Here we are creating an endpoint exception using the information that would prevent a malware detection alert from being generated if the preceding criteria are met:

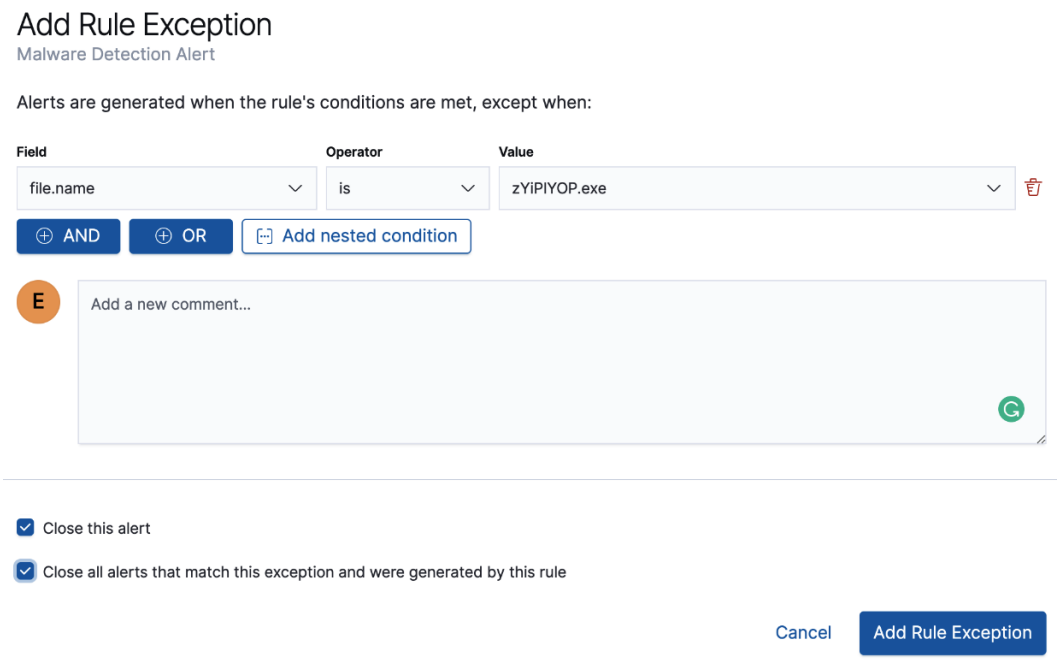

Figure 8.46 – Malware event – Add Rule Exception

We can also make a rule exception to not create an alert when the filename is `zYiPlYOP.exe`.

When making exceptions, we should look back at the *Pyramid of Pain* that we discussed in *Chapter 2*, *Hunting Concepts, Methodologies, and Techniques*. Remember that the higher up in the pyramid we go, the harder it is for an adversary to adapt. Using that as a critical thinking point, creating a rule based on a filename would be an exception that could easily be circumvented, so ensure you're making exceptions that have a high level of specificity, such as a file hash, code-signing information, or process, network, or registry associations.

The exception framework is a powerful tool in adapting the security solution to your environment. Many files or events could be considered malicious in some environments but benign in yours.

The detections engine is a tremendous part of the Elastic Security solution. We discussed alerts, individual events, organizing, the **Resolver** interface, creating timelines and cases, basic alert management, and creating both endpoint and rule exceptions.

Next, we'll be moving onto the **Hosts** tab to focus on host-specific information.

Hosts

The **Hosts** section of the Security solution allows you to get a high-level view of the endpoints that are reporting into your stack. This can be helpful to get ecosystem-wide metrics about your environment, such as the number of hosts, operating systems, authentication statistics, and so on.

Our lab environment will likely be sparsely populated with data because we only have one host (our victim machine). Looking at a larger analysis environment, we can see how this view can provide an overview of your hosts:

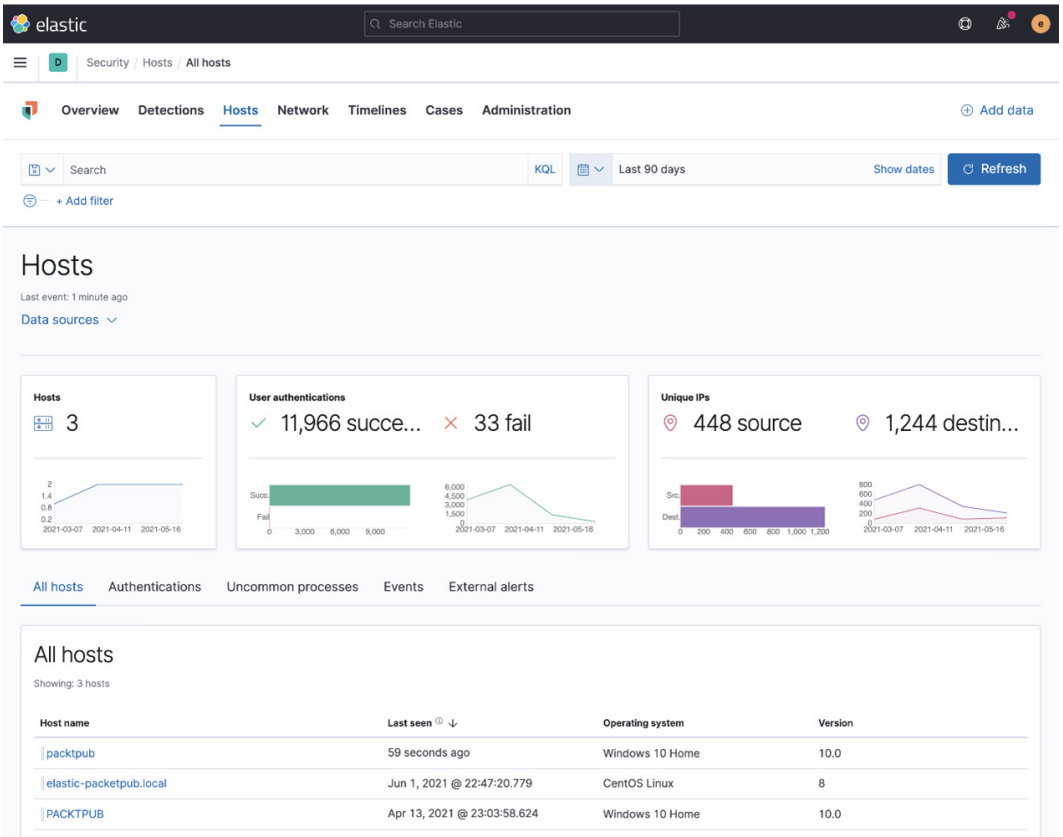

Figure 8.47 – Hosts overview

When we built our lab in *Chapter 4*, *Building Your Hunting Lab – Part 1*, we configured our victim to use the Elastic Agent, Packetbeat, and Winlogbeat. We can see those data sources reflected in the **Hosts** section. If you want to remove specific data sources, you can do that in **DATA SOURCES SELECTION**:

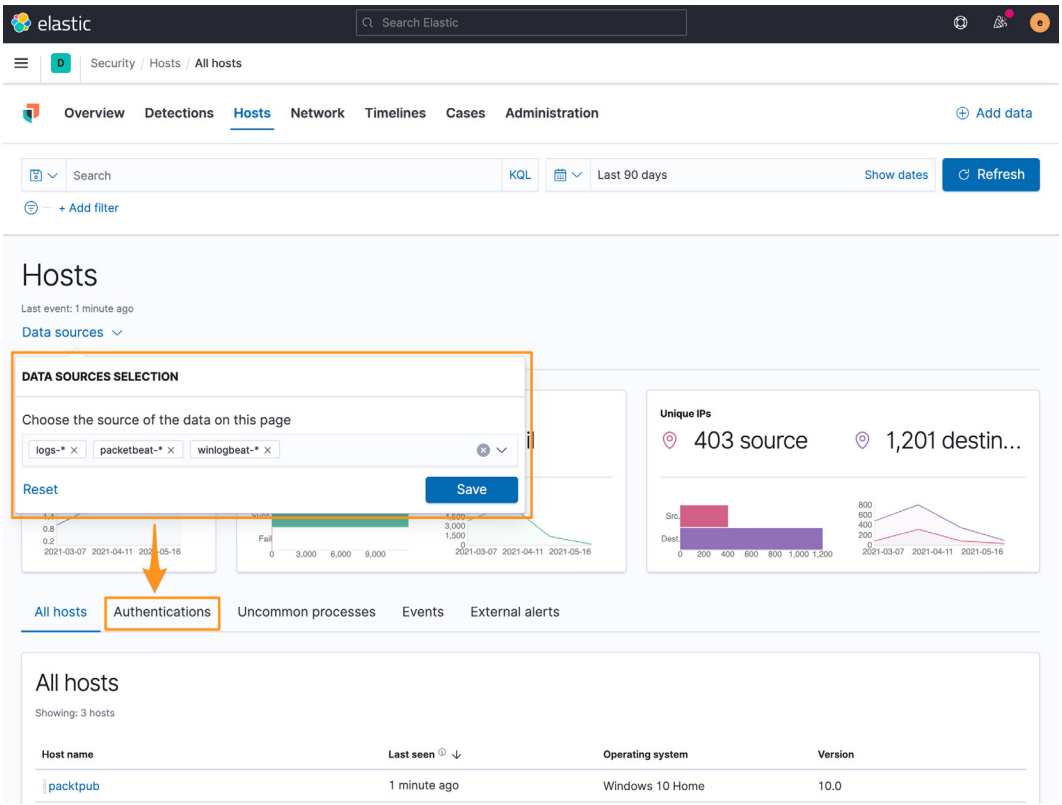

Figure 8.48 – DATA SOURCES SELECTION

Now that we've reviewed the different data source options, we can click on the **Authentications** tab to view an overview of the authentication events:

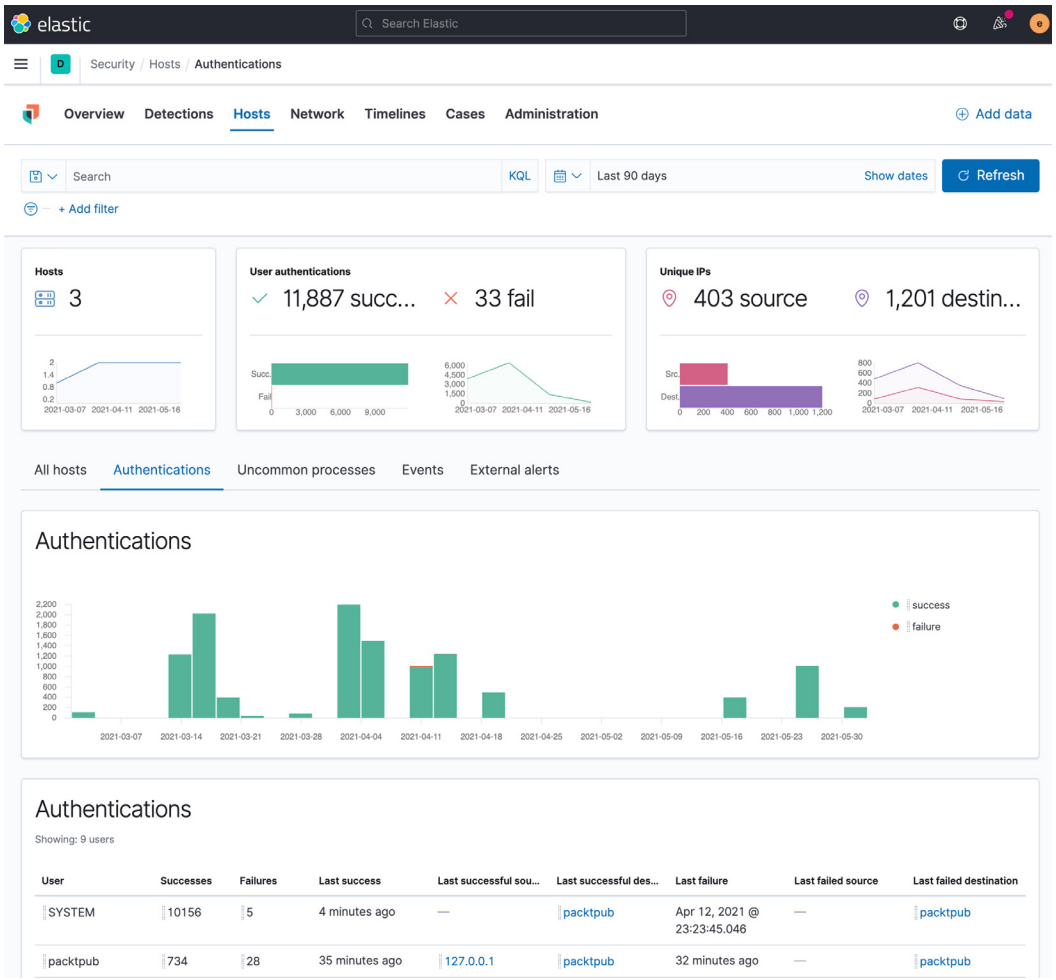

Figure 8.49 – Authentications

We can see additional information about authentications from our environment.

