Event Log Tampering Part 1: Disrupting the EventLog Service

By svch0st

Published: 2020-10-03 · Archived: 2026-04-05 21:35:26 UTC

6 min read

Oct 1, 2020

Windows event logs are a fundamental source of data and evidence for incident response. Attackers will target this source to slow down the response by clearing or tampering logs (T1070). Although there are other artefacts that these activities would not be able to hide from, it is still a popular anti-forensic technique.

> You Can't See Me

For this set of blog posts, I wanted to focus on slightly more involved **anti-forensic methods** and look at examples for each method**,** such as**:**

Part 1: Disrupting the EventLog Service

- **Service Host Thread Tampering** (*Invoke-Phant0m*)
- **Patching the Event Service** (*Mimikatz*)
- **Downgrading Windows Components** (Adding MiniNT key)

Part 2: Manipulating individual event logs

- **Evtx Structure & Manual Event Editing** (*A must-read to understand the following sections)*
- **Event Record Unreferencing** (Shadow Brokers Tools *DanderSpritz/eventlogedit)*
- **Rewriting Logs with WinAPI EvtExportLog** (3gstudent's evolutions of *eventlogedit)*

Part 3: Combining Techniques

Just for completeness sake, the more common and already heavily documented methods are:

Clear the Log
Example: `wevtutil cl Security` or `Clear-EventLog`
Detected by: `Security Event ID 1102, System Event ID 104` or command line usage of `wevtutil`

Disable the Event Log Service
Example: `sc stop EventLog`
Detected by: `Service Control Manager Event ID 7035` or command line usage

Although some of the more advanced methods will use these steps, I wanted to put these aside and focus on the more involved techniques.

Disrupting the EventLog Service

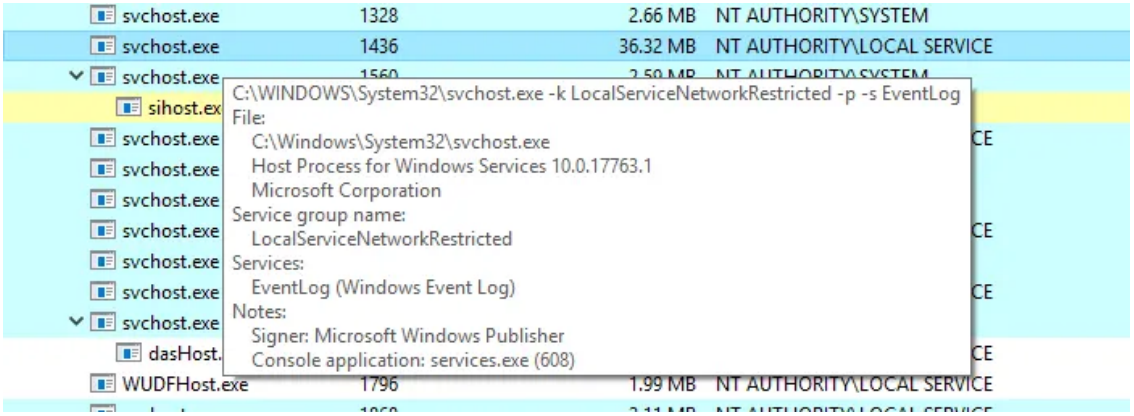
The goal of the methods I will go over below are to impact the service responsible for Event Logging that will **result in no logs recorded.** This will leave a hole in your timelines or be used to clear the event log without being recorded.

Service Host Thread Tampering

Let's quickly have a look at how the EventLog service runs. Each service will be associated with an instance of `svchost.exe` so we need to find which one EventLog uses.

