Chapter 9: Creating a Breach Detection Dashboard in Kibana

Introduction

Event log collection and correlation systems can bring in a ton of data. Our simple lab setup, for example, is pulling in around 1,000,000 events every 24 hours, and that is for a handful of systems and devices. The way we can deal with this much data is by using it to pinpoint areas of concern, events, or trends that seem suspicious. That is the kind of information we want to visualize so that an analyst can quickly assess if something fishy is going on. At this point, they will use all the other data we have been accumulating to find the smoking gun that proves an incident is occurring or as supporting data to perform forensics and incident response activities. The other way we can use this tremendous amount of detailed information is during threat-hunting exercises, which we will cover in Section 3, Threat Hunting.

Throughout this exercise, we will be adding widgets and visualizations to a custom dashboard within Security Onion's Kibana. To get an initial (blank) dashboard started, log in to the Security Onion web portal and open the Kibana tool, then navigate to **https://172.25.100.250/kibana/app/dashboards**. From the Dashboards list page, click on **Create Dashboard**, which will open a blank **Editing New Dashboard**.

We are now ready to start adding widgets and visualizations to the new dashboard (**Breach Detection Portal**).

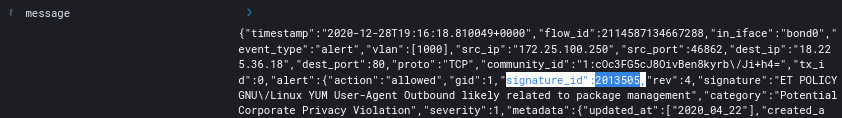
**Submission:** You need to submit a detailed lab report, with screenshots, to describe what you have done and observed. Questions will be defined as you progress through the lab. The lab report will be compiled as a Word document and submitted on Brightspace by **MONTH DAY at TIME AM/PM.**

NIDS Alerts

The first data we want to visualize are NIDS alerts. If tuned properly, NIDS alerts are some of the most definitive indicators of malicious activities or content traversing a network.

Proper tuning of an IDS install is beyond the scope of this book; however, as a rule of thumb, tuning comes down to eliminating noise and false positives. Within Security Onion, you can disable noisy rules by adding their signature identifier (**SID**) under the idstools section in the **/opt/so/saltstack/local/pillar/minions/ind-securityonionv2\_eval.sls** Salt configuration file and run **sudo salt ind-securityonionv2\_eval state.apply idstools**.

As an example, in my lab setup, I was seeing a lot of **ET POLICY GNU/Linux YUM User-Agent Outbound** **likely related to package management alerts**. This is normal traffic in environments with Linux (CentOS) systems. In order to eliminate these false positives from clogging up the database, I added the alerts' SID to the **ind-securityonionv2\_eval.sls** file, under the **idstools** section. You can find the SID for an alert of interest by looking at the detailed alert message, as illustrated in the following screenshot:

*Figure 9.44 – Exercise 6: Suricata alerts – message details*

A SID is a piece of information in the message identified with **signature\_id (2013505)**. This number is what we add to the **idstools** section (**sudo vi /opt/so/saltstack/local/pillar/minions/ind-securityonionv2\_eval.sls**), as illustrated in the following screenshot:

idstools:

config:

ruleset: 'ETOPEN'

oinkcode: ' '

urls:

sids:

enabled:

disabled:

- 2013505

modify:

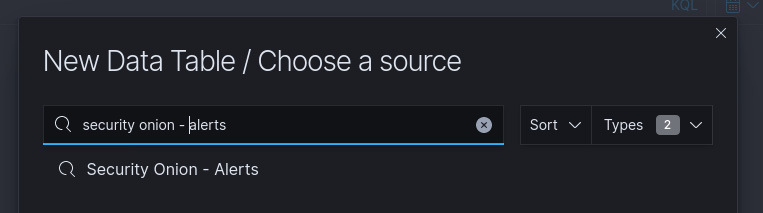
Save the file and run a **sudo salt ind-securityonionv2\_eval state.apply idstools** Salt update command. The Suricata alert with SID **2013505** will no longer be logged.

As a side note, if you obtained a Snort **oinkcode** from https://snort.org/ that allows you to download the community rules from their site, the idstools section is where you would enter that code and have Security Onion pull Snort rules along with **Emerging Threats (ET)** rules.

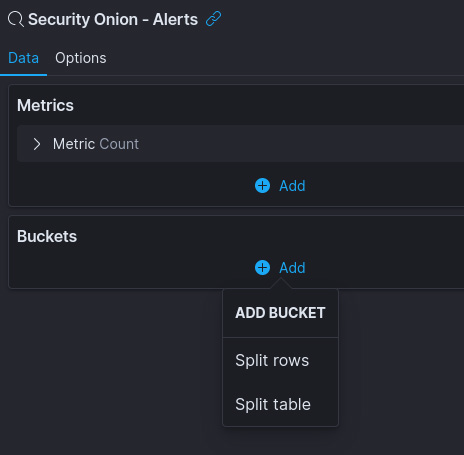
Tuning of IDS rules is a crucial part of creating a usable security monitoring environment. The more benign or bogus alerts our analysts must sift through to get to the real stuff, the more likely it is that they will miss actual alerts. The bulk of any type of signature-based tool deployment is in fine-tuning the rules.

We are going to add a NIDS (Suricata) alert data table visualization to our breach detection dashboard. Follow along with these instructions:

1. Click on the **Create new object** to this dashboard button on the **Editing New Dashboard page**.
2. Select **Data Table** from the **New Visualization** **selection** screen that pops up.
3. Search for **Alerts** in the **New Data Table / Choose a source** screen that follows, and select the **Security Onion – Alerts** data source, as illustrated in the following screenshot:

*Figure 9.45 – Exercise 6: Adding Snort alerts data table*

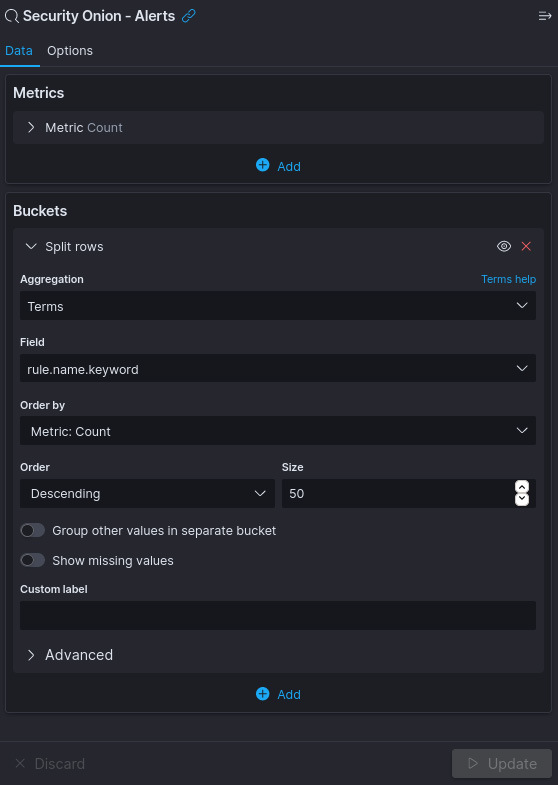
4. We are taken to the visualization editing page. This is where we define what data visualizations (widgets) will display and how they will be displayed. Data is displayed in **buckets**; we need to add a bucket of data for viewing NIDS alert messages. Click on the **ADD BUCKET** button under the **Data** panel on the right of the configuration screen, and select **Split rows**, as illustrated in the following screenshot:



*Figure 9.46 – Exercise 6: Snort alerts data table – adding a data bucket*

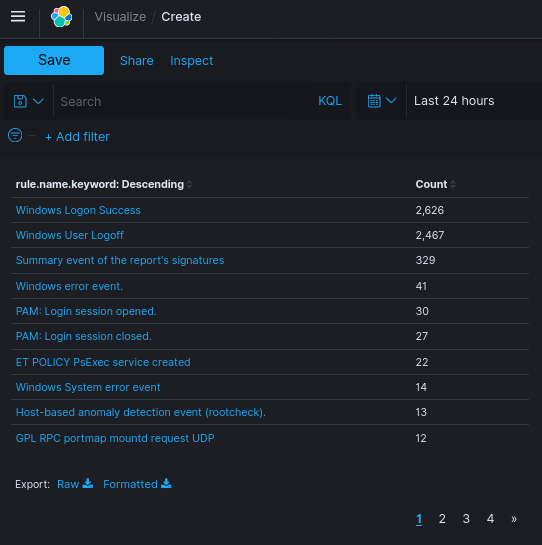
5. Under the **Split rows** panel, select **Terms** from the **Aggregation** drop-down menu and for **Field** (the database field we want the data from), search for and select **rule.name.keyword**. Finally, change the **Size** setting to **50** to show the 50 most often occurring alerts.

