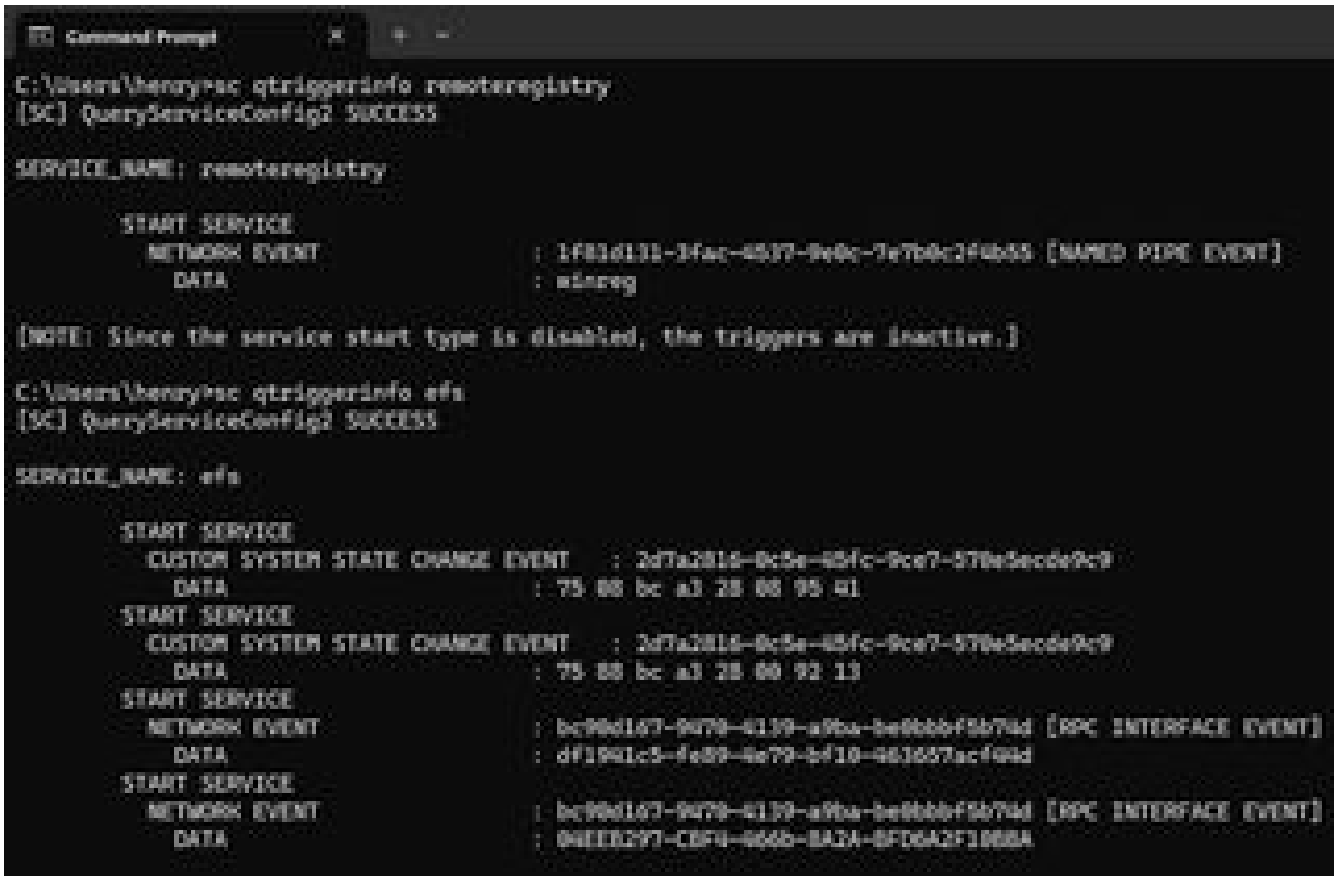
There's More than One Way to Trigger a Windows Service

trustedsec.com/blog/theres-more-than-one-way-to-trigger-a-windows-service

October 16, 2025



There are several components and technologies on Windows that are easy to overlook. I believe service triggers are one of those. I've observed individuals discovering service triggers multiple times over the last few years. To that end, I thought it may be beneficial to write a short blog on the different types of service triggers that exist and how they can be activated with little to no code required.