Just like on the **Detections** tab, if we want to know more about the failures that I generated, I can simply click and drag the failures down to the **Timeline** slide-out at the bottom of the screen, or click the **Timeline** button, to investigate more about these events:

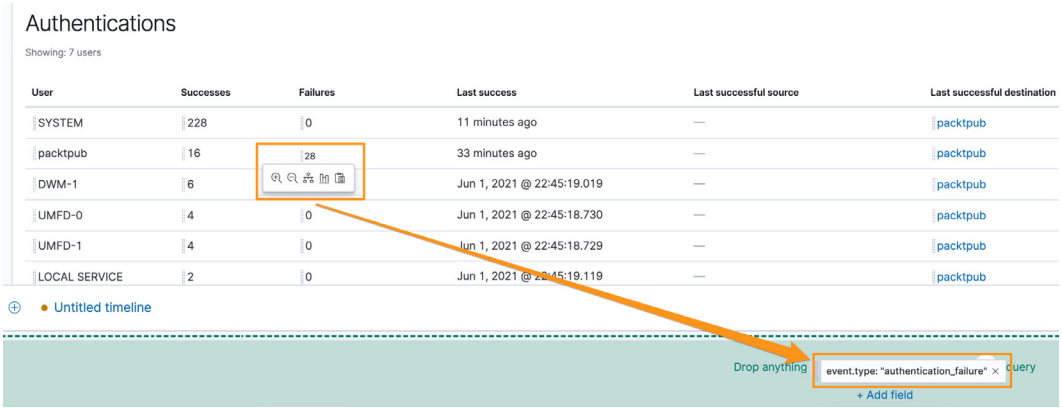

Figure 8.50 – Drag event to timeline

Moving on from **Authentications**, we can click on the **Uncommon processes** tab. This will show us processes that are occurring the least amount of times on the least amount of hosts. Our lab has just one host, so this will have a lot of processes.

Next, we can click on the **Events** tab. This will have a tremendous amount of data, from endpoint events to network events. This can be very valuable, but it should be a place that we search when we have an idea of what we are looking for. As an example, if we search for a previously identified suspicious process, we can do that here and greatly narrow down our aperture:

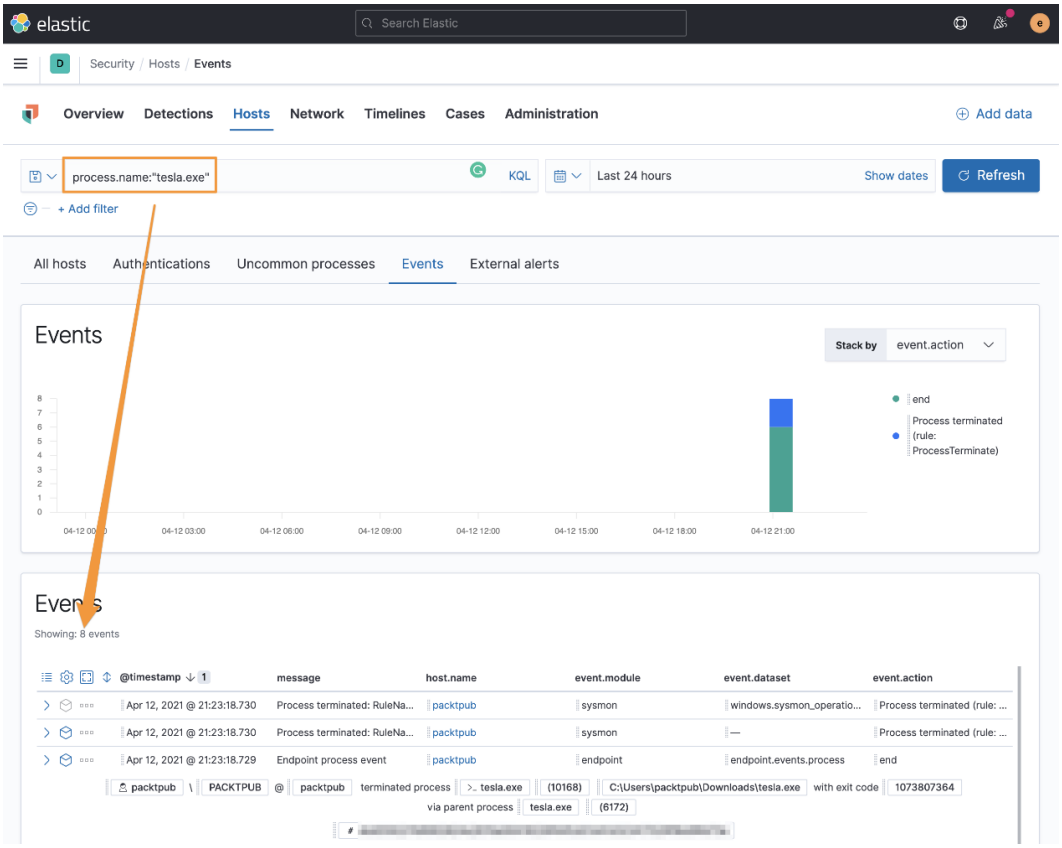

Figure 8.51 – Searching for tesla.exe

The **External alerts** tab will include endpoint alerts that are generated by third parties that are sending data into the Elastic Stack using ECS. Examples could be `osquery` (`https://osquery.io`), Tanium (`https://www.tanium.com`), and others. For our lab environment, we don't have any third-party sources.

Narrowing down the events that are displayed makes this a helpful view; again, as with all things in the Security solution, we can drag events into the timeline or analyze them in **Resolver**.

The **Hosts** section allows you to focus on just host-specific data. While you can use this hosts data to identify network events, having a narrow view while analyzing large amounts of data is helpful to identify abnormal events.

Next, we'll discuss analyzing network-specific events on the **Network** tab of the Security solution.

Network

Clicking on the **Network** tab will take you to an overview of the network data that is provided by the endpoints sending data into our Elastic Stack.

Similar to the **Hosts** section, there are protocol sections to allow you to review the more common network protocols, such as DNS, HTTP, and TLS. **Flows** are display data that doesn't fall into a parsed protocol, but is still recorded by Packetbeat.

Also, like the **Hosts** tab, you'll notice an **External alerts** section. This is where third-party network security solutions would report observations, such as Zeek or Suricata:

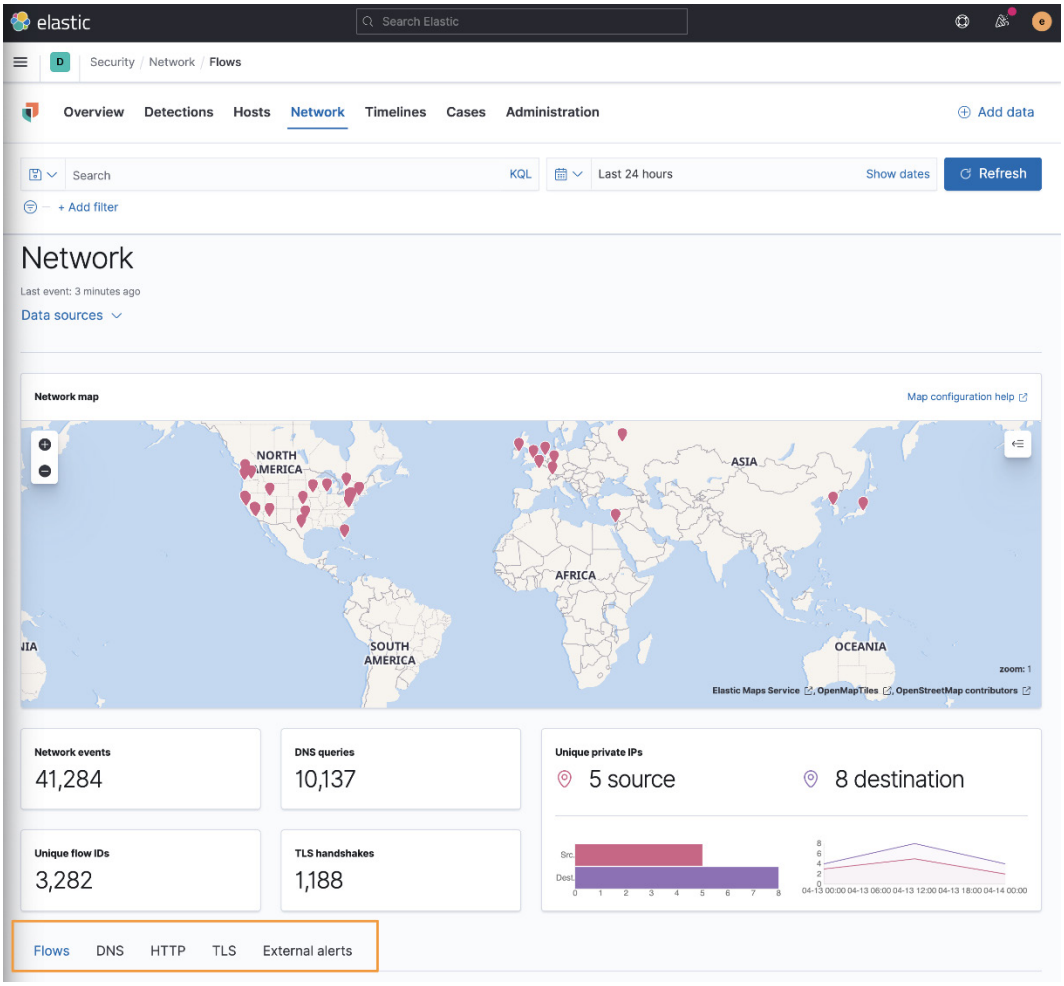

Figure 8.52 – Network overview of the Security solution

In this section, we introduced you to the **Network** section of the Security solution. Next, we'll explore **Timelines**, which is a powerful searching feature from within the Security solution.

Timelines

In the **Detection alerts** section earlier in the chapter, we discussed how to add events to the **Timelines** section as a query, either from the **Alerts** window or from the **Timelines** section by dragging fields onto the query panel.

There is another section in Timelines, and that is where you can write EQL queries. This is a huge benefit because the only other places that you can use the powerful EQL queries are against the Elasticsearch API or correlation detection rules.

Creating a very simple query to correlate events from the endpoint that show the cURL process starting a malicious destination domain we used in the indicator match rule:

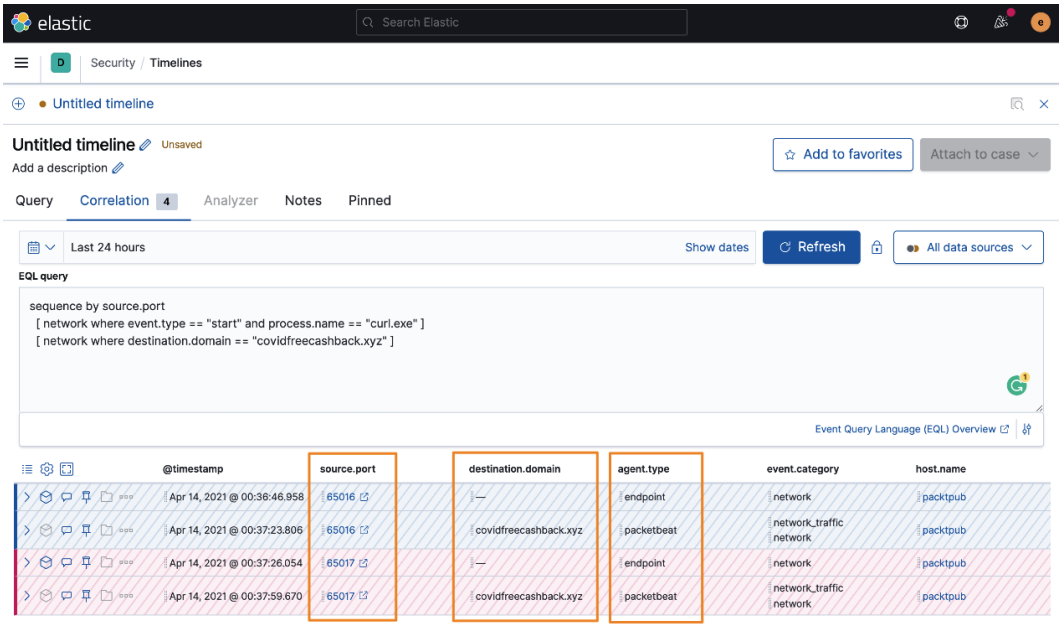

Figure 8.53 – Correlating endpoint and Packetbeat data together

The events are color-coded to visually associate them together. The blue endpoint events go with the blue Packetbeat data, and the same goes for the red events. You can see that the `sequence by` syntax for the `source.port` is reflected in source ports of `65016` and `65017`.

In this section, we covered timelines, which are a powerful tool used to query our security data using EQL for advanced queries.

Next, we'll discuss the **Cases** tool, which is used for basic event tracking.

