Labs - Threat Hunting & Hunting With Elastic

Skills Assessment

Hunting For Stuxbot (Round 2)

Scenario

- Recently uncovered details shed light on the operational strategy of Stuxbot's newest iteration.
  1. The newest iterations of Stuxbot are exploiting the C:\Users\Public directory as a conduit for deploying supplementary utilities.
  2. The newest iterations of Stuxbot are utilizing registry run keys as a mechanism to ensure their sustained presence within the infected system.
  3. The newest iterations of Stuxbot are utilizing PowerShell Remoting for lateral movement within the network and to gain access to domain controllers.

The Tasks

`Hunt 1`: Create a KQL query to hunt for "Lateral Tool Transfer" to C:\Users\Public. Enter the content of the `user.name` field in the document that is related to a transferred tool that starts with "r" as your answer.

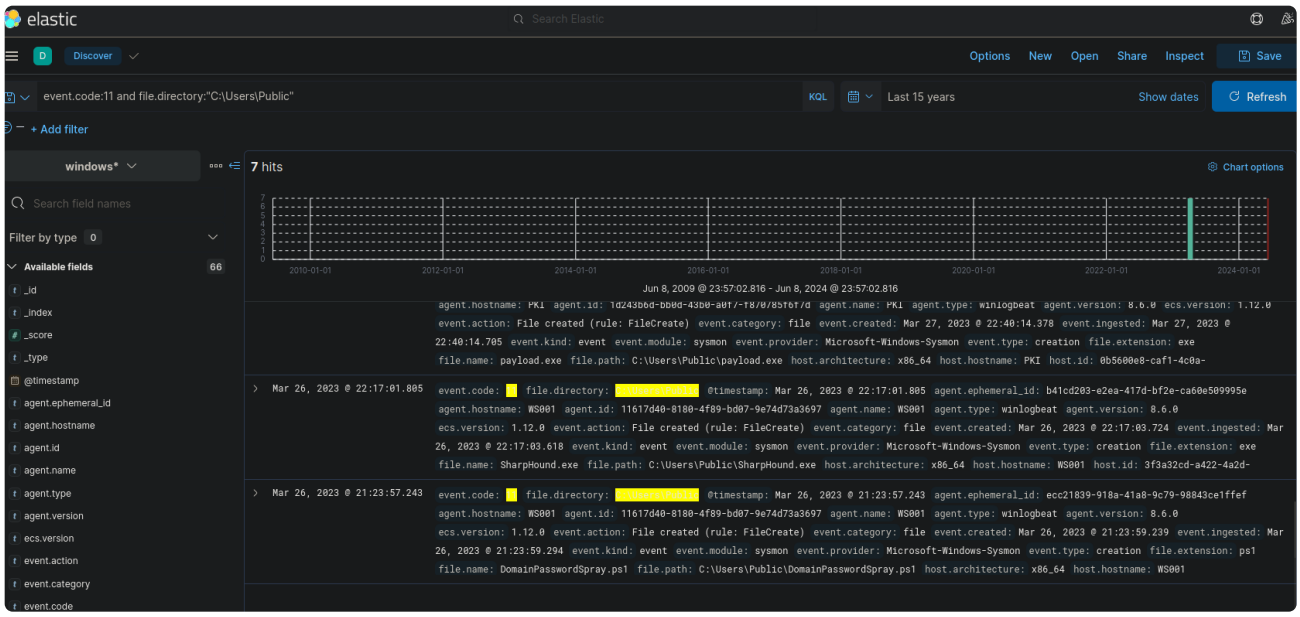
`Hunt 2`: Create a KQL query to hunt for "Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder". Enter the content of the `registry.value` field in the document that is related to the first registry-based persistence action as your answer.

`Hunt 3`: Create a KQL query to hunt for "PowerShell Remoting for Lateral Movement". Enter the content of the `winlog.user.name` field in the document that is related to PowerShell remoting-based lateral movement towards DC1.

Question

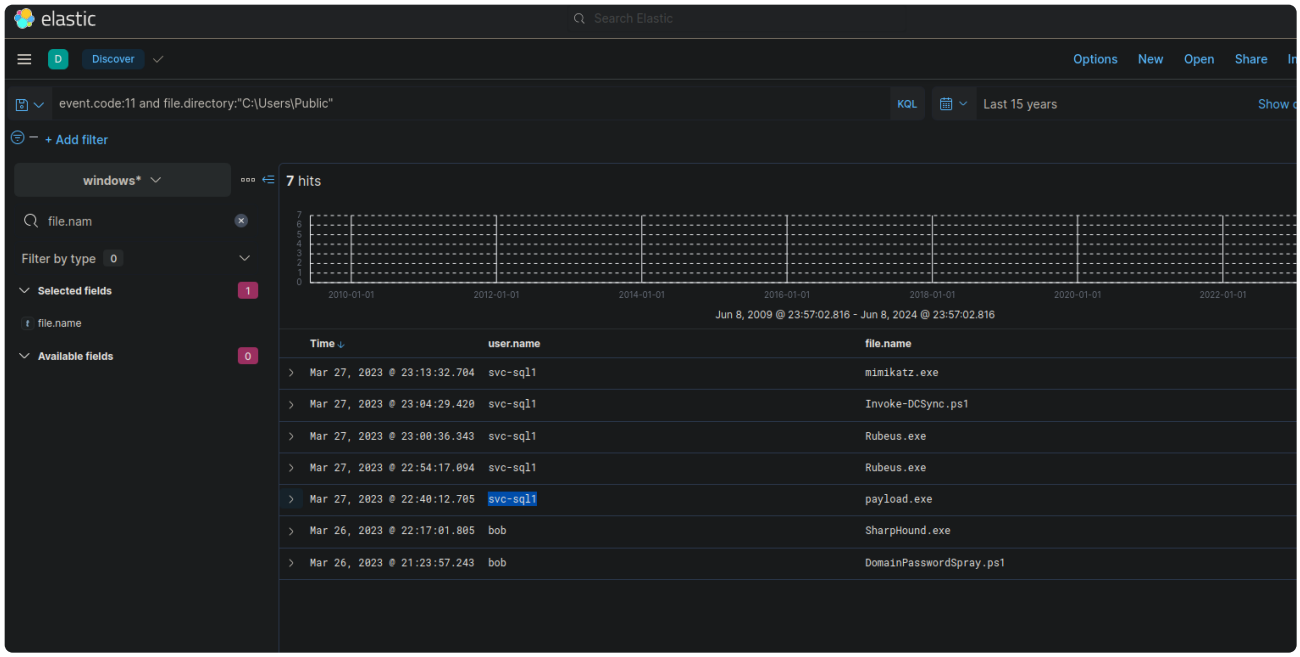
- Enter your answer for Hunt 1.
  -> We know that the tool used for lateral movement is loaded on the directory C:\Users\Public. This means it creates a file on that directory (event.code=11 for sysmon logs)

```
event.code:11 and file.directory:"C:\Users\Public"
```



-> We see alot of tools loaded on to this directory for lateral movement, with the first one being the popular domain spray tool that we as penetration tester always uses in an active directory environment.

-> Adding some filters in the log, we see that:

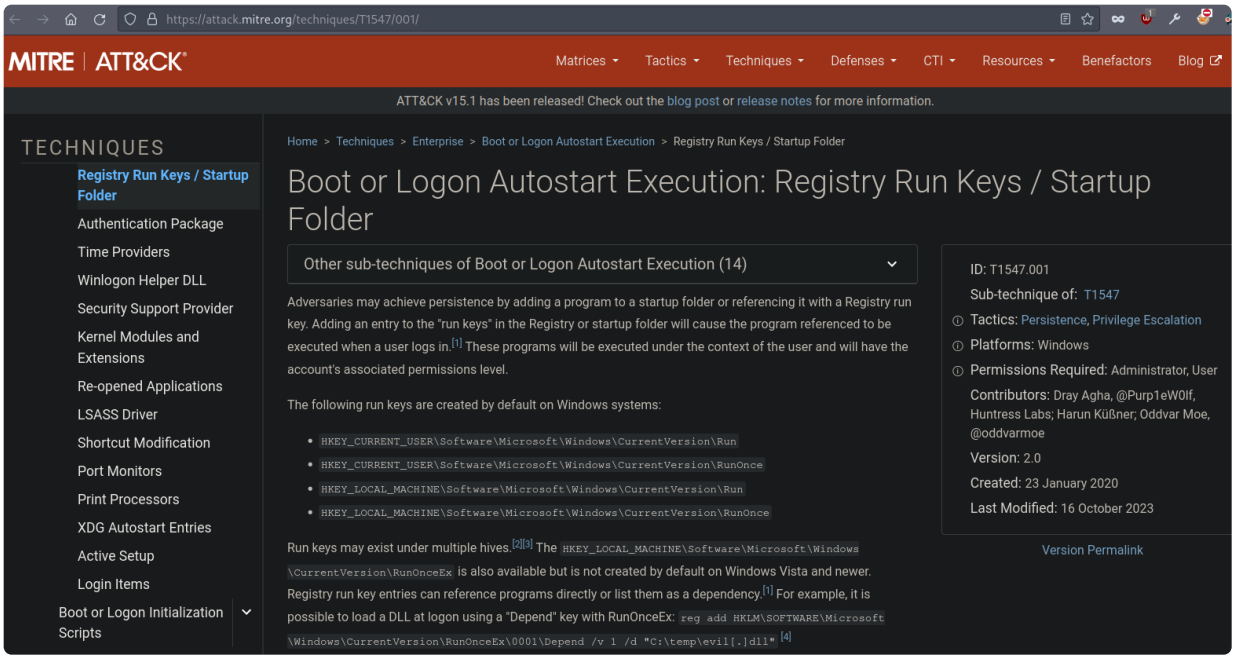


-> We see some impactful tools being loaded, with Rubeus being able to perform various Kerberoast attack.

