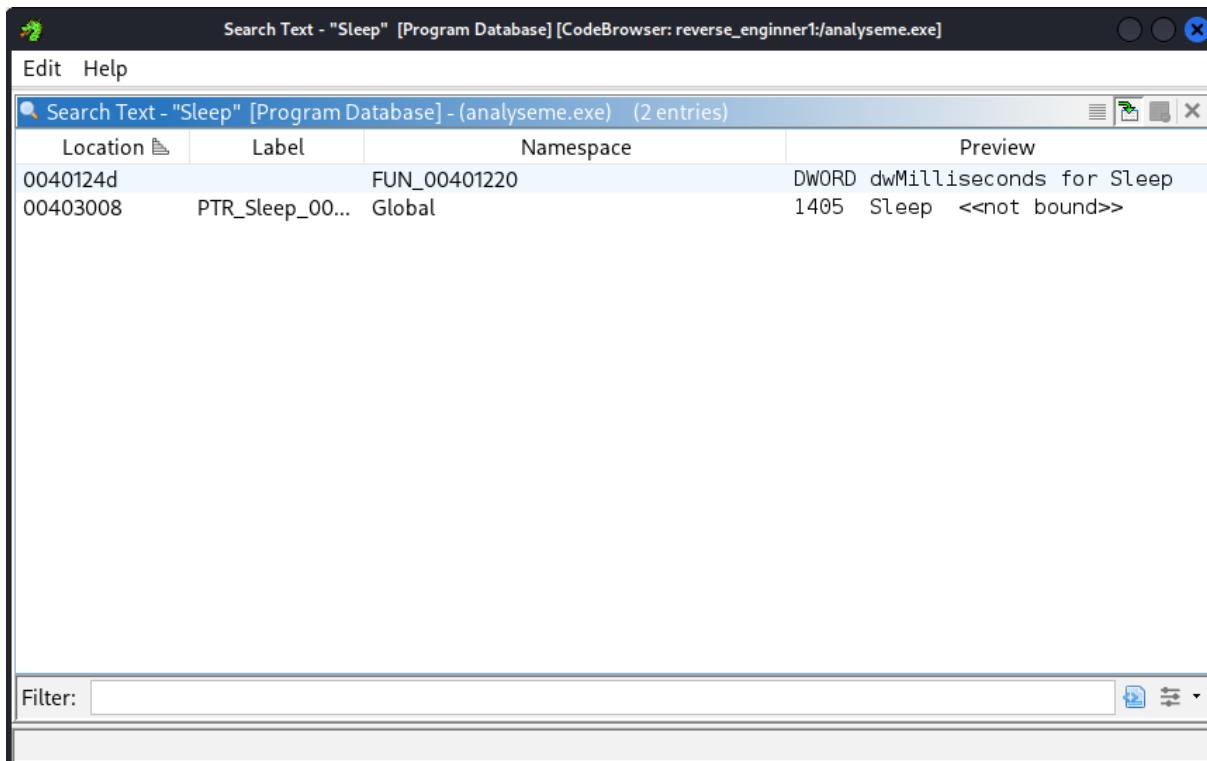
What is the name of the compiler used to generate the EXE? (1 points)

Microsoft Visual C++

QUESTION 2

This malware, when executed, sleeps for some time. What is the sleep time in minutes?

The objective of the second question was to determine the sleep duration used by the Sleep function within the executable. Ghidra was used for this analysis.



A screenshot of the Ghidra Code Browser interface. The title bar reads "Search Text - 'Sleep' [Program Database] [CodeBrowser: reverse_enginner1:/analyseme.exe]". The main window shows a table titled "Search Text - 'Sleep' [Program Database] - (analyseme.exe) (2 entries)". The table has columns: Location, Label, Namespace, and Preview. There are two entries:

| Location | Label | Namespace | Preview |
|----------|-----------------|--------------|--------------------------------|
| 0040124d | | FUN_00401220 | DWORD dwMilliseconds for Sleep |
| 00403008 | PTR_Sleep_00... | Global | 1405 Sleep <<not bound>> |

At the bottom left, there is a "Filter:" input field and some toolbar icons.

