

# Weapon System Training - 2

---

## LAB 4: QUERIES AND FILTERS





Maj Michael Lester and Capt Jon Bynum  
32 WPS/DOA | NELLIS AFB, NEVADA

## Weapon System Training – 2: Queries and Filters

### CONTENTS

Lab 4: Queries and Filters .....	2
Overview .....	2
Procedures .....	3
Step 1.    Create Hypotheses to Test .....	3
Step 2.    Create Filter for Scripting Processes .....	5
Step 3.    Create DSL Query for Suspicious PowerShell Arguments .....	8
Appendix .....	15
Appendix A: Kibana Query Language (KQL) .....	15
4688 – Child Processes of another Process .....	15
4688 – Scripting Child Processes .....	15
Appendix B: Domain Specific Language (DSL) .....	16
4688 – PowerShell Contains -ExecutionPolicy Bypass .....	16
4688 – Commandline Contains Base64 with Exceptions .....	16

### Symbols Table

Symbol	Name	Meaning
	<b>Note</b>	Detailed information that is required to fully understanding the concept or to be able to execute a procedure but is not necessarily related to a key learning objective.
	<b>Learning Point</b>	Information related to key learning objectives.
	<b>Warning</b>	Important information related to safety and security.
	<b>Raise Hand</b>	Raise your hand for instructor assistance. This is often used at critical points to validate your understanding of the material.

## Weapon System Training – 2: Queries and Filters

### LAB 4: QUERIES AND FILTERS

#### OVERVIEW

**Summary:** The purpose of this lab is to acquire the necessary knowledge and skills to create and apply queries and filters using Kibana Query Language (KQL) and Domain Specific Language (DSL) within Kibana to search for specific adversary TTPs.

**Outcomes:** By the end of the lab, you will be able to perform the following:

- Create KQL queries.
  - Keyword searches against one and many fields.
  - Wildcard searches.
  - Boolean logic (and/or operations).
- Save KQL queries.
- Create DSL queries.
  - Filters on one field.
  - Regular expressions.
  - Exception lists.
- Save DSL queries.


## Weapon System Training – 2: Queries and Filters

### PROCEDURES

#### STEP 1. CREATE HYPOTHESES TO TEST

The first step in creating KQL and DSL queries and filters that are effective at detecting malicious activity is to create a good hypothesis about how an adversary will achieve their objectives in the cyber domain. Let's look at a basic hypothesis about how an adversary might gain initial access via phishing.

<b>Hypothesis</b>	An adversary will attempt to gain initial access using an embedded macro within a Microsoft Office document which will be opened by winword.exe, excel.exe, powerpnt.exe, etc. Upon execution, the macro will drop some type of executable file to disk (e.g. .js, .vb, .vbe, .js, .ps1, .bat, .exe, .dll, .scr) and spawn a new process to execute that file (e.g. the exe itself, powershell.exe, cscript.exe, wscript.exe, rundll32.exe, mshta.exe, regsvr32.exe).
<b>Artifacts</b>	Event Id 4688 Process Creation
<b>Analysis</b>	Frequency Analysis on Child Processes of Microsoft Office
<b>Implementation</b>	Threat Hunting Dashboard with Filter on Parent Process Path

 **LEARNING POINT: A hypothesis needs to get specific in order to develop robust analytic techniques to detect the hypothesized activity. We all know that adversaries love to use phishing attacks with malware embedded in attached documents, but what does that look like? What types of file extensions would the adversary drop? Does it have to be specific extensions? What types of processes would it spawn? Your hypothesis should answer some of these questions.**

Once you have a decent hypothesis, the next step is to consider what artifacts the adversary would create on the system during that attack. In the case above, the attacker would generate Event Id 4688 Process Creation events (among others). We could look for the specific child processes outlined in our hypothesis, or we could use frequency analysis (count of unique computer names by child process sorted in ascending order) to identify the least common child processes of Microsoft Office products. This technique has the advantage of being more generic and robust to adversary defense evasion because the technique is not based on specific IOCs. We already have this technique implemented in the Threat Hunting dashboard, but we need a simple query to analyze Microsoft Office processes.

Start developing filters by executing very simple queries and then work your way up to more complex queries by adding one component at a time.

1. Navigate to the Threat Hunting Dashboard you created in Lab 2.
2. Execute a search for all events where the field winlog.event\_data.ParentProcessName exists. This helps us verify that we have the field name spelled correctly and that there is data available to build our filter against.

```
winlog.event_data.ParentProcessName:*
```

There are multiple Microsoft Office executables. We could create a query that looks for each one of them and combines them together in a single query, or we could look for all applications running out of the

## Weapon System Training – 2: Queries and Filters

Microsoft Office installation directory. The benefit is that the later query is less process intensive, and it potentially covers more products than a single list. We can always tailor our query down later if we need to.