-> We also see the user svc-sql1, an service account, maybe this shows they have compromised the sql server?

- Enter your answer for Hunt 2.

  -> We first consult the related documentation.

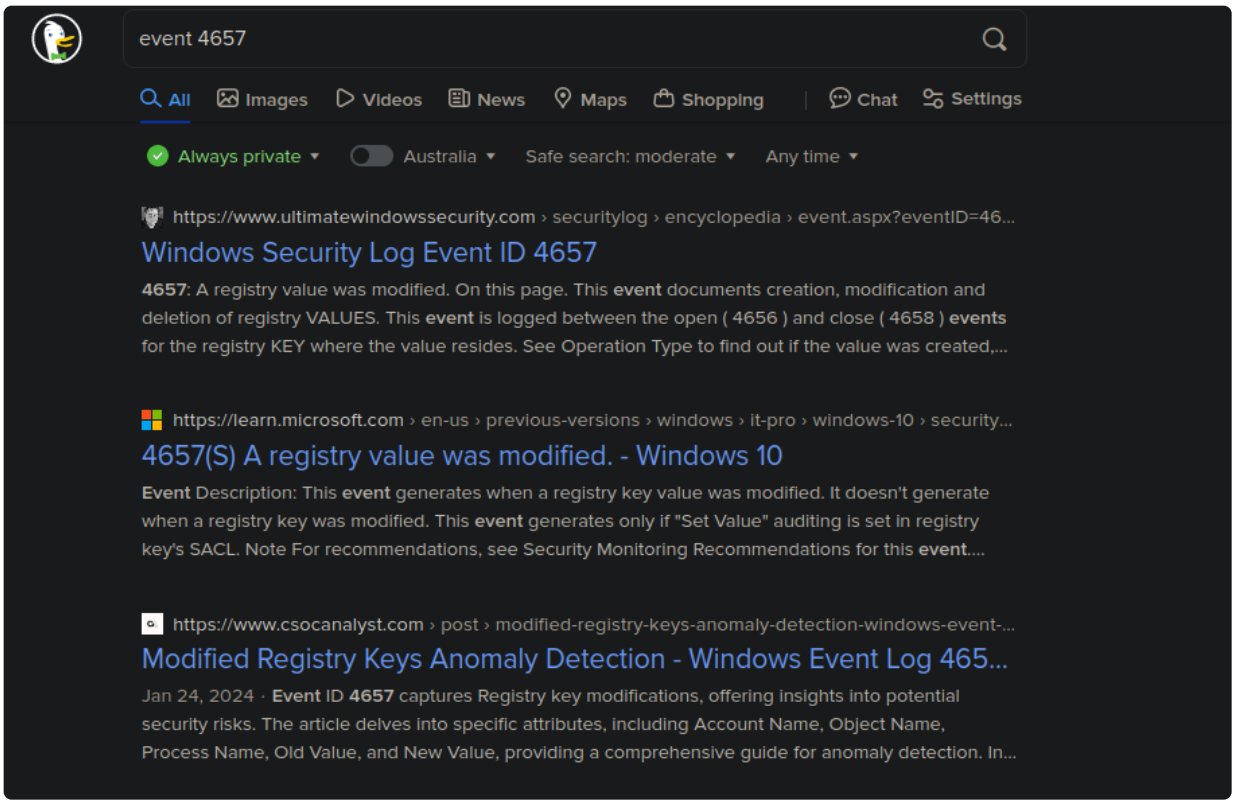


  -> There a few key points to be noted:

  -> There are default and non-default keys in Windows system that executes when a user login, where they have the common pattern of "*CurrentVersion\Run*" *in regex term.*

  *-> We can see from an example that to perform this persistence attack, you generally need to add an register value under some locations of "*CurrentVersion\Run*"*, with the give away of the malicious application being the value of the string.

  -> Hence, We know that we need to look for an sysmon event of 13 or windows every code of 4657.



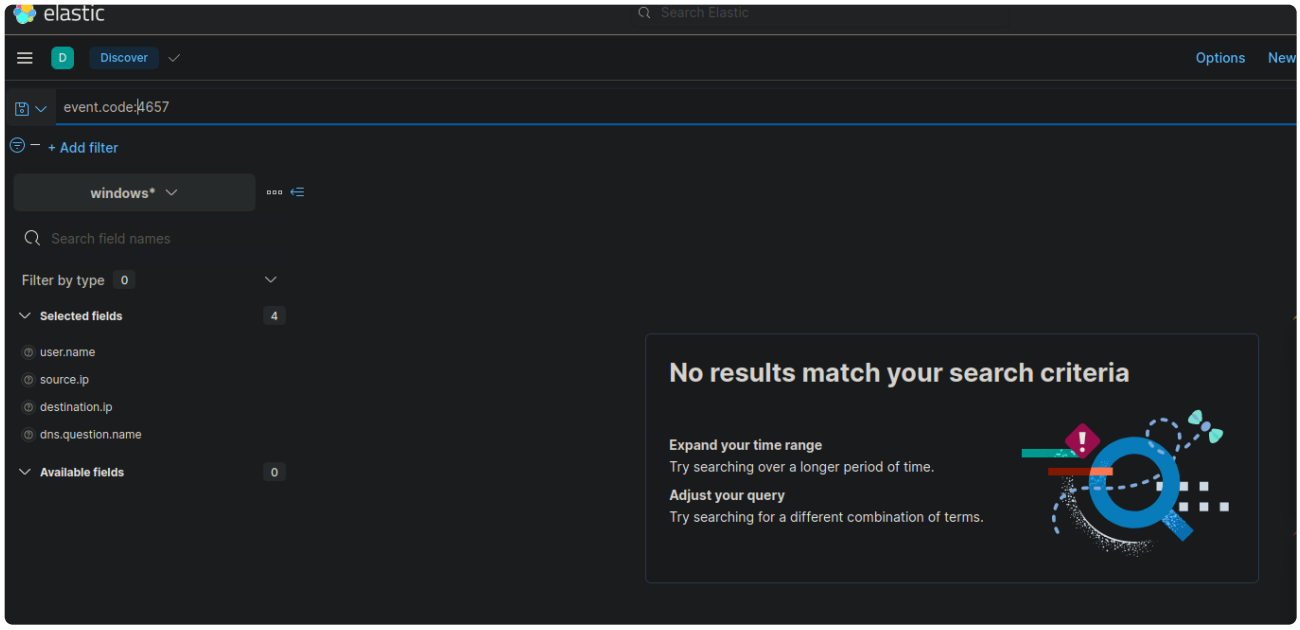
Event ID 13: RegistryEvent (Value Set)

This Registry event type identifies Registry value modifications. The event records the value written for Registry values of type `DWORD` and `QWORD`.

- Hence, our full kql query would be the following:

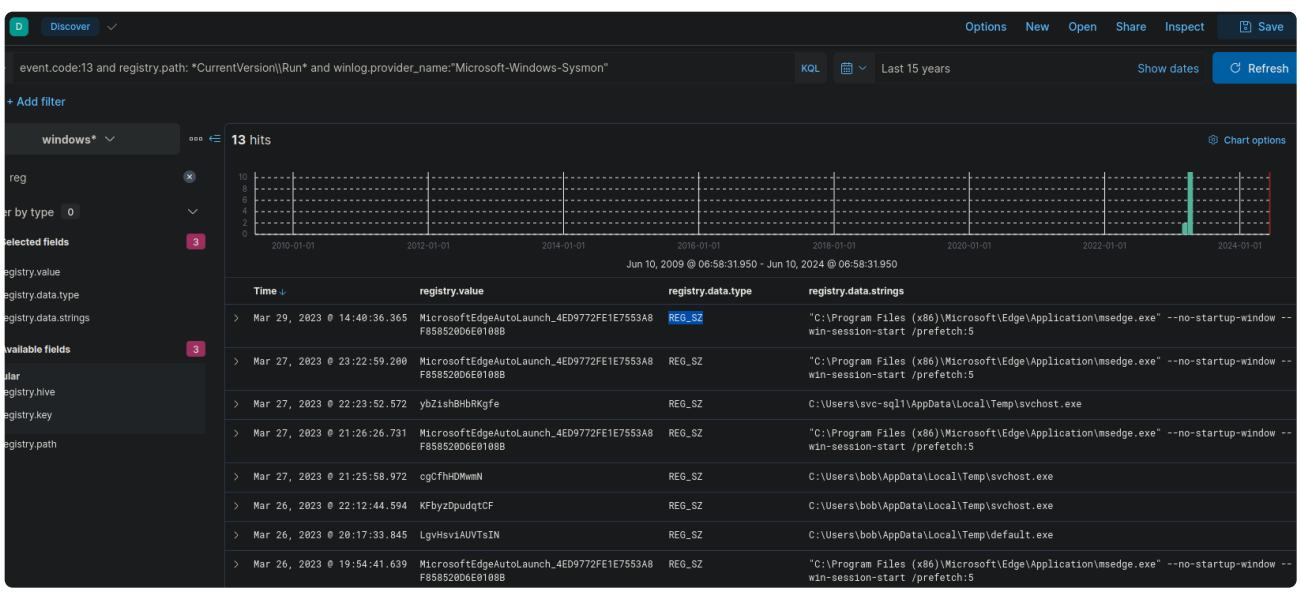
```
(event.code:13 and registry.path: *CurrentVersion\\Run* and
winlog.provider_name:"Microsoft-Windows-Sysmon") or (registry.path:
*CurrentVersion\\Run* and event.code:4657 and
winlog.provider_name:"Microsoft-Windows-Security-Auditing")
```

