All.Net Analyst Report and Newsletter

Welcome to our Analyst Report and Newsletter

Escalating and de-escalating incident response

In recording one of the lectures for one of my online classes late last year I decided to describe the escalation and de-escalation process associated with detecting and responding to network attacks. The specifics are specific of course, but the general approach is worth describing. So here we go.

The structure of detection and response

In substantial entities and those who work with others, there are effectively three levels of the detection and response regimen.

- **Local** sensor and actuator systems:
  - Detect a wide range of **Events**, some of which they
    - Store for a limited time locally
    - Forward to the next higher echelon
    - Act on locally
  - Change their storage, action, and reporting regimens per entity-level criteria
- **Entity** level aggregation and analysis sites:
  - Receive Event information for Local sensor systems which they
    - Selectively store for a limited time
    - Analyze and
      - Aggregate into "**incidents**" that they manage
      - Selectively report on to the next higher echelon
      - Gather more stored data from local sites
  - Update local sites to change their local detection and response
  - Change their approaches based on information from Global entities
- **Global** information sharing and analysis entities:
  - Receive information from Entities
  - Analyze the information to provide wider scale understanding
  - Transmit resulting information to Entities

Of course this sometimes happens in 4 or more levels with multiple Entity levels, and the overall picture is a bit more complicated in some cases with Global and Entities interacting, but for the purposes of this analysis, this description is adequate.

Urgency and importance

The mechanisms of detection, transmission, analysis, and response cannot store and share all of the information. There is too much of it. For that reason, the system has attention limits. Increased attention (and retention) are associated with increased perceived urgency and importance.

- Importance stems from the perceived potential consequences of the Event or Incident
  - Higher consequences → more important
- Urgency is associated with the perceived time before consequences take place
  - Less time to respond to mitigate consequences → more urgent

Of course this is driven in turn by the time to respond and the change in effect from response.

The requirements of such a system

For such a system to operate properly, it is necessary to:

- Model consequences associated with Events sequences and Incidents
  - Consequence modeling is necessary to associate importance
- Model changes in consequences associated with available response alternatives
  - Consequence modeling not taking response into effect renders time to respond irrelevant
- Model time to respond with associated effect
  - Time to respond drives latest effective time to act and thus urgency
- Model available resources and the effect of their use on the other parameters
  - With limited total resources, responses must be earlier to limit the effects of subsequent events and incidents in averting undesired consequences
- Provide the means to analyze all of this to determine urgency and importance
  - All of this has to be taken into account in order to prevent systemic failures

Of course few if any such systems meet these requirements. And for that reason we continue to have failures. But perhaps more importantly, resource limitations on detection and response systems imply the need to trade off. Spending more on incident detection, analysis, and response means less is available for seeking rewards.

Escalation and De-escalation

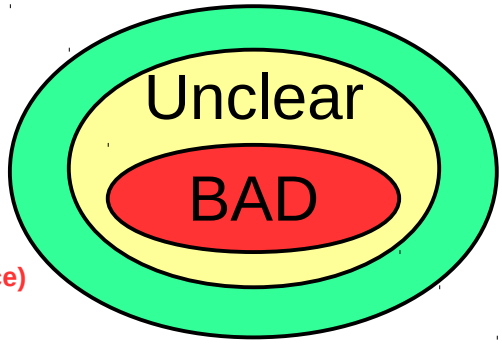
Events and incidents are not static, in that they may have different urgency and importance over time. In particular, as the time to consequence shortens, urgency necessarily increases. But many Event sequences that can lead to undesired consequences do not do so, and as they move further in time or away from the capability to produce these consequences, their urgency and importance should decrease. In order to deal with this effectively, it is necessary to keep measuring the situation so that Events and Incidents are kept in context. Response is then driven by the most urgent and important taking precedence with some resource allocation process to prevent deadlock. At some point, action must be taken to interdict.

An example may help

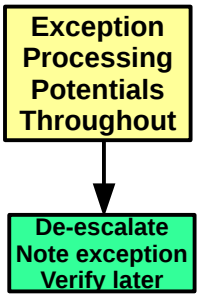
Or not… Here is one from the course. We start with the rational:

- **Suppose I think telnet to login to port 23 is bad**
  - **Suppose it is a known attack method**
    - **Suppose serious negative consequences if they login**
      - **Suppose we know that "login:" is often the login prompt**
        - **Suppose we can react automatically in 0.1 seconds**
        - **→ We need to detect and decide with >0.1 s left**
      - **We can look for the "login:" prompt**
        - **And watch what follows**
          - **Detection gets more certain as we watch longer**
            - **Till we must act or suffer potential consequences**
        - **But if no "Valid" thing follows, we can alarm now**
      - **There may be other prompts**
        - **Suppose we watch for other activity as well**
          - **Detecting things that might produce the consequences**
            - **Detection gets better as we watch longer**
              - **Till we must act or suffer potential consequences**
  - **We can watch port 23**
    - **Detection gets more certain as we watch longer**

Now that we have the rational, we can support escalation and de-escalation of detection:



- **Suppose we detect TCP port 23 anything**
  - **Notify (low urgency and importance)**
  - **Watch further**
    - **If "login:"**
      - **Notify (higher urgency and importance)**
      - **Watch further**
        - **If sequence gets low on time**
          - **Notify (higher urgency and importance)**
          - **Watch further**
            - **If time gets to critical**
              - **Notify (critical urgency and relevant importance)**
              - **Watch further**
                - **If bad things happen**
                  - **Notify (harm underway – or done)**
                  - **…**
                - **If NOT →**
                  - **Notify of near miss for future investigation**
            - **If NOT →**
              - **Notify of possible attempt/indicator/false positive**
        - **If NOT → watch further**
      - **…**



Note that this is ONLY the process for the detection side of things. The response process can act or not on these notifications based on available resources and importance. Most things are unclear until we run out of time.

Gaining clarity and extending our time

One of the ways to gain clarity and extend the time to respond more definitively is through the use of delays and deceptions. This may, of course, produce reduced overall performance, but on the other hand, perhaps performance reduction for more dangerous things is a good idea in some cases.

- **Delay is a good thing:** A systems administrator is using login to fix a problem requiring root privileges on a critical server and it is worth the risk doing it outside of the normal maintenance window. However, a delay of a minute or two to call them and verify that it is them and that they have permission and that it is a reasonable risk to take is no real problem. On the other hand, if this was an attacker, there could be serious harm, and thus the delay allowed an effective response to block the bad thing.
- **Delay a bad thing:** An emergency response to a critical operational technology incident requires that a valve control be overridden before there is an explosion. The delay of 30-90 seconds for a phone call to verify will stop the emergency response from preventing the explosion, killing the responder and destroying the facility. On the other hand, if this was a bad actor in the same location, at this point, the explosion was going to happen and kill them unless they stopped it in time. The delay cannot improve the situation in either case, but perhaps the bad actor will act to save their own life.

The challenge and the solution is to have the proper failsafe modes in place when the consequences justify them.

How far do you go?

That last point is key. The consequences have to justify the level of effort to even figure all of this out. Most detection and response processes do not go to this level of depth, nor should they. But this is done in near-real-time as assessments are updated in process for terrorist attack. This is because the consequences are indeed high. And you would imagine that the same is true for nuclear attack and defense, except of course that the time frames for ICBMs is short enough that the decisions have to be made in advance with very limited time for national command authority to choose between limited alternatives.

In the cyber-arena, where time is often very short, consequences should be considered in advance, and automated decisions are called for in many cases. Otherwise, events will overtake the ability to respond in