- Execute the following query looking for all Event Id 4688 logs where the ParentProcessName field starts with the base path to the Microsoft Office installation directory.

```
winlog.event_data.ParentProcessName:"C:\Program Files\Microsoft Office\*" 
```

You should see results similar to those shown below:

ParentProcessName	Count
C:\Program Files\Microsoft Office\root\Office16\WINWORD.exe	1

4688 - Lens Table Detailed					
@timestamp per hour	computer_name	user_name	process.pid	process.child.command_line	Count of records
2022-07-19 12:00	WKST-001.lab.net	Administrator	7,864	"C:\Program Files (x86)\Mozilla Firefox\firefox.exe" https://careers.mitre.org/us/en	1
2022-07-19 12:00	WKST-001.lab.net	Administrator	7,864	\?C:\Windows\system32\conhost.exe 0xffffffff -ForceV1	1
2022-07-19 12:00	WKST-001.lab.net	Administrator	8,324	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -executionpolicy bypass -nopprofile -noexit -encodedcommand VwByAGkAdABIACOASABvAHMAdAAgACcAaABIAgWABA...	1
2022-07-19 12:00	WKST-001.lab.net	Administrator	8,324	\?C:\Windows\system32\conhost.exe 0xffffffff -ForceV1	1

**LEARNING POINT:** Looking at parent child process relationships tells a story about what was happening on the system, and specifically what a user was doing on that system.

**RAISE HAND:** What story does the first event above in the “4688 – Lens Table Detailed” panel tell? Explain to the instructor what the user did that caused this event.

The filter and dashboard shown above combine two analytic techniques:

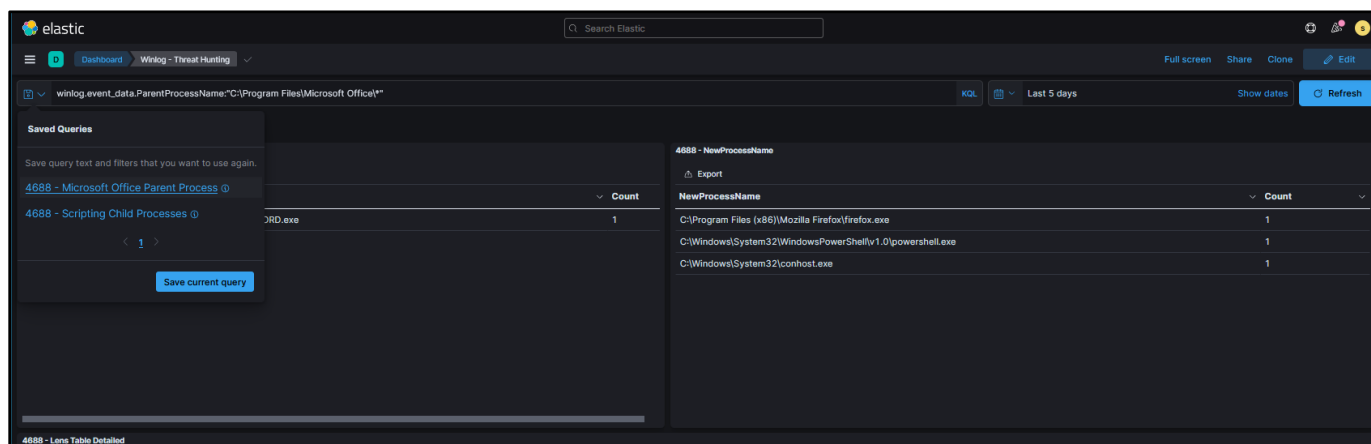
- Parent/child process relationships
- Frequency Analysis of child processes to identify anomalies

The least common child processes of Microsoft Office products will be shown in the panel in the top, right-hand corner of the screen. We can then easily investigate each one of those by creating a quick filter by clicking on the “+” symbol for each of those values. The quick filter will show examples of child processes with commandline arguments in the Lens Table at the bottom of the page.

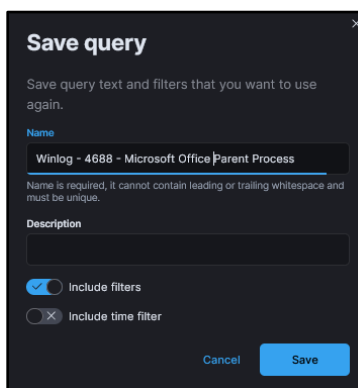
This query is sufficient for what we need it for. We can go ahead and save it as a Search Query that we can reference later in combination with one or more dashboards.

- Click on the “save” icon on the left-hand side of the search bar, then click on the blue “Save current query” button on the popup.