-> Testing the query bit of bit, we see that



-> There are no windows security logs related to event code 4657, so we only have to look at the sysmon log:

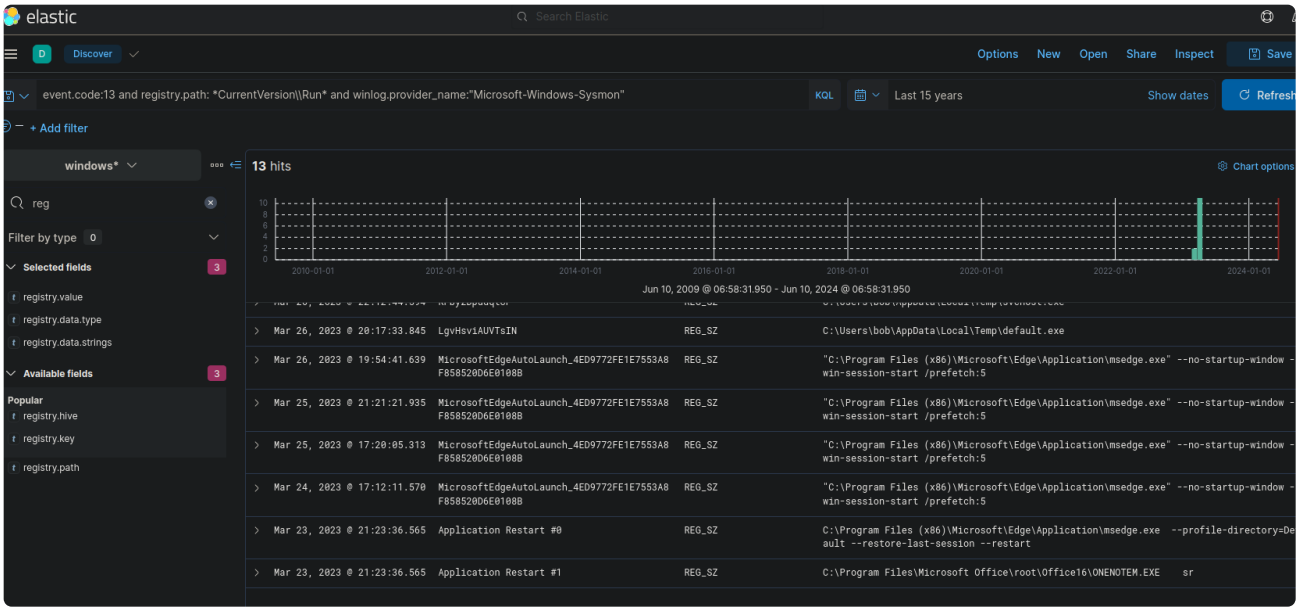
```
event.code:13 and registry.path: *CurrentVersion\\Run* and
winlog.provider_name:"Microsoft-Windows-Sysmon"
```



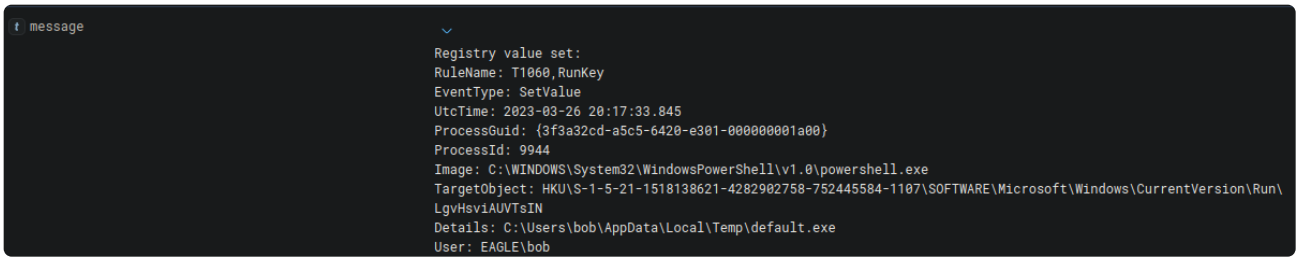
-> Consulting the documentation for `Reg_SZ`, we see that it refers to an null-terminated string:

`REG_SZ`	A null-terminated string. It's either a Unicode or an ANSI string, depending on whether you use the Unicode or ANSI functions.

-> Scrolling back down the log, we see an suspicious value for the value name LgvHsviAUVTsIN



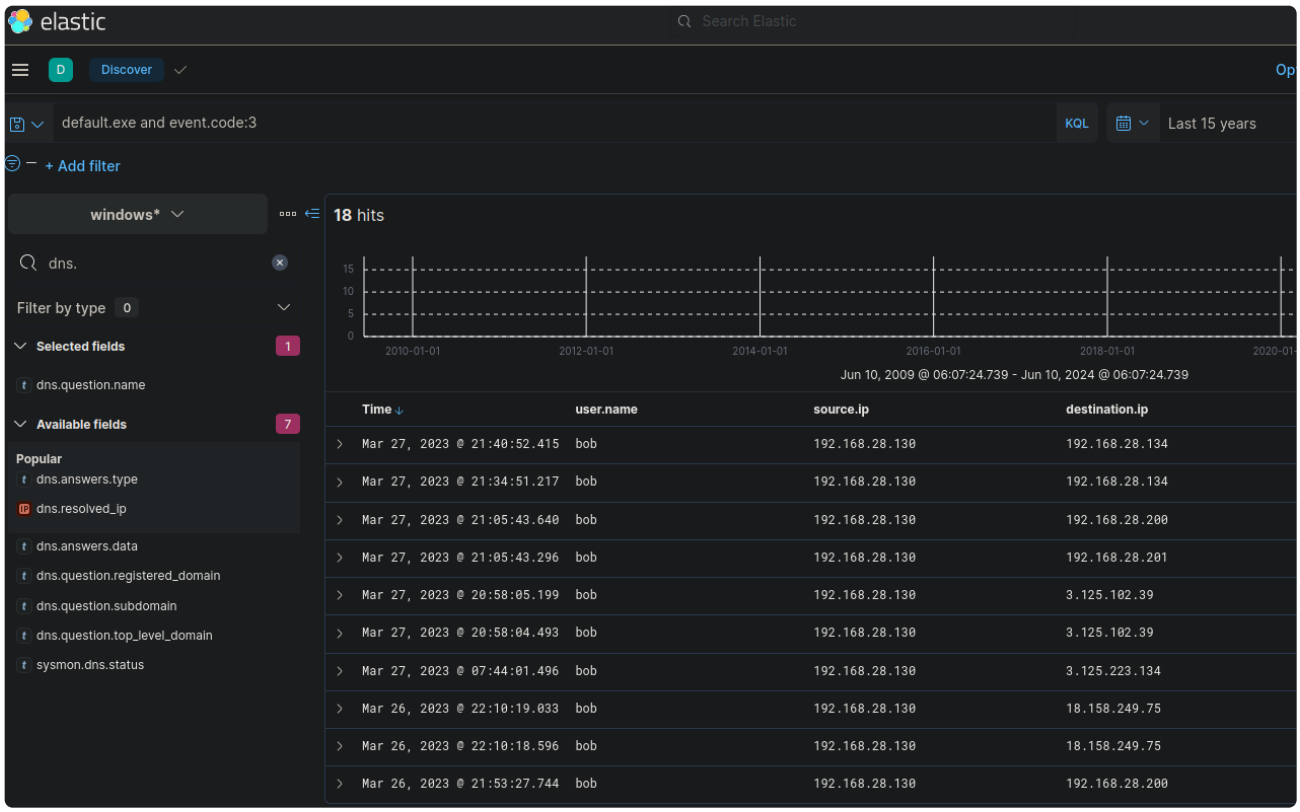
-> Looking at the message, we see that this is just like the example we see in Mitre and Attack Framework:



-> We see that an it is powershell that executed this command, and adds default.exe to the "run key", certainly an unusual behaviour.