A service trigger is a configuration option for a Windows service that allows that service to be started when some other activity or condition is met. Some of these conditions are largely static, for example the computer being domain joined or having an IP address allocated. Other conditions are more dynamic in nature, for instance receiving a connection request for a given named pipe or processing a specific trace message.

What makes service triggers interesting is several of them allow a low-privilege user to start a service they might not otherwise have permissions to start. When considering built-in components, Remote Registry, WebClient, and Encrypting File System (EFS), all are services that a pentester may have interest in starting but may not have the appropriate permissions to do so.

1 Listing Service Triggers

There are a variety of tools that allow setting and listing Windows service triggers. Let's walk through those options now.

1.1 Native Utilities

A preinstalled utility, ***sc.exe***, is for interacting with the service control manager (SCM). It's been around for a very long time and many applications that manipulate services through command line calls leverage it.

It can show triggers on a given service by using the subcommand `qtriggerinfo`

Figure 1 - qc qtriggerinfo Output

Like many configuration settings on Windows, service triggers are housed in the registry. We can query the registry for these values using `reg query`, targeting the service subkey

TriggerInfo recursively.

Figure 2 - Registry Listing of EFS Service Triggers

1.2 Win32 API

Service configurations including trigger information are exposed via the Win32 API. To retrieve the service triggers using the Windows API a developer needs to call ***QueryServiceConfig2***. In this call, SERVICE_CONFIG_TRIGGER_INFO (8) should be passed as the second parameter. An example can be found in our Beacon Object File (BOF) repo here.

Figure 3 - sc_qtriggerinfo BOF

This output isn't the best, when comparing it to ***sc.exe***, namely because the data element is missing and the undocumented Type of 0x7 isn't named. If anyone reading this is interested in contributing to our repo and adding the missing information I would love to review and accept such a pull request!

1.3 MS-SCMR

It is possible to interact remotely with the SCM. Under the hood this is largely what the Windows API is doing, just in a more developer-friendly manner. For service trigger specifically, [MS-SCMR 3.1.4.48] outlines the required raw RPC message used for listing.

Currently, ***Impacket*** has the backing structures defined but does not have an example that utilizes them.

TrustedSec has also released our own framework for initiating remote RPC communications named Titanis. Sample output of `Scm.exe qtriggers` is below.

Figure 4 Titanis Service Trigger Listing

2 Types of Service Triggers

Before we can talk about how a given service trigger can be abused, we need to understand what triggers exist in the first place. Below we review each of the existing service trigger types as of the creation of this post. Most of the triggers we discuss are documented here.

2.1 Device Interface Arrival

This type of trigger starts a service when a new device of the matching class and hardware ID is attached. You are not required to specify a data item when defining a trigger of this type. If a data item is not provided the trigger will fire when any device matching the class provided in ***pTriggerSubtype*** is connected. Devices matching the specified type are checked on startup and will fire triggers at that time. Triggers will also fire if a device is connected after the computer has started if it matches the given type and hardware ID.

2.2 Domain Join

This service trigger is documented in a rather confusing manner. Microsoft Developer Network (MSDN) states the event is triggered when the computer (leaves || joins) a domain. This may or may not be correct but very likely is not the intended use case for this trigger. It triggers startup based on if the computer is or isn't domain joined. Setting ***DOMAIN_JOIN_GUID*** as the subtype will trigger the service to start at boot if the computer is domain joined.

DOMAIN_LEAVE_GUID will only trigger the service to start at boot if the computer is not domain joined.

2.3 Firewall Port Event

As far as I was able to determine, this trigger does not work as advertised.

If I add a service trigger for port 9999/tcp to a service, that service is started when ANY change is made to the Windows Firewall. This could be disabling an existing firewall rule, deleting a firewall rule, adding a new firewall rule, etc. The rules I created could be TCP/UDP and refer to any port, not just those specified for the filtering logic.

Further, and perhaps more interestingly, if the rule was misconfigured by providing a port and not a protocol, it would prevent the Windows Firewall from properly working. This appears to be due to the Base Filtering Engine (BFE) service failing to start on subsequent boots when it encounters this invalid data. This causes every service depending on BFE to also fail startup.