The following screenshot shows how this is done:



*Figure 9.47 – Exercise 6: Configuring the Snort alerts data table data bucket*

6. When you click **Update**, the data table will populate on the left of the screen, showing the results for the data we just configured, as illustrated in the following screenshot:



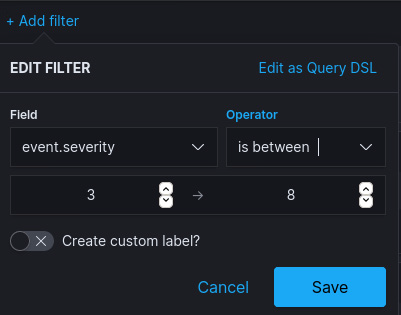
*Figure 9.48 – Exercise 6: Snort alerts widget*

If the Snort engine is well tuned, this view should show very little output. The reason mine is showing a large list is because I am displaying data from the last 7 days, as well as having done some network scanning and probing to have the data show up here as a pretty picture.

Our intention is to create a dashboard with data that is indicative of attacks in progress—in other words, data that can concisely and accurately tell us we are under attack. Having a bunch of false positives show up on any of the widgets on this page is not going to give us that definitive alert overview.

7. To trim down some of the noisy alerts, click on the minus (-) symbol (filter out) next to a **rule.name** when you hover over an alert you want to suppress. You can come back later (**Edit visualization**) to add any annoying noisy alerts.

8. Additionally, we can filter on event severity. To do this, click on the **Add Filter** button on the top of the editor screen and set **Field** to **event.severity**. Now, set **Operator** to **is between** and enter the values **3** and **8** for the bottom and top values, as illustrated in the following screenshot:



*Figure 9.49 – Filtering on severity*

Click on **Save** to apply the filter.

9. With that, we just created a data table that shows us a summary of the NIDS alerts seen within the timeframe we specify (**7 days**). Click on **Save**, specify the name for this visualization as **Breach Detection – NIDS Alerts Summary**, and click on **Save and return**. This saves the visualization and adds it to our new dashboard.

10. We can resize and move the visualization to our liking while we are in **Edit Mode** for the dashboard.

11. Save the dashboard with the Save button on the top left of the editor screen, and name our new creation **Breach Detection Portal**.

We just created our initial breach detection dashboard, which for the moment only shows the NIDS alerts, but we will be adding some more data here soon, starting with Zeek notices next.

Zeek notices

Zeek notices logs are Zeek's method of alerting us on interesting and pressing findings. An example notice would be the discovery of a log tied to a defined intel threat. This is a major finding and needs to be alerted on.

We can add notices by following these instructions:

1. Navigate to our **Breach Detection** dashboard (**Kibana** | **Dashboards** | **Breach Detection**) and click on **Edit** in the top left of the dashboard screen.

2. Click on the **Create New** button to add a new widget (visualization).

3. Select the **Data Table** visualization type and search for and add the **\*:so-\*** data source (this is the data source for all logs).

4. Enter an **event.dataset:notice AND event.module: zeek** search term to filter out notices logs only. Hit **Refresh**.

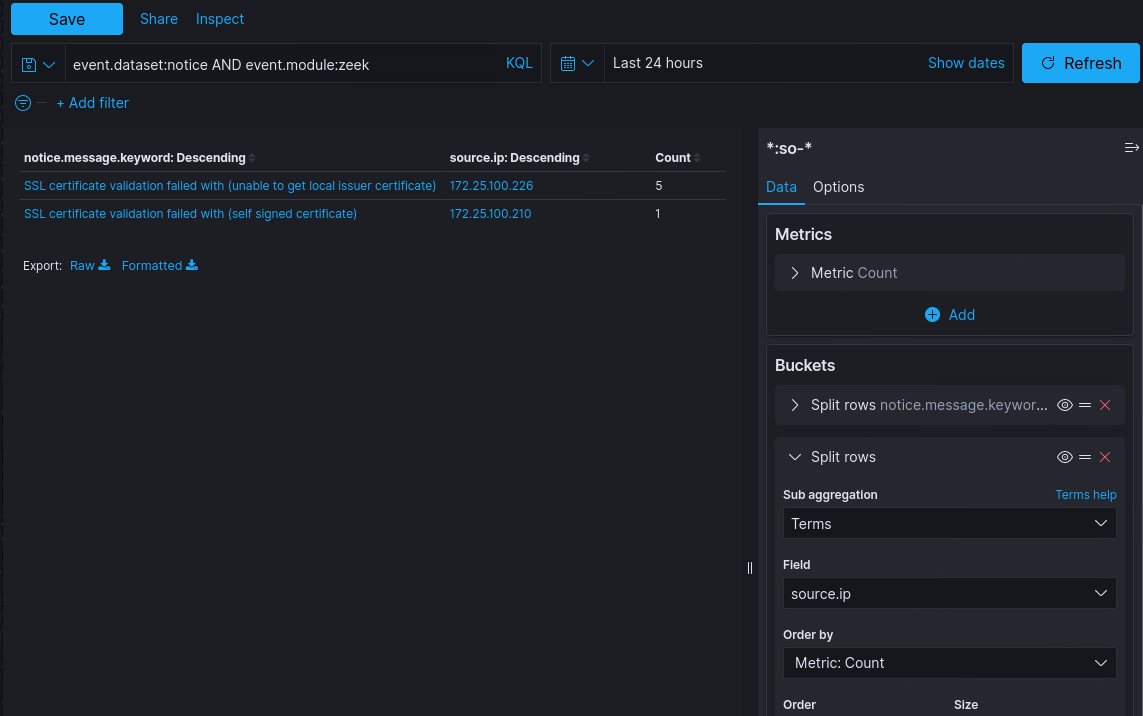
5. Add a **Split Rows** data bucket and set **Sub aggregation** to **Terms**, with **notice.message.keyword** as the **Field** selection.

6. Set the **Size** option to **50** and click **Update**.

7. We will be adding the source IP address to show with the notice messages. To do this, add an additional **Split Rows** data bucket and set **Sub aggregation** to **Terms**, with **source.ip** as the **Field** selection.

8. Set the **Size** option to **50** and click **Update**.

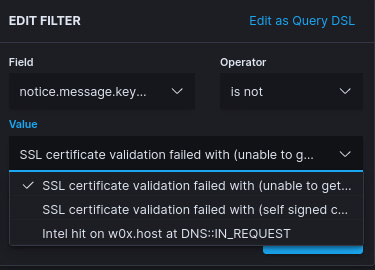
The following screenshot shows how this should be done:



*Figure 9.50 – Exercise 6: Zeek notices – creating a visualization*

By default, Zeek flags self-signed certificates as invalid and creates a notice for them. Although this is a significant finding for IT environments, on the OT side we are less concerned with this discovery. The shielded architecture of a typical industrial environment makes the management of a **public-key infrastructure (PKI)** difficult to maintain, and hence certificate errors are not uncommon. Ignoring these certificate errors in our **Breach Detection** dashboard view is OK. We will not eliminate them from the database, just hide them from our view.

9. To exclude **SSL certificate validation failed with (unable to get local issuer certificate)** notices from our view, hover your mouse over the area just to the left of the notice count (**14**), and click on the – (minus) symbol that pops up **Filter out value**. Click on this button. Alternatively, you can click on the **Add filter** button along the top of the screen. In the **Add filter** pop-up screen, type in **notice.message.keyword** for **Field**, set the **operator** to **is not**, and select **SSL certificate validation failed with (unable to get local issuer certificate)** from the **Value** dropdown. Click **Save**. The following screenshot illustrates this process:



*Figure 9.51 – Exercise 6: Zeek notices – adding data filter*

10. In the same way, filter out any other **Secure Sockets Layer (SSL)** certificate notices or notices that clutter the view.

