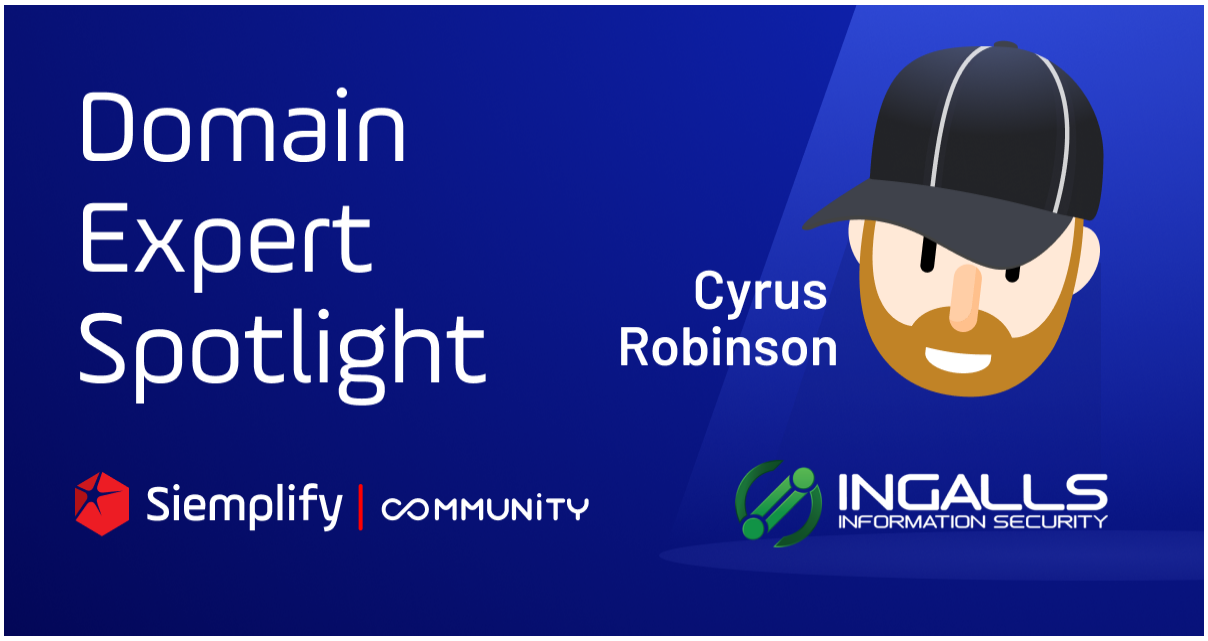
Noise Reduction

Cyrus Robinson

Cybersecurity Professional - SOC Director - IR Lead at Ingalls and Siemplify SOAR Community domain expert



Introduction

Have you ever noticed trees that are marked with spray paint? Now, I'm no tree spray paint marking expert, but it's my understanding that different colors or symbols can signify different things such as trees that need to be removed, are damaged but may survive, need to be treated, or are a danger to public utilities. If you're walking through a neighborhood, trees with these markings can be easily identified along the street, but this gets much trickier when you're walking through a forest and need to find the trees that need to be addressed. I know, I know! This is supposed to be a SecOps article, not an arborist article, but hear me out. To entertain an analogy here, the trees represent alerts coming into your SOAR platform (or your "forest"), and the spray paint markings are actionable IoCs.

There are currently over 250 supported integrations in the Siemplify Marketplace, and with Siemplify's IDE, the potential for nearly an endless number of custom-developed integrations and log sources. This has the potential for giving unprecedented visibility into your environment, and the ability to integrate and then orchestrate and automate data from so many different sources can be a SOC analyst's dream. However, without implementing a tuning and noise reduction strategy, it can leave analysts wandering through a forest of alerts. When this happens, a SOC analyst can begin to experience what we refer to as Alert Fatigue or Alert Blindness.

So how can a SecOps team improve their visibility while eliminating the noise?

Clearing Trees: Internal Noise Reduction within Siemplify

When an alert is brought into Siemplify, Siemplify compares the defined entities of the alert to other alerts in the environment within a specified timeframe (half-hour increments up to a max of 24 hours). Siemplify's orchestration will then group the alert into a case with other alerts (up to 30 per case) that have shared entities in common within that timeframe.