By searching for the string “Sleep” in the Code Browser, Ghidra revealed that the Sleep function was invoked within the function FUN_00401220, with an argument value of 180,000 milliseconds.

```
29 local_14 = DAT_00405008 ^ (uint)&stack0xfffffffffc;
30 ExceptionList = &local_10;
31 Sleep(180000);
32 local_1f8 = 0;
33 local_1f4 = 0xf;
```

Convert time: 180000 ms (millisecond) to other units

Convert

Select input unit of time:

ns (nanosecond)
us (microsecond)
ms (millisecond)
s (second)
min (minute)
h (hour)
d (day)
week
year
decade
century
millennium

180000 ms (millisecond) equals to:

180000000000 ns (nanosecond)
180000000 us (microsecond)
180000 ms (millisecond)
180 s (second)
3 min (minute)
0.05 h (hour)
0.0020833333 d (day)
0.00029761905 week
5.70776E-6 year
5.7078E-7 decade
5.708E-8 century
5.71E-9 millennium

This malware, when executed, sleeps for some time. What is the sleep time in minutes? (1 points)

3

Correct! ✓

QUESTION 3

After the sleep time, it prompts for user password, what is the correct password?

Beneath the Sleep function call, the following code fragment was identified.

```
46 }  
47 pbVar4 = (byte *)&DAT_00403210;  
48 while (uVar1 = uVar5 - 4, 3 < uVar5) {  
49     if (*pppbVar2 != *(byte ****)pbVar4) goto LAB_004012e6;  
50     pppbVar2 = pppbVar2 + 1;  
51     pbVar4 = pbVar4 + 4;  
52     uVar5 = uVar1;  
53 }  
54 if (uVar1 != 0xffffffffc) {  
55 LAB_004012e6.
```

This fragment appears to compare values and copy the contents of the variable pbVar4. The variable is assigned a value originating from the program's data section, suggesting that pbVar4 may be used to store a hidden password.

The disassembly responsible for assigning the value to pbVar4 is shown below.

| | |
|--|--|
| 004012a8 8b bd fc MOV EDI,dword ptr [EBP + local_208] 004012ae ba 04 00 MOV EDX,0x4 004012b3 8b b5 0c MOV ESI,dword ptr [EBP + interesting_value] ff ff ff 004012b9 0f 43 cf CMOVNC ECX,EDI 004012bc 3b f2 CMP ESI,EDX 004012be 0f 47 f2 CMOVNA interesting_value_copy,EDX 004012c1 ba 10 32 MOV EDX,DAT_00403210 40 00 004012c6 83 ee 04 SUB interesting_value_copy,0x4 004012c9 72 16 JC LAB_004012e1 004012d0 48 8b 00 00 NNP dword ptr [EAX + FAV0v1] | 42 } 43 interesting_value_copy = interesting_value; 44 if (4 < interesting_value) { 45 interesting_value_copy = 4; 46 } 47 pbVar4 = (byte *)DAT_00403210; 48 while (uVar1 = interesting_value_copy - 4, 3 < interesting_value_copy) { 49 if (*pppbVar2 != *(byte ****)pbVar4) goto LAB_004012e6; 50 pppbVar2 = pppbVar2 + 1; 51 pbVar4 = pbVar4 + 4; 52 interesting_value_copy = uVar1; 53 } 54 if (uVar1 != 0xffffffffc) { 55 LAB_004012e6; 56 bVarR = *(byte *)nnnnVar2 < *nbVar4; |
|--|--|

Analysis of the disassembly indicates that the value stored at DAT_00403210 is 0x6F6C7462.

Moving on to CyberChef, the value 0x6F6C7462 was analyzed and found to represent the ASCII string “oltb” encoded in hexadecimal format.