## Weapon System Training – 2: Queries and Filters



5. Configure the following options, then click the blue “Save” button.
  - 5.1. **Name:** Winlog – 4688 – Microsoft Office Parent Process (<lastname>)
  - 5.2. **Include filters:** Checked
  - 5.3. **Include time filter:** Unchecked



### STEP 2. CREATE FILTER FOR SCRIPTING PROCESSES

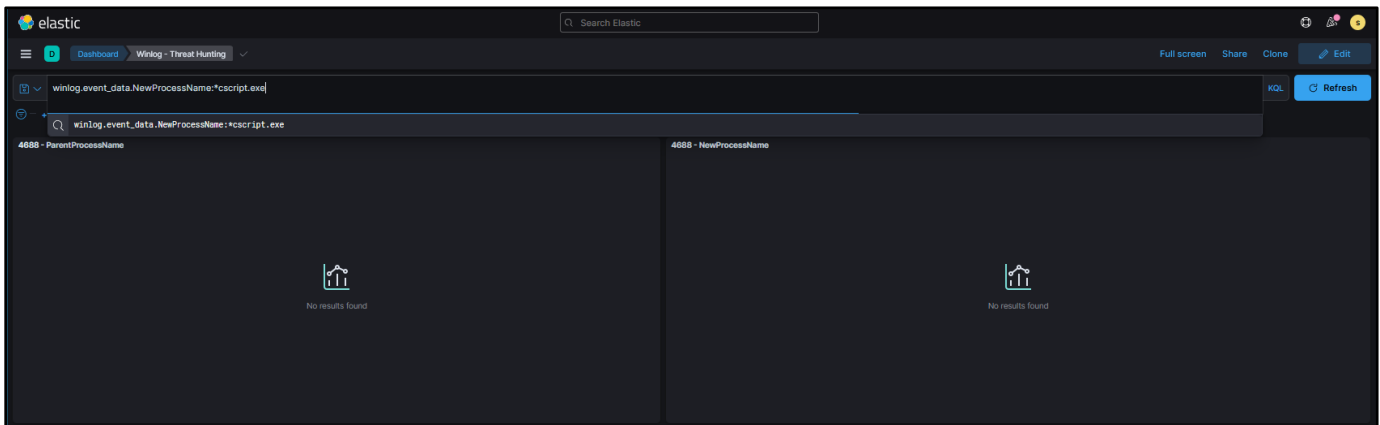
Another part of our hypothesis includes an adversary creating a scripting child process that is native to Windows. We could create a saved search that narrows the dashboard down to only display child scripting processes. There are several, so we will combine them together with Boolean logic operators. We'll add them one at a time to make the complex query easier to construct.

1. Start by running a query for all process creation events where the child process was `cscrip.exe`.

```
winlog.event_data.NewProcessName:*cscrip.exe
```

You most likely will get blank results such as in the screenshot below. Did we mess up the query? Let's try with a binary that is more likely to be executed.

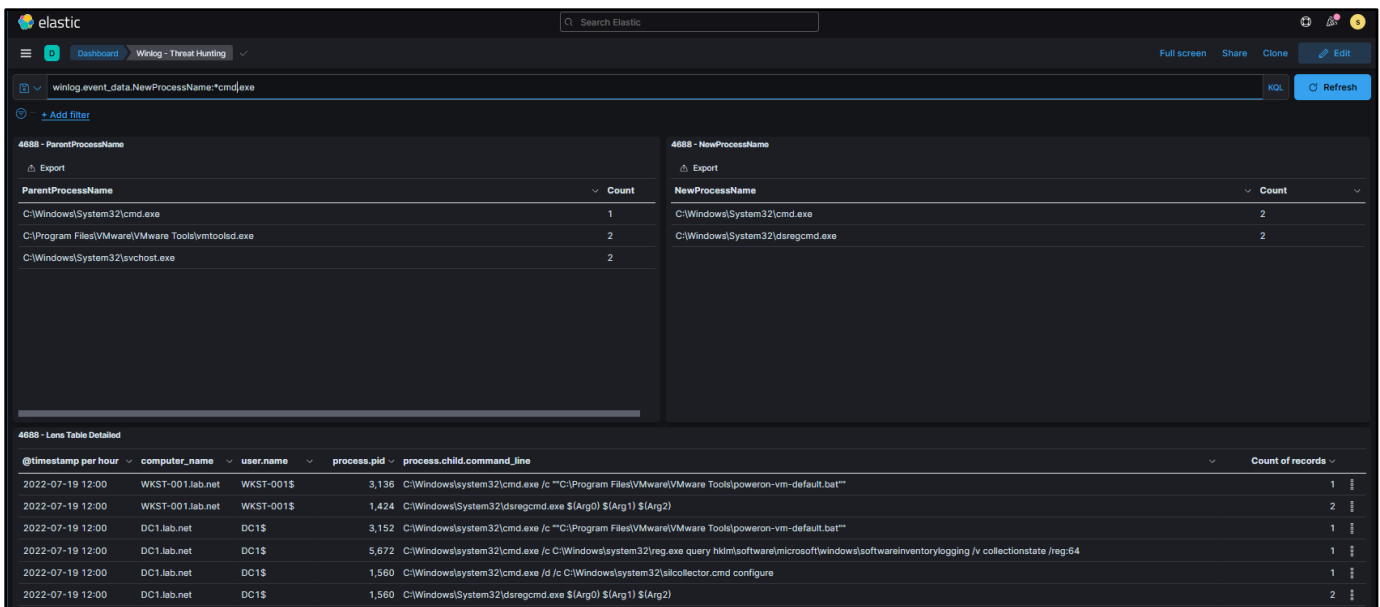
## Weapon System Training – 2: Queries and Filters



2. Change the previous search to look for `cmd.exe` as that is more likely to have been run in the environment and can help us verify that our query is correct.

```
winlog.event_data.NewProcessName:*cmd.exe
```

You should see some results now.