Cases

The Elastic cases feature is used to manage basic workflow and processes for observed events. This is not a full-blown case management solution; it is basic, with the intention that third-party connections are used for a proper case-management solution.

Cases can be created from the **Alerts** section by clicking on the folder icon, from a timeline, or from the **Cases** tab:

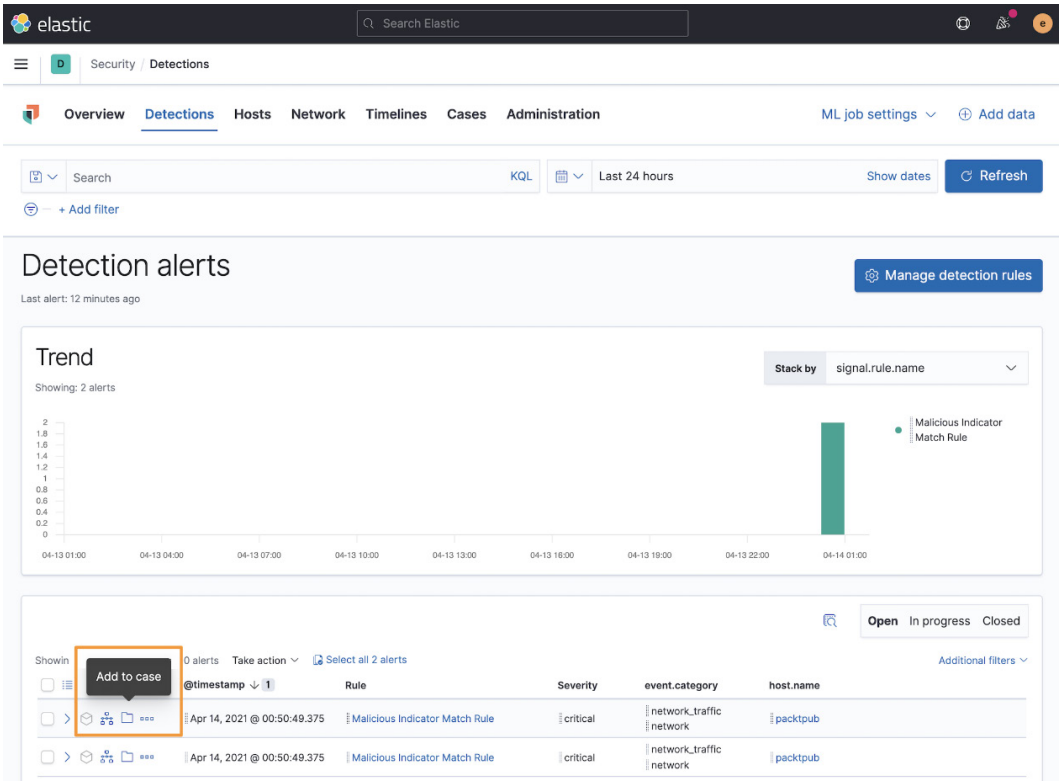

Figure 8.54 – Create cases from the Alerts page

Cases can also have templates added to them that aid in the investigation of events:

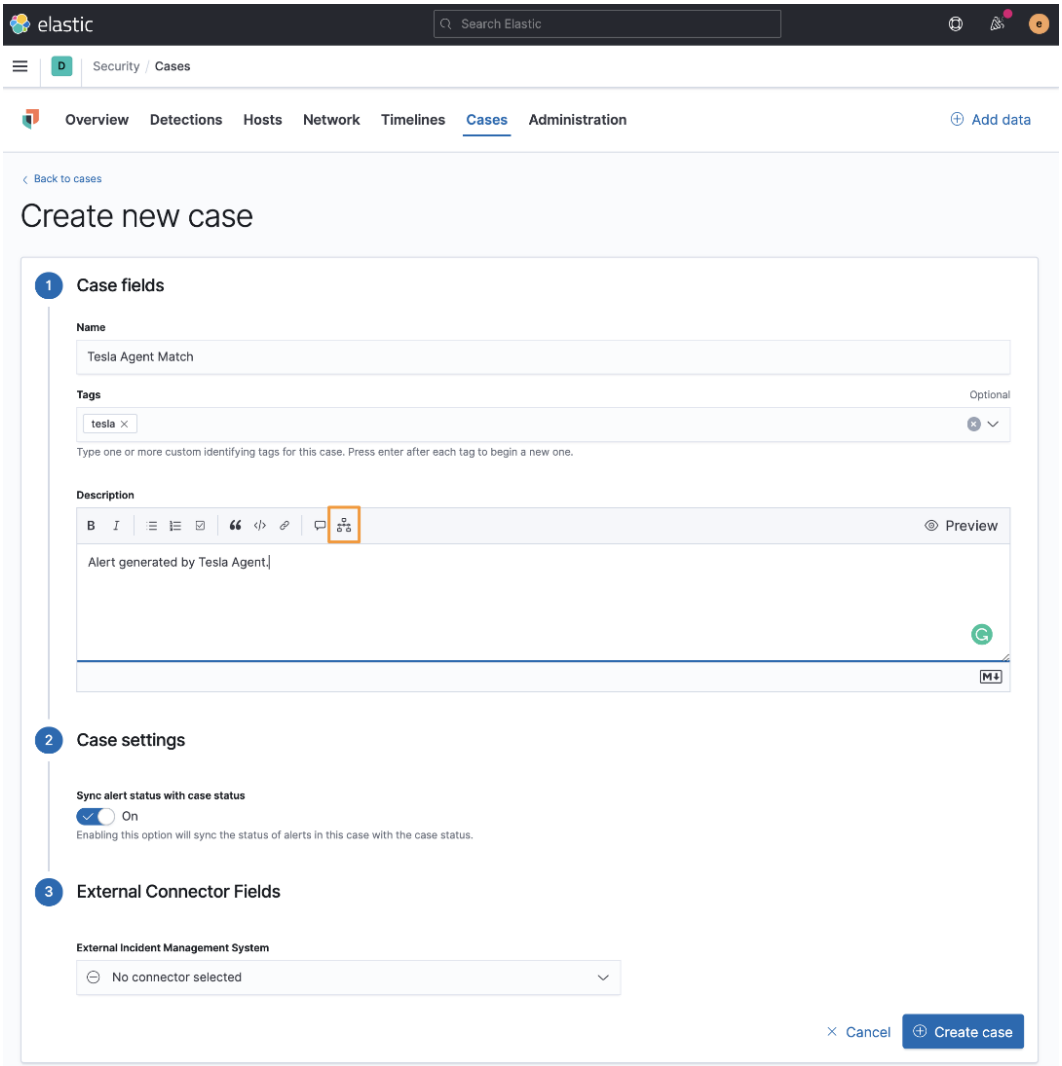

Figure 8.55 – Cases with timeline icon

Clicking on the timeline icon will open a window that will allow you to select any available timeline:

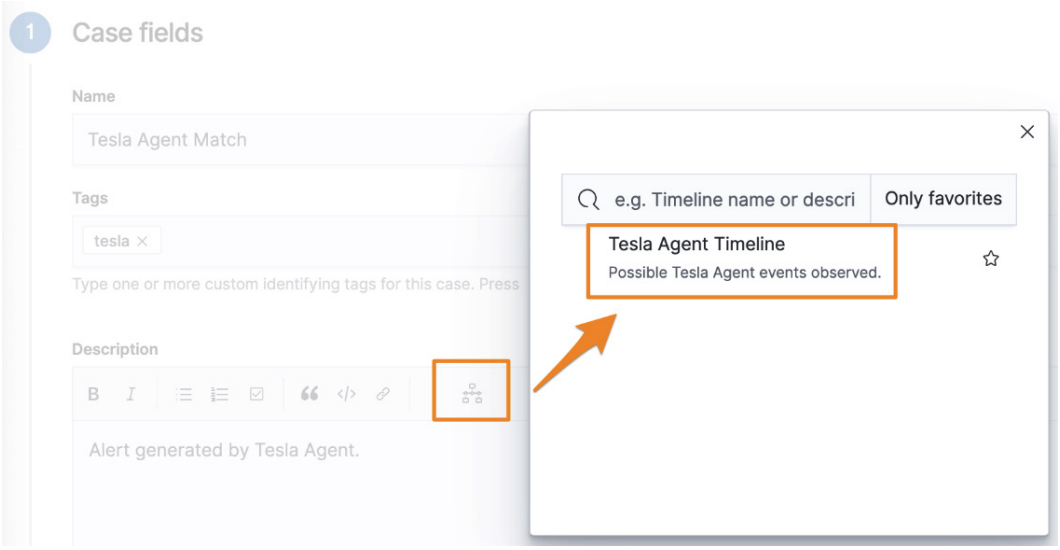

Figure 8.56 – Adding a timeline to a case

We can add the timeline we created for the previously observed Agent Tesla infection. This adds the timeline as a Markdown hyperlink.

Once the case is created, we can make basic annotations and notes during our investigation. All of the comments render Markdown. Once you've completed your investigation, you can close the case from here:

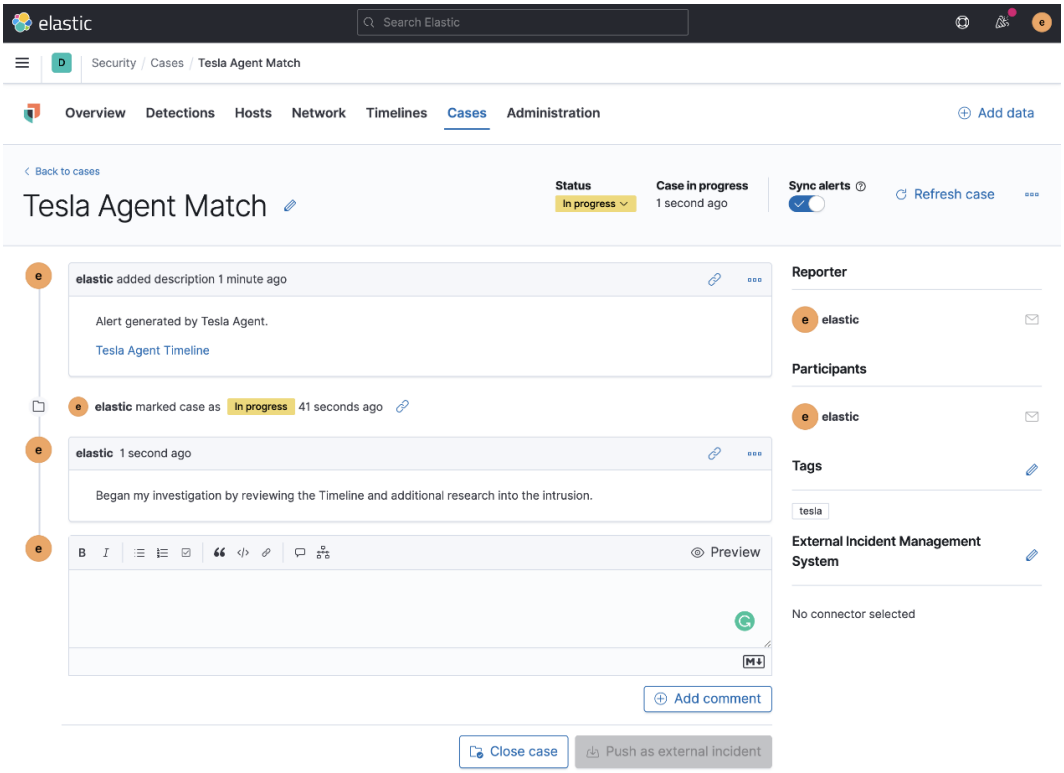

Figure 8.57 – Responding to an event using a case

Using cases, we can manage basic responses to identified events from within the Security solution.

Next, we'll review the **Administration** section of the Security solution.

Administration

The **Administration** tab allows us to review the status of all of the endpoints that are reporting to our Elastic Stack.