The screenshot shows the CyberChef interface. In the 'Input' section, the value '6F6C7462h' is entered. The 'Recipe' section is set to 'From Hex'. The 'Output' section displays the ASCII string 'oltb'. A status bar at the bottom indicates 'asc 9' and '1'.

Due to the little-endian byte order, where the least significant byte is stored first, the resulting text appears reversed.

By applying the reverse operation to this string, the password was obtained:

The screenshot shows a terminal window with the following interface:

- Recipe** tab:
 - From Hex**: Input field contains "6F6C7462h".
 - Delimiter: Auto
 - Reverse**: Input field contains "By Character".

On the right side of the terminal window, there is an **Input** field containing "6F6C7462h" and an **Output** field containing "btlo".

After the sleep time, it prompts for user password, what is the correct password? (1 points)

btlo

Correct! ✓

QUESTION 4

What is the size of the shellcode?

To obtain this information, additional research was conducted. The [Red Team Notes Site](#) resource indicates that one of the commonly used functions in shellcode injection techniques particularly for determining shellcode size is VirtualAllocEx.

Syntax

```
C++  
LPVOID VirtualAllocEx(  
    [in]         HANDLE hProcess,  
    [in, optional] LPVOID lpAddress,  
    [in]         SIZE_T dwSize,  
    [in]         DWORD  flAllocationType,  
    [in]         DWORD  flProtect  
) ;
```

Copy

This function was subsequently identified in the Code Browser within Ghidra.

```
lpStartAddress =  
    (LPTHREAD_START_ROUTINE)VirtualAllocEx(local_25c.hProcess, (LPVOID)0x0, 473, 0x3000, 0x40);
```

According to the function's syntax, the third parameter specifies the size of the memory region to be allocated, which corresponds to the shellcode size.

What is the size of the shellcode? (1 points)

473

Correct! ✓

QUESTION 5

Shellcode injection involves three important windows API. What is the name of the API Call used?

The strings utility was once again used on the executable, revealing additional readable ASCII strings. Further research was conducted to determine the significance of each identified string.

```
.rsrc$02  
WriteProcessMemory  
WaitForSingleObject  
Sleep  
VirtualAllocEx  
CreateProcessW  
CreateRemoteThread  
KERNEL32.dll  
?? 1_Lockit@std@@QAE@XZ
```

Shellcode injection involves three important windows API. What is the name of the API Call used? (2 points)

CreateRemoteThread

Correct! ✓

QUESTION 6

What is the name of the victim process?

According to the [Red Team Notes Site](#), the CreateProcess function can be used to spawn a new process. A reference to this function was identified in the Ghidra Code Browser.

```
CreateProcessW(L"C:\\\\Windows\\\\System32\\\\nslookup.exe", (LPWSTR)0x0, (LPSECURITY_ATTRIBUTES)0x0,
               (LPSECURITY_ATTRIBUTES)0x0, 0x80000000, (LPVOID)0x0, (LPCWSTR)0x0, &local_24c,
               &local_25c);
```

WaitForSingleObject(local_25c.hProcess, 1000);

Syntax

C++

```
BOOL CreateProcessA(
    [in, optional]    LPCSTR          lpApplicationName,
    [in, out, optional] LPSTR           lpCommandLine,
    [in, optional]    LPSECURITY_ATTRIBUTES lpProcessAttributes,
    [in, optional]    LPSECURITY_ATTRIBUTES lpThreadAttributes,
    [in]              BOOL            bInheritHandles,
    [in]              DWORD           dwCreationFlags,
    [in, optional]    LPVOID          lpEnvironment,
    [in, optional]    LPCSTR          lpCurrentDirectory,
    [in]              LPSTARTUPINFOA   lpStartupInfo,
    [out]             LPPROCESS_INFORMATION lpProcessInformation
);
```

Copy

Based on the function's syntax, the process name is specified in the first parameter.

