Managing cyclicality in metric values

It is relatively common for the values of a metric to exhibit a weekly periodicity. For example, if most of the users of an application live in a certain timezone, and the metric is influenced by the quantity of user activity, then normal Monday afternoons will tend to look similar to each other, and different from Thursday evenings or Sunday mornings. For such metrics, sensible static thresholds may be difficult to define, as the notion of “normal behavior” itself changes with time.

Use timeshifting

Timeshifting is the best strategy to deal with cyclicality in metric values. For example, you can divide the 30-minute rolling mean of a metric by the 30-minute rolling mean of the metric shifted by one week. You’d then expect the value of the quotient to be relatively close to 1, and you alert when the value is very different from 1, with the exact discrepancy determined by your tolerance for deviation.

While this dynamic threshold is generally better than comparing to a static threshold, comparing current values to a single timeshifted window has the potential to be quite noisy. Alerts are likely to “echo” one week after they happen. For example, you might see a flurry of activity due to a large customer being onboarded or a change in metrics due to a code deployment, and either would cause your alerts to fire as intended. Unfortunately, unless similar events happen precisely one week later, you will receive alerts again the following week because normal activity does not match the behavior observed during the event the week before.

To deal with this phenomenon, use multiple timeshifted windows and discard outliers in order to calculate the dynamic threshold to compare against. For example,

1. Calculate 30-minute rolling means one, two, three, and four weeks ago.
2. Discard the highest and lowest values.
3. Alert when the current value is greater than 20%, for example, above the larger remaining value, or less than 20% below the smaller remaining value.

With more data, you can more confidently construct a band of normal values. Unless two past outliers align perfectly (in this example, that means they are spaced exactly one week apart), this strategy eliminates their influence on the dynamic threshold.

Variations on this strategy can be obtained by summarizing past data in different ways. For example, instead of constructing a +/- 20 percent band around a rolling mean, construct a band using standard deviations. This replaces the static value of 20% with a dynamic estimate of the typical spread of values. While you are free to implement this strategy yourself, this routine is packaged in the SignalFlow library and goes by the name “Historical Anomaly” in the detector UI.
Next steps

These additional Splunk resources might help you understand and implement these recommendations:

- Splunk Docs: Timeshift
- Signal Flow Library: against_periods.flow
- Blog: A deep dive into built-in anomaly detection: How the algorithm works