Additionally, we can add trusted applications that we don't want to generate alerts from. Great examples could be legacy anti-virus, asset management tools, or vulnerability scanners:

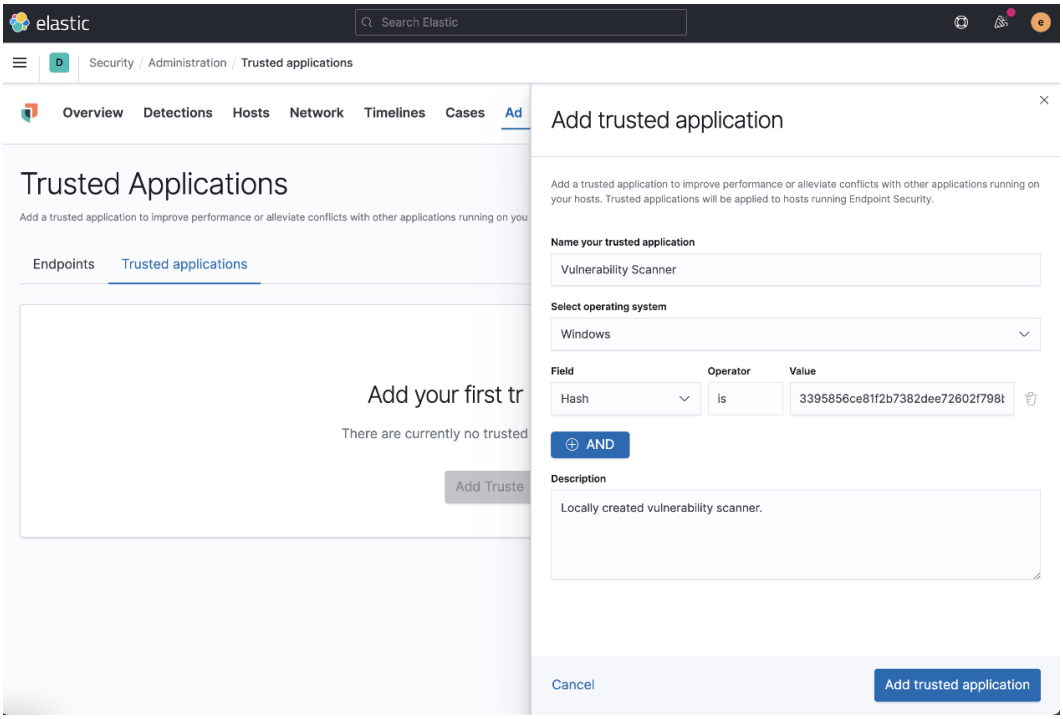

Figure 8.58 – Adding trusted applications

We can name the application, select the appropriate operating system, and define a path, filename, or hash value. We can select multiple fields, so if there is a file that is trusted but could also be abused, we could define the name, hash, and location:

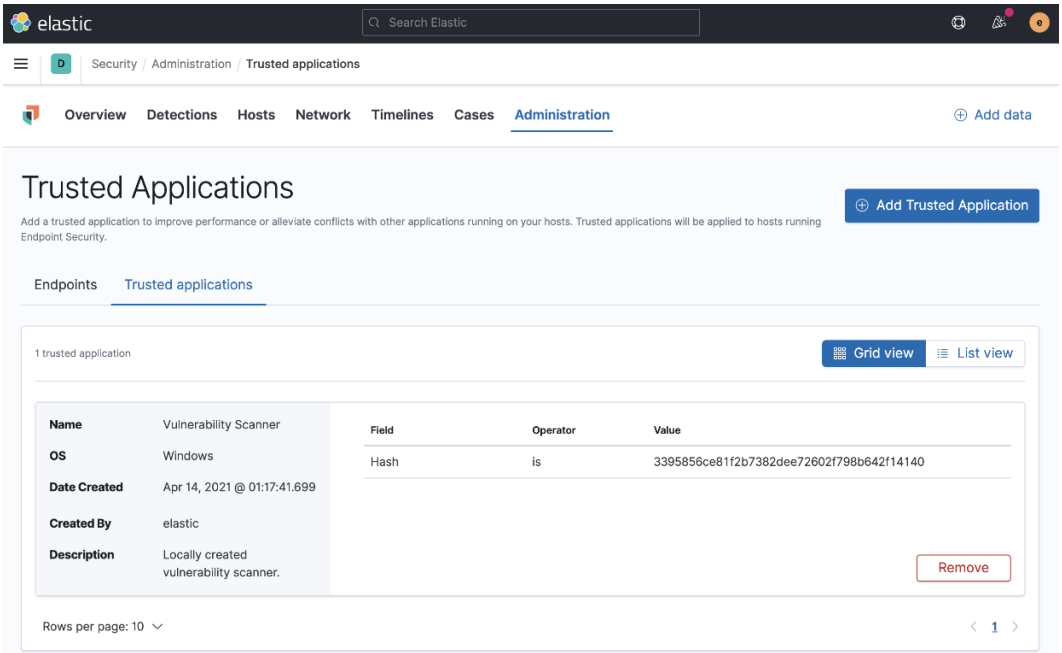

Figure 8.59 – Adding trusted applications

We can monitor all of our trusted applications and see some information about them.

In this section, we discussed the administration of the Elastic Security solution, specifically the endpoints and the trusted applications.

Summary

In this chapter, we thoroughly explored the Elastic Security app. We dug into each of the app sections and explored the detection engine. From the detection engine, we created five different types of rules and generated sample data for analysis. We also explored specific host and network sections that display security-related information. We created timelines for events using EQL. We used cases to track events in combination with timelines. Finally, we explored the administration of the Security solution, looking at adding trusted applications.

The skills you gained in this chapter will allow you to identify malicious events, correlate endpoint and network data together, and begin the analysis process.

In the next chapter, we'll spend even more time in the Security solution, specifically leveraging timelines to further investigate the Tesla Agent event we observed in this chapter.

Questions

As we conclude, here is a list of questions for you to test your knowledge regarding this chapter's material. You will find the answers in the *Assessments* section of the *Appendix*:

1. External host alerts can be collected from where?

   a. Osquery

   b. Zeek

   c. Suricata

   d. Filebeat

2. External network alerts can be collected from where?

   a. Osquery

   b. Zeek

   c. Tanium

   d. Filebeat

3. Indicator match rules can be fed from what module?

   a. Filebeat System Module

   b. Packetbeat

   c. Auditbeat

   d. Filebeat Threat Intel Module

4. Which of the following query languages can timelines use for correlations?

   a. KQL

   b. SQL

   c. EQL

   d. Lucene

5. What is the name of the tool that allows you to visually explore alerts?

   a. Resolver

   b. Hosts

   c. Network

   d. Timelines

Further reading

To learn more about the topics in this chapter, see the following:

- Elastic detection rules: `https://github.com/elastic/detection-rules`
- Building block rules: `https://www.elastic.co/guide/en/security/7.12/building-block-rule.html`
- Tesla Agent: `https://malpedia.caad.fkie.fraunhofer.de/details/win.agent_tesla`
- `Schtasks.exe`: `https://docs.microsoft.com/en-us/windows/win32/taskschd/schtasks`
- attrib: `https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/attrib`

Section 3: Operationalizing Threat Hunting

Now that the methodologies and theory have been discussed and the tools have been explored, this part will stitch processes and technology together so that you can go beyond just reading alerts and actually hunt for advanced adversaries.

This part of the book comprises the following chapters:

- *Chapter 9*, *Using Kibana to Pivot through Data to Find Adversaries*
- *Chapter 10*, *Leveraging Hunting to Inform Operations*
- *Chapter 11*, *Enriching Data to Make Intelligence*
- *Chapter 12*, *Sharing Information and Analysis*

9

Using Kibana to Pivot Through Data to Find Adversaries

Now that we've learned about the individual apps within Kibana, introduced various query languages, experimented with visualizations and dashboards, and explored security solutions, we can begin to stitch various data sources together to move beyond detection to identify how an adversary got inside the endpoint and what the goal of their intrusion was. This is extremely helpful when looking at operational and strategic intelligence assessments, as discussed in *Chapter 1*, *Introduction to Cyber Threat Intelligence, Analytical Models, and Frameworks*.

In this chapter, you'll learn how to use timelines in the Security app, use observations to connect network and endpoint data, and create detection logic using information derived from previous observations.

In this chapter, we'll go through the following topics:

- Connecting events with a timeline
- Using observations to perform targeted hunts
- Generating tailored detection logic

Technical requirements

In this chapter, you will need to have access to the following:

- The Elastic and Windows virtual machines that you built in *Chapter 4*, *Building Your Hunting Lab – Part 1*
- A modern web browser with a UI

Check out the following video to see the Code in Action:
`https://bit.ly/2VDveDx`

Connecting events with a timeline

In *Chapter 8*, *The Elastic Security App*, we were introduced to an information stealer known as Tesla Agent. We were able to see a bit of data surrounding the Tesla execution, with me even staging the malware onto the victim box. As promised, we're going to dig a bit deeper into this malware infection to demonstrate how to use the Security app to perform targeted hunts for observed events. Let's get right into it.

As I mentioned in the previous chapter, I am obscuring the malware identifying marks because it is live malware, which can damage a network, and adversary-controlled infrastructure could have innocent victims that I don't want to expose.

As a brief reminder, we detonated a malware sample on our victim machine. I used a two-day-old Agent Tesla sample, but any will do. Once you've detonated your malware, you should see it in the **Alert View** of the Security app. From here, we can click on the **Resolver** button.

Moving on, we can click on `tesla.exe` and then the **4 file** button. We should be able to see some more useful data about the actual malware execution:

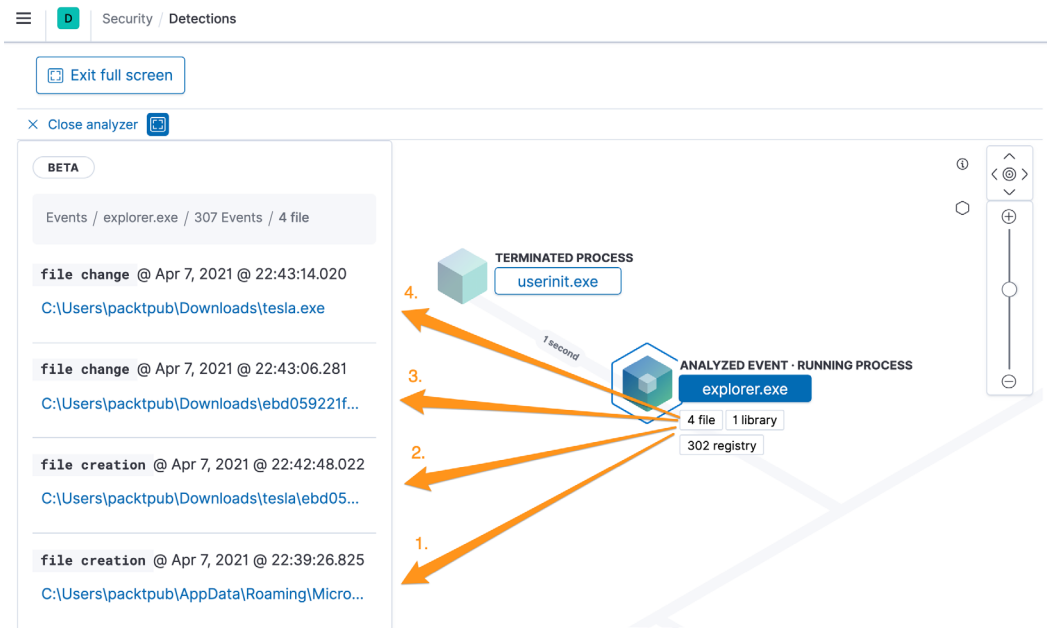

Figure 9.1 – Tesla.exe – the Resolver view

Here are the details of the numbered items listed in the preceding screenshot:

1. This is a file that was created in `C:\Users\packtpub\AppData\Roaming\zYiPlYOP.exe`. We'll explore this in more detail going forward when we exit the **Resolver** view and move on to the endpoint.
2. This is the file that was just created being altered; we're not exactly sure what has been changed *yet*, but we'll get to that later.
3. This is another file that was created called `C:\Users\packtpub\AppData\Local\Temp\tmp203.tmp`. We'll explore this more going forward.
4. A Microsoft log file was also created after the execution.

If we click on a file that was created in the `Temp` directory (number *3* from the preceding list), we'll see that it was overwritten, so we won't have that to analyze. After doing some searches of `tmp203.tmp` on the internet, we can see that it is exclusively associated with malware events. That's helpful, but we're not interested in what the internet is telling us. Instead, we're interested in what we can find.