What is the name of the victim process? (1 points)

nslookup.exe

Correct! ✓

QUESTION 7,8,9

What is the file created by the sample

What is the message in the created file

What is the program that the shellcode used to create and write this file

The executable was observed to pause execution for approximately three minutes before performing further actions. Since the Any.Run free plan limits dynamic analysis to 60 seconds, the executable was patched to bypass this delay.

The following code fragment is responsible for setting up the argument for the Sleep function. The argument is passed via the stack in accordance with the x86 calling convention, where the value is pushed onto the stack prior to invoking the function.

```
0040123e 89 45 fo    MOV     dword ptr [EBP + local_14],EAX
00401241 56          PUSH    ESI
00401242 57          PUSH    EDI
00401243 50          PUSH    EAX
00401244 8d 45 f4    LEA     EAX=>local_10,[EBP + -0xc]
00401247 64 a3 00    MOV     FS:[0x0]=>ExceptionList,EAX
00 00 00
0040124d 68 20 bf    PUSH    0x2bf20
02 00
00401252 ff 15 08    CALL    dword ptr [->KERNEL32.DLL::Sleep]
30 40 00
00401258 c7 85 0c    MOV     dword ptr [EBP + local_1f8],0x0
fe ff ff
00 00 00 00
25
26 local_8 = 0xffffffff;
27 puStack_c = &LAB_00402aa0;
28 local_10 = ExceptionList;
29 local_14 = DAT_00405008 ^ (*
30 ExceptionList = &local_10;
31 Sleep(180000);
32 local_1f8 = 0;
33 local_1fa = 0xf;
34 local_208[0] = (byte ***)((
35 local_8 = 0;
36 FUN_004015f0((basic_ostream<
37 FUN_00401a40((basic_istream<
38 pppbVar3 = (byte ****)local
```

The Patch Instruction option in Ghidra was used to modify this behavior.

| Comments > | | | |
|---|-------------------------------------|------|-----------|
| Instruction Info | | | |
| Modify Instruction Length... Patch Instruction Ctrl+Shift+G | | | |
| 1401243 | 50 | PUSH | EAX |
| 1401244 | 8d 45 f4 | LEA | EAX=>lo |
| 1401247 | 64 a3 00 00 00 00 | MOV | FS: [0x0] |
| 140124d | 68 20 bf 02 00 | PUSH | 0x2bf20 |
| 1401252 | ff 15 08 30 40 00 | CALL | dword p |
| 1401258 | c7 85 0c fe ff ff 00 00 00 00 | MOV | dword p |
| 1401262 | c7 85 10 fe ff ff | MOV | dword p |

Originally, the stack contained the value corresponding to the Sleep function delay.