3. Now that we have verified our basic query is correct, we can add some additional terms by copying and pasting, then modifying our previous query.

```
winlog.event_data.NewProcessName:*cscrip.exe or winlog.event_data.NewProcessName:*wscript.exe or  
winlog.event_data.NewProcessName:*mshta.exe or winlog.event_data.NewProcessName:*powershell.exe or  
winlog.event_data.NewProcessName:*pwsh.exe
```

You should now see some results similar to the following:

## Weapon System Training – 2: Queries and Filters

elastic

Search Elastic

Dashboard Winlog - Threat Hunting

Full screen Share Clone Edit

(winlog.event\_data.NewProcessName:\*cscript.exe or winlog.event\_data.NewProcessName:\*wscript.exe or winlog.event\_data.NewProcessName:\*mshta.exe or winlog.event\_data.NewProcessName:\*powershell KQL Last 5 days Show dates Refresh

+ Add filter

4688 - ParentProcessName

Export

ParentProcessName	Count
C:\Program Files\Microsoft Office\Root\Office16\WINWORD.exe	1
C:\Windows\explorer.exe	1
C:\Windows\System32\CompatTelRunner.exe	2

4688 - NewProcessName

Export

NewProcessName	Count
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	2

4688 - Lines Table Detailed

@timestamp per hour	computer_name	user_name	process.pid	process.child.command_line	Count of records
2022-07-19 12:00	DC1.lab.net	DC1\$	1,320	powershell.exe -ExecutionPolicy Restricted -Command \$res = 0; if(\$? -and \$?) {get-vmswitch}   Where (\$_.NetAdapterInterfaceDescription -ne \$null -and \$_.NetAdapterInterfaceDescription -eq (Get-Netb...	1
2022-07-19 12:00	DC1.lab.net	DC1\$	1,320	powershell.exe -ExecutionPolicy Restricted -Command Write-Host 'Final result: 1';	1
2022-07-19 12:00	WKST-001.lab.net	Administrator	8,324	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -executionpolicy bypass -noexit -encodedcommand YwByAGkA2ABAC0ASABVAHMAgAAgACcAaABAGwABvAC...	1
2022-07-19 12:00	WKST-001.lab.net	WKST-001\$	1,728	powershell.exe -ExecutionPolicy Restricted -Command Write-Host 'Final result: 1';	1
2022-07-19 13:00	WKST-001.lab.net	Administrator	5,160	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"	3

This query is starting to shape up, but there is one more thing to consider. In our hypothesis, the parent process of the scripting host application is winword.exe or excel.exe; however, our current query includes events where “C:\Windows\explorer.exe” is a parent process. Those events are most likely from a standard user running PowerShell from their desktop and then manually typing commands. Those are false positives that aren’t directly related to our hypothesis, so we can filter those out by requiring that explorer.exe not be the parent process.

4. Run the following query in the search bar to eliminate false positives with explore.exe as the parent process.

```
(winlog.event_data.NewProcessName:*cscript.exe or winlog.event_data.NewProcessName:*wscript.exe or winlog.event_data.NewProcessName:*mshta.exe or winlog.event_data.NewProcessName:*powershell.exe or winlog.event_data.NewProcessName:*pwsh.exe) and not winlog.event_data.ParentProcessName.keyword:*explorer.exe
```

You should see results similar to those shown below:



## Weapon System Training – 2: Queries and Filters

The screenshot shows the Elastic SIEM dashboard with a KQL query: `(winlog_event_data.NewProcessName:*csript.exe or winlog_event_data.NewProcessName:*wscript.exe or winlog_event_data.NewProcessName:*mshta.exe or winlog_event_data.NewProcessName:*powershell.exe or winlog_event_data.NewProcessName:*pwsh.exe) and not winlog_event_data.ParentProcessName.keyword:*explorer.exe`. The results are split into two panels: 'ParentProcessName' and 'NewProcessName'. The 'ParentProcessName' panel shows two entries: 'C:\Program Files\Microsoft Office\Root\Office16\WINWORD.exe' with a count of 1, and 'C:\Windows\System32\CompatTelRunner.exe' with a count of 2. The 'NewProcessName' panel shows one entry: 'C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe' with a count of 2. Below these panels is a detailed table for the 'NewProcessName' results.