Let's move on to the actual running process, `tesla.exe`:

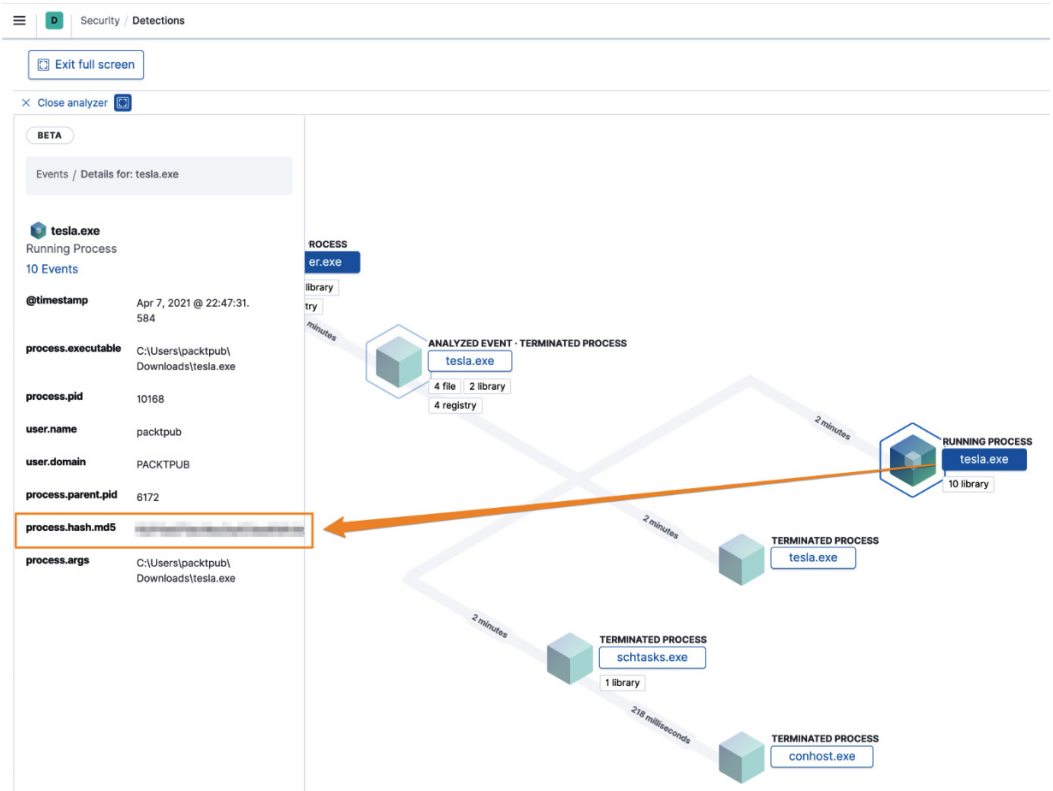

Figure 9.2 – Tesla.exe – the MD5 hash

When we click on `tesla.exe`, we can view the MD5 file hash. Using the file hash, you can collect additional information from third-party research sites that allow you to search for information based on file hashes. That's useful, but we can also see that the **Analyzed Event** of `tesla.exe` has spawned another process, `schtasks.exe`. If we click on `schtasks.exe`, we can see additional data in the details.

These details show us that there appears to have been a scheduled task created in the **Updates** library. This is a common persistence technique that specifically uses the MITRE ATT&CK persistence tactic, the Scheduled Task/Job technique, and the Scheduled Task/Job: Scheduled Task sub-technique (`https://attack.mitre.org/techniques/T1053/005/`):

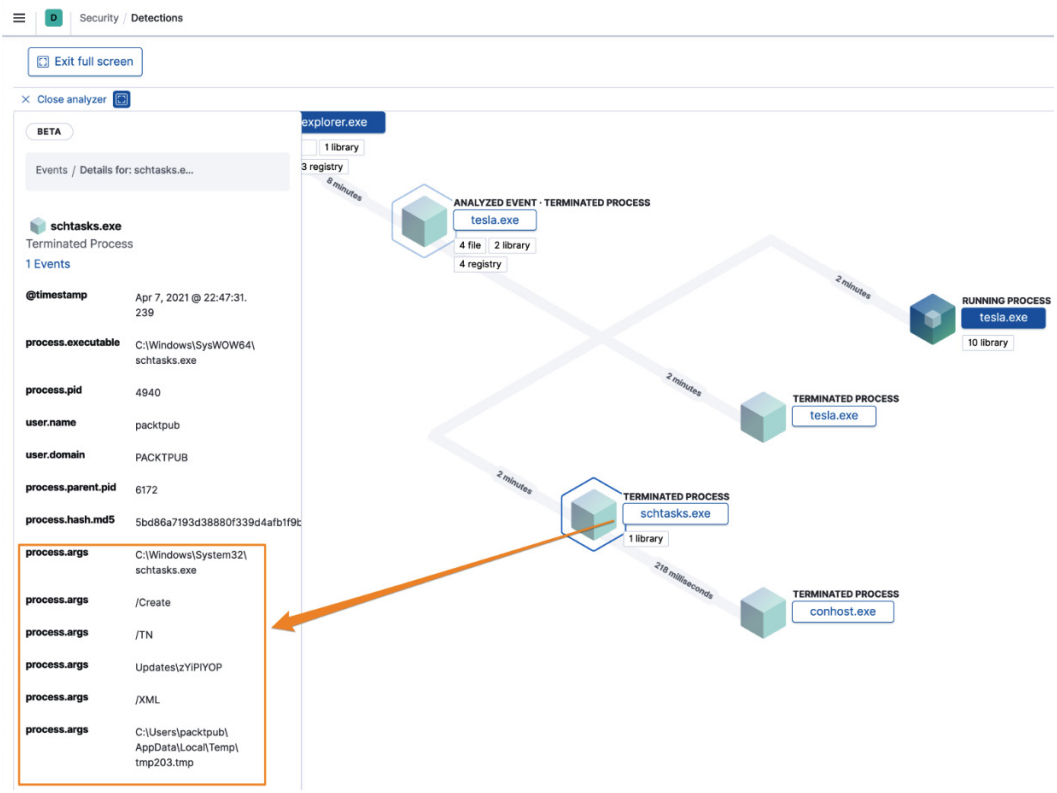

Figure 9.3 – Malware event – Task Scheduler persistence

Looking at the previous argument process, we can view our mysterious `tmp203.tmp` file again. After seeing it in this context, we have a better idea of what might have been in that file.

If we line those process arguments up, we can see pretty clearly how `schtasks.exe` is being used:

```
C:\Windows\System32\schtasks.exe /Create /TN Updates\zYiPlYOP /
XML C:\Users\packtpub\AppData\Local\Temp\tmp203.tmp
```

Here are details of the syntax of the preceding `schtasks.exe` command:

- `C:\Windows\System32\schtasks.exe`: This calls the **Task Scheduler** program.
- `/Create`: This creates a new scheduled task.
- `/TN`: This defines the scheduled **Task Name** (**TN**).

- `Updates\zYiPlYOP`: This puts the scheduled task in the **Updates** library and names it `zYiPlYOP`.
- `/XML`: This is used to create the scheduled task from an XML file.
- `C:\Users\packtpub\AppData\Local\Temp\tmp203.tmp`: This is the XML file used to create the task.

So far, we know the following main facts about this malware event without even touching the box:

- Based on the Detection Engine alert, a malware event has occurred.
- A file was created in the `AppData\Roaming` folder and then changed.
- We have an MD5 hash of the `tesla.exe` file.
- A scheduled task was created from a removed XML file.

Using that information, I think this warrants a closer inspection of the endpoint. Let's take a look at the victim system and see whether we can find any additional information.

First, let's see if we can find the file that is in the `AppData\Roaming` folder. Open up `cmd.exe` and navigate to the directory. In my case, that is `C:\Users\packtpub\AppData\Roaming`. Next, run the `dir` command to list the contents of the directory:

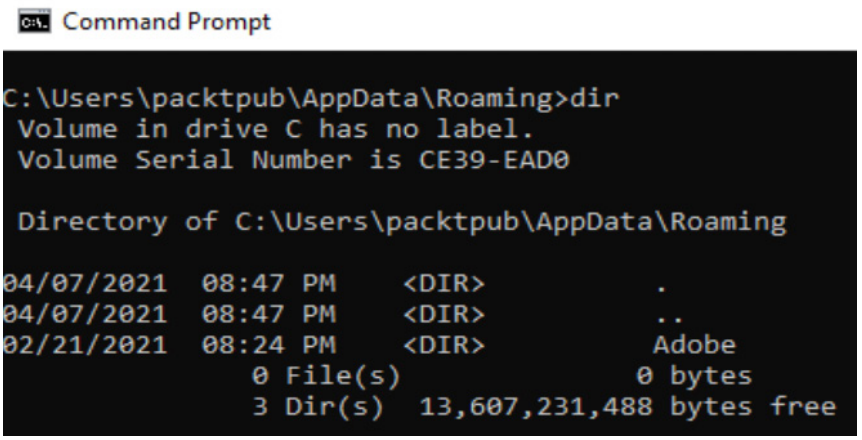

Figure 9.4 – The directory of AppData\Roaming

That doesn't look good. The file is supposed to be here, right? If you remember back in the **Resolver** view of **4 file**, we saw that `C:\Users\packtpub\AppData\Roaming\zYiPlYOP.exe` had a **File change** event. We know the file is here because we haven't come across a file overwrite or delete event. Let's see whether there are any files here that have special attributes set. File attributes can be used to hide files.

We can run `attrib` and we see that the `C:\Users\packtpub\AppData\Roaming\zYiPlYOP.exe` file has the `SHRI` attributes set! This is a Defense Evasion tactic from the MITRE ATT&CK framework, which specifically uses the Hide Artifacts technique and the Hidden Files and Directories sub-technique (`https://attack.mitre.org/techniques/T1564/001/`):

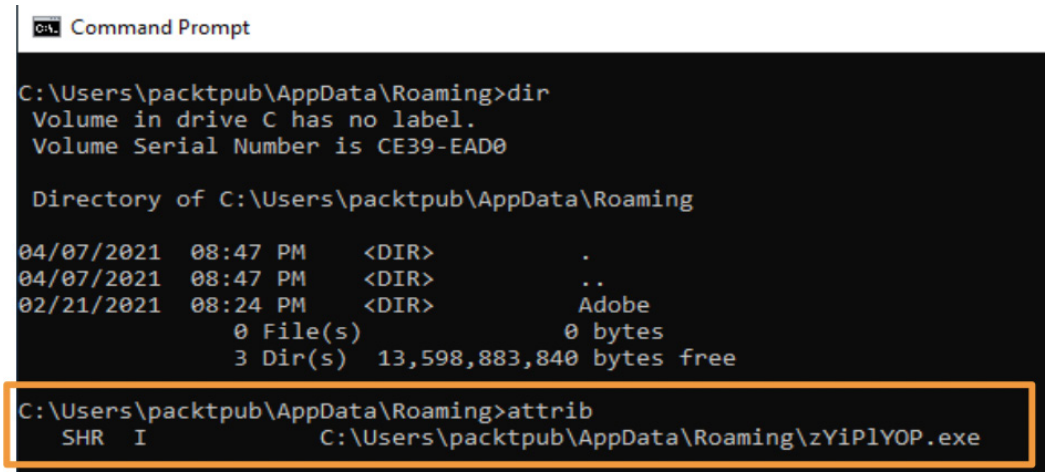

Figure 9.5 – Displaying the file attributes

When looking at the attributes, we can see what they mean:

- `S`: This is the **System** file attribute. It means that an adversary is attempting to mark the file as being used exclusively by the operating system, meaning it cannot be modified or deleted.
- `H`: This is the **Hidden** file attribute. It means that the file cannot be viewed under normal conditions.
- `R`: This is the **Read-Only** file attribute. It means that the file cannot have changes saved to it by other programs, but this will not prevent it from being deleted.
- `I`: This refers to **Not Content-Indexed**, meaning the file won't be indexed or searchable in Windows.

As we're already inside the box and my presence could be made known to the adversary, I will copy the file so that I have a copy in the event that the adversary starts to burn their implants.

To do that, you need to remove the system and hidden attributes and then copy the file somewhere away from the infected system.

You can again use the `attrib` program with the `-s` and `-h` switches to remove the system and hidden attributes. Then, you can copy the file elsewhere for analysis:

```
attrib -s -h zYiPlYOP.exe
```

The following screenshot shows the removal of the system and hidden file attributes:

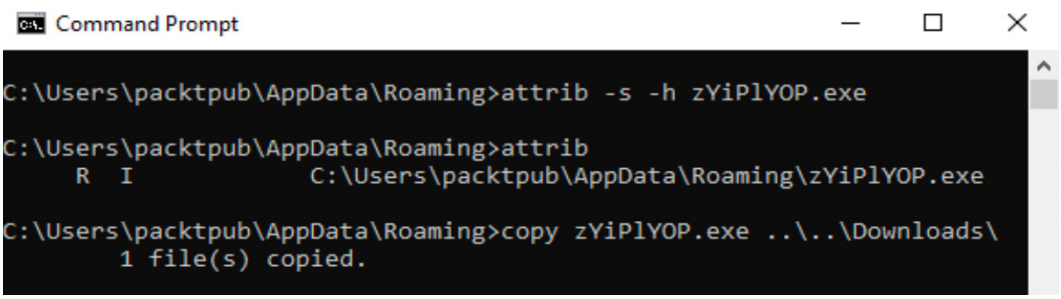

Figure 9.6 – Removing the system and hidden attributes and copying the malware

Let's also grab a file hash with PowerShell, as follows:

```
PowerShell.exe Get-FileHash -Algorithm sha1 zYiPlYOP.exe
```

The following screenshot shows the file hash of the implant:

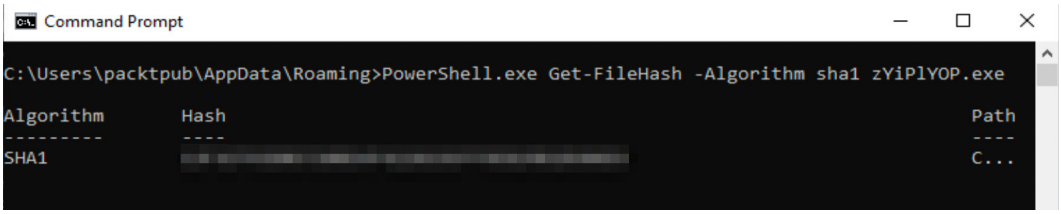

Figure 9.7 – Collecting the file hash

Now we can take that hash and compare it to the one that we observed in the **Resolver** view. In this example, they are the same. So, `tesla.exe` is copied here as `zYiPlYOP.exe`.