11. The result is a Zeek notice logs view, with minimal excessive alerts, to the point and streamlined. Click on Save, saving the widget as **Breach Detection – Zeek Notices Summary.**

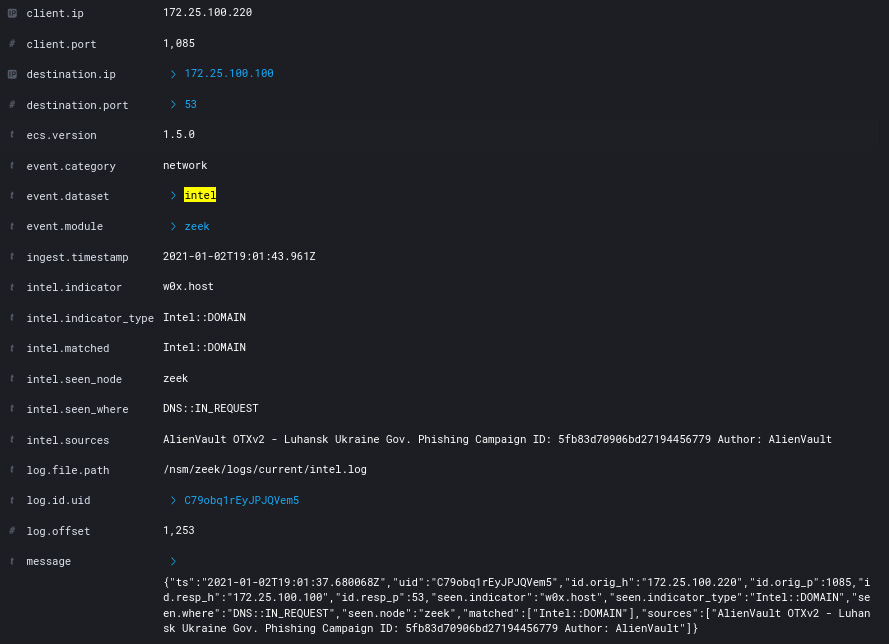
Next, we are going to add Zeek intel logs to the **Breach Detection** dashboard.

Zeek Intel logs

Zeek Intel logs are created when Zeek correlates a discovered artifact with a corresponding entry in the Intel file. In Chapter 8, Industrial Threat Intelligence, we added the **AlienVault** threat information feed to Security Onion. Have a look at some entries in that feed file (**intel.dat**) here:

 *Figure 9.52 – Exercise 6: Zeek Intel logs – Intel file*

We can see there is an entry in the Intel file that correlates the (**Intel::DOMAIN**) **w0x.host** domain name with **Luhansk Ukraine Gov. Phishing Campaign.** Now, when Zeek comes across the **w0x.host** domain name while monitoring network traffic, it creates an Intel log entry, as is the case here:



*Figure 9.53 – Exercise 6: Zeek Intel – Intel log details*

Depending on the quality of the threat feed going into the Intel data file, Intel logs can be a very clear IOC, or at least an indicator of very suspicious activity on your network. We can add the Zeek Intel log to our Breach Detection dashboard by following these instructions:

1. Navigate to our **Breach Detection** dashboard (**Kibana** | **Dashboards** | **Breach Detection**) and click on **Edit** in the top left of the dashboard screen.

2. Click on the **Create New** button to add a new widget (visualization).

3. Select the **Data Table** visualization type and search for and add the **\*:so-\*** data source (this is the data source for all logs).

4. Enter an **event.dataset:intel** search term to filter out notice logs only. Hit **Refresh.**

5. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **intel.sources.keyword** as the **Field** selection.

6. Set the **Size** option to **50** and click **Update**.

7. We will be adding the source IP address to show with the Intel notice messages. To do this, add an additional Split Rows data bucket and set **Aggregation** to **Terms**, with **source.ip** as the **Field** selection.

8. Set the **Size** option to **50** and click **Update**.

9. Click on **Save**, saving the widget as **Breach Detection – Intel Logs Summary.**

Next, we are going to look at suspicious processes with Sysmon logs.

Suspicious process and file creation

Creation of files and starting of processes is a very common practice on Windows machines. However, there are a few indicators that alert on foul play when creating processes or creating (saving files). One of those indicators is the location an executable is started from or saved to. By filtering out most of the common locations for these actions, we can generate a view that shows unusual executable paths (locations).

Follow these instructions to create a visualization that allows us to pinpoint suspicious process creation:

1. Navigate to our **Breach Detection** dashboard (**Kibana** | **Dashboards** | **Breach Detection**) and click on **Edit** in the top left of the dashboard screen.

2. Click on the **Create New** button to add a new widget (visualization).

3. Select the **Data Table** visualization type and search for and add the **Security Onion Sysmon** data source.

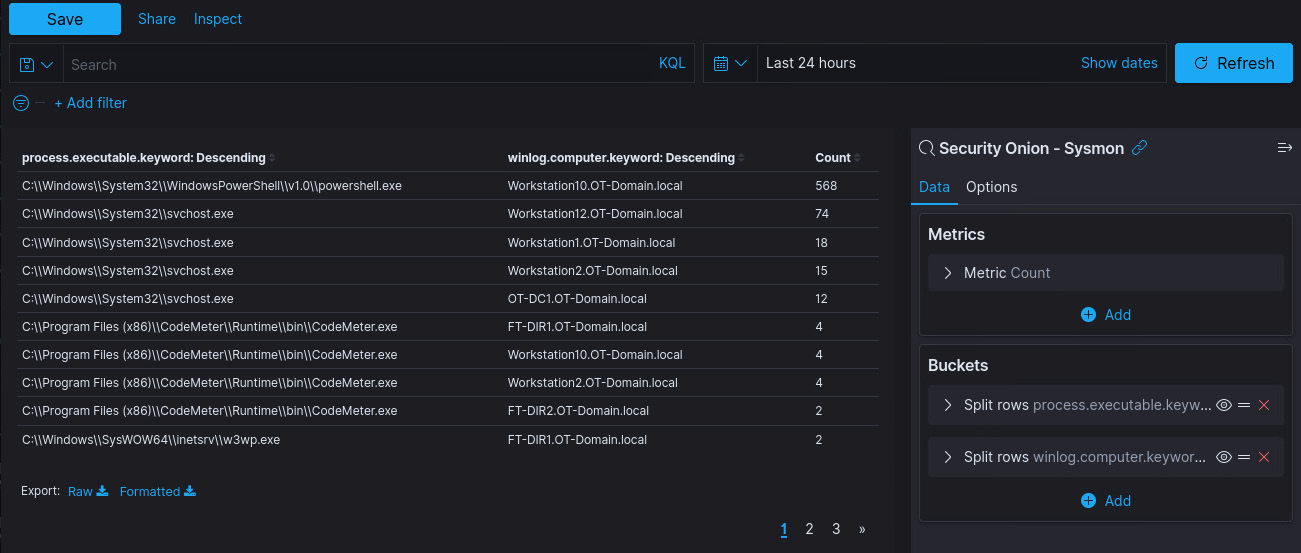
4. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **process.executable.keyword** as the **Field** selection.

5. Set the **Size** option to **50** and click **Update**.

6. We will be adding the source computer name to show this with the process event entries. To do this, add an additional **Split Rows** data bucket and set **Aggregation** to **Terms**, with **winlog.computer.keyword** as the **Field** selection.

7. Set the **Size** option to **50** and click **Update**.

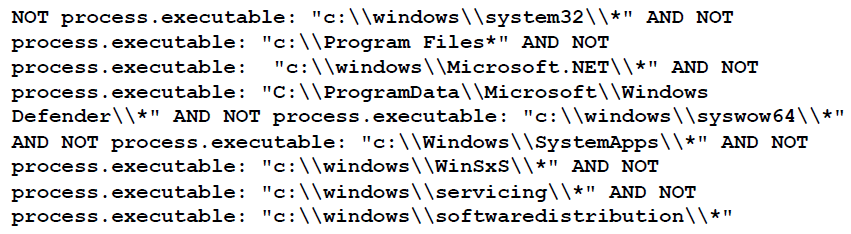
The following screenshot shows how this is done:



*Figure 9.54 – Exercise 6: Suspicious process creation – all logs*

At this point, we have created a view that shows a summary of the location (**image\_path**) of every process that is created on all the endpoints in our environment. Most of these will be legitimate actions. What we are going to do is filter out all the usual locations processes should start from, such as **c:\windows\system(32), c:\Program Files,** and so on. Let's proceed, as follows:

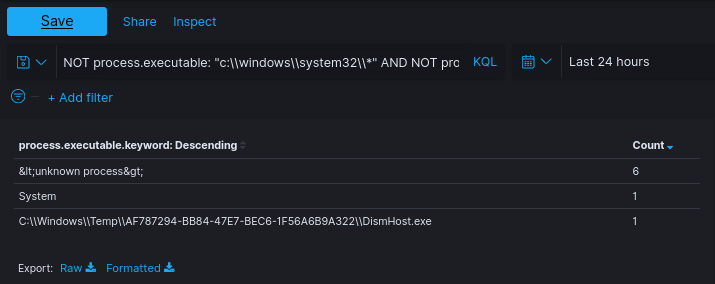
8. Enter the following code as the search query in the **Visualization Editing** page:



This is a good starting list of common locations that Windows executable files start from. Over time, you will probably come across additional ones, and this is the spot to add them.