So, as a simple example, let's assume that you have the following alerts in your environment within your specified grouping timeframe:

- Alert 1: Suspicious Network Connection EDR Alert
    - Source Hostname: Computer1
    - Parent Process: TotallyNotABackdoorWinkWinkNudgeNudge.exe

    - Parent Process (MD5) Hash: 075ff2fb2e33a319e56a8955fade154e
    - Destination IP Address: 101.100.146[.]147 (which happens to be a known TOR exit node).
- Alert 2: Suspicious Network Connection EDR Alert
    - Source Hostname: Computer2
    - Parent Process: TotallyNotABackdoorWinkWinkNudgeNudge.exe
    - Parent Process (MD5) Hash: 075ff2fb2e33a319e56a8955fade154e
    - Destination IP Address: 101.100.146[.]147
- Alert 3: Known TOR Exit Node IDS Alert
    - Source Hostname: Server3
    - Source IP Address: 10.1.1.1 (Server3)
    - Destination IP Address: 101.100.146[.]147

If you have your entities defined appropriately, Siemplify would group these 3 alerts into a single case based on the common entity, 101.100.146[.]147. Based on this data an analyst might rightly wonder whether Server3 also has TotallyNotABackdoorWinkWinkNudgeNudge.exe, and if so why was there no EDR alert for TotallyNotABackdoorWinkWinkNudgeNudge.exe on that device. This would be a good place to begin investigating, but we all know that the reality is that alerts tend to be much noisier than this.

Instead of a single EDR alert on Computer1, you may have 10 alerts. Instead of a single EDR alert on Computer2, you may have 15 alerts. As a result, if your maximum alerts per case are set to 25 alerts, then the analyst may not see the relationship between the traffic on Server3. That may end up looking something like this…

- Alert 1: Suspicious Network Connection EDR Alert
    - Source Hostname: Computer1
    - Parent Process: TotallyNotABackdoorWinkWinkNudgeNudge.exe
    - Parent Process (MD5) Hash: 075ff2fb2e33a319e56a8955fade154e
    - Destination IP Address: 101.100.146[.]147 (which happens to be a known TOR exit node).
- Alert 2: Suspicious Network Connection EDR Alert
    - Source Hostname: Computer1
    - Parent Process: TotallyNotABackdoorWinkWinkNudgeNudge.exe
    - Parent Process (MD5) Hash: 075ff2fb2e33a319e56a8955fade154e
    - Destination IP Address: 101.100.146[.]147
- Alert 3: Suspicious Network Connection EDR Alert

    - Source Hostname: Computer2
    - Parent Process: TotallyNotABackdoorWinkWinkNudgeNudge.exe
    - Parent Process (MD5) Hash: 075ff2fb2e33a319e56a8955fade154e
    - Destination IP Address: 101.100.146[.]147

  ...Alerts 4 through 25 are duplicates of the above alerts.
- Alert 26: Known TOR Exit Node Traffic IDS Alert
    - Source Hostname: Server3
    - Source IP Address: 10.1.1.1 (Server3)
    - Destination IP Address: 101.100.146[.]147

This is where some internal noise reduction within Siemplify can come into play! Different SOCs can use different strategies to accomplish this, but I'm going to give a high-level example of how this can be done using Siemplify's Custom Lists.

1. Alert 1 comes into Siemplify.
2. During your noise reduction Siemplify playbook, you check whether your EDR_CustomList already has a duplicate alert.
3. A duplicate alert does not exist, so you use placeholders to add an item identifier to the EDR_CustomList for that alert that looks something like this:
   [Event.SoureHostname].[Event.ParentProcessHash].[Event.DestinationIP] which is actually added to the EDR_CustomList as
   Computer1.075ff2fb2e33a319e56a8955fade154e.101.100.146[.]147
4. Alert 2, which is identical to Alert 1 except for the timestamp being a second later, comes into Siemplify.
5. During your noise reduction Siemplify playbook, you check whether the EDR_CustomList already has a duplicate alert. It does because Alert 1 was just added to EDR_CustomList.
6. Siemplify automation closes Alert 2.
7. Alert 3, which is the first Computer2 EDR alert comes into Siemplify.
8. During your noise reduction Siemplify playbook, you check whether the EDR_CustomList already has a duplicate alert. The Source Hostname in Alert 3 is different from the Source Hostname in Alert 1, so there is no duplicate entry in the EDR_CustomList for this alert yet.
9. ***Computer2.075ff2fb2e33a319e56a8955fade154e.101.100.146[.]147*** is added to the EDR_CustomList.
10. Alerts 4 through 25 are all duplicates of one of these two entries and will be automatically closed.
11. Alert 26, the Known TOR Exit Node Traffic IDS alert, comes into Siemplify.