Now that we've identified this file, collected some metadata in the hash, and copied it for further analysis, let's move on to the persistence mechanism.

From the victim machine, open up **Task Scheduler** and click on the **Updates** library. We can see there is a new task called `zYiplYOP`. Does this look familiar? Take a look at the following screenshot:

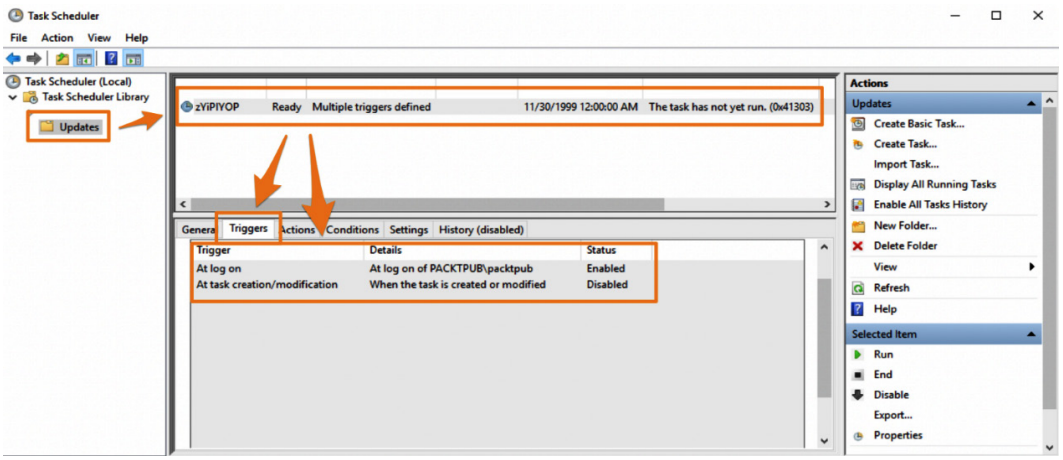

Figure 9.8 – Malware event – Task Scheduler Triggers

After examining the triggers, we can see that the task executes when the user logs on, but it has not run yet.

Next, by clicking on **Actions**, we can observe what the task does:

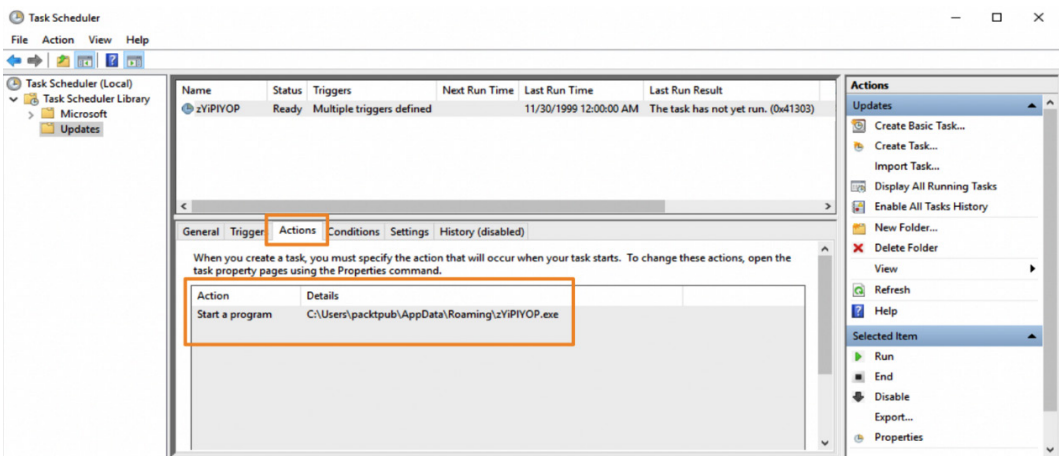

Figure 9.9 – Malware event – Task Scheduler Actions

The **Actions** tab tells us that `zYiPlYOP.exe` executes when the trigger of a user login occurs.

We could log off and then back in again so that the scheduled task executes, or we could delete it to remove the observed persistence mechanism. My recommendation would be to take the file we copied earlier and perform reverse engineering to understand more about it before trying to evict it from the infected system.

Using the Security app, we have learned a tremendous amount about a piece of malware that we detonated on our victim machine. We were able to use the **Resolver** view to pinpoint specific information about the malware event. This information led us to the infected endpoint, and we were able to collect metadata about the malware sample, collect the sample for future analysis, and even identify the persistence mechanism.

Next, we'll use the information we have collected to identify other systems that might be infected with the same malware sample.

Using observations to perform targeted hunts

As we explored in the previous section, using an Elastic Agent to detect and track malware samples is a great way to collect observations and security events that are happening on a system. However, what happens when we want to search through historic data to identify any previously infected systems? We can use the information we've collected to identify previously undetected infections.

There are several reasons as to why a system might have been infected without detection. It could be as simple as the system not having an Elastic Agent on it, the malware sample could be using bleeding-edge capabilities to evade detection at the time of infection, or it could also be that an alert just wasn't responded to.

Now that we've identified some malware samples on the victim machine, let's discuss the process to use that metadata to identify additional infections.

Pivoting to find more infections

Now that we've identified an observation, let's use that to search through the historical data to identify any missed infections.

As mentioned in the previous section, we were able to collect the file hash from both **Resolver** and the endpoint, a host artifact, a network artifact, and a TTP (persistence mechanism).

If you remember the *Pyramid of Pain* from *Chapter 2*, *Hunting Concepts, Methodologies, and Techniques*, hash values are trivial for an adversary to change, but host artifacts and TTPs (such as persistence mechanisms) are a bit more challenging. Difficult or not, we have them all, and we can use them all.

To safely illustrate this example, we'll use the EICAR MD5 hash (`44d88612fea8a8f36de82e1278abb02f`) as our hash but the Agent Tesla host artifact and persistence mechanism. For reference, the EICAR test file is a harmless file that can be used to test whether an anti-virus program is working in a variety of situations.

Let's navigate back to **Discover** and search for the MD5 hash and check whether we have any additional information we can use.

Searching for `file.hash.md5: 44d88612fea8a8f36de82e1278abb02f` gives us the SHA256 hash in addition to the MD5 hash. Great, let's add that to our list:

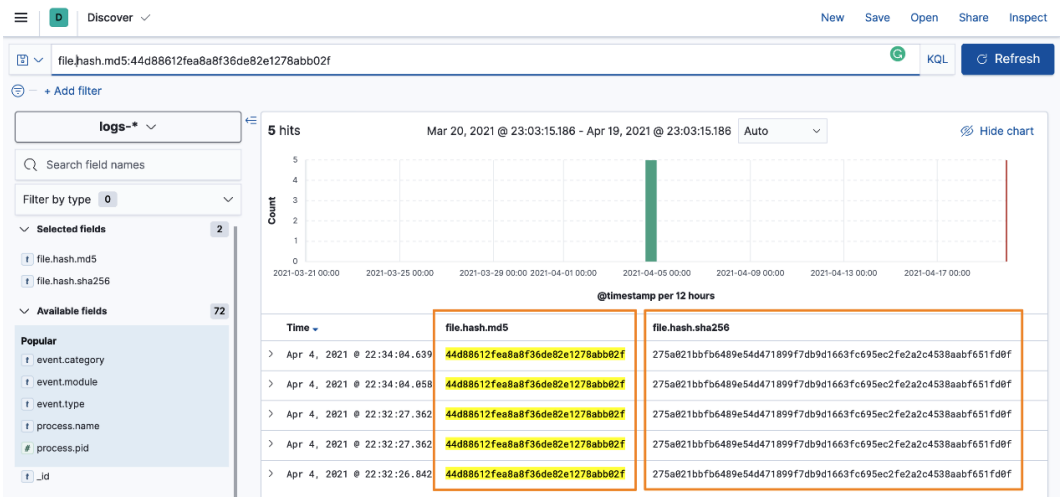

Figure 9.10 – Collecting additional hash information on the files

Next, let's change to the Winlogbeat Index Pattern to search for hosts that may not have had an Elastic Agent but could have had possible infections:

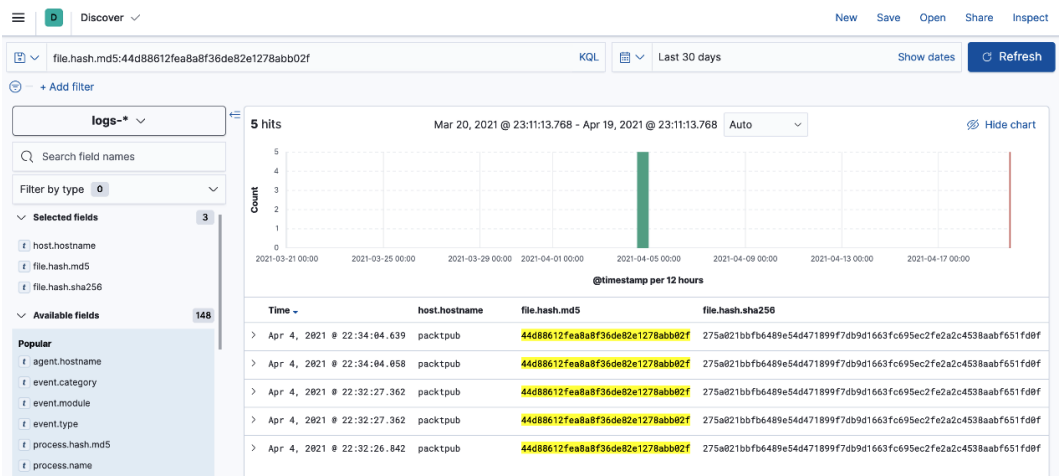

Figure 9.11 – Identifying other possible infections by file hash

I also added the `host.hostname` field to identify the infected hostname.

That's great – but we also discussed host artifacts, such as the directory where the file is being hidden.

If you remember, the file was created in the `C:\Users\packtpub\AppData\Roaming` directory. We can do some tailored searches for what's going on there. This could be noisy, but in our example, the files were made directly in the `Roaming` directory, not in `C:\Users\packtpub\AppData\Roaming\Microsoft\...`, which is more common. Again, this is the value of using a victim machine that we control: the malicious events are relatively easy to track when we control everything about them in comparison with what you might encounter on systems with real users.

So, let's run a search for the files created in our directory of interest:

```
file.directory:"C:\Users\packtpub\AppData\Roaming" and event.
type:creation
```

The following screenshot shows search results of the files created in the directory where we observed the implant:

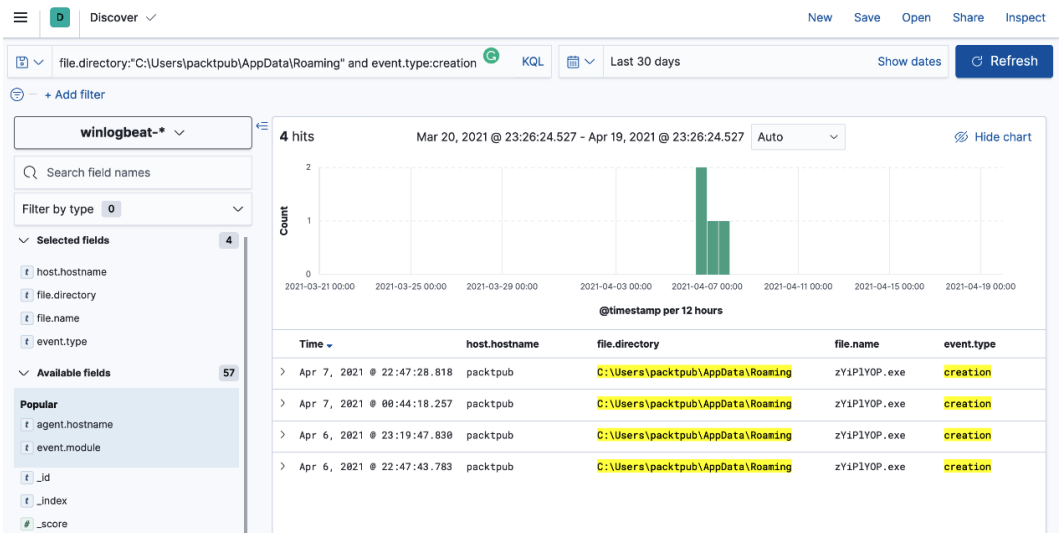

Figure 9.12 – Identifying other possible infections by the host artifact

After we've identified other possible infections based on where the implants are housed, let's consider the network artifacts.

Looking at network traffic from the Tesla agent, we can identify two network artifacts that we can use and one other interesting observation, SMTP traffic, that we'll explore in *Chapter 11*, *Enriching Data to Make Intelligence*:

Figure 9.13 – Identifying other possible infections by network artifacts

Here are some indicators that we can examine in the Packetbeat data. Let's change to Packetbeat again, because we're demonstrating how to search for observations without the Elastic Agent.