Figure 5 - Some Services Depending on BFE

In this state, all ports will remain filtered for inbound connections and the firewall itself cannot be configured. The error below occurred when attempting to reenable the firewall through the Settings menu.

Figure 6 - Error Reenabling Firewall

This bug was reported to Microsoft on September 04, 2025, and they deemed it to not be a security vulnerability.

2.4 Group Policy

This service trigger runs when the named subtype (***Machine*** or ***User***) Group Policy is updated. This service trigger doesn't run at boot on a non-domain-joined machine. The service trigger will run at boot on a domain-joined machine if the corresponding machine or user Group Policy is configured. It won't trigger when `gpupdate` is run if no changes are present. It will run using `gpupdate /force` regardless of if there is a change or not, assuming there exists a policy of the given type.

2.5 IP Address Available

This trigger type does exactly what it says it will. It will initiate its trigger when the first IP address is obtained on a network adapter or when the last one is lost. This does mean in most normal environments the service will trigger on boot. There doesn't appear to be any meaningful way of linking the trigger with a specific interface.

2.6 Network Endpoint

This is the trigger type that I believe most individuals have the largest interest in. It is broken into two categories: named pipes and RPC endpoints. For named pipes, the given service is started when an application attempts to connect to the given named pipe. Notably, the service does not need to be listening at that pipe. For RPC endpoints, the given service is started when the endpoint mapper is asked where the specified UUID is located. This gives the

service a chance to start up, and register with the endpoint mapper and the client application to be able to connect to it, without having already needed to have the service running.

2.7 System State Change - WNF

This provider gets a little trickier to test as it isn't formally documented. It relates to messages published using Windows Notification Facility (WNF). I've done some casual experimentation with this trigger but not enough that I want to make declarative statements about how this trigger works. I largely want to point out that it exists. If you'd like to learn more about WNF, Unknown Known DLLs is one of the most comprehensive reviews I've found so far.

2.8 Custom - ETW

Event Tracing for Windows (ETW) is a component of Windows that provides a framework for registering, raising, and consuming events from a variety of providers. An application can register as a provider of ETW messages and generate events as that provider when enabled. Other applications can attempt to register a tracing session and enable the previously registered provider. They are then able to view the messages being generated.

WebClient is a popular service to be started using an ETW message. Let's locate that specific trigger.
First, we can list it using `sc.exe qtriggerinfo webclient`:

Figure 7 - sc.exe qtriggerinfo webclient Output

This tells us the provider ID, which will raise this message. To see if this provider is currently registered/discoverable, we can run a command such as `logman query providers | findstr /I 22b6d684-fa63-4578-87c9-effcbe6643c7`.

This command will output:

```
Microsoft-Windows-WebdavClient-LookupServiceTrigger {22B6D684-FA63-4578-87C9-EFFCBE664
```

That seems to be a fitting provider name for the trigger we're looking at.

If any additional filtering was done on this message, we would have observed it in our `sc.exe qtriggerinfo` output. In this case, any message raised by this provider should start the interface in question. Binary, string, keyword, and level filtering are all viable on ETW triggers and control what a matching provider-raised message would need to contain to start the service.

2.9 Aggregate Service Triggers

There is another undocumented service trigger type named Aggregate. An example service using this trigger type on Windows 11 is CDPSvc. Listing it with `sc.exe qtriggerinfo` gives rather confusing results, presumably because ***sc.exe*** is not coded to handle this trigger type.

Figure 8 - sc.exe qtriggerinfo CDPSvc Output

Triggers of this type refer to a GUID:

Figure 9 - CDPSvc Trigger via Registry View

If properly registered, this GUID should have a corresponding entry under ***HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\ServiceAggregatedEvents***:

Figure 10 - ServiceAggregatedEvents Subkey

The Trigger value has data that is equal to a GUID. This GUID is found to be a subkey and has the following values:

Figure 11 - Further Related Registry Subkey

Searching for the (Default) value yields a few results at the following locations:

- ***Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\NetworkServiceTriggers\Triggers\bc90d167-9470-4139-a9ba-be0bbbf5b74d\D09BDEB5-6171-4A34-BFE2-06FA82652568:fdd099c6-df06-4904-83b4-a87a27903c70***
- ***Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\BackgroundModel\Brokers\{fdd099c6-df06-4904-83b4-a87a27903c70}***
- ***Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows NT\CurrentVersion\BackgroundModel\Brokers\{fdd099c6-df06-4904-83b4-a87a27903c70}***

I do not fully understand how this trigger would work though, and this likely would be an area for future research.

