Sign in

This repository has been archived by the owner on Jan 18, 2024. It is now read-only.

hlldz / Phant0m Public archive

Notifications | Fork 297 | Star 1.8k

Code | Issues | Pull requests | Actions | Projects | Security | Insights

master | Go to file | Code

- images
- phant0m
- README.md
- phant0m.cna

README

About

Windows Event Log Killer

windows cpp powershell cobalt-strike eventlog reflective-dll eventlog-service

Readme

Activity

1.8k stars

61 watching

297 forks

Report repository

Releases

No releases published

Packages

No packages published

Languages

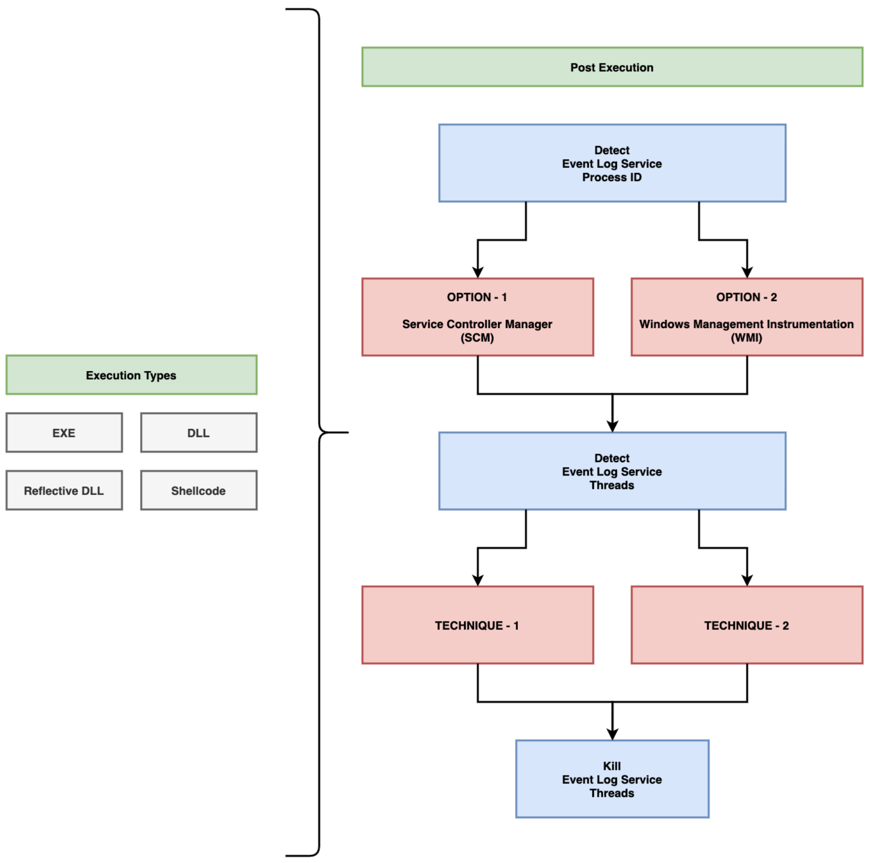
● C 89.3% ● C++ 10.7%

Phant0m | Windows Event Log Killer

Svchost is essential in the implementation of so-called shared service processes, where a number of services can share a process in order to reduce resource consumption. Grouping multiple services into a single process conserves computing resources, and this consideration was of particular concern to NT designers because creating Windows processes takes more time and consumes more memory than in other operating systems, e.g. in the Unix family.[1]

This means briefly that; On Windows operating systems, svchost.exe manages the services and services are actually running under svchost.exe's as threads. Phant0m targets the Event Log service and finding the process responsible for the Event Log service, it detects and kills the threads responsible for the Event Log service. Thus, while the Event Log service appears to be running in the system (because Phant0m didn't kill process), it does not actually run (because Phant0m killed threads) and the system does not collect logs.

How It Works & How To Use


Execution Types
EXE
DLL
Reflective DLL
Shellcode
Post Execution
Detect Event Log Service Process ID
OPTION - 1 Service Controller Manager (SCM)
OPTION - 2 Windows Management Instrumentation (WMI)
Detect Event Log Service Threads
TECHNIQUE - 1
TECHNIQUE - 2
Kill Event Log Service Threads

Detecting Event Log Service

Phant0m uses two different options to detect the Process ID of the Event Log service. The first is to detect via the SCM (Service Control Manager) and the second is to detect via WMI (Windows Management Instrumentation). With which method you want Phant0m to detect the Process ID of the Event Log service, change the following lines in the main.cpp file.

For example, if you want the Process ID to be detected via SCM, you should edit it as follows. (Do not set all values at the same time, set only the one technique you want.)

```
// PID detection techniques configuration secti
#define PID_FROM_SCM 1 // If you set it to 1, tl
#define PID_FROM_WMI 0 // If you set it to 1, tl
```

For example, if you want threads to be killed using Technique-1, you should edit it as follows. (Do not set all values at the same time, set only the one technique you want.)

```
// TID detection and kill techniques configurat
#define KILL_WITH_T1 1 // If you set it to 1, T
#define KILL_WITH_T2 0 // If you set it to 1, T
```

Detecting and Killing Threads

Phant0m uses two different options to detect and kill the threads of the Event Log service.