Performing a query for the DNS question domain and answer returns the results I was looking for and possible additional infected systems:

Time	host.hostname	dns.answers.data	dns.question.name
Jun 1, 2021 @ 23:35:10.431	packtpub	[redacted].191	[redacted].com
May 27, 2021 @ 20:12:12.108	packtpub	[redacted].191	[redacted].com
May 26, 2021 @ 23:38:49.970	packtpub	[redacted].191	[redacted].com
Apr 19, 2021 @ 22:22:14.705	packtpub	[redacted].191	[redacted].com
Apr 14, 2021 @ 00:42:42.099	packtpub	[redacted].191	[redacted].com
Apr 13, 2021 @ 23:57:40.224	packtpub	[redacted].191	[redacted].com
Apr 8, 2021 @ 01:03:31.819	packtpub	[redacted].191	[redacted].com
Apr 8, 2021 @ 00:18:31.840	packtpub	[redacted].191	[redacted].com
Apr 7, 2021 @ 23:33:33.430	packtpub	[redacted].191	[redacted].com
Apr 7, 2021 @ 00:05:46.992	packtpub	[redacted].191	[redacted].com

Figure 9.14 – Identifying other possible infections with Packetbeat

Now that we've identified other possible infections based on the network connections, let's consider TTP and its persistence mechanism.

With the Winlogbeat Index Pattern, let's discuss how to search the Task Scheduler with Kibana.

Using the information we collected regarding how scheduled tasks are created and structured, let's perform a simple search to identify when `schtasks.exe` is used in the same way that we observed.

We don't need this entire search. However, if we want to get really specific, perhaps for an enterprise deployment, we could perform a very granular search:

```
process.name:"schtasks.exe" and process.args:("/Create" and "/
TN" and Updates* and "/XML" and *\\AppData\\Local\\Temp\\tmp*.
tmp) and process.parent.executable:C\:\\Users\\*\\AppData\\
Roaming\\*.exe
```

The following screenshot shows a focused search where the scheduled task is being created:

	Time	host.hostname	process.args	process.parent.executable
›	Jun 3, 2021 @ 23:08:48.396	packtpub	C:\Windows\System32\schtasks.exe, /Create, /TN, Updates\zYiPlYOP, /XML, C:\Users\packtpub\AppData\Local\Temp\tmp6013.tmp	C:\Users\packtpub\AppData\Roaming\zYiPlYOP.exe
›	Jun 2, 2021 @ 20:57:41.115	packtpub	C:\Windows\System32\schtasks.exe, /Create, /TN, Updates\zYiPlYOP, /XML, C:\Users\packtpub\AppData\Local\Temp\tmp739B.tmp	C:\Users\packtpub\AppData\Roaming\zYiPlYOP.exe
›	Jun 1, 2021 @ 22:49:05.277	packtpub	C:\Windows\System32\schtasks.exe, /Create, /TN, Updates\zYiPlYOP, /XML, C:\Users\packtpub\AppData\Local\Temp\tmp92AC.tmp	C:\Users\packtpub\AppData\Roaming\zYiPlYOP.exe
›	May 27, 2021 @ 19:25:35.014	packtpub	C:\Windows\System32\schtasks.exe, /Create, /TN, Updates\zYiPlYOP, /XML, C:\Users\packtpub\AppData\Local\Temp\tmp7B2.tmp	C:\Users\packtpub\AppData\Roaming\zYiPlYOP.exe

Figure 9.15 – Identifying other possible infections by the persistence mechanism

If we break this search down, we can observe the following:

- `process.name`: `schtasks.exe` – This is used to identify the Task Scheduler process.
- `process.args`: `("/Create" and "/TN" and Updates* and "/XML" and *\\AppData\\Local\\Temp\\tmp*.tmp)` – We use the Task Scheduler process to create a scheduled task in a directory that has been previously identified as suspicious.
- `process.parent.executable`: `C\:\\Users\\*\\AppData\\Roaming\\*.exe` – This tells us that the scheduled task was created by an executable running from the Roaming directory, which was previously identified as suspicious.

Hunting involves the process of dialing in observations to identify previously undetected malicious events. This is done by identifying one abnormality, pulling the thread, collecting additional information, then painting a whole picture of how an intrusion might have occurred, and leveraging this information to evict an adversary.

Next, we'll create a detection rule to identify this activity in the future.

Generating tailored detection logic

It's great that we've identified a good search query to identify this type of malicious activity, but let's take that a step further to generate detection events in the Security app so that we aren't continually having to run a query in the Discover app.

Using what we learned in the *Creating detection rules* section of *Chapter 8*, *The Elastic Security App*, we can create a custom query detection rule to identify this activity:

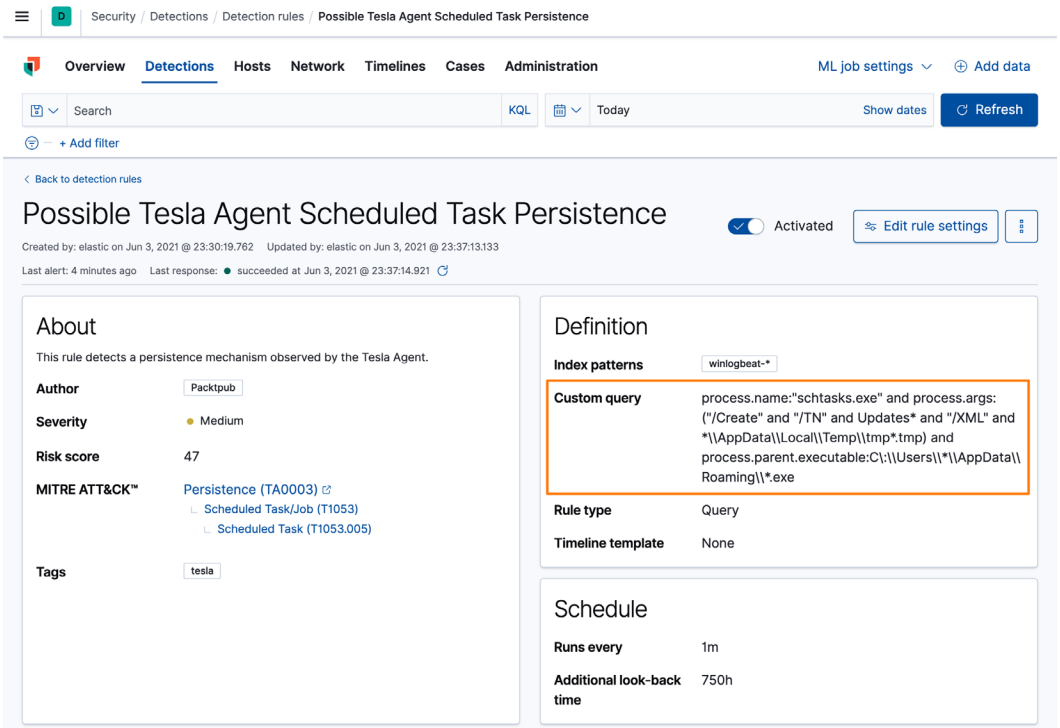

Figure 9.16 – Tailored detection logic for an observed activity

In the preceding screenshot, we can see the completed detection rule that will generate an event when this activity is observed in the future:

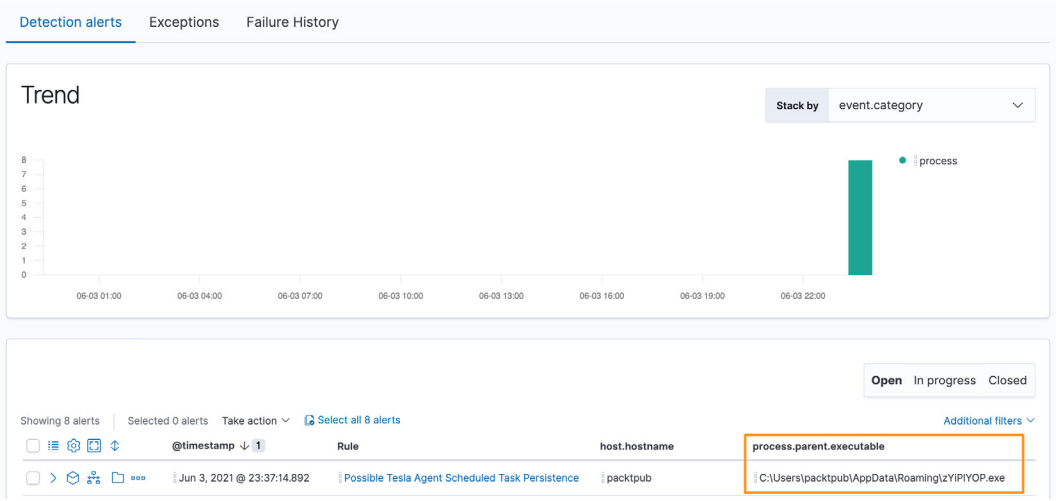

Figure 9.17 – Tailored detection logic for an observed activity

In the preceding screenshot, we can see that the detection rule was triggered based on the persistence detection logic that we just created.

In this section, we created tailored detection logic based on the information that we collected during our malware investigation of the victim machine.

Summary

In this chapter, we took a deep dive into a live malware sample to identify how to take an alert from Kibana, pivot down to the infected host and collect additional information, and then use all of this with Kibana to identify previously undetected infections using three infection elements: a hash, a host artifact, and a persistence mechanism. Finally, we created tailored detection logic in the Security app to allow us to detect this activity in the future.

In the next chapter, we'll use this technical information to inform incident responses and enduring operations. By doing so, we can enhance the security posture of an organization and prioritize additional visibility.

Questions

As we conclude, here is a list of questions for you to test your knowledge regarding this chapter's material. You will find the answers in the *Assessments* section of the *Appendix*:

1. What was the persistence technique observed in the example in this chapter?
   a. Attributes
   b. Scheduled task
   c. Deleting files
   d. Running in memory
2. What was the defense evasion technique observed in the example in this chapter?
   a. Scheduled task
   b. Deleting files
   c. Setting attributes
   d. Running as SYSTEM
3. What is an EICAR file?
   a. A harmless anti-virus test file
   b. A process hooking technique
   c. A memory injection tool
   d. A Kibana template
4. What do Actions in Task Scheduler show us?
   a. Who to run the scheduled task as
   b. Who to connect to and report the status of the scheduled task
   c. When a scheduled task runs
   d. What the scheduled task does
5. How do you remove a hidden attribute from a file?
   a. `attrib -h`
   b. `attrib +h`
   c. `attrib -d`
   d. `attrib -all`

Further reading

To learn more about the subject, you can refer to the following resources:

- *Defense Evasion, MITRE ATT&CK*: `https://attack.mitre.org/tactics/TA0005/`
- *Hide Artifacts, MITRE ATT&CK*: `https://attack.mitre.org/techniques/T1564/`
- *Hide Artifacts: Hidden Files and Directories, MITRE ATT&CK*: `https://attack.mitre.org/techniques/T1564/001/`
- *Persistence, MITRE ATT&CK*: `https://attack.mitre.org/tactics/TA0003/`
- *Scheduled Task/Job, MITRE ATT&CK*: `https://attack.mitre.org/techniques/T1053/`
- *Scheduled Task/Job: Scheduled Task, MITRE ATT&CK*: `https://attack.mitre.org/techniques/T1053/005/`

10
Leveraging Hunting to Inform Operations

In the previous few chapters, we have focused in-depth on leveraging the Elastic Stack to perform hunt operations. This was done by searching through your data using the Discover App, creating rich and contextual visualizations and dashboards, and leveraging the Security App to explore malicious endpoint and network activities.

A key aspect of the success of hunt operations is how they are incorporated into traditional security and IT operations. Let's now explore how to enhance the protective posture of organizations. In this chapter, you'll learn about the incident response process, how threat hunters can fold into that process, how threat hunters can do more than just find adversaries, and finally, some useful third-party sources to help keep your skills sharp.

In this chapter, we're going to cover the following main topics:

- An overview of the incident response process
- Using threat hunting information to assist incident response
- Using IR and threat hunting to identify and prioritize improvements to the security posture
- Using external information to drive hunting techniques

Technical requirements

In this chapter, you will need to have access to the following:

- The Elastic and Windows virtual machines built in *Chapter 4*, *Building Your Hunting Lab – Part 1*
- A modern web browser with a UI

An overview of incident response

This book does not involve defining **Incident Response** (**IR**) processes or building an IR plan, although it is important to understand the basic tenets of IR.

In the same way that the Cyber Kill Chain is a process that adversaries use in their execution of campaigns, there is a companion process in dealing with incidents. There are various approaches to the IR processes, but, by and large, they fall into the following six steps:

1. Preparation
2. Detection and analysis
3. Containment
4. Eviction
5. Recovery
6. Lessons learned

Let's look at each of these steps in detail.