3 Activating Service Triggers

For this section we'll talk through how we can activate each service trigger, excluding WNF and Aggregate.

3.1 Device Interface Arrival

I was unable to find a method to trigger interfaces of this type without plugging in something like a USB drive. At most, I could see setting a subtype that you expect to always have connected (such as GUID_DEVINTERFACE_KEYBOARD - Windows drivers | Microsoft Learn). At that point you would have a service that appeared to start on demand and would essentially always start.

3.2 Domain Join

Like above, I would see this being used to auto start a service without changing its start type. Got a service implanted on a domain-joined box or want to change an existing service to always start? Add this trigger type.

3.3 Firewall Port Event

Given findings that filtering for this rule did not work properly, it could be used like above. As Windows starts and the firewall configuration is applied, this rule seems to trigger, leading to another way of auto starting a service on boot.

3.4 Group Policy

This specific method would not be useful for automatically starting on system startup. For security testing I can't think of a logical reason to install a service using this trigger type. If you happen to run across a service that you want to start that uses this trigger type you can trigger it as a low-privilege user by executing `gpupdate /force`.

3.5 IP Address Available

I could see using this trigger type if you had an implant that only ran while there was an active IP address available. It is important to note that this trigger is looking for an IP address, not the Internet. It's entirely possible that this trigger could be configured and a non-Internet connected network is what shows up as the available IP address.

3.6 Network Endpoint

These triggers are very interesting to us as pentesters, largely because they should let us start a service remotely and likely as a low-privileged user.

Named pipe triggers are started when a program attempts to speak to that pipe. We can perform this triggering from something as simple as a ***cmd. exe*** session. First, ensure the service in question isn't disabled. Triggers won't work on a disabled service. This can be checked with a command such as:

Figure 12 - sc.exe qc remoteregistry Output

In the above instance we see the START_TYPE is set to ***4*** (DISABLED) so normally this wouldn't work. I've changed it to ***3*** (DEMAND) to continue this example.

We need to know what endpoint we are hitting, which we can find from `sc.exe qtriggerinfo`. In this case we see it's a named pipe event for ***winreg***.

Figure 13 - sc.exe qtriggerinfo remoteregistry Output

Finally, we can trigger this service to start using a basic PowerShell `ls` alias.

Figure 14 - Starting remoteregistry via Named Pipe

RPC endpoints are similar but a little bit harder to trigger. Let’s look at ClipSVC as an example.

Figure 15 - Review of ClipSVC State

This service is currently stopped, demand start, and should start when the endpoint mapper is asked about interface 64D1D045-F675-460B-8A94-570246B36DAB. We can try to start this service using `rpcping`.

Figure 16 - Starting ClipSVC via rpcping

I would like to point out that for both examples above, the connection itself did not need to meaningfully complete. It was the act of attempting to initiate a connection to the given named pipe or asking the endpoint mapper where the given interface was that caused these services to start.

3.7 Custom - ETW

Locally triggering a service with an ETW trigger shouldn't cause too much heartache. Let's look at a service that isn't in every other example, ***edgeupdate***. To trigger this service, we will need to write a bit of code. We can walk it all the way back to 2015 and check out Tyranid's Lair: Starting WebClient Service Programmatically for a basic code outline. Flip out the GUID for the one in question (5ef81e80-ca64-475b-b469-485dbc993fe2) and BOOM!…***edgeupdate*** service starts. Note that if you try this, it does immediately stop, so a query afterwards will likely show the service stopped. If you would like more information on the complete programmatic flow of the event Will WebClient Start - SpecterOps provides a nice outline of how the message gets routed and potential failure points for getting to the consumer that would start the service based on this trigger.

4 Closing

We reviewed Windows service triggers and how to utilize low/no code solutions to activate those triggers. Hopefully some of you testers out there found this informative and can

potentially use it as a jumping off point for future research. Until next time, happy hacking, and triggering!