Hexadecimal to Decimal converter

From To

Hexadecimal Decimal

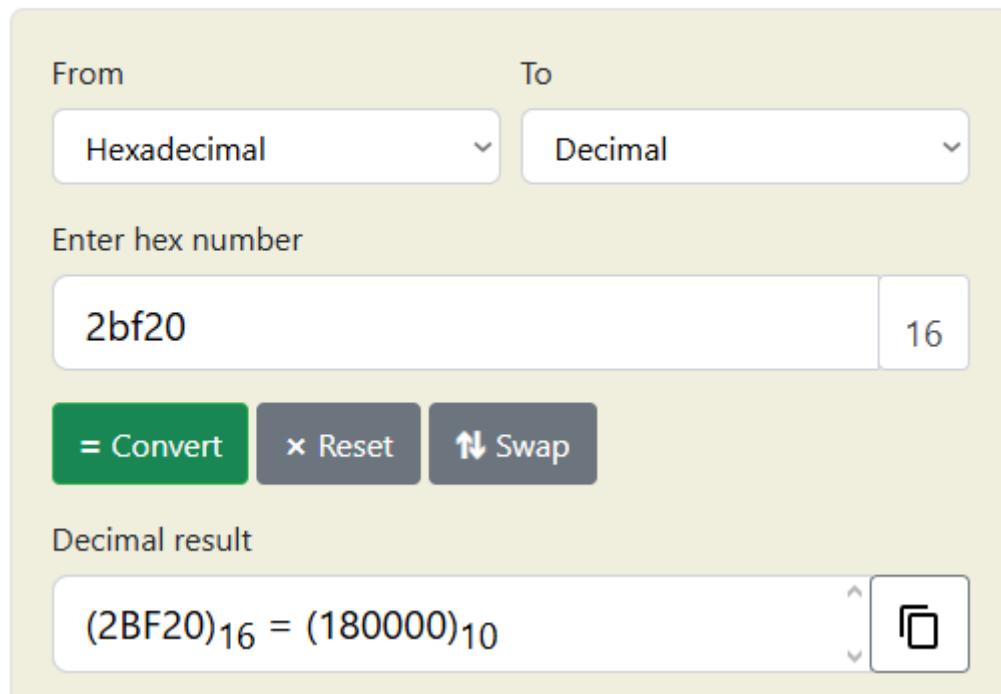
Enter hex number

2bf20 16

= Convert x Reset Swap

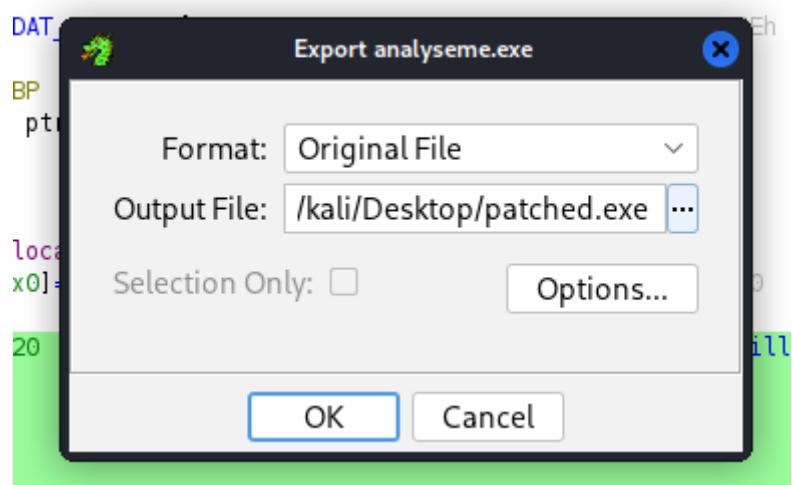
Decimal result

(2BF20)₁₆ = (180000)₁₀



To prevent the execution of the Sleep function, the original call was replaced with NOP instructions, effectively disabling the delay.

| | | | |
|--|------|----------------------------------|--------------------------------|
| 00401247 64 a3 00 00 00 00 | MOV | FS: [0x0] => ExceptionList, EAX | = 00000000 |
| 0040124d 68 20 bf 02 00 | PUSH | 0x2bf20 | DWORD dwMilliseconds for Sleep |
| 00401252 90 | NOP | | |
| 00401253 90 | NOP | | |
| 00401254 90 | NOP | | |
| 00401255 90 | NOP | | |
| 00401256 90 | NOP | | |
| 00401257 90 | NOP | | |
| 00401258 c7 85 0c fe ff ff 00 00 00 00 | MOV | dword ptr [EBP + local_1f8], 0x0 | |



The patched executable was then exported and uploaded into the Any.Run environment for further analysis.