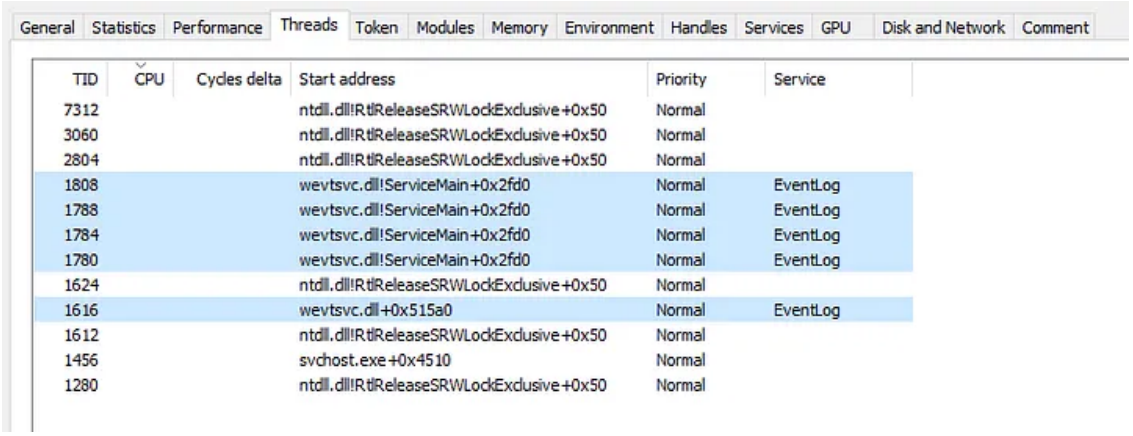
You can see below the EventLog service is running in the `svchost.exe` with a PID of `1436`.

Press enter or click to view image in full size



Below are the threads that are related to the EventLog service in the svchost.exe process.

Press enter or click to view image in full size



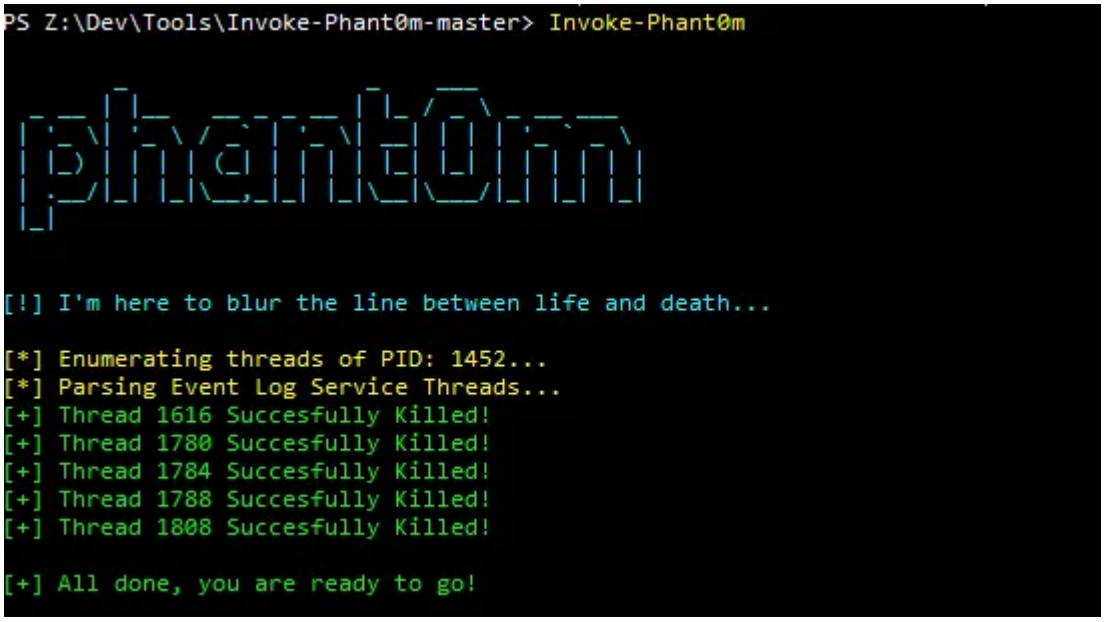
These are basically the worker threads of the service. If we can tamper with these, then we can affect the event log. We will use the tool Invoke-Phant0m to showcase this method.