@timestamp per hour	computer_name	user_name	process.pid	process.child.command_line	Count of records
2022-07-19 12:00	DC1.lab.net	DC1\$	1,320	powershell.exe -ExecutionPolicy Restricted -Command \$res = 0; if(\$get-vmswitch) { Where (\$_.NetAdapterInterfaceDescription -ne \$null -and \$_.NetAdapterInterfaceDescription -eq (Get-NetB...	1
2022-07-19 12:00	DC1.lab.net	DC1\$	1,320	powershell.exe -ExecutionPolicy Restricted -Command Write-Host 'Final result: 1';	1
2022-07-19 12:00	WKST-001.lab.net	Administrator	8,324	"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -executionpolicy bypass -nopprofile -noexit -encodedcommand VwByAGkAdABAC0ASABVAHMAgAAgACcAbABIAgWAbABVAC...	1
2022-07-19 12:00	WKST-001.lab.net	WKST-001\$	1,728	powershell.exe -ExecutionPolicy Restricted -Command Write-Host 'Final result: 1';	1

5. This query is now sufficient for our needs. Go ahead and save the query with the following options:
  - 5.1. **Name:** Winlog – 4688 – Scripting Child Process
  - 5.2. **Include filters:** Checked
  - 5.3. **Include time filter:** Unchecked

The 'Save query' dialog box is shown with the following fields and options:

- Name:** Winlog - 4688 - Scripting Child Process
- Description:** (Empty text box)
- Include filters:** ☒ (Checked)
- Include time filter:** ☐ (Unchecked)
- Buttons:** Cancel, Save

### STEP 3. CREATE DSL QUERY FOR SUSPICIOUS POWERSHELL ARGUMENTS

Next, let's focus in on PowerShell in our hypothesis. One of the ways an attacker could break out of the Microsoft Office application after gaining initial execution through the embedded VBA script is to run powershell.exe with the -ExecutionPolicy Bypass argument. This argument is commonly used by attackers and system administrators to bypass policies that prevent execution of scripts in the environment. We could create a filter to only show PowerShell processes where these arguments were specified to eliminate some false positives.

Unfortunately, as a defender, there are a lot of ways an attacker could call those arguments. With powershell, you generally only need to provide as many leading characters of an argument name as necessary to uniquely distinguish that argument from the other arguments, so in this case the full string "-

## Weapon System Training – 2: Queries and Filters

ExecutionPolicy” bypass is not *really* necessary. It could be shorted to “-Exe b” or “-ex b” or “-exec byp”. There are a lot of possible permutations here. Too many to list them all out individually. To get around that, we are going to use a regular expression query on process commandline arguments.

Just like before, we are going to start out with a very basic query and then work our way up to more advanced queries.

1. Start by navigating back to our Threat Hunting dashboard and clearing out all of the filters and searches. You should see something similar to what is shown below.
2. Click on the “+Add filter” button in the top, left-hand corner of the screen.

The screenshot shows the Elastic Threat Hunting dashboard. On the left, the 'Add filter' dialog is open, showing a list of fields to choose from. The 'Field' dropdown is set to 'process.command\_line'. The 'Operator' is 'Contains'. The 'Value' is 'C:\Program Files (x86)\Mozilla Firefox\uninstall\helper.exe'. The 'Count' is 1. The 'Add filter' button is highlighted. On the right, the 'NewProcessName' table is displayed, showing a list of process names and their counts. The table has columns for 'NewProcessName' and 'Count'. The first row is 'C:\\$WinREAgent\Scratch\89014E78-06B8-431F-B303-22BE688BD964\DiskHost.exe' with a count of 1. The second row is 'C:\Program Files (x86)\Microsoft\EdgeUpdate\install\{4816018C-C8F9-4750-8E49-4EB5E42D70A5}\EDGEIMTP\_E0B87.tmp\set...' with a count of 1. The third row is 'C:\Program Files (x86)\Microsoft\EdgeUpdate\install\{4816018C-C8F9-4750-8E49-4EB5E42D70A5}\MicrosoftEdge\_X64\_103.0.1...' with a count of 1. The fourth row is 'C:\Program Files (x86)\Microsoft\EdgeUpdate\MicrosoftEdgeUpdate.exe' with a count of 1. The fifth row is 'C:\Program Files (x86)\Microsoft\Edge\Application\103.0.1264.62\Installer\setup.exe' with a count of 1. The sixth row is 'C:\Program Files (x86)\Mozilla Firefox\default-browser-agent.exe' with a count of 1. The seventh row is 'C:\Program Files (x86)\Mozilla Firefox\Firefox.exe' with a count of 1. The eighth row is 'C:\Program Files (x86)\Mozilla Firefox\uninstall\helper.exe' with a count of 1. The ninth row is 'C:\Program Files (x86)\Mozilla Firefox\update.exe' with a count of 1.

3. Next, click on the blue “Edit as Query DSL” button in the top, right-hand corner of the popup.
4. Copy and paste the following query into panel and click “Save”.

```
{
  "query": {
    "bool": {
      "filter": [
        {
          "term": {
            "winlog.event_id": {
              "value": "4688"
            }
          }
        }
      ]
    }
  }
}
```

**LEARNING POINT:** The filter above executes a query that is a Boolean combination of filters with one filter that is a term filter where winlog.event\_id is 4688.

You should see output similar to the what is shown below.