Deep analysis Safebrowsing beta

Simple mode Pro mode New VM video streaming beta Analyses: Public

URL or file upload

patched.exe

Start object from Desktop Open in browser Microsoft Edge

Change extension to valid On Hide source

Command line Type or choose a preset

Duration, sec 60

Network Connected

HTTPS MITM PROXY Fake net

Route internet traffic through (optional):

Route via TOR Residential proxy User VPN (0/100)

Fastest geo Choose Add a config +

Preset configuration (1/100)

Default + Autosave changes

Operating system Windows 10 (64 bit)

Auto confirm UAC On

Pre-installed soft set Complete Locale (OS Language) United States (en-US)

Applications Hot fixes Tools collection

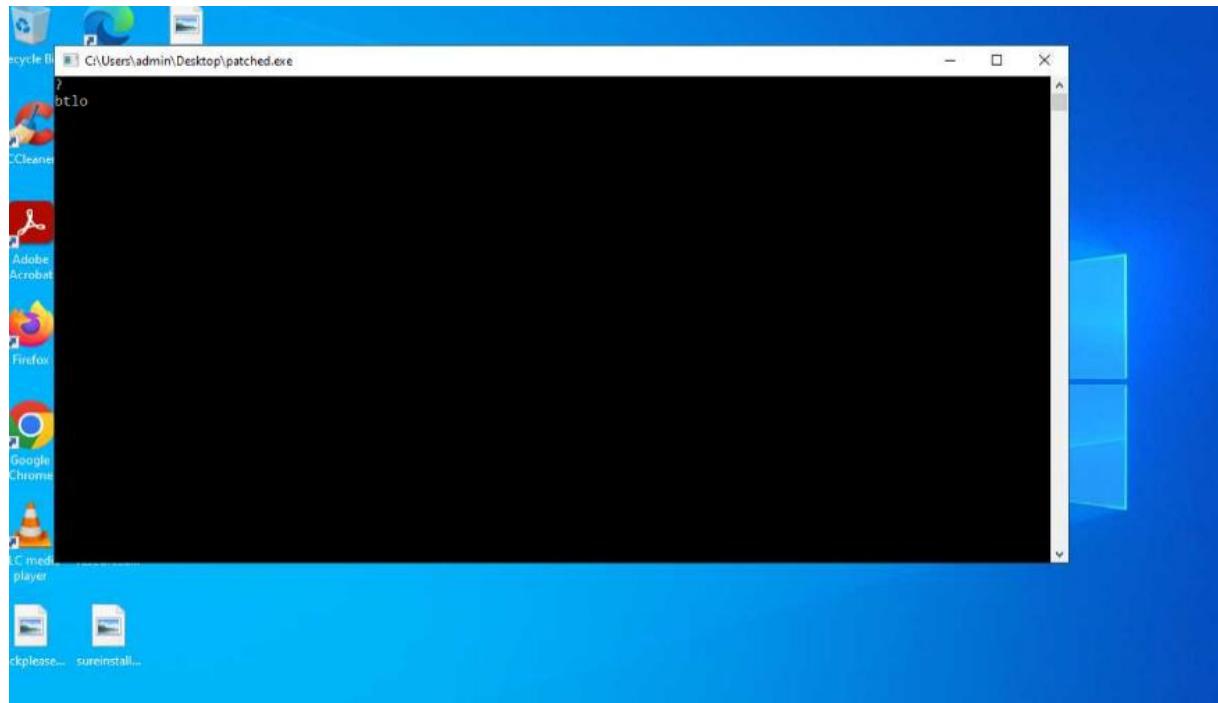
Additional settings Automated Interactivity (ML)

Privacy Only me Team Who has a link Public

The report will be deleted in 2 weeks

Run a public analysis Auto

Upon execution, the modified executable prompted the user for a password.