Preparation

The preparation phase is used to define the time between active intrusions. This isn't intended to be binary in that if you're performing an IR engagement, you shouldn't still be preparing for other intrusions.

This is the planning phase and includes ensuring that employees (both IT and non-IT) have been properly trained through security awareness, and that security policies and IR plans exist and have been tested with table-top exercises (practicing the execution of your policies and IR plans), and that security teams have the appropriate resources to defend your infrastructure.

Detection and analysis

In this phase, you have detected that an intrusion has occurred.

In this phase, you are trying to understand what happened: how did the adversary get in; how many additional entry points do they have; does this affect your ability to generate revenue for your organization or team; how many systems are impacted; how long has this event been going on; and how are they persisting in your environment.

It is vitally important to understand this phase before moving on to the next phase. If you do not take your time here, an adversary will simply regain control of your assets. At best, you'll have to start your IR process over; at worst, the adversary will regain access and you'll not know it.

Frequently, if you identify that intellectual property, regulatory data, or high-value data is at risk of being stolen, the response is to stop the observed events. This simply alerts the adversary that you're aware of their presence. It is important to fully understand all the ingress and egress points and the affected systems before any attempt to contain or evict the adversary is made.

Containment

The containment phase is when you make attempts to stop the adversary from advancing in their intrusion or break their Kill Chain. This is the first time that the adversary should be made aware that you've identified the intrusion.

When I think of containment, I think of stopping the following MITRE ATT&CK tactics that we defined in *Chapter 1*, *Introduction to Cyber Threat Intelligence, Analytical Models, and Frameworks*:

- Impact
- Exfiltration
- Command and control
- Collection
- Lateral movement
- Discovery
- Credential access
- Defense evasion
- Privilege escalation
- Persistence

In containment, speed is a factor. The adversary will likely go quiet and wait to retain access to the contested environment. If you have taken your time, observed as much as possible in the detection phase, and leveraged your practiced containment processes, it shouldn't matter what the adversary does because you're ready to stop them in their tracks. Examples of containment could be blocking adversary network traffic or removing infected systems from the network.

Eviction

Once you've contained the intrusion and stopped its progression, now is the time to remove their access from your network.

When I think about eviction, I think about stopping the following MITRE ATT&CK tactic:

- Execution

Most commonly, eviction is performed through rebuilding affected systems with known good media. For large campaigns, this can be impactful. Eviction can also be achieved through the very specific and methodical removal of adversarial control of your assets.

Recovery

The recovery phase is used for returning systems to production and, most importantly, validating that the entry points into your environment have been addressed.

When I think about recovery, I think about stopping the following MITRE ATT&CK tactic:

- Initial access

In the same way that the detection phase is often rushed, this phase is often overlooked. On multiple occasions, I have observed systems identified as needing to be rebuilt following an intrusion, or specific security controls needing to be put in place, but never completed. The infected systems are returned to service or holes in the system not closed and adversaries are able to regain access. When this happens, like rushing through discovery, at best you're starting over, and at worst, you don't know that you've been reinfected. Validating that the steps identified in the eviction phase have been carried out is paramount for this phase.

Lessons learned

Once an adversary has been evicted and systems have been to production, it is important to document any lessons that were observed to improve the overall protective posture of the environment. This includes additional investments in the security ecosystem, enhancements to the IR team's capabilities and tools, identifying and completing new training requirements, and documenting and rectifying process and documentation gaps. We'll talk more about how to identify those priorities later in the chapter.

This phase should incorporate improvements into the preparation phase to avoid having to perform multiple IR engagements for the same type of intrusion. Hopefully, lessons only have to be learned once.

In this section, we discussed a standard six-step IR process and briefly explored the different phases, to include several examples in each phase. While this book is not a deep dive into the IR process, this knowledge will assist in your understanding of how hunting can be leveraged in IR.

Using threat hunting information to assist IR

While performing hunt operations, you'll likely identify events that will require an IR operation. Beyond the identification of potential intrusion events, threat hunters have an additional context that can assist in the response efforts.

As we discussed in *Chapter 1*, *Introduction to Cyber Threat Intelligence, Analytical Models, and Frameworks*, there are several models that we can use to inform response decisions by the IR team members.

Hunt and IR teams frequently work together during a response. It is important to remember to navigate this situation sensitively. While it may appear to the responders or traditional security teams that you've identified a defect in their defense in the network, you should always underscore that you are part of the team. You have helped to identify a potential intrusion against a large network that has many entry and exit points. The intrusion wasn't necessarily accomplished through a lack of skill, passion, or knowledge by the defenders. If intrusions weren't largely successful, they wouldn't be so prevalent.

While there may be gaps and mistakes that led to a successful intrusion, you shouldn't allow yourself to be drawn into a debate about how this happened or who is to blame. There will be plenty of time to identify what led to an intrusion once the adversary has been evicted.

Next, let's explore a real-world supply chain example.

Supply chain compromise example

A supply chain compromise is when an adversary manipulates the environments or tools used to create or manage hardware or software. This is an attempt to manipulate trusted code and gain access to ecosystems that would be difficult to exploit traditionally.

As an example, in late 2020, a network monitoring company announced that an insecure password had led to a compromise. The compromise allowed malicious code to be inserted into the company's update process, resulting in a large-scale intrusion opportunity into their products. Famously, the company blamed an intern for setting this insecure password *(SolarWinds blaming intern for a leaked password is a symptom of 'security failures', SCMagazine*: `https://www.scmagazine.com/access-control/solarwinds-blaming-intern-for-leaked-password-is-symptom-of-security-failures`). This was widely received as an attempt to deflect the real issue – an insecure password had remained unchanged for 3 years.

Using this example, how can we, as analysts, assist in the IR operations? We can use the MITRE ATT&CK framework we discussed in *Chapter 1*, *Introduction to Cyber Threat Intelligence, Analytical Models, and Frameworks*, to identify some detection opportunities as well as assist in response operations.

MITRE ATT&CK framework

Looking at the supply chain compromise (*Supply Chain Compromise, MITRE*: `https://attack.mitre.org/techniques/T1195`), we can see that there are three sub-techniques that we can analyze to assist the IR team in focusing their responses:

- Compromise software dependencies and development tools
- Compromise the software supply chain
- Compromise the hardware supply chain

If we look at our example, the adversary abused the update process to introduce malicious code, so that fits squarely into the description of the compromise software supply chain sub-technique. *"Adversaries may manipulate application software prior to receipt by a final consumer for the purpose of data or system compromise."* (*Supply Chain Compromise: Compromise Software Supply Chain, MITRE*: `https://attack.mitre.org/techniques/T1195/002`).

Staying within the model, we can observe how adversaries have manipulated software, groups that have been observed using this sub-technique, and mitigations that can be leveraged to defend or respond to this type of intrusion. We can even continue to research the different adversaries to identify what types of industry verticals the adversary has been observed targeting.

> **Important note**
>
> Grouping and attribution is a tremendously vast and candidly debated topic in the cyber threat intelligence space. It is important to note that a single observed tactic, technique, or sub-technique is not enough information to attribute an intrusion with any level of confidence to a specific group. While this can be information to assist in hunting and response, it should not be used solely as attribution.

Using this example, we will be able to use our analysis techniques and tools to focus response and recovery operations on tactics, techniques, and sub-techniques that have been observed in previously reported campaigns. Additionally, we may be able to use the adversary observations to identify other techniques they may attempt that can inform other hunting and IR operations.

In this section, we explored an example of a supply chain compromise and showed how to use a threat model covered in *Chapter 1*, *Introduction to Cyber Threat Intelligence, Analytical Models, and Frameworks*, to identify additional threat hunting opportunities and provide information for the IR team.

Next, let's discuss how threat hunters can go beyond just finding adversaries and assist in the strengthening of an organization's overall proactive defense.

Prioritizing improvements to the security posture

Improving the security posture of an organization is daunting. What will make the biggest impact? Where to start? Will the changes stop adversaries? Are the necessary defensive technologies in place, but need to be reconfigured? The questions can go on and on. Organizations only have so many resources to apply, so it is important to prioritize these investments.

We discussed in the previous section how to use the MITRE ATT&CK™ model to assist not only in hunting, but also in IR operations. We can use additional models, such as the Lockheed Martin Cyber Kill Chain.

Lockheed Martin Cyber Kill Chain

As we discussed in *Chapter 1*, *Introduction to Cyber Threat Intelligence, Analytical Models, and Frameworks*, the Lockheed Martin Cyber Kill Chain is a response model for identifying activities that an adversary must conclude in order to complete a campaign. We can use this model to assist in the improvement of the security posture of an organization as we recover from an intrusion.

Using this model, we can work with security operations and incident responders to prioritize security and visibility changes to prevent the adversary (or other adversaries sharing their TTPs) from regaining access to a contested environment after they are successfully evicted.

The goal of using this model is to identify where the intrusion was detected and ensure that it wasn't *luck*, but the result of properly executed technology, analysis, and processes, and then move to the previous step to identify what can be improved to identify intrusions earlier:

Figure 10.1 – Lockheed Martin Kill Chain for prioritization

As you push the adversary back through this model, you'll discover that the earlier you can detect a specific intrusion, the more intrusions you'll detect overall.

As an example, if you can identify the adversary at the command and control phase using a specific **Domain Generation Algorithm** (**DGA**), that's great for detecting that specific adversary. However, if you can detect the adversary at the exploitation phase using a software exploit, you'll be able to prevent other adversaries from using that exploit, not just that specific adversary.

Using the hunting steps we've outlined in the previous chapters, you should try to understand how far into the Kill Chain you can observe an adversary and what resources could be applied for additional visibility earlier in the Kill Chain.

As an example, we can make a basic table that maps the phases of the Kill Chain to visibility that we currently have, whether the controls are effective at stopping an adversary, and what the data collection source is:

Phase	Have Visibility	Effective Controls	Collection Opportunities
Reconnaissance	No	No	N/A
Weaponization	No	No	Security research
Delivery	Yes	No	Email/TLS inspection, security awareness training, Elastic Beats/Endpoint
Exploitation	No	No	Email/TLS inspection, security awareness training, Elastic Beats/Endpoint
Installation	No	No	Email/TLS inspection, security awareness training, Elastic Beats/Endpoint
Command and Control	Yes	Yes	Email/TLS inspection, security awareness training, Elastic Beats/Endpoint
Actions on Objective	No	No	Email/TLS inspection, security awareness training, Elastic Beats/Endpoint

Table 10.1 – Example phase of effective control mapping

Using the preceding table, we can map this to an intrusion to identify what opportunities there are for prioritization. We can see that there is visibility in the delivery phase, but there are no effective controls. In this example, this could be that while they may have email and TLS inspection, the organization may not have the ability to block malicious emails or websites, or perhaps the security awareness training is ineffective.

Looking further, the organization has visibility and effective controls in the command and control phase, so if we go back one phase to installation, we see that they have no visibility and no effective controls. This could be a good opportunity to leverage the items in the collection opportunities column to gain visibility and then apply effective controls. Repeat this to move earlier and earlier into the Kill Chain.

In this section, we discussed that there are limited resources to apply to security posture improvements, so we can use hunting and threat models to identify existing visibility, effectiveness, and opportunities for improvement to make the largest impact on the adversary's ability to effectively complete campaign objectives.

Next, we'll cover some external sources that can help augment your threat hunting skills and domain knowledge.

Using external information to drive hunting techniques

It is important to remember that threat hunting is a journey. One of the most useful resources in threat hunting are your peers in the infosec community. While you're spending your time honing your skills to learn how adversaries are attempting to infiltrate your network, there are millions of other practitioners who are doing the same thing.

These practitioners don't just do their work and move on; they want to share the tactics they've identified, show how they've mitigated them, and share lessons they've learned along the way.

Use the open source community to drive and enhance your war chest of techniques.

Not only do defenders share their knowledge, but penetration testers, hackers, and security engineers publish their research and exploits. This information should also be used to learn more about what malicious adversaries may be attempting to use during their next campaign.

There are so many great examples of professional security organizations that frequently release bleeding-edge tactics and campaigns. Using the teachings in these publications, you can learn about new adversary techniques, emerging threats, and a litany of indicators to search for in your environment using what we learned in *Chapter 8*, *The Elastic Security App*, and *Chapter 9*, *Using Kibana to Pivot through Data to Find Adversaries*.

Again, there are many great resources, but a few that put out high-caliber and reliable research include the following:

- *Palo Alto's Unit 42*: `https://unit42.paloaltonetworks.com`
- *FireEye's Threat Research Blog*: `https://www.fireeye.com/blog.html`
- *Cisco's Talos*: `https://talosintelligence.com`
- *The Hackers Choice*: `https://www.thc.org`
- *Cybersecurity Infrastructure Security Agency (CISA – US government)*: `https://www.cisa.gov`
- *Information Sharing and Analysis Centers (ISAC)*: `https://www.nationalisacs.org`
- *Exploit DB*: `https://www.exploit-db.com/`