Invoke-Phant0m uses the following steps:

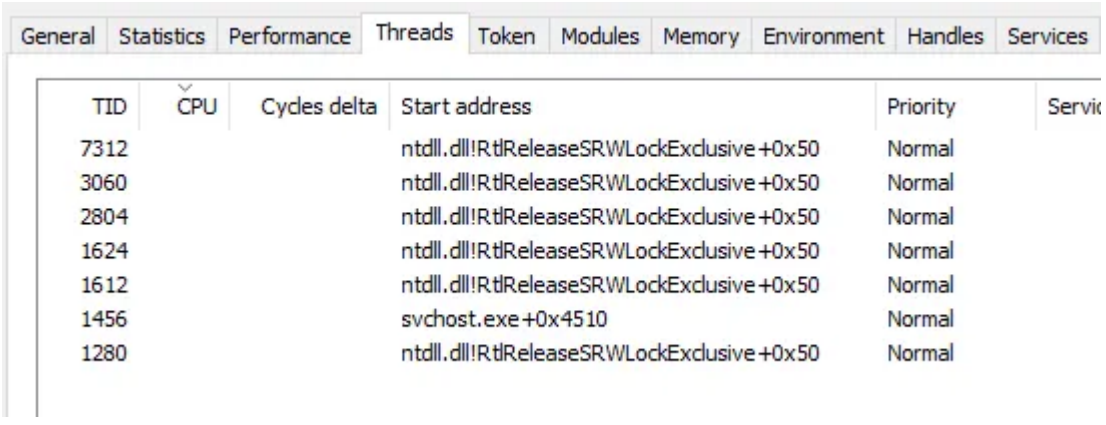
> 1. Detect the process of the Windows Event Log Service in the target
> 2. Get thread list and identify the Windows Event Log Service thread IDs.
> 3. Kill all threads about the Windows Event Log Service.

This list is from the following blog which also is a more detailed write up on how Phant0m works:

Once we run `Invoke-Phant0m`, it will locate all the threads associated with the EventLog service (*see in the pic above to compare thread IDs*) and kill them.

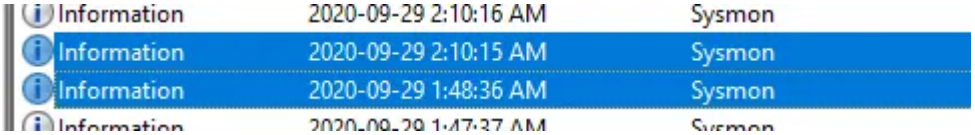


You can now see that there are no threads in the svchost.exe process and there will no longer be events written to the log.



If we restart the service using `net stop/start eventlog`, the threads return and the eventing starts to return with no indication of what happened in between.

Press enter or click to view image in full size



A time gap in logs from where I started Phant0m and restarted the service.

This technique is quieter than simply disabling the event service. During the time the threads are killed, you could clear the event log **without** leaving behind the `Security Event ID 1102` indicator. Later on, we will also look at just **suspending the threads** instead of killing them with Phant0m.

Patching the Event Service

Mimikatz currently has a module to be able to patch the event log service and then clear the log.

```
mimikatz # privilege::debug
Privilege '20' OK

mimikatz # event::
ERROR mimikatz_doLocal ; "(null)" command of "event" module not found !

Module :        event
Full name :     Event module

           drop  -  [experimental] patch Events service to avoid new events
          clear  -  Clear an event log

mimikatz # _
```

This method is simple but effective because it doesn't leave behind the `Security Event ID 1102` indicator when you go to clear the log just like the first example.

Get svch0st's stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

Mimikatz targets `wevtsvc.dll` (the Windows Event Service DLL) that is loaded in the svchost.exe responsible for the EventLog service.

Here is a link to the source code of the module.

First, Mimikatz will find the function `Channel::ActualProcessEvent` depending on the Windows version using predefined patterns. This function is responsible for writing the events to the log.