Any Run Summary:

ANY RUN
INTERACTIVE MALWARE ANALYSIS

General Behavior MalConf Static information Video Screenshots System events Network  

General Info

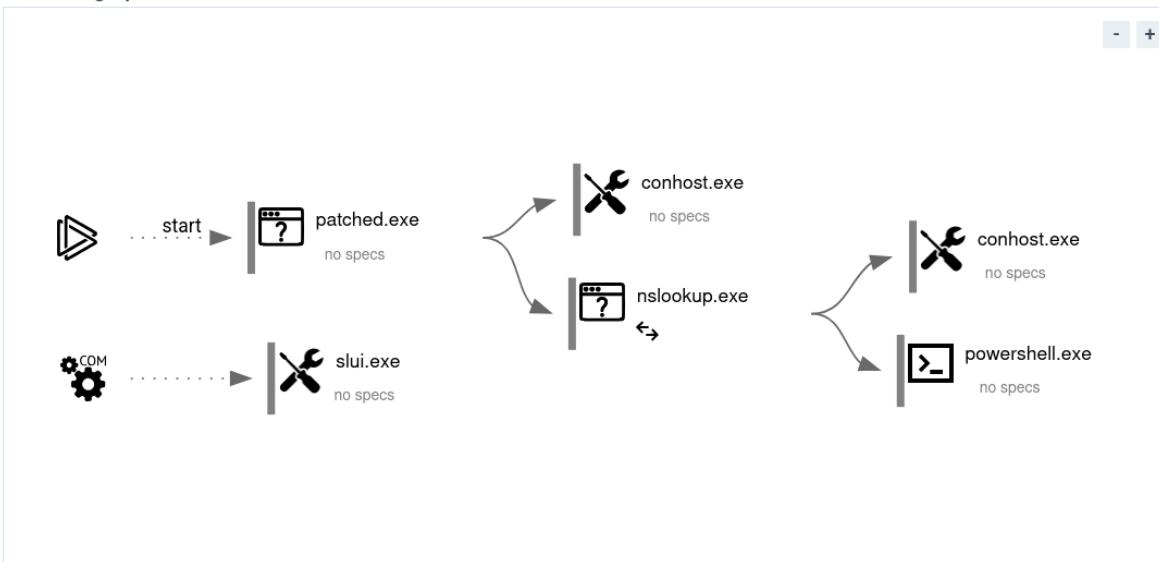
Add for printing ▾

| | |
|----------------|---|
| File name: | patched.exe |
| Full analysis: | https://app.any.run/tasks/5da5b035-c7e3-40a8-9af8-826c2586fb26 |
| Verdict: | No threats detected |
| Analysis date: | December 17, 2025 at 12:32:33 |
| OS: | Windows 10 Professional (build: 19044, 64 bit) |
| Indicators: |  |
| MIME: | application/vnd.microsoft.portable-executable |
| File info: | PE32 executable (console) Intel 80386, for MS Windows, 5 sections |
| MD5: | 2A05ECE7A0042F3F29E12834BCE02DCA |
| SHA1: | F3FBF9AAC138A793E063E4241412F253DD9DEA87 |
| SHA256: | E38F4D0963EF7FD6ADF049B4DC35F809161317DEB903F666AC81CE739A8EF29 |
| SSDeep: | 192:TGGy///7GG4Z48MOA5sbdh3uKAEViQuwR4vhlpZmd2wE3imRwysLAZrgoo0P7GN:MoishheKAEKplpZcs3ih7K |

ANY.RUN is an interactive service which provides full access to the guest system. Information in this report could be distorted by user actions and is provided for user acknowledgement as it is. ANY.RUN does not guarantee maliciousness or safety of the content.

Behavior graph

Click at the process to see the details - +



```
graph LR; Start(( )) -- start --> Patched[patched.exe  
no specs]; COM[COM] -.-> Slui[slui.exe  
no specs]; Patched --> Conhost1[conhost.exe  
no specs]; Patched --> Nslookup[nslookup.exe]; Nslookup --> Conhost2[conhost.exe  
no specs]; Nslookup --> Powershell[powershell.exe  
no specs]
```

The behavior graph illustrates the execution flow of the malware. It starts with a 'start' event leading to the execution of 'patched.exe'. Simultaneously, a 'COM' object interacts with 'slui.exe'. The 'patched.exe' process then initiates two parallel tasks: one leading to 'conhost.exe' and another leading to 'nslookup.exe'. Finally, 'nslookup.exe' triggers two more parallel tasks, both leading to 'conhost.exe' and 'powershell.exe' respectively.