## Weapon System Training – 2: Queries and Filters

elastic

Search Elastic

Dashboard Winlog - Threat Hunting

Full screen Share Clone Edit

Search

["bool":{"filter":[{"term":{"winlog.event\_id":{"value":"4688"}}]}]}]

4688 - ParentProcessName

Export

ParentProcessName	Count
C:\Program Files (x86)\Microsoft\EdgeUpdate\install\{4816018C-C8F9-4750-8E49-4EB5E42D70A5}\EDGEIMTP_E0B87.tmp\set...	1
C:\Program Files (x86)\Microsoft\EdgeUpdate\install\{4816018C-C8F9-4750-8E49-4EB5E42D70A5}\MicrosoftEdge_X64_103.0.1...	1
C:\Program Files (x86)\Microsoft\EdgeUpdate\MicrosoftEdgeUpdate.exe	1
C:\Program Files (x86)\Mozilla Firefox\firefox.exe	1
C:\Program Files (x86)\Mozilla Firefox\uninstall\helper.exe	1
C:\Program Files (x86)\Mozilla Firefox\update.exe	1
C:\Program Files\Microsoft Office\Root\Office16\WINWORD.exe	1
C:\Users\administrator\AppData\Local\Microsoft\OneDrive\OneDriveStandaloneUpdater.exe	1
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\ngen task.exe	1

4688 - NewProcessName

Export

NewProcessName	Count
C:\\$WinREAgent\Scratch\89014E78-0688-431F-B303-22BE688D964\DisHost.exe	1
C:\Program Files (x86)\Microsoft\EdgeUpdate\install\{4816018C-C8F9-4750-8E49-4EB5E42D70A5}\EDGEIMTP_E0B87.tmp\set...	1
C:\Program Files (x86)\Microsoft\EdgeUpdate\install\{4816018C-C8F9-4750-8E49-4EB5E42D70A5}\MicrosoftEdge_X64_103.0.1...	1
C:\Program Files (x86)\Microsoft\EdgeUpdate\MicrosoftEdgeUpdate.exe	1
C:\Program Files (x86)\Microsoft\Edge\Application\103.0.1264.62\Installer\setup.exe	1
C:\Program Files (x86)\Mozilla Firefox\default-browser-agent.exe	1
C:\Program Files (x86)\Mozilla Firefox\firefox.exe	1
C:\Program Files (x86)\Mozilla Firefox\uninstall\helper.exe	1
C:\Program Files (x86)\Mozilla Firefox\update.exe	1

4688 - Lens Table Detailed

@timestamp per hour	computer_name	user.name	process.pid	process.child.command_line	Count of records
2022-07-19 12:00	DC1.lab.net	LOCAL SERVI...	3,360	\\?C:\Windows\system32\conhost.exe 0xffffffff -ForceV1	1
2022-07-19 12:00	DC1.lab.net	DC1\$	400	C:\Windows\system32\MpSigStub.exe /stub 1.1.18500.10 /payload 1.371.455.0 /MpWUStub /program C:\Windows\SoftwareDistribution\Download\install\AM_Delta.exe WD /q	1

```
{
  "query": {
    "bool": {
      "filter": [
        {
          "term": {
            "winlog.event_id": {
              "value": "4688"
            }
          }
        }
      ]
    }
  }
}
```

elastic

Search Elastic

Dashboard Winlog - Threat Hunting

Full screen Share Clone Edit

Search

["bool":{"filter":[{"term":{"winlog.event\_id":{"value":"4688"}}]}]}]

4688 - ParentProcessName

Export

ParentProcessName	Count
C:\Program Files (x86)\Microsoft\EdgeUpdate\install\{4816018C-C8F9-4750-8E49-4EB5E42D70A5}\EDGEIMTP_E0B87.tmp\set...	1
C:\Program Files (x86)\Microsoft\EdgeUpdate\install\{4816018C-C8F9-4750-8E49-4EB5E42D70A5}\MicrosoftEdge_X64_103.0.1...	1
C:\Program Files (x86)\Microsoft\EdgeUpdate\MicrosoftEdgeUpdate.exe	1
C:\Program Files (x86)\Mozilla Firefox\firefox.exe	1
C:\Program Files (x86)\Mozilla Firefox\uninstall\helper.exe	1
C:\Program Files (x86)\Mozilla Firefox\update.exe	1
C:\Program Files\Microsoft Office\Root\Office16\WINWORD.exe	1
C:\Users\administrator\AppData\Local\Microsoft\OneDrive\OneDriveStandaloneUpdater.exe	1
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\ngen task.exe	1

## Weapon System Training – 2: Queries and Filters

- Next, add another filter that is a regular expression filter with a catch all such as “.\*”. This regular expression should match on everything at this point.
- Copy and paste the following code into your DSL query.

```
{
  "query": {
    "bool": {
      "filter": [
        {
          "term": {
            "winlog.event_id": {
              "value": "4688"
            }
          }
        },
        {
          "regexp": {
            "winlog.event_data.CommandLine": {
              "case_insensitive": true,
              "flags": "ALL",
              "value": ".*"
            }
          }
        }
      ]
    }
  }
}
```

You should see output similar to what is shown below (i.e. nothing should change!).

The screenshot shows the Elastic Security console interface. At the top, the search bar contains the query: `["bool":{"filter":[{"term":{"winlog.event_id":{"value":"4688"}}}, {"regexp":{"winlog.event_data.CommandLine":{"case_insensitive":true,"flags":"ALL","value":".*"}}]}}]`. Below the search bar, there are two panels showing search results for event ID 4688.