-> To verify that default.exe is used for persistence and is indeed malicious, we search examine whether it performs any network connection:

```
default.exe and event.code:3
```

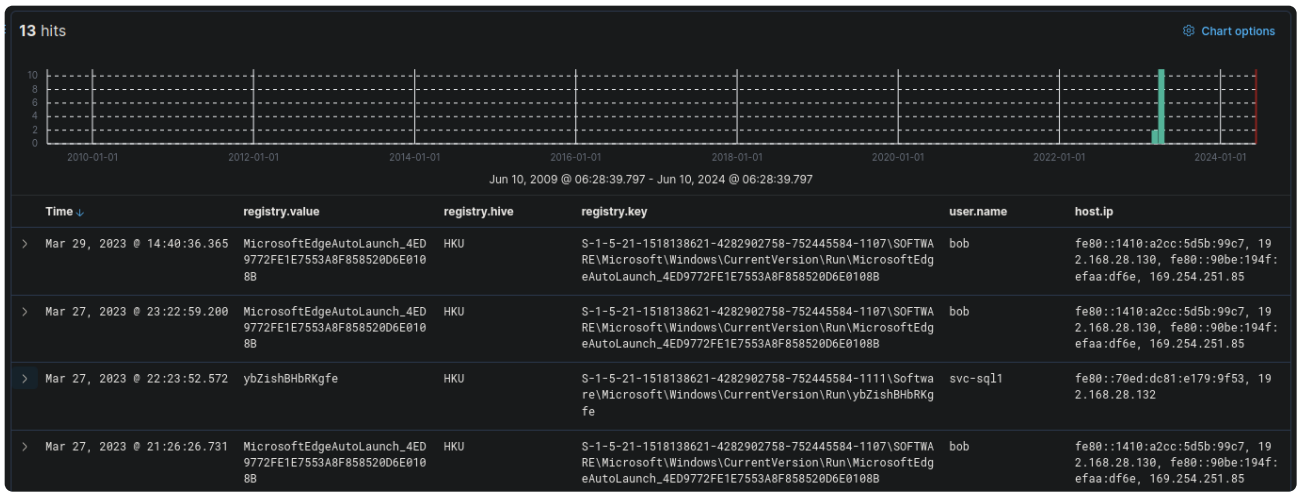


-> Indeed, default.exe is making connection to various IP's. We can see it makes connection to both external (3.125.223.134, 18.192.31.165) and internal IP addresses (192.168.28.134).

-> Something that needed to be started up at run time, with the fact that it is added manually, as well as connecting various internal and external IP, this is very likely to be an persistence C2.

-> Hence, the answer for initial registry.value for persistence is LgvHsviAUVTsIN.

-> Furthermore, looking at the logs for the user.name and hostname, we see that is has also been placed on other computers with under the svc-sql1 user:



-> As an extra side note, the extra registry value looks similar to random character names, like those generated from running psexec or meterpreter exploit, which can also raise our alert when we see values like those.

- Enter your answer for Hunt 3.
  -> Given that we have to hunt for powershell remote techniques for lateral movement, it would be ideal to think of what commands attacker typical utilise.
  -> We consult an red team notes and we see that attackers typically utilise `Enter-PSSession` command.

  ## WinRM for Lateral Movement

  PowerShell remoting for lateral movement.

  ### Execution

  Attacker establishing a PSRemoting session from a compromised system `10.0.0.2` to a domain controller `dc-mantvydas` at `10.0.0.6`:

  attacker@10.0.0.2

  ```
  New-PSSession -ComputerName dc-mantvydas -Credential (Get-Credential)

    Id Name            ComputerName    ComputerType    State         ConfigurationName
   -- ----            ------------    ------------    -----         -----------------
    1 Session1        dc-mantvydas    RemoteMachine   Opened        Microsoft.PowerShell

  PS C:\Users\mantvydas> Enter-PSSession 1
  [dc-mantvydas]: PS C:\Users\spotless\Documents> calc.exe
  ```

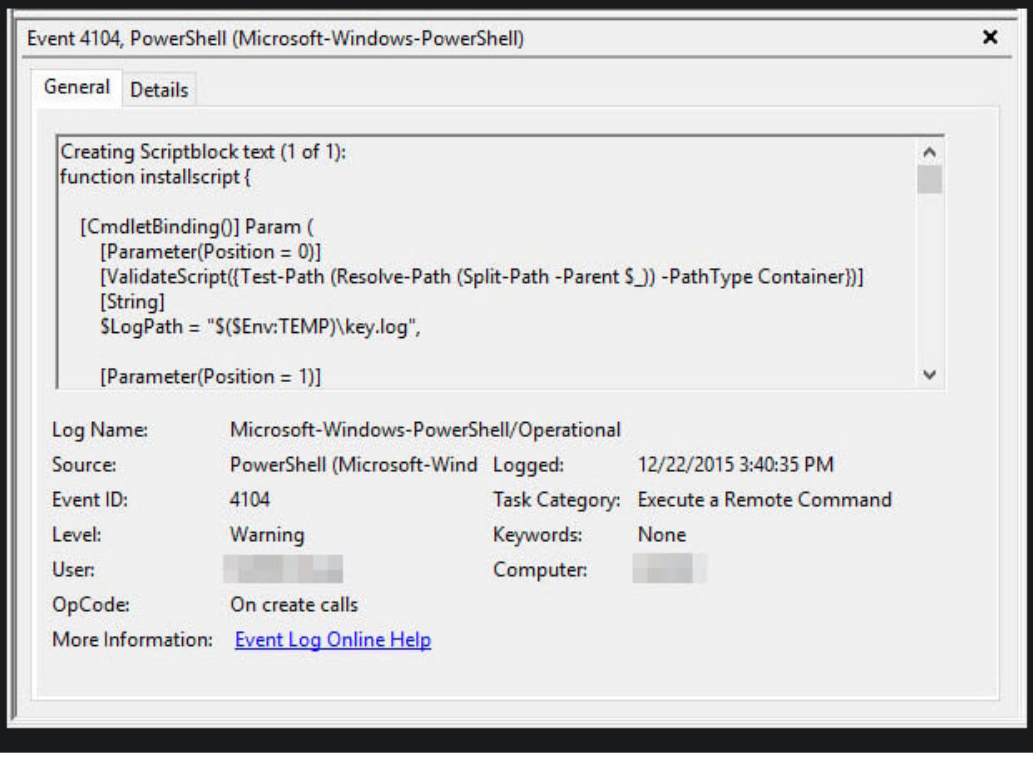
  -> Alternatively, common techniques such as evil win-rm also exists for lateral movement, which we can keep in our back of the mind.

-> Also, we know that when comands such as `Enter-PSSession` is executed for the first time, it creates an event 4104, as it is executing an function for the first time, from the Powershell log source provider:

CROWDSTRIKE | BLOG

Featured

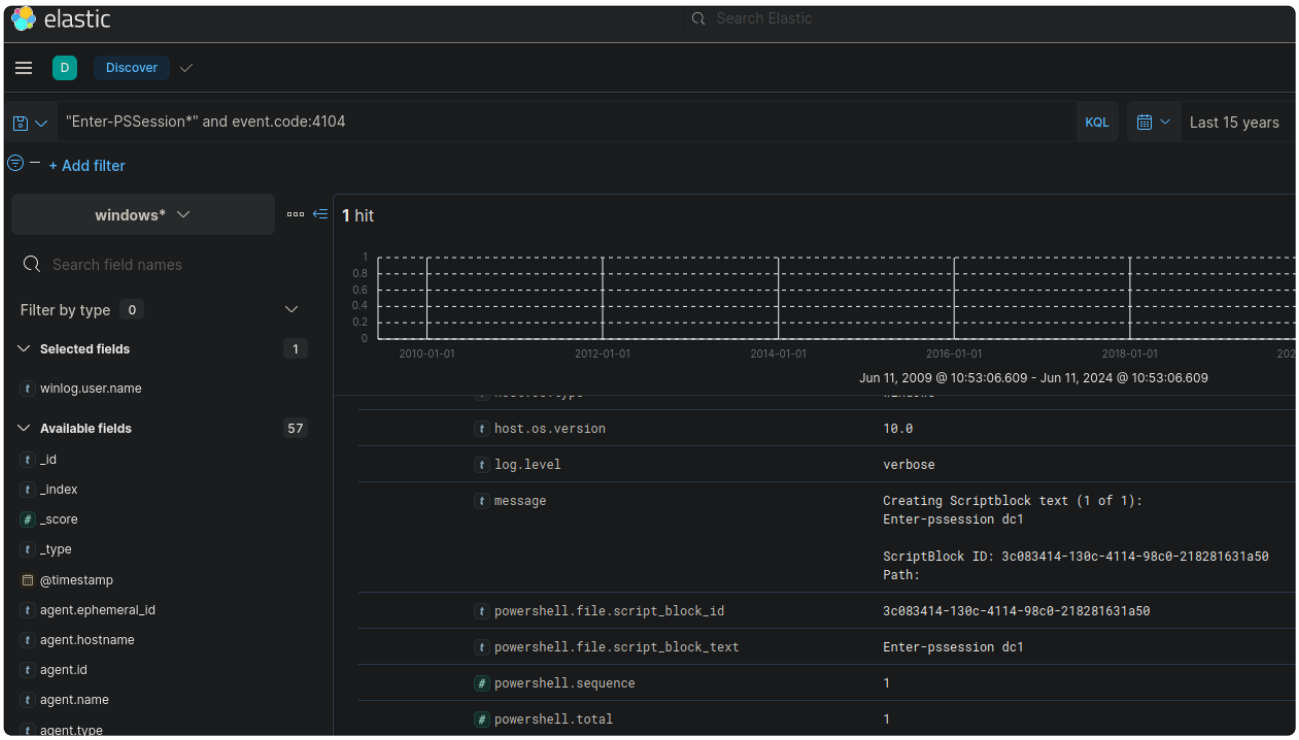
A script block can be thought of as a collection of code that accomplishes a task. Script blocks can be as simple as a function or as full-featured as a script calling multiple cmdlets. Script block auditing captures the full command or contents of the script, who executed it, and when it occurred. Audits are recorded as event log entries in the Microsoft-Windows-PowerShell/Operational log regardless of how PowerShell was executed – from a command shell, the integrated scripting environment (ISE), or via custom hosting of PowerShell components. Event ID 4104 records the script block contents, but only the first time it is executed in an attempt to reduce log volume (see Figure 2).