During dynamic analysis, it was observed that nslookup.exe executed a PowerShell command. The following encoded command was identified:

The screenshot shows a debugger interface with two windows. The top window is titled 'nslookup.exe' and shows a command-line input field with the following text:

```
powershell.exe -enc TgBlAHcALQBjAHQAZQBtACAAQwA6AFwAVwBpAG4AZABvAHcAcwBcAHQAZQBtAHAXXABiAHQAbABvAC4AdAB4AHQACgBTAGUAdAAtAEMAbwBuAHQAZQBuAHQAIABDADoAXABXAGkAbgBkAG8AdwBzAFwAdABIAG0AcABCAGIAdABSAG8ALgB0AHgAdAAgACC AVwB1AGwAYwBvAG0AZQAgAHQAbwAgAEIAVABMAE8AIQAnAA==
```

The bottom window is titled 'Information' and displays the following details about the process:

| User: | admin | Company: | Microsoft Corporation |
|------------------|--------|--------------|-------------------------------------|
| Integrity Level: | MEDIUM | Description: | Windows PowerShell |
| Exit code: | 0 | Version: | 10.0.19041.1 (WinBuild.160101.0800) |

The 'Modules' section lists several system DLLs and the PowerShell executable:

- c:\windows\syswow64\windowspowershell\v1.0\powershell.exe
- c:\windows\system32\ntdll.dll
- c:\windows\syswow64\ntdll.dll
- c:\windows\system32\wow64.dll
- c:\windows\system32\wow64win.dll
- c:\windows\system32\wow64cpu.dll
- c:\windows\syswow64\kernel32.dll
- c:\windows\syswow64\kernelbase.dll
- c:\windows\syswow64\msvcrt.dll
- c:\windows\syswow64\oleaut32.dll

```
powershell.exe -enc TgBlAHcALQBjAHQAZQBtACAAQwA6AFwAVwBpAG4AZABvAHcAcwBcAHQAZQBtAHAXXABiAHQAbABvAC4AdAB4AHQACgBTAGUAdAAtAEMAbwBuAHQAZQBuAHQAIABDADoAXABXAGkAbgBkAG8AdwBzAFwAdABIAG0AcABCAGIAdABSAG8ALgB0AHgAdAAgACC AVwB1AGwAYwBvAG0AZQAgAHQAbwAgAEIAVABMAE8AIQAnAA==
```

The presence of the `-enc` flag indicates an encoded PowerShell command, which must be provided in Base64 format using UTF-16LE encoding.

The image consists of two vertically stacked screenshots of the CyberChef web application. Both screenshots show the same input: a long string of characters including null bytes (represented by the character 'NUL').

Top Screenshot:

- Recipe:** From Base64
- Alphabet:** A-Za-z0-9+=
- Input:** The base64-encoded string.
- Output:** The output is a large string of null bytes (NUL) repeated many times.

Bottom Screenshot:

- Recipe:** From Base64
- Alphabet:** A-Za-z0-9+=
- Input:** The same base64-encoded string.
- Operations:**
 - From Base64
 - Remove null bytes
- Output:** The output is the PowerShell command `New-Item C:\Windows\temp\btlo.txt` followed by `Set-Content C:\Windows\temp\btlo.txt 'Welcome to BTLO!'`.

Using CyberChef, the command was decoded by applying the “From Base64” operation followed by “Remove Null Bytes”, revealing the following commands:

```
New-Item C:\Windows\temp\btlo.txt
Set-Content C:\Windows\temp\btlo.txt 'Welcome to BTLO!'
```

What is the file created by the sample (1 points)

C:\Windows\temp\btlo.txt

Correct! ✓

What is the message in the created file (1 points)

'Welcome to BTLO!'

Correct! ✓

What is the program that the shellcode used to create and write this file (1 points)

powershell.exe

Correct! ✓