**4688 - ParentProcessName**

ParentProcessName	Count
C:\Program Files (x86)\Microsoft\EdgeUpdate\Install\{4816018C-CBF9-4750-8E49-4E85E42D70A5}\EDGEMITMP_E0B87.tmp\set...	1
C:\Program Files (x86)\Microsoft\EdgeUpdate\Install\{4816018C-CBF9-4750-8E49-4E85E42D70A5}\MicrosoftEdge_X64_103.0.1...	1
C:\Program Files (x86)\Microsoft\EdgeUpdate\MicrosoftEdgeUpdate.exe	1
C:\Program Files (x86)\Mozilla Firefox\firefox.exe	1
C:\Program Files (x86)\Mozilla Firefox\uninstall\helper.exe	1
C:\Program Files (x86)\Mozilla Firefox\update.exe	1
C:\Program Files\Microsoft Office\Root\Office16\WINWORD.exe	1
C:\Users\Administrator\AppData\Local\Microsoft\OneDrive\OneDriveStandaloneUpdater.exe	1
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\ngen task.exe	1

**4688 - NewProcessName**

NewProcessName	Count
C:\WinREAgent\Scratch\89014E78-0688-431F-B303-22BE6888D964\DisHost.exe	1
C:\Program Files (x86)\Microsoft\EdgeUpdate\Install\{4816018C-CBF9-4750-8E49-4E85E42D70A5}\EDGEMITMP_E0B87.tmp\set...	1
C:\Program Files (x86)\Microsoft\EdgeUpdate\Install\{4816018C-CBF9-4750-8E49-4E85E42D70A5}\MicrosoftEdge_X64_103.0.1...	1
C:\Program Files (x86)\Microsoft\EdgeUpdate\MicrosoftEdgeUpdate.exe	1
C:\Program Files (x86)\Microsoft\Edge\Application\103.0.1264.62\Installer\setup.exe	1
C:\Program Files (x86)\Mozilla Firefox\default-browser-agent.exe	1
C:\Program Files (x86)\Mozilla Firefox\firefox.exe	1
C:\Program Files (x86)\Mozilla Firefox\uninstall\helper.exe	1
C:\Program Files (x86)\Mozilla Firefox\update.exe	1

**4688 - Lens Table Detailed**

@timestamp per hour	computer_name	user.name	process.pid	process.child.command_line	Count of records
2022-07-19 12:00	DC1.lab.net	LOCAL SERVI...	3,360	[77C:\Windows\system32\conhost.exe 0xfffffff -ForceV1	1
2022-07-19 12:00	DC1.lab.net	DC1\$	400	C:\Windows\system32\ImpSigStub.exe /stub 1.1.18500.10 /payload 1.371.455.0 /MpwUStub /program C:\Windows\SoftwareDistribution\Download\Install\AM_Delta.exe WD /q	1
2022-07-19 12:00	DC1.lab.net	DC1\$	500	"fontdrvhost.exe"	1

- Next, add a regular expression that looks for all possible combinations of ways to execute the - ExecutionPolicy Bypass arguments.

```
{
```

## Weapon System Training – 2: Queries and Filters

```
"query": {
  "bool": {
    "filter": [
      {
        "term": {
          "winlog.event_id": {
            "value": "4688"
          }
        }
      },
      {
        "regexp": {
          "process.child.command_line.keyword": {
            "case_insensitive": true,
            "flags": "ALL",
            "value": ".*powershell.exe.*-ex([ecutionply]+)?[ \\t]+b.*"
          }
        }
      }
    ]
  }
}
```

You should now see output similar to the following. If you don't, log into one of the systems and run PowerShell with the -ExecutionPolicy Bypass arguments and change your time filter to reflect the current time.

ParentProcessName	Count
C:\\Program Files\\Microsoft Office\\root\\Office16\\WINWORD.exe	1

NewProcessName	Count
C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe	1

@timestamp	per hour	computer_name	user.name	process.pid	process.child.command_line	Count of records
2022-07-19 12:00	WKST-001.lab.net	Administrator	8,324	C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe	-executionpolicy bypass -nopprofile -noexit -encodedcommand VwByAGkAdABAC0ASABVAHMAAAGAgACcAaABIAQwAbABvAC...	1

Next, we are going to add a term to reduce false positives and the number of events the regular expression needs to match.

8. Copy and paste the following query in the DSL query in kibana.

```
{
  "query": {
    "bool": {
```

## Weapon System Training – 2: Queries and Filters

```
"filter": [
  {
    "term": {
      "winlog.event_id": {
        "value": "4688"
      }
    }
  },
  {
    "regexp": {
      "process.child.command_line.keyword": {
        "case_insensitive": true,
        "flags": "ALL",
        "value": ".*powershell.exe.*-ex([ecution|y|+)?[ \\t]+b.*"
      }
    }
  }
],
"must_not": {
  "wildcard": {
    "winlog.event_data.ParentProcessName": {
      "case_insensitive": true,
      "value": "*explorer.exe"
    }
  }
}
}
```

You should see output similar to the following.