-> Hence, our kql query would look like the following, incoporating event id of 4104 and looks for command executing the command Enter-PSSession

```
"Enter-PSSession*" and event.code:4104
```

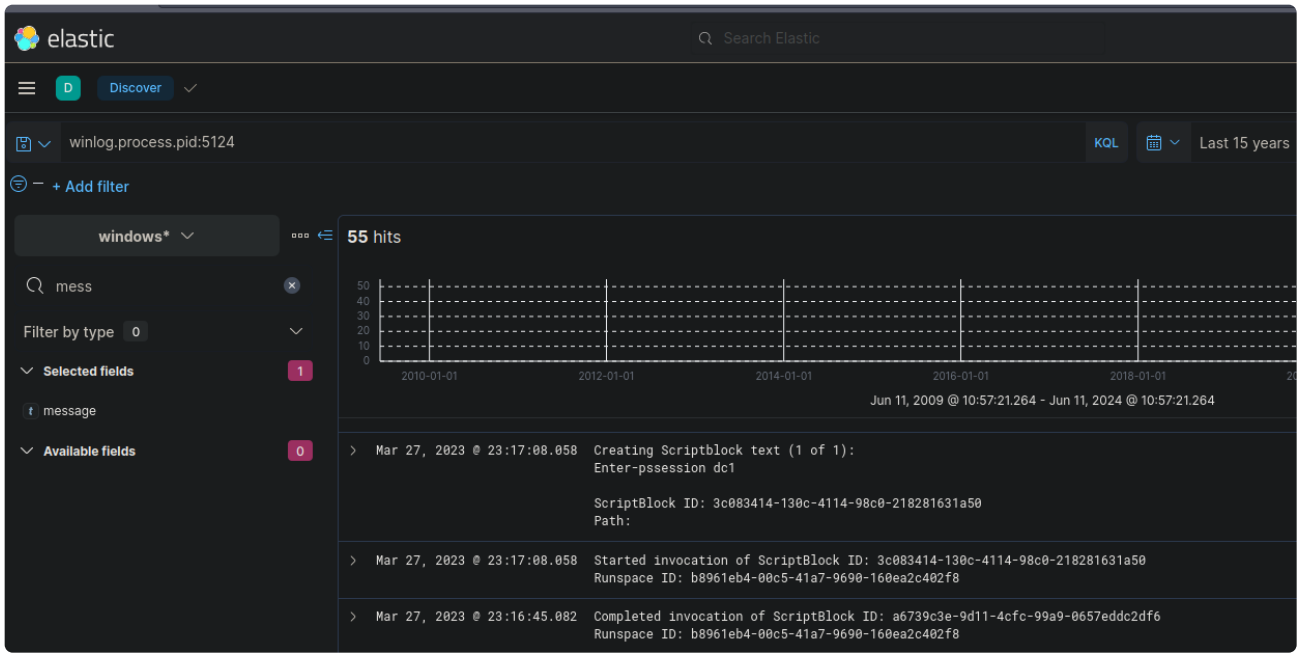
-> We see we have one result, which is very suspicious as service accounts would never perform an power-shell remoting.

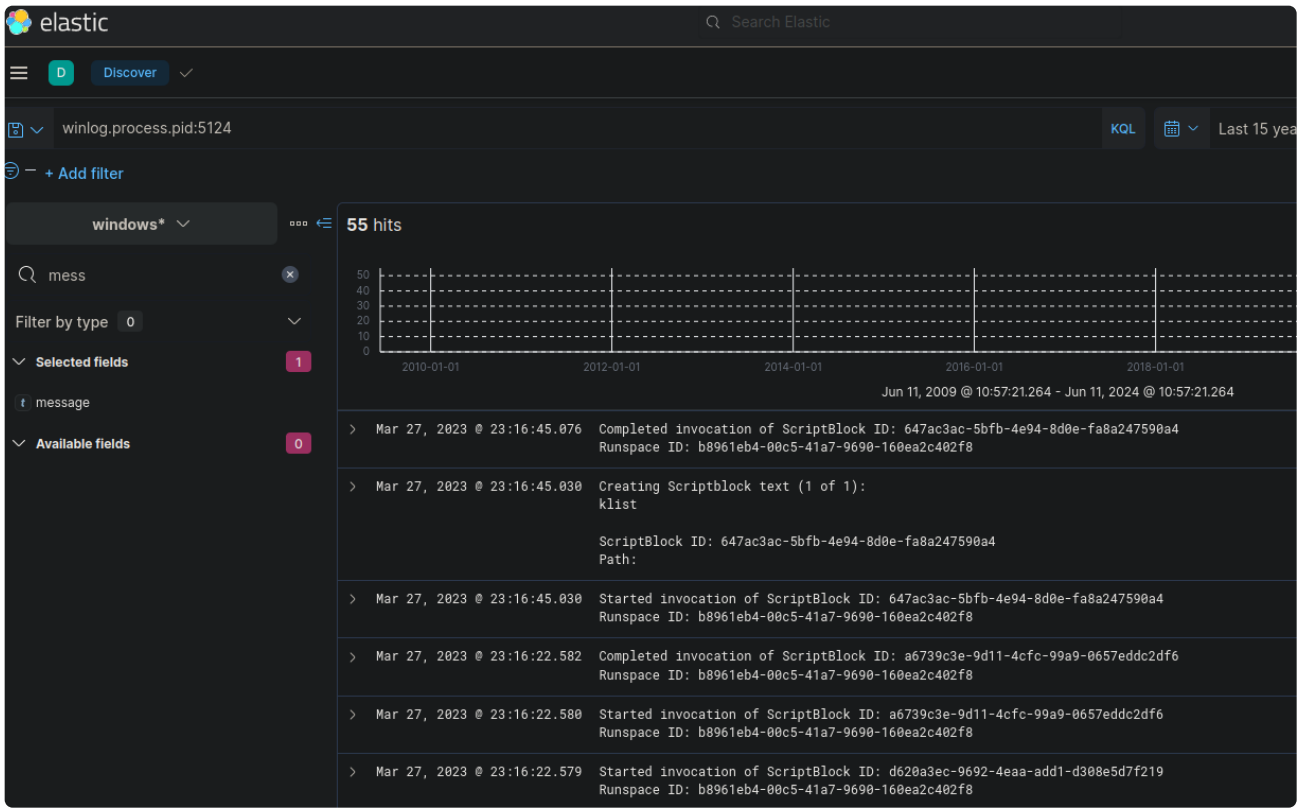


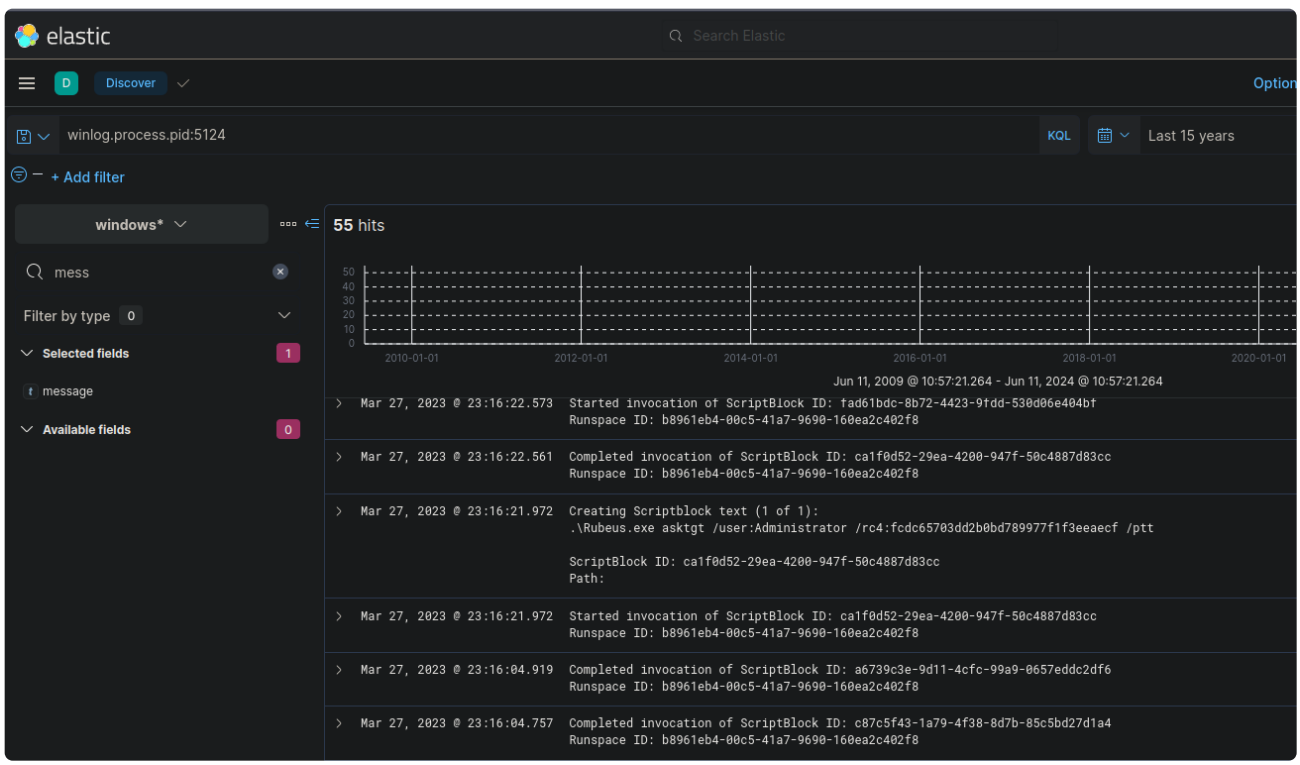
-> Diving deeper into the result, we see that it is indeed performing an lateral movement towards dc1.