9. Click **Refresh** and eliminate any other obvious known safe executables (such as **c:\windows\explorer.exe)** by using the **Filter out value** button. This is how you can fine-tune the view going forward.

10. What remains should be a very cut-down output of processes that started from executables in weird places, as illustrated in the following screenshot:



*Figure 9.55 – Exercise 6: Suspicious process creation – filtered view*

Processes that start from the **c:\windows\temp** location are always suspicious but not always malicious!

11. Click on **Save**, saving the widget as **Breach Detection – Suspicious Image Paths**.

Next, we are going to add PowerShell Script Block Logging to show interesting PowerShell activity in our environment.

Suspicious PowerShell commands

PowerShell is a very convenient and—well—powerful way to manage Windows. Many attackers adapting known tools build these around the scripting language. This makes attacks stealthier—as no external executables need to be run—and also convenient, as PowerShell is installed on all modern Windows versions by default.

In Exercise 2 – Using Wazuh for PowerShell Script Block Logging, we added PowerShell logging capabilities to our security monitoring tool bag. We will now see how we can leverage that functionality to detect potentially dangerous commands within PowerShell. For this, we are going to add a data table that only displays commands that can be potentially harmful. Follow these instructions to get a data table created:

1. Navigate to our **Breach Detection** dashboard (**Kibana** | **Dashboards** | **Breach Detection**) and click on **Edit** in the top left of the dashboard screen.

2. Click on the **Create New** button to add a new widget (visualization).

3. Select the **Data Table** visualization type and search for and add the **\*:so-\*** data source.

4. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **winlog.event\_data.scriptBlockText.keyword** as the **Field** selection.

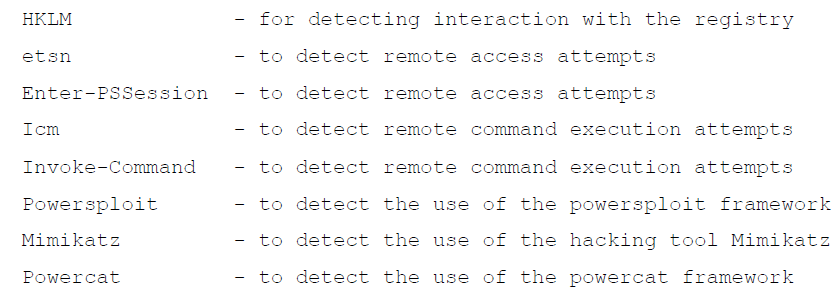
5. Set the **Size** option to **50** and click **Update**.

6. We will be adding the source computer name to show this with the ScriptBlock event entries. To do this, add an additional **Split Rows** data bucket and set **Aggregation** to **Terms**, with **winlog.computer.keyword** as the **Field** selection.

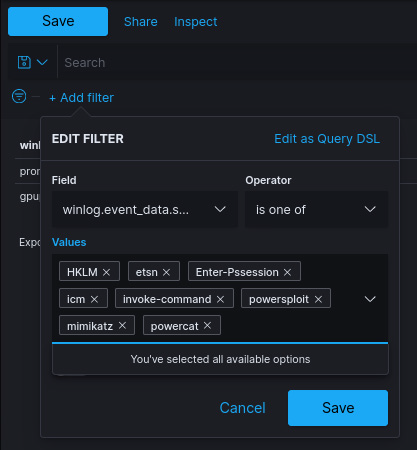
7. Set the **Size** option to **50** and click **Update**.

8. This generates a complete list of PowerShell Script Block logs, grouped by computer name. We are only going to filter for the interesting ones. Click on the **Add Filter** button and set the **Field** option to **winlog.event\_data.scriptBlockText** and the **Operator** option to **is one of**.

9. Now, one at a time, add in the following values and click **Save** when finished:



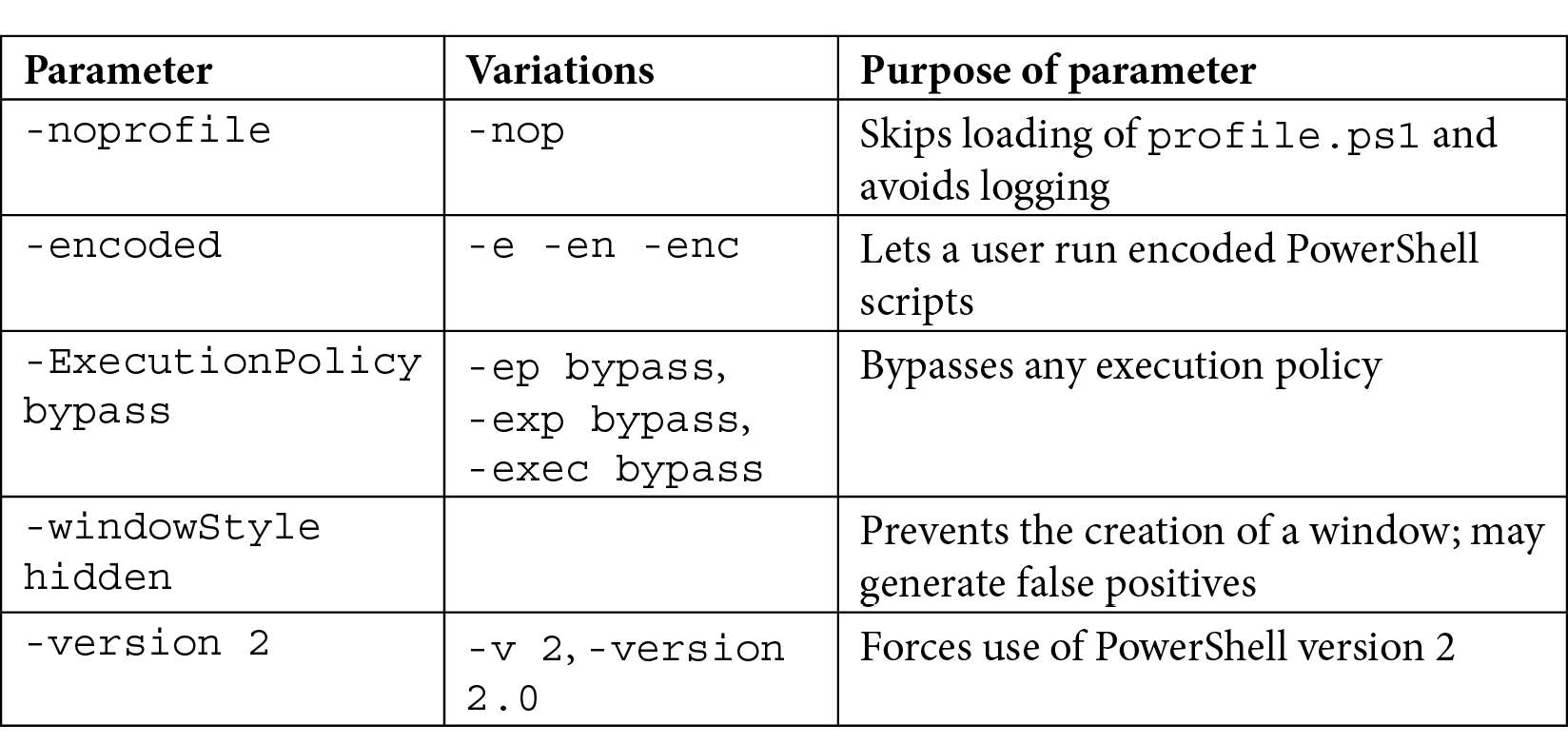
Again, this is just a list to get you started, so do some research to find out the commands that are most often used in PowerShell attacks and then add them here, as shown:



*Figure 9.56 – Exercise 6: Suspicious PowerShell commands*

10. Click on **Save**, saving the widget as **Breach Detection – Suspicious PowerShell Commands.**

The way PowerShell is started can give away any malicious intent or intended abuse as well. The following table summarizes some suspicious ways PowerShell can be invoked (started):