12. Alert 26 is grouped into a case with Alerts 1 and 3 because the duplicate EDR alerts were all automatically closed by Siemplify.

In this example, by using Siemplify's automation and Custom Lists, we have eliminated 23 duplicate alerts and have allowed Siemplify's orchestration to group these 3 alerts into a single case rather than having them potentially divided into 2 separate cases. This gives the analyst the visibility needed to identify the truly unique events occurring within the environment without having to wander aimlessly through a forest of alerts looking for events that stand out. Depending on the type of alerts in your environment, you may need to be more or less granular with the identifiers added to the Custom List, and one thing that does need to be accounted for with this strategy is ensuring that you include automation to remove the identifiers from the Custom List towards the end of your playbook (or after a specific time window) to ensure that important alerts aren't unintentionally automatically closed in the future.

Pruning Trees: Noise Reduction in your Alerting Pipeline

Leaning on our analogy a bit, another way to reduce noise would be to prune the trees before the canopy blocks the sunlight and visibility within the forest of alerts. While the Internal Noise Reduction method listed above is a great opportunity to reduce noise once the alerts come into Siemplify, the closer to the source of the alerts that noise is reduced, the better.

So, what are the benefits of tuning the noise out closer to the source of the alerts? Depending on how your Siemplify platform is configured (SaaS vs on-prem) or how efficient your automation and playbooks are, even after noise reduction, there's still potential for a large number of alerts being attached to a large number of cases within Siemplify. The flexibility of the Siemplify platform allows for alerts to be brought in from a wide variety of ways. Your organization might be feeding alerts to a SIEM use Siemplify Marketplace Integrations and Connectors for that SIEM to generate alerts within Siemplify. Your organization might be using marketplace or custom integrations to generate alerts from remote agents within Siemplify. Your organization might be using a custom integration and the Siemplify IDE to feed alerts from a SIEM into Siemplify. Whatever the case, if there are alerts that can be pruned before being brought into Siemplify, this will not only increase the visibility of high-value alerts for your analysts but would also improve the performance of your Siemplify platform as a whole.

An example of this might be a known good, legitimate process that frequently triggers a false-positive EDR detection. For example, perhaps you have Sysmon installed on endpoints within your organization, and your EDR platform flags legitimate network connections to/from Sysmon as suspicious. If your integration, SIEM, or alert pipeline allows for you to out the "suspicious network

connections" from a file image of c:\windows\sysmon.exe (or wherever you have sysmon.exe installed), then you can effectively increase the visibility of your analyst with alerts that need to be investigated while also reducing the performance impact within your Siemplify platform.

Clearing the Brush: Tuning within your Endpoint Tools

Removing the unnecessary trees (alerts) from the forest (Siemplify) is a great place to start, and pruning healthy trees before their canopy obscures your visibility makes identifying the spray paint markings (actionable IoCs) a lot easier. Building on the second strategy, another strategy is to clear the brush and overgrowth reducing visibility in your forest! This noise reduction strategy is really an extension of the alerting pipeline strategy.

As previously mentioned, the closer to the source of the alerts that noise is reduced, the better. So taking the previous strategy a step further, the best way to reduce noise and to improve visibility is to reduce the noise in your endpoint tools themselves. Using the same sysmon.exe EDR detection example described above, if this detection can be tuned in your EDR platform before the alert is sent into your pipeline in the first place, then this can improve visibility and performance within your SIEM or alert pipeline and also within Siemplify. Another example would be if you know have a known false positive malware quarantine, marking this as a false positive or Safelisting the file in your organization's Antivirus (AV) console can reduce the number of alerts for this file making their way into Siemplify, allowing your analysts to focus on alerts that matter, alerts that have actionable IoCs. In fact, your Siemplify integration for your AV solution may allow this from directly within the Siemplify platform itself! If you are an MSSP or an organization with multiple environments in Siemplify you may be able to leverage your AV and EDR solution APIs to do this for all of the environments that your SecOps teams monitor to clear out the brush and overgrowth within your forest.

Every environment is different, and these strategies may not match your specific use cases and requirements. However, the robust integrations, playbooks, and IDE built into the Siemplify platform allow SecOps teams to customize their own tuning and noise reduction strategy to fit their needs, and hopefully, this article can provide teams with a starting point for that effort. Having increased visibility can provide analysts with the insight needed to make more well-informed determinations with alerts in the environments they monitor, but a noise reduction strategy is an integral part of ensuring that analysts don't miss the forest for the trees.