-> Looking deeper at where the attack started, it started from a PKI service.

-> Examining the log more deeply, we see that:

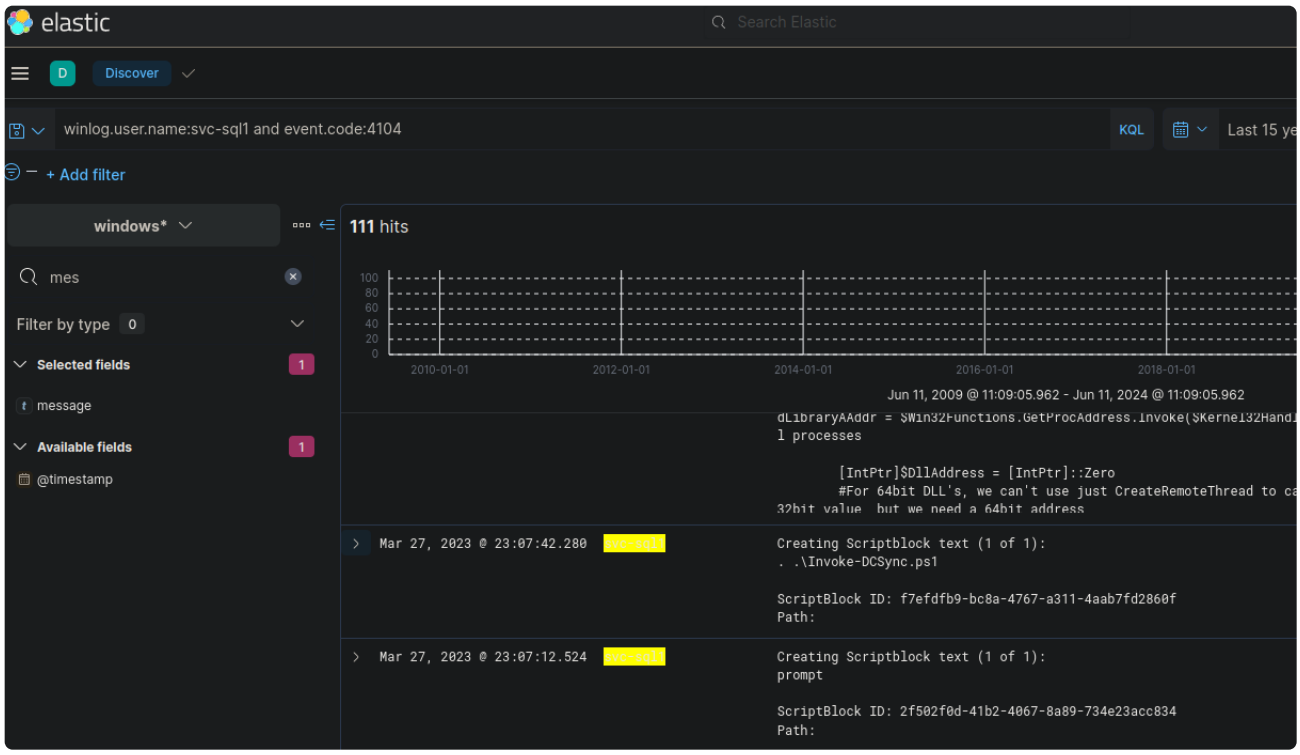






-> So the part of the picture becomes clear now, the attacker utilised an pass the ticket attack on the default domain admin user (administrator) using rubeus, checked that it has the ticket and moved laterally dc1.

-> A good follow up question is how the user obtained the hash of admin?



-> We see that it is importing scripts capable of performing DCSync, which is likely the technique it used to get the hash of the domain.
-> Now we can keep on going back to the question how it is able to do that to uncover the whole attack chain.

Threat Hunting & Threat Intelligence Fundamentals

Threat Hunting Fundamentals

Question

- Threat hunting is used ... Choose one of the following as your answer: "proactively", "reactively", "proactively and reactively".
  -> Threat hunting is used proactively and reactively, it is often initiated when a new vulnerability of application appears in our system our information on new adversary is discovered. It also works closely with incident response team and has an interdependence relationship with it.
- Threat hunting and incident handling are two processes that always function independently. Answer format: True, False.
  -> False, threat hunting and incident handling work closely together and threat hunting team often do work in the incident handling process.
- Threat hunting and incident response can be conducted simultaneously. Answer format: True, False.
  -> True, it can be conduct simulatenously, while incident response is performing, the

threat intelligence can look/hunt for other threat that might be trying to break in (e.g. drive by attacker).

The Threat Hunting Process

Question

- It is OK to formulate hypotheses that are not testable. Answer format: True, False.
  -> No, it is not ok to formulate hypothesis that are not testable. To elaborate, an important aspect of threat hunting is Evaluating Findings and Testing hypothesis, which would not be possible without an hypothesis.

Threat Intelligence Fundamentals

Question

- It’s useful for the CTI team to provide a single IP with no context to the SOC team. Answer format: True, False.
  -> No, CTI should comprise of 4 aspect, relevance, time, actionable and accuracy. Just providing an IP address provides no relevance to the company.
- When an incident occurs on the network and the CTI team is made aware, what should they do? Choose one of the following as your answer: "Do Nothing", "Reach out to the Incident Handler/Incident Responder", "Provide IOCs on all research being conducted, regardless if the IOC is verified".
  -> Reach out to the Incident Handler/Incident Responder, as timeliness is one of the factors in threat intelligence.
- When an incident occurs on the network and the CTI team is made aware, what should they do? Choose one of the following as your answer: "Provide IOCs on all research being conducted, regardless if the IOC is verified", "Do Nothing", "Provide further IOCs and TTPs associated with the incident".
  -> We would need to provide further IOCs and TTPs associated with the incident, as IOC needs to be verified for accuracy before disseminating any intelligence.
- Cyber Threat Intelligence, if curated and analyzed properly, can ... ? Choose one of the following as your answer: "be used for security awareness", "be used for fine-tuning network segmentation", "provide insight into adversary operations".
  -> It would provide insight into adversary operations. This is because the mission of cyber intelligence is to predict the location of the intended attack, the timing of thh attack, the operational strategies the adversary will employ and the ultimate objectives of the adversary.