Technique-1

When each service is registered on a machine running Windows Vista or later, the Service Control Manager (SCM) assigns a unique numeric tag to the service (in ascending order). Then, at service creation time, the tag is assigned to the TEB of the main service thread. This tag will then be propagated to every thread created by the main service thread. For example, if the Foo service thread creates an RPC worker thread (note: RPC worker threads don't use the thread pool mechanism more on that later), that thread will have the Service Tag of the Foo service.[2]

So, in this technique Phant0m will detect threads of Event Log service with NtQueryInformationThread API to get the thread's TEB address and read the SubProcessTag from the TEB. Then it kills the threads related to the Event Log service. The codes for this technique are in `the technique_1.h` file.

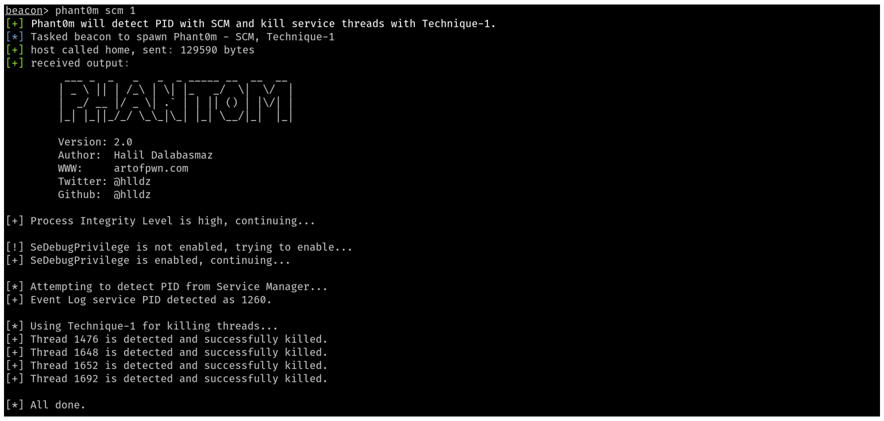
Technique-2

In this technique, Phant0m detects the names of DLLs associated with threads. Windows Event Log Service uses `wevtsvc.dll`. Full path is `%WinDir%\System32\wevtsvc.dll`. If the thread is using that DLL, it is the Windows Event Log

Service's thread and then Phant0m kills the thread. The codes for this technique are in `the technique_2.h` file.

Usage

You can use Phant0m both as a standalone EXE and as a Reflective DLL. Open the project in Microsoft Visual Studio, make the settings (select the detection and kill techniques) and compile. You can also use the Reflective DLL version with Cobalt Strike, for this there is an Aggressor Script file (phant0m.cna) in the repository.



```
beacon> phant0m scm 1
[+] Phant0m will detect PID with SCM and kill service threads with Technique-1.
[*] Tasked beacon to spawn Phant0m - SCM, Technique-1
[+] host called home, sent: 129590 bytes
[+] received output:

        Version: 2.0
        Author:  Halil Dalabasmaz
        WWW:     artofpwn.com
        Twitter: @hlldz
        Github:  @hlldz

[+] Process Integrity Level is high, continuing...

[!] SeDebugPrivilege is not enabled, trying to enable...
[+] SeDebugPrivilege is enabled, continuing...

[*] Attempting to detect PID from Service Manager...
[+] Event Log service PID detected as 1260.

[*] Using Technique-1 for killing threads...
[+] Thread 1476 is detected and successfully killed.
[+] Thread 1648 is detected and successfully killed.
[+] Thread 1652 is detected and successfully killed.
[+] Thread 1692 is detected and successfully killed.

[*] All done.
```

Fork and inject method was used with `bdllspawn` in the execution type of Aggressor Script (phant0m.cna) for Cobalt Strike. If you want to inject Phant0m into your existing process and run it, you can review this project (https://github.com/rxwx/cs-rdll-ipc-example) and you can do it easily. You can also convert the code to DLL and then to Shellcode with Donut.

NOTE: The project only supports x64 architecture.

Special Thanks to Those Who Mentioned Phant0m

- Detecting in-memory attacks with Sysmon and Azure Security Center - https://azure.microsoft.com/tr-

tr/blog/detecting-in-memory-attacks-with-sysmon-and-azure-security-center/

- Experiments with Invoke-Phant0m - http://www.insomniacsecurity.com/2017/08/27/phant0m.html
- Event Log Tampering Part 1: Disrupting the EventLog Service - https://medium.com/@7a616368/event-log-tampering-part-1-disrupting-the-eventlog-service-8d4b7d67335c
- Flying under the radar - https://www.exploit-db.com/docs/english/45898-flying-under-the-radar.pdf?rss
- Denetim ve Log'lamanın Elli Tonu - https://gallery.technet.microsoft.com/Denetim-ve-Loglamann-Elli-cbed0000
- Disabling Windows Event Logs by Suspending EventLog Service Threads - https://www.ired.team/offensive-security/defense-evasion/disabling-windows-event-logs-by-suspending-eventlog-service-threads
- Event Log Service – Between Offensive And Defensive - https://blog.cybercastle.io/event-log-service-between-offensive-and-defensive/