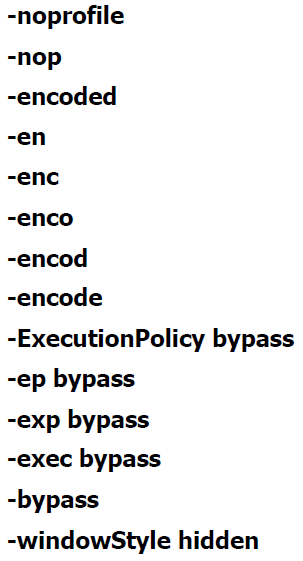


Detecting suspicious invocation of the PowerShell engine can show malicious intent. We will add a data table that lists any process creation logs that contain these suspicious command-line parameters. The logs this data is in are provided by the Sysmon logging engine that we deployed earlier, in Exercise 1 – Using Wazuh to add Sysmon logging. Follow these steps to get the data table added to our dashboard:

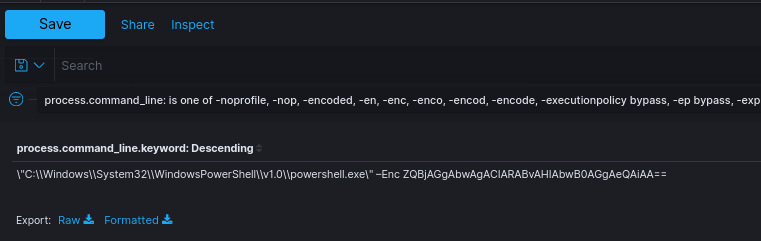
1. Navigate to our **Breach Detection** dashboard (**Kibana** | **Dashboards** | **Breach Detection**) and click on **Edit** in the top left of the dashboard screen.
2. Click on the **Create New** button to add a new widget (visualization).
3. Select the **Data Table** visualization type and search for and add the **Security Onion Sysmon** data source.
4. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with

**process.command\_line.keyword** as the **Field** selection.

1. Set the **Size** option to **50** and click **Update**.
2. We will be adding the source computer name to show this with the PowerShell invocation event entries. To do this, add an additional **Split Rows** data bucket and set **Aggregation** to **Terms**, with **winlog.computer.keyword** as the **Field** selection.
3. Set the **Size** option to **50** and click **Update**.
4. We will be adding the username that started the PowerShell process to show with the PowerShell invocation event entries. To do this, add an additional **Split Rows** data bucket and set **Aggregation** to **Terms**, with the **user.name.keyword** as the **Field** selection.
5. Set the **Size** option to **50** and click **Update**.
6. This generates a complete list of command-line logs. We are only going to filter for the interesting ones. Click on the **Add Filter** button and set the **Field** option to **process.command\_line** and the **Operator** option to **is one of**.
7. Now, one at a time, add in the following values and click **Save** when finished:



The next screenshot shows the configured widget:



*Figure 9.57 – Exercise 6: Suspicious PowerShell invocation*

12. Click on **Save**, saving the widget as **Breach Detection – Suspicious PowerShell Invocation**.

We now have a visualization around suspicious PowerShell commands and how PowerShell is invoked. Next, we are going to look at suspicious network connections.

Suspicious Egress Connections

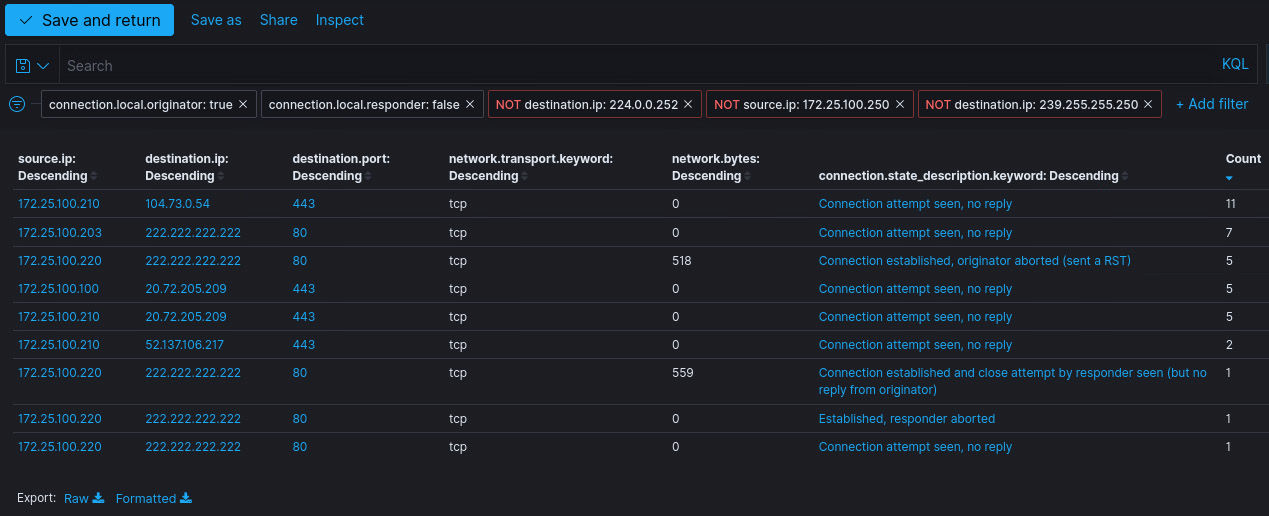
Seeing repeated connections to an external address (the Enterprise Zone or internet) can be an indication of **bot** activity on the network or exfiltration of data from your internal network (the Industrial Zone). Bots are types of malware that turn the hosts they infect into **zombie** machines, mindless drones to a master (a **command and control** (**C&C**) server, run by a **botmaster**), performing actions on behalf of that master. Bots need to make regular connections back to their C&C server to check in with the botmaster, asking for instructions or updates. We can detect these check-ins by monitoring for repetitive connections between an internal and an external IP address. Often, these connections are of the same type and size and are aimed at the same IP address. What we will do next is add a visualization that quickly depicts this typical type of connection in a summary table. To add a data table that summarizes outbound established connection counts, follow these instructions:

1. Navigate to our **Breach Detection** dashboard (**Kibana** | **Dashboards** | **Breach Detection**) and click on **Edit** in the top left of the dashboard screen.
2. Click on the **Create New** button to add a new widget (visualization).
3. Select the **Data Table** visualization type and search for and add the **Security Onion – Connections** data source.
4. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **source.ip** as the **Field** selection. Set the **Size** option to **50**.
5. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **destination.ip** as the **Field** selection. Set the **Size** option to **50**.
6. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **destination.port** as the **Field** selection. Set the **Size** option to **50**.
7. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **network.transport.keyword** as the **Field** selection. Set the **Size** option to **10**.
8. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **network.bytes** as the **Field** selection. Set the **Size** option to **50**.
9. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **connection.state\_description.keyword** as the **Field** selection. Set the **Size** option to **50**.
10. This generates a summary view of all connection logs. We want to view established connections, but egress only. In order to do this, add the following filters:

* Filter the data table on **connection.local.originator: true**—this filters the view to show connections originating from the local network (Industrial Zone) only.
* Filter the data table on **connection.local.responder: false**— this filters the view to show connections to external destinations (enterprise and internet) only.

These two filters will eliminate connection *attempts* and effectively hide non- established (non-successful) connections from this view, but it cuts down on noise significantly, and arguably, we are only looking for ongoing incidents and eminent risk in our breach detection view. Finding lingering malware is a function of threat hunting, something we will cover in *Section 3* of this book: *Threat Hunting*.

At this point, we have a filtered connections summary view, as illustrated in the following screenshot, showing us the count of connections between the same source and destination IP, over the same network protocol, using the same service, and having the same total connection size. We should filter out the broadcast address, IP addresses that have a legitimate reason to make egress connections, and other fluff, to clean up the view as much as possible:



*Figure 9.58 – Exercise 6: Suspicious egress connections*

11. Make sure to sort by **Count** (descending) and click on **Save**, saving the widget as **Breach Detection – Suspicious Egress Connections**.

Next, we are going to reverse our logic and view suspicious connections originating from outside the industrial network.

Suspicious ingress connections

Another good indicator based around network connections is ingress connection detection. Especially in an industrial environment, external connections to assets on the ICS network are suspicious, even more so if they originate from the internet. We will now create a visualization that displays an ingress network connections summary. Follow these instructions to create a data table that summarizes external connections into an industrial network:

1. Navigate to our **Breach Detection** dashboard (**Kibana** | **Dashboards** | **Breach Detection**) and click on **Edit** in the top left of the dashboard screen.
2. Click on the **Create New** button to add a new widget (visualization).
3. Select the **Data Table** visualization type and search for and add the **Security Onion – Connections** data source.
4. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **@timestamp** as the **Field** selection. Set the **Size** option to **50**.
5. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **source.ip** as the

**Field** selection. Set the **Size** option to **50**.

1. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **destination.ip** as the **Field** selection. Set the **Size** option to **50**.
2. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **destination.port**

as the **Field** selection. Set the **Size** option to **50**.