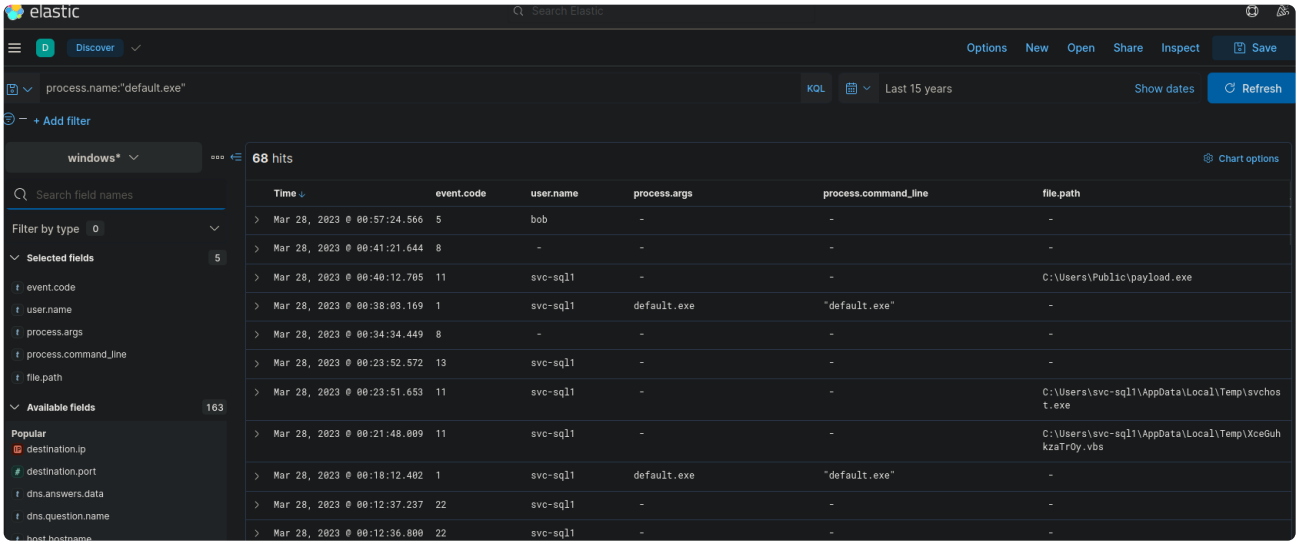
Threat Hunting With The Elastic Stack

Hunting For Stuxbot

Question

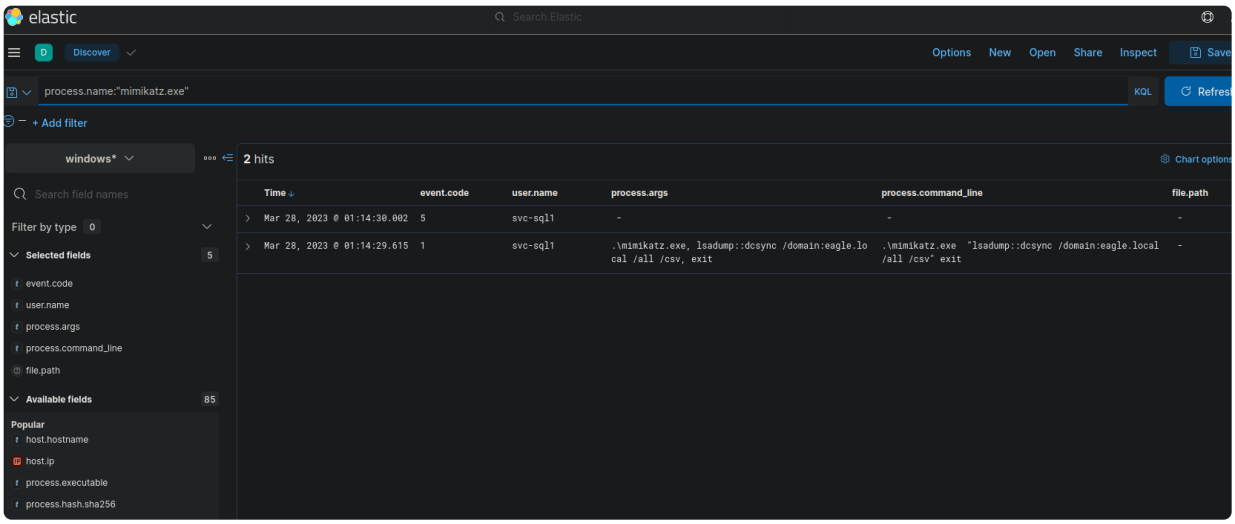
- Navigate to http://[Target IP]:5601 and follow along as we hunt for Stuxbot. In the part where default.exe is under investigation, a VBS file is mentioned. Enter its full name as your answer, including the extension.
  -> We search for default.exe with appropriate filters on selected field

```
process.name:"default.exe"
```



-> We see that `default.exe` uploaded `XceGuhkzaTrOy.vbs` as the vns file

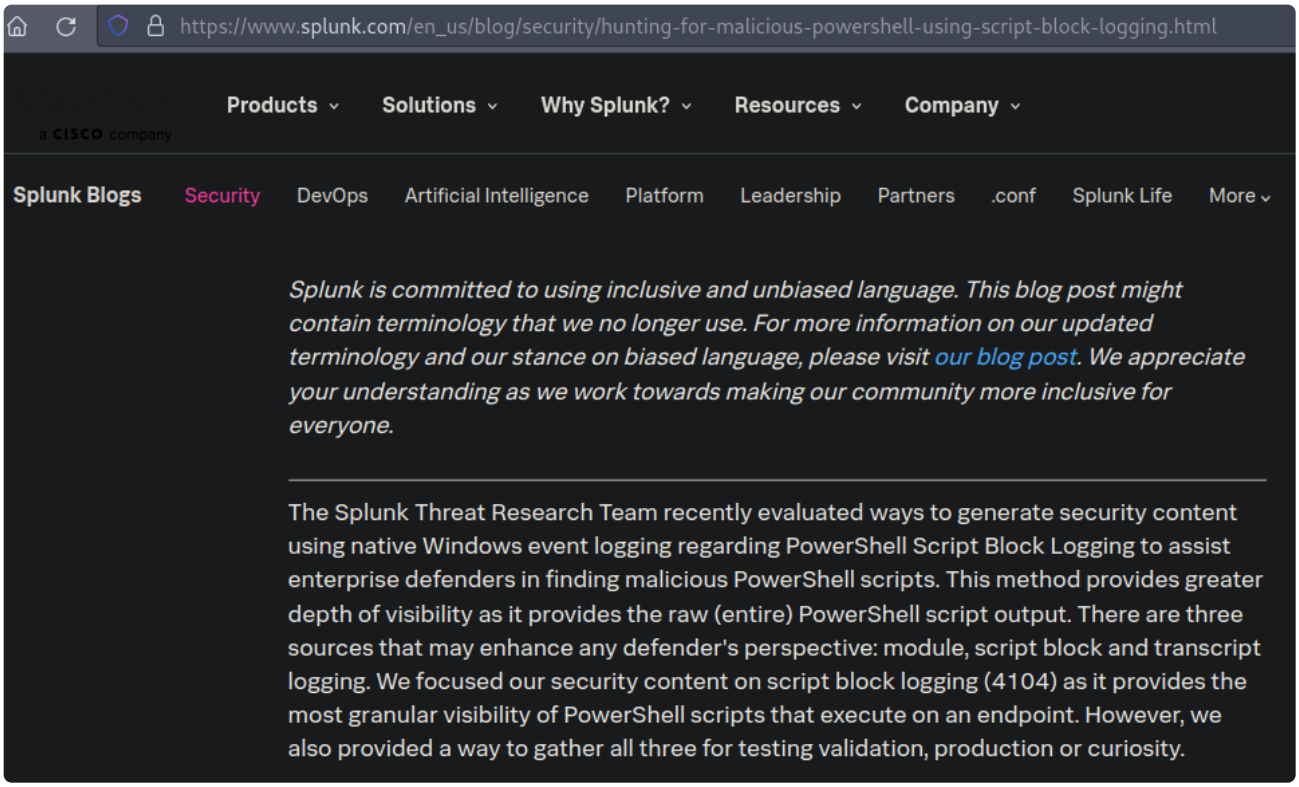
- Stuxbot uploaded and executed mimikatz. Provide the process arguments (what is after .\mimikatz.exe, ...) as your answer.
  -> We search for mimiktaz.exe accordingly for the process



-> and we see that `.\mimikatz.exe, lsadump::dcsync /domain:eagle.local /all /csv, exit` is being executed

- Some PowerShell code has been loaded into memory that scans/targets network shares. Leverage the available PowerShell logs to identify from which popular hacking tool this code derives. Answer format (one word): P__**V**_

-> We need to identify how to start filtering for relevant for PowerShell logs. We first being by consulting the relevant link in the section:

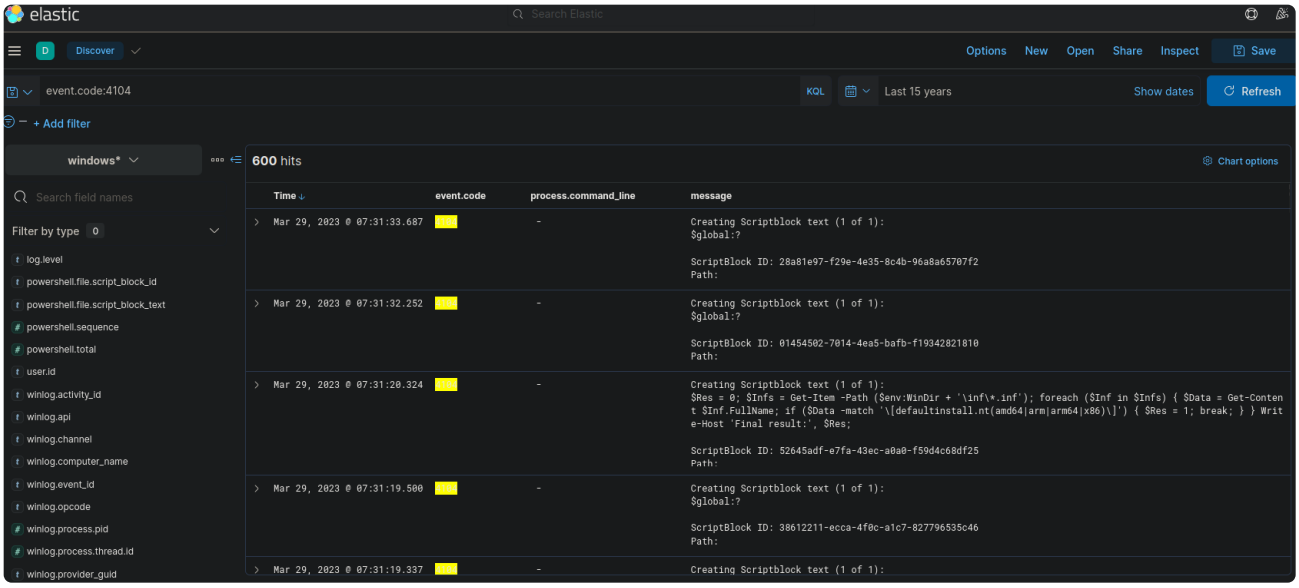


-> We see from the splunk post that script block logging provides the most granular visibility of PowerShell scripts that executes on an endpoint.