Press enter or click to view image in full size

```
#if defined(_M_X64) || defined(_M_ARM64) // TODO:ARM64
BYTE PTRN_WNT5_PerformWriteRequest[]                  = {0x49, 0x89, 0x5b, 0x10, 0x49, 0x89, 0x73, 0x18};
BYTE PTRN_WN60_Channel__ActualProcessEvent[]    = {0x48, 0x89, 0x5c, 0x24, 0x08, 0x57, 0x48, 0x83, 0xec, 0x20, 0x48, 0x8b, 0xf9, 0x
BYTE PTRN_WIN6_Channel__ActualProcessEvent[]    = {0xff, 0xf7, 0x48, 0x83, 0xec, 0x50, 0x48, 0xc7, 0x44, 0x24, 0x20, 0xfe, 0xff, 0x
BYTE PTRN_WI10_Channel__ActualProcessEvent[]    = {0x48, 0x8b, 0xc4, 0x57, 0x48, 0x83, 0xec, 0x50, 0x48, 0xc7, 0x40, 0xc8, 0xfe, 0x
BYTE PTRN_WN10_1607_Channel__ActualProcessEvent[]       = {0x40, 0x57, 0x48, 0x83, 0xec, 0x40, 0x48, 0xc7, 0x44, 0x24, 0x20, 0xfe,
BYTE PTRN_WN10_1709_Channel__ActualProcessEvent[]       = {0x48, 0x89, 0x5c, 0x24, 0x08, 0x57, 0x48, 0x83, 0xec, 0x40, 0x48, 0x8b,
BYTE PTRN_WN10_1803_Channel__ActualProcessEvent[]       = {0x40, 0x57, 0x48, 0x83, 0xec, 0x40, 0x48, 0xc7, 0x44, 0x24, 0x20, 0xfe,
BYTE PTRN_WN10_1809_Channel__ActualProcessEvent[]       = {0x40, 0x57, 0x48, 0x83, 0xec, 0x40, 0x48, 0xc7, 0x44, 0x24, 0x20, 0xfe,
BYTE PTRN_WN10_1909_Channel__ActualProcessEvent[]       = {0x40, 0x57, 0x48, 0x83, 0xec, 0x40, 0x48, 0xc7, 0x44, 0x24, 0x20, 0xfe,
BYTE PTRN_WN10_2004_Channel__ActualProcessEvent[]       = {0x48, 0x89, 0x5c, 0x24, 0x08, 0x48, 0x89, 0x74, 0x24, 0x10, 0x57, 0x48,

BYTE PATC_WNT6_Channel__ActualProcessEvent[]    = {0xc3};
```

It will then write `0xc3` (which is a `ret`, or x64 return instruction) or `0xC20400` (which is `ret 4` for x86) at the start of `Channel::ActualProcessEvent` using an offset and the position of the instructions found in the previous step.

Mimikatz will have now modified `Channel::ActualProcessEvent` to always return before any action is taken place.

This is only an in-memory modification so once the service is restarted or computer is rebooted, the EventLog service will return to normal.

Downgrading Windows Components

The existence of the `MiniNT` registry key will result in various Windows components thinking the environment is WinPE (Preinstallation Environment).

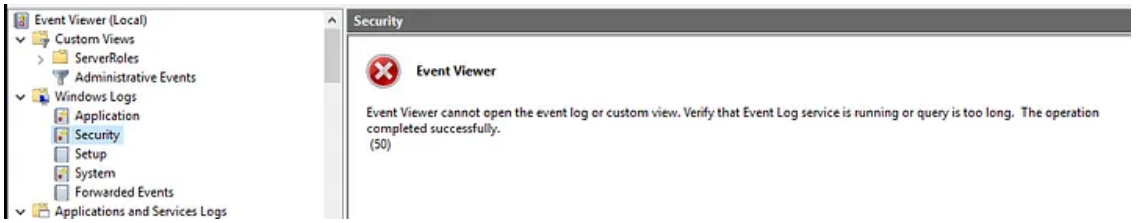
One of these components is the Event Log service! By adding the key below, we can test how the service reacts.

```
reg add “HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\MiniNt”
```

```
C:\Dev>reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\MiniNt"
The operation completed successfully.
```

Once we add the key and restart to load changes in the hive, and then open the event viewer, we are presented with this error for any log:

Press enter or click to view image in full size



If we manually check the file, all events up until the restart remain.

When the key was active, the EventLog Service still ran, but the svchost.exe for it didn't have a file handle on any of the .evtx files.

When I deleted the MiniNT key and restarted the EventLog service (also tried rebooting), all of the events in the period that it was disabled were populated in `security.evtx`. They must be stored somewhere… Something to investigate further!

This method wouldn't hide activity when restarting the service, unlike the other methods but does release the handle access to the file for editing.