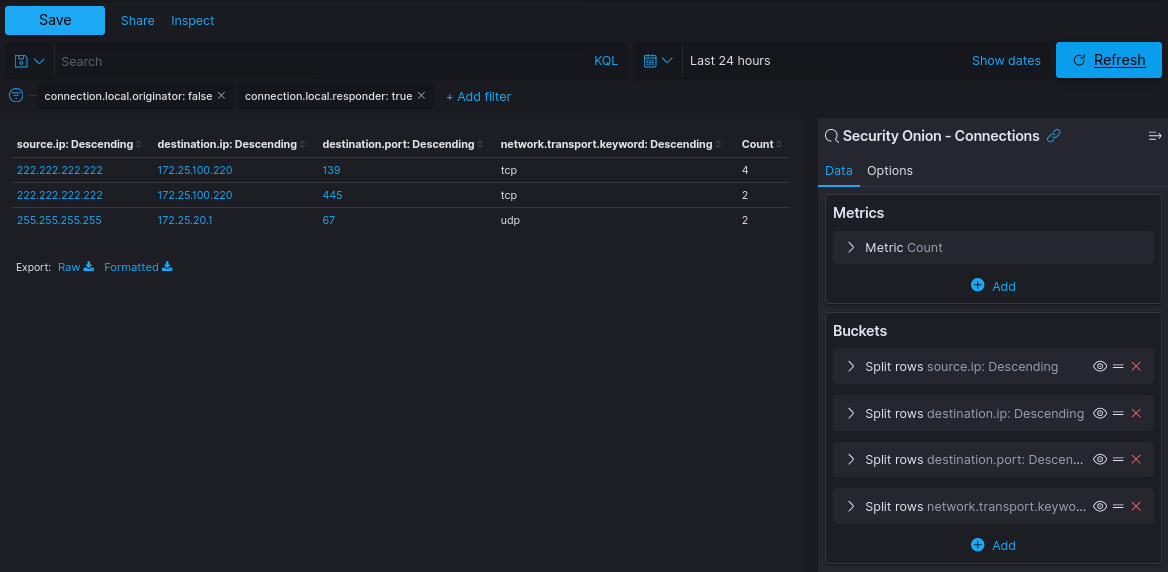
1. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with

**network.transport.keyword** as the **Field** selection. Set the **Size** option to **50**.

1. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **event.duration** as the **Field** selection. Set the **Size** option to **50**.
2. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **network.bytes** as the **Field** selection. Set the **Size** option to **50**.
3. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **connection.state\_description.keyword** as the **Field** selection. Set the **Size** option to **10**.
4. This generates a summary view of all connection logs. We want to view established connections, but egress only. In order to do this, add the following filters:

* Enter the following search string for the view: **NOT source.ip: 172.25.100.0/24**. This will filter out any IP addresses that are local to the industrial network (adapt this to your subnet).
* Filter the data table on **connection.local.responder: true**—this filters the view to show connections to local destinations (Industrial Zone) only.

13. Seeing as we are dealing with the industrial network, we should see few to no external connections at all in the data table view we just created. As a matter of fact, seeing connections pop up here is extremely suspicious. If you do have enterprise systems that have legitimate reasons to connect to industrial assets, you can whitelist those by filtering them out. You can see a sample output in the following screenshot:



*Figure 9.59 – Exercise 6: Suspicious ingress connections*

14. Make sure to sort by **Duration** (descending) and click on **Save**, saving the widget as **Breach Detection – Suspicious Ingress Connections.**

Next, we are going to look at some user account activity.

Failed user login attempts

Failed user login attempts happen all the time. People fat-finger their password, forget their password, or are locked out for some reason. This is normal activity for any network; however, certain patterns in failed login attempts stand out. Excessive login attempts from or to a certain IP address (host) are almost guaranteed to be malicious. As many network administrators have implemented account lockout procedures for excessive login attempts, attackers have changed their approach to only try for an X amount of time per machine, then move on to the next. For this, a few failed login attempts, from a single machine to several other machines, should also be considered suspicious.

To capture both types of login attacks (brute-force attacks), we are going to create a visualization that trends failed login attempts over time. Follow along with these instructions to create a graph widget:

1. Navigate to our **Breach Detection** dashboard (**Kibana** | **Dashboards** | **Breach Detection**) and click on **Edit** in the top left of the dashboard screen.
2. Click on the **Create New** button to add a new widget (visualization).
3. Select the **Vertical Bar** visualization type and search for and add the **Security Onion – Alerts** data source.
4. Add a **Split Series** data bucket and set **Aggregation** to **Terms**, with **winlog.event\_data.targetUserName.keyword** as the **Field** selection. Set the **Size** option to **50**.
5. Add an **X-axis** data bucket and set **Aggregation** to **Data Histogram**, and then click **Update**.
6. This sets the aggregation as a timeline display, viewing all alerts' target username logs over time.
7. Now for the magic: click on the **Add Filter** button, then set the **Field** option to **event.code** and the **Operator** option to **is one of**. Set the values to **4625**, **4771**, and **539**, three well-known Windows event IDs that indicate failed login attempts. A sample output is shown in the following screenshot:



*Figure 9.60 – Exercise 6: Failed login attempts*

There are more event IDs that could indicate a failed login attempt, so I will leave it as a homework task for you to research additional event IDs that are related to failed logins and add them here (think domain login versus local login).

8. Click on **Save**, saving the widget as **Breach Detection – Failed Login Attempts over time**.

We now have a very convenient view of failed login attempts over time. Spikes in the graph indicate activity, and with us pulling logs from several machines, the spike can indicate either of the two attack scenarios we discussed at the beginning of this section.

Next, we will look at user creation and change logs.

New user creation and changes to user accounts

We will now visualize alerts around new user creation and changes to user accounts. User creation or changes to a user account are great indicators of foul play in an environment, and alerts around these actions should be kept a close eye on. Follow along with these instructions to get a data table widget created that shows new or changed user account alerts:

1. Navigate to our **Breach Detection** dashboard (**Kibana** | **Dashboards** | **Breach Detection**) and click on **Edit** in the top left of the dashboard screen.
2. Click on the **Create New** button to add a new widget (visualization).
3. Select the **Data Table** visualization type and search for and add the **Security Onion – Alerts** data source.
4. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with

**rule.name.keyword** as the **Field** selection. Set the **Size** option to **50**.

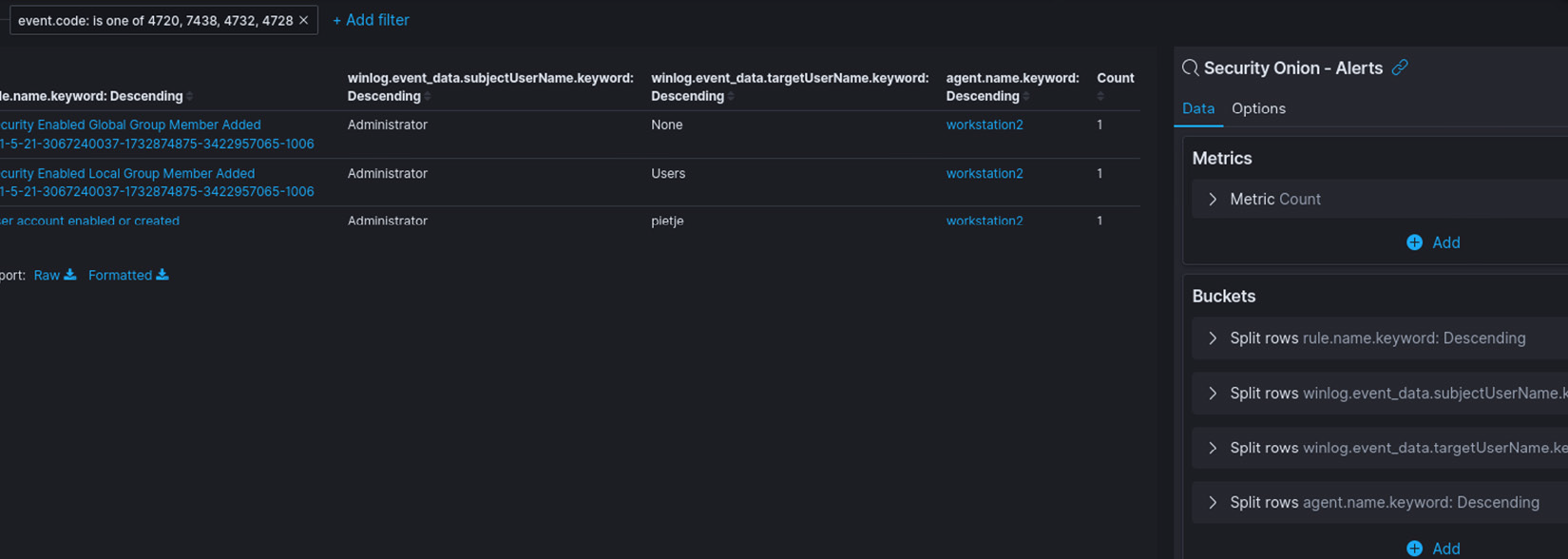
1. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **winlog.event\_data.subjectUserName.keyword** as the **Field** selection. Set the **Size** option to **50**.
2. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **winlog.event\_data.targetUserName.keyword** as the **Field** selection. Set the **Size** option to **50**.
3. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with