The screenshot shows the Elastic Search Kibana interface. The search bar contains the query: `("bool":{"filter":[{"term":{"winlog.event_id":{"value":"4688"}}},{"regexp":{"process.child.command_line.keyword":{"case_insensitive":true,"flags":"ALL","value":".*powershell.exe.*-ex([ecution|y|+)?[ \\t]+b.*"}}}],"must_not":{"wildcard":{"winlog.event_data.ParentProcessName":{"case_insensitive":true,"value":"*explorer.exe"}}}}})`. The search results are displayed in two tables. The first table, titled "4688 - ParentProcessName", shows a single row with "ParentProcessName" as "C:\Program Files\Microsoft Office\Root\Office16\WINWORD.exe" and a count of 1. The second table, titled "4688 - NewProcessName", shows a single row with "NewProcessName" as "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" and a count of 1. Below these tables is a "Lens Table Detailed" view showing columns for @timestamp, computer\_name, user.name, process.pid, process.child.command\_line, and Count of records. The first row shows a timestamp of 2022-07-19 12:00, computer\_name WKST-001.lab.net, user.name Administrator, process.pid 8,324, and process.child.command\_line C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe, with a count of 1 record.

This query is looking pretty good. Let's go ahead and save it.

9. Click on the blue "Save" button on the DSL edit screen with the following options:
  - 9.1. **Name:** Winlog – 4688 – PowerShell with Base64

## Weapon System Training – 2: Queries and Filters

Edit filter Edit filter values

Elasticsearch Query DSL

```
1 {
2   "query": {
3     "bool": {
4       "filter": [
5         {
6           "term": {
7             "winlog.event_id": {
8               "value": "4688"
9             }
10          }
11        },
12      ]
13    }
14  }
```

☒ Create custom label?

Custom label

Winlog - 4688 - PowerShell with Base64

Cancel Save

10. Save the query so that it can be reused by following the same steps we used to save KQL queries.

## Weapon System Training – 2: Queries and Filters

### APPENDIX

#### APPENDIX A: KIBANA QUERY LANGUAGE (KQL)

##### 4688 – CHILD PROCESSES OF ANOTHER PROCESS

<b>Description:</b>	This is a filter to show only processes that are created by Microsoft Office processes.
<b>Query:</b>	<code>winlog.event_data.ParentProcessName:"C:\Program Files\Microsoft Office\*"</code>

##### 4688 – SCRIPTING CHILD PROCESSES

<b>Description:</b>	This filter shows only events where the child process is a scripting host application such as powershell.exe, cscript.exe, wscript.exe, or mshta.exe and the parent process is not explorer.exe. The last part ensures that the application wasn't opened by a currently logged on user as that is less suspicious event.
<b>Query:</b>	<code>(winlog.event_data.NewProcessName:*cscript.exe or winlog.event_data.NewProcessName:*wscript.exe or winlog.event_data.NewProcessName:*mshta.exe or winlog.event_data.NewProcessName:*powershell.exe or winlog.event_data.NewProcessName:*pwsh.exe) and not winlog.event_data.ParentProcessName.keyword:*explorer.exe</code>



## Weapon System Training – 2: Queries and Filters

### APPENDIX B: DOMAIN SPECIFIC LANGUAGE (DSL)

#### 4688 – POWERSHELL CONTAINS -EXECUTIONPOLICY BYPASS

<b>Description:</b>	This filter looks for powershell executions with various permutations of the -ExecutionPolicy Bypass commandline arguments, but only if the ParentProcessName does not include explorer.exe.
<b>Query:</b>	<pre>{   "query": {     "bool": {       "filter": [         {           "term": {             "winlog.event_id": {               "value": "4688"             }           }         },         {           "regexp": {             "process.child.command_line.keyword": {               "case_insensitive": true,               "flags": "ALL",               "value": ".*powershell.exe.*-ex([ecutionply]+)?[ \\t]+b.*"             }           }         }       ],       "must_not": {         "wildcard": {           "winlog.event_data.ParentProcessName": {             "case_insensitive": true,             "value": "*explorer.exe"           }         }       }     }   } }</pre>

#### 4688 – COMMANDLINE CONTAINS BASE64 WITH EXCEPTIONS

<b>Description:</b>	This filter looks for any process creation event containing at least 66 continuous Base64 characters in it.
<b>Query:</b>	<pre>{   "query": {     "bool": {       "filter": [         {           "term": {             "winlog.event_id": {</pre>

## Weapon System Training – 2: Queries and Filters

```
        "value": "4688"
      }
    },
    {
      "regexp": {
        "process.child.command_line.keyword": {
          "case_insensitive": true,
          "flags": "ALL",
          "value": ".*[A-Za-z0-9+/=]{65,}.*"
        }
      }
    }
  ],
  "must_not": {
    "term": {
      "process.child.executable": {
        "case_insensitive": true,
        "value": "C:\\Program Files (x86)\\Microsoft\\EdgeUpdate\\MicrosoftEdgeUpdate.exe"
      }
    }
  }
}
```