-> Hence, we look for filtering event 4014. We see from other posts that event 4014 refers to remote command execution using Powershell:

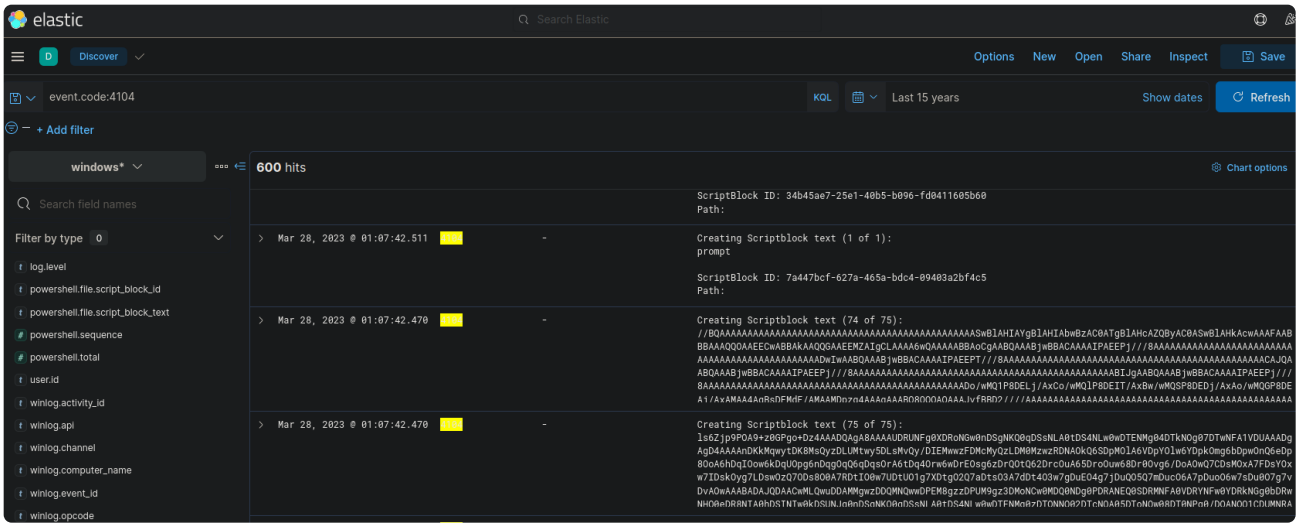
The first PowerShell code example below filters the event log entries using specific event IDs. In this example, event ID 4104 refers to the execution of a remote command using PowerShell.

The second PowerShell example queries an exported event log for the phrase "PowerShell."



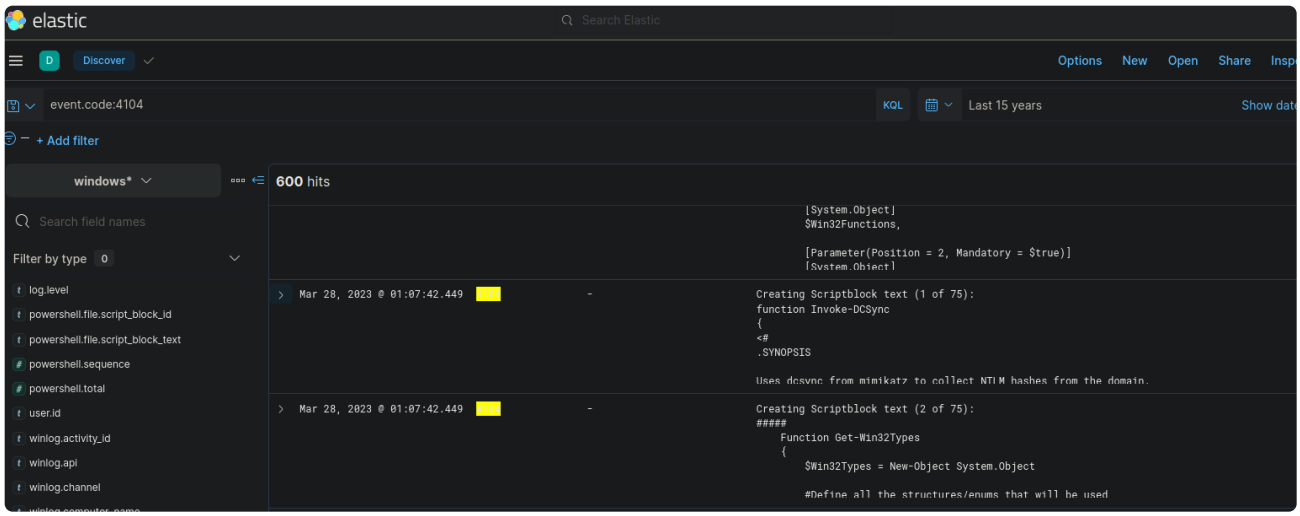
-> After querying for event 4104, we see 600 logs, not too much but also not too little. We can have a look here.

-> Scrolling down a bit, we see the following big chunk of ScriptBlock text:



-> This will be our target to look at for now.

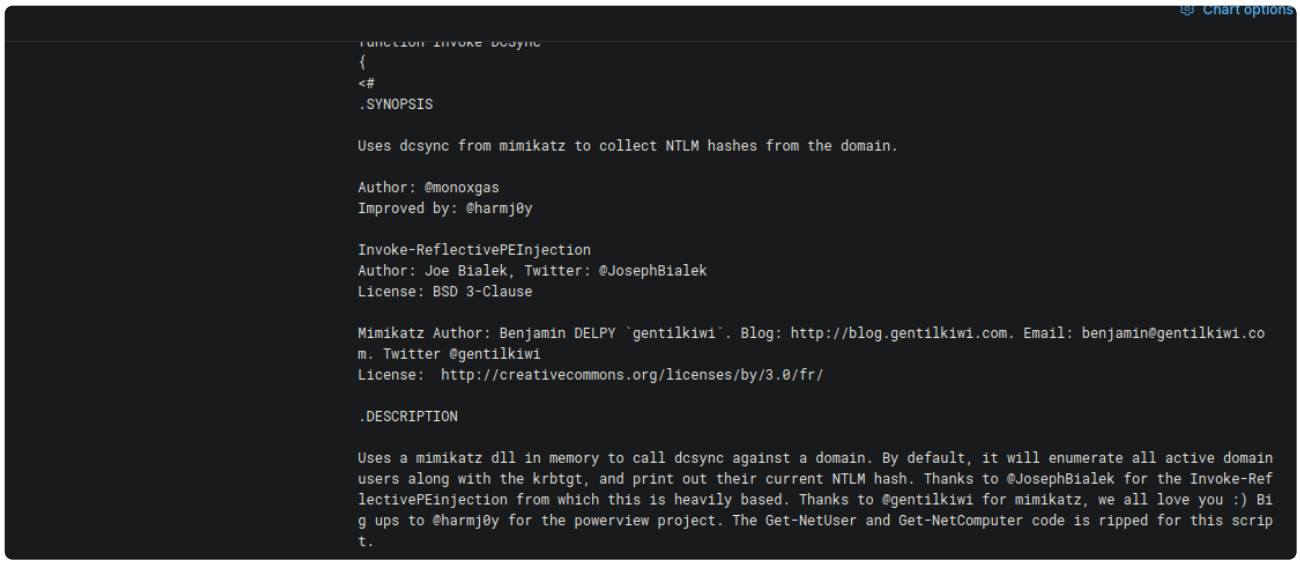
-> We'll look at the first script block as it may reveal the most important information, like what tool the code is derivsed from.



-> Now looking at this, we can tell from pentesting knowledge that this is the Powerview module, the Invoke-DCSync command, with reference to the relevant section shown below.

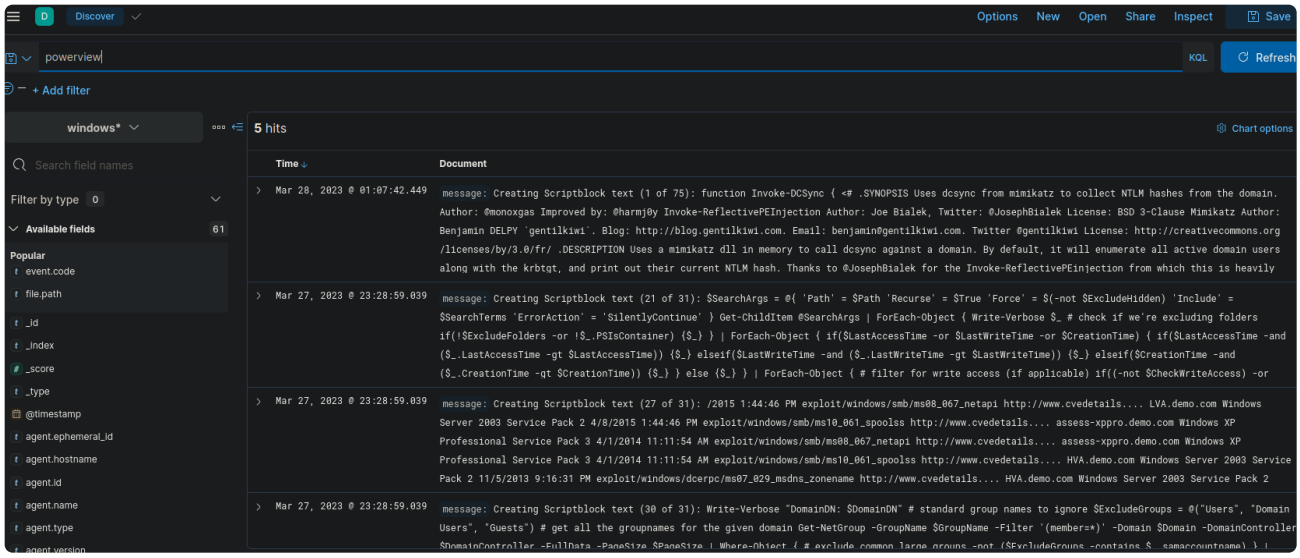
> If we had certain rights over the user (such as WriteDacl), we could also add this privilege to a user under our control, execute the DCSync attack, and then remove the privileges to attempt to cover our tracks. DCSync replication can be performed using tools such as Mimikatz, Invoke-DCSync, and Impacket's secretsdump.py. Let's see a few quick examples.

-> Furthermore, looking in detail of the script block section, we see from the description section:



-> We see that this snippet of code is from the PowerView project that uses a mimikatz dll in memory.

-> We can verify this by looking up powerview in the kql query



-> We can see that powerview is being referenced throughout the big chunk of code being loaded in the memory through Scriptblocks shown in the log.

-> Hence, the code being loaded derives from the powerview tool.