**agent.name.keyword** as the **Field** selection. Set the **Size** option to **50**.

1. Click **Update** to apply data buckets.
2. Now for the magic: click on the **Add Filter** button, then set the **Field** option to

**event.code** and the **Operator** option to **is one of**. Set the values to the following:

* **4720**—This is the event ID for new user creation
* **4738**—This is the event ID for account changes
* **4732**—This is the event ID indicating a user was added to a local privileged group
* **4728**—This is the event ID indicating a user was added to a global privileged group
* **4722**—This is the event ID indicating a user account was enabled You can see a sample output in the following screenshot:



*Figure 9.61 – Exercise 6: User account alerts summary*

10. Click on **Save**, saving the widget as **Breach Detection – User Account Alerts Summary.**

So far, we have looked at alerts, processes, and users. Next, we are going to look at files that were downloaded.

Downloaded files

The last visualization we will add to our **Breach Detection** dashboard will show us files downloaded from the internet. We will be creating a visualization around the **bro\_http** logs, showing us interesting **Uniform Resource Identifier** (**URI**) strings such as **.exe**, **.bat**, **.cmd**, and so on. Follow these instructions to build the data table widget:

1. Navigate to our **Breach Detection** dashboard (**Kibana** | **Dashboards** | **Breach Detection**) and click on **Edit** in the top left of the dashboard screen.
2. Click on the **Create New** button to add a new widget (visualization).
3. Select the **Data Table** visualization type and search for and add the **\*:so-\*** data source (this is the data source for all logs).
4. Enter an **event.dataset:http** search term to filter out HTTP logs only, then hit **Update**.
5. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **source.ip** as the

**Field** selection.

1. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with

**http.virtual\_host.keyword** as the **Field** selection.

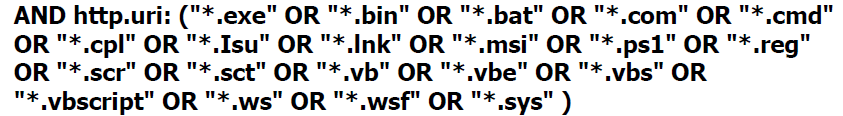
1. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **http.uri.keyword**

as the **Field** selection.

1. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with

**http.user\_agent.keyword** as the **Field** selection.

1. Set the **Size** option to **50** and click **Update**.
2. We will now search for interesting (executable) file extensions. Enter the following search query after the existing search term:



11. Click on **Save**, saving the widget as **Breach Detection – Download of Executable Files**.

There are many more widgets we could add, but for the sake of brevity we will leave it at these fundamental ones.

SilentDefense alerts

The final alert summary widget we will cover in this exercise is around showing SilentDefense alerts. We want to show interesting events that our **OT-IDS** alert is catching. To do this, follow this procedure:

1. Navigate to our **Breach Detection** dashboard (**Kibana** | **Dashboards** | **Breach Detection**) and click on **Edit** in the top left of the dashboard screen.
2. Click on the **Create New** button to add a new widget (visualization).
3. Select the **Data Table** visualization type and search for and add the **\*:so-\*** data source.
4. Enter an **event.dataset:ot-ids** search term to filter out SilentDefense logs only, then hit **Update**.
5. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **source.ip** as the **Field** selection. Set the **Size** option to **50**.
6. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **rule.category.keyword** as the **Field** selection. Set the **Size** option to **50**.
7. Add a **Split Rows** data bucket and set **Aggregation** to **Terms**, with **rule.name.keyword** as the **Field** selection. Set the **Size** option to **50**.
8. Click **Update** to apply data buckets.
9. Filter out any noisy (uninteresting) alert categories such as **NameResolution** and alerts such as **FileRead**, to clean up the view.
10. Click on **Save**, saving the widget as **Breach Detection – SilentDefense Alerts Summary.**

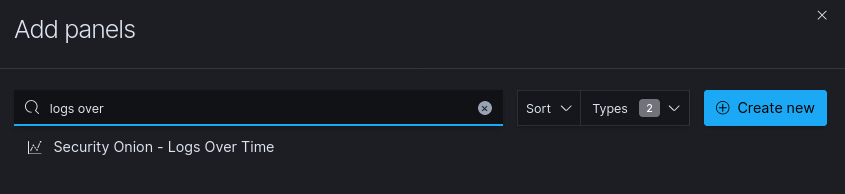
We now have a widget that shows us interesting OT-IDS alerts.

There are many more widgets we could cover here, and additionally, there are many more views, columns, and filters we could apply to customize the overall **Breach Detection** dashboard. Over time, when you start using this information, you will adapt the views and data to fit your needs. For now, you should have a solid knowledge of Security Onion and Kibana's capabilities to get you started. Let's finish up the dashboard with some standard widgets that should be present on every dashboard.

Finishing up the dashboard

To finish up our dashboard, we are going to add some visualizations that should be part of every dashboard. The first one is a **log count-over-time bar graph**, showing us the log disperse over the time period we set our dashboard for. To add this visualization, follow these steps:

1. Navigate to our **Breach Detection** dashboard (**Kibana** | **Dashboards** | **Breach Detection**) and click on **Edit** in the top left of the dashboard screen.
2. Click on the **Add** button at the top of the dashboard and search for and add the **Security Onion - Logs Over Time** widget, as illustrated in the following screenshot:

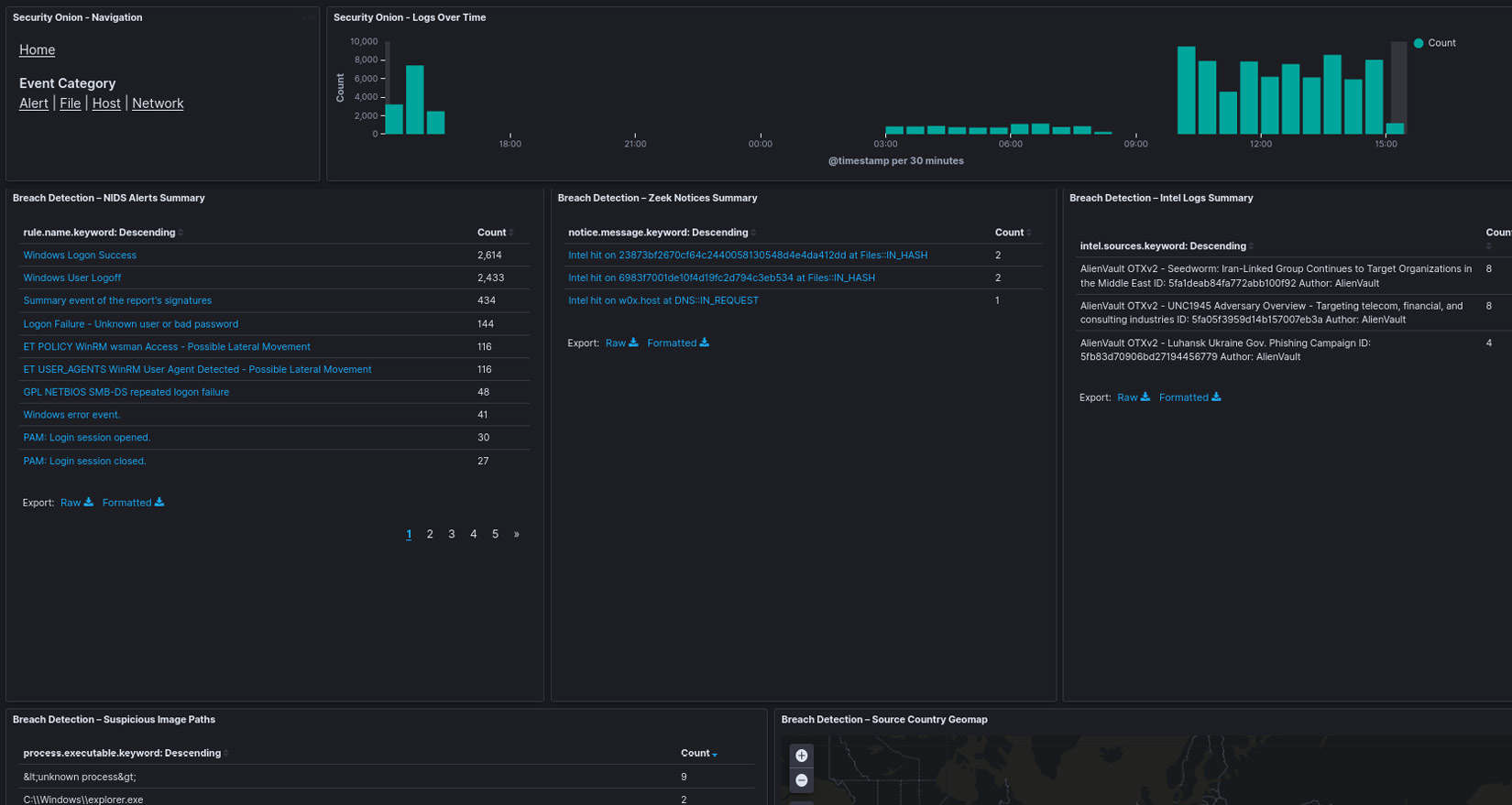


*Figure 9.62 – Exercise 6: Finishing up – adding log count over time*

1. Close the **Add Panels** screen and drag the newly added visualization to the top of the screen.

The second widget we are going to add is a **navigation side panel** that allows us to get to Security Onion dashboards quickly. To add this visualization, follow these steps:

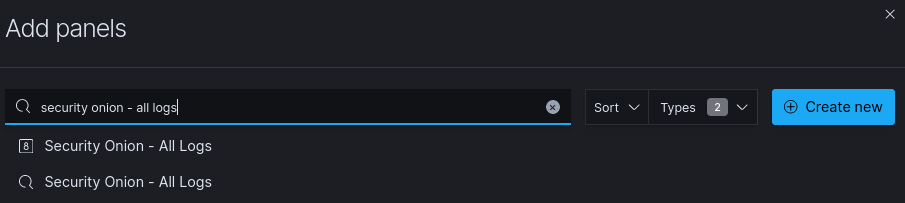
1. Navigate to our **Breach Detection** dashboard (**Kibana** | **Dashboards** | **Breach Detection**) and click on **Edit** in the top left of the dashboard screen.
2. Click on the **Add** button at the top of the dashboard and search for and add the **Security Onion – Navigation** panel.
3. Close the **Add Panels** screen and drag the newly added visualization to the top left of the screen, right next to the log count-over-time widget. Fit all the other visualizations to your liking. A sample output is shown in the following screenshot:



*Figure 9.63 – Exercise 6: Overview of a possible layout of the finished dashboard*

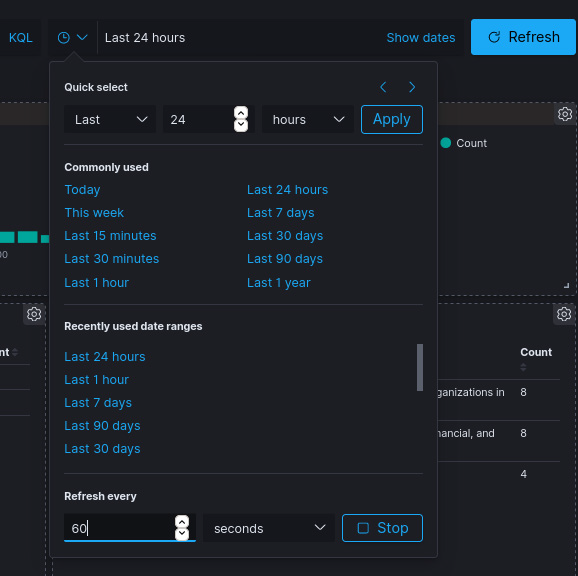
One final visualization we will add is a **log viewer**.

Click on **Add** and search for and add the **Security Onion – All Logs** saved search. This adds a panel way at the bottom that displays the raw log entries, as illustrated in the following screenshot:



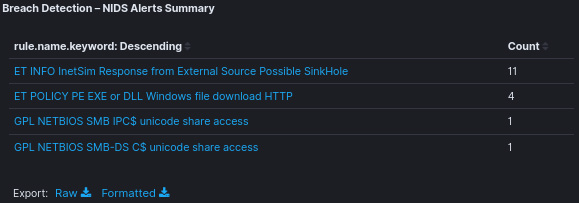
*Figure 9.64 – Exercise 6: Finishing up – adding an event viewer*

Open one of the logs and add any interesting data field to the table view by clicking on the **Toggle column in table** button that appears when you hover over a field. Once you are happy with how things look, save the dashboard and pat yourself on the shoulder—we now have created a dashboard that allows us to quickly assess the security state of our environment. We should set the dashboard to show a 24-hour timespan and automatically refresh every minute, as illustrated in the following screenshot:



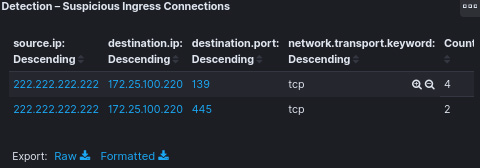
*Figure 9.65 – Exercise 6: Finishing up – automatic refresh*

So, now, when an interesting piece of data catches our eye, we can interactively look for information surrounding the artifact. For example, the IP address **222.222.222.222** has been popping up all over my dashboard. I can filter on that IP address by hovering over it and clicking Filter for value. The entire dashboard will now show data filtered on **source\_ip: 222.222.222.222**. We can see NIDS alerts in the following screenshot:



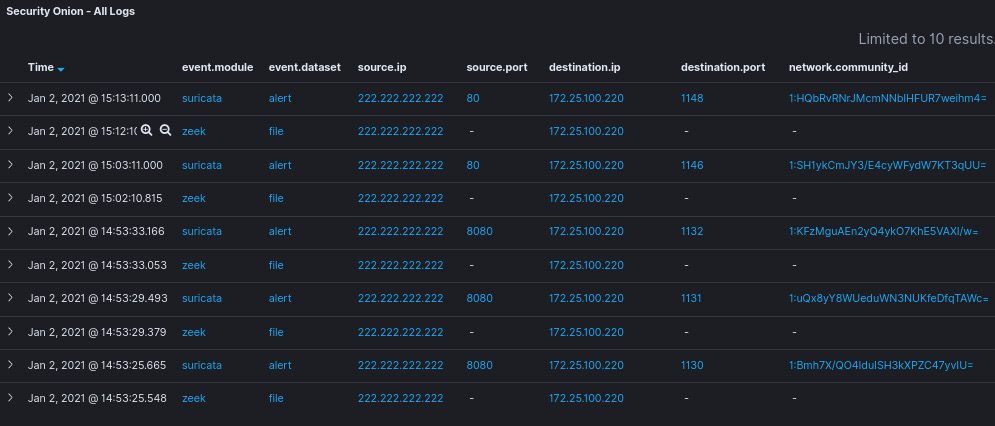
*Figure 9.66 – Breach Detection dashboard: filtered NIDS alerts*

We can also see suspicious ingress connections from that IP address, as illustrated in the following screenshot:



*Figure 9.67 – Breach Detection dashboard: filtered suspicious ingress connections view*

Additionally, we can look at the details of any log at the bottom of the screen, as illustrated in the following screenshot:



*Figure 9.68 – Breach Detection dashboard: filtered event viewer*

That is all we will cover for now. This dashboard can function as a starter to get a **security operations center (SOC)** view in place. This will likely change, so add and modify the dashboard to make it fit your needs, but this is a great starting point for your journey into a holistic security monitoring approach.

Summary

In this chapter, we took a mostly passive approach to security monitoring, where we had our tools do the digging and exposing for us. We looked at how to combine all of our (passive) security monitoring tools and combine them into a single, interactive dashboard view that allows us to quickly assess the security status of our environment. You should now be able to add, change, or extend the functionality of the Kibana **Breach Detection** dashboard or any other dashboard that ships with Security Onion, to make the best use out of the data we have been collecting. In the next chapter, the start of Section 3, Threat Hunting, we will be rolling up our sleeves to start digging around in the environment to see if we can find some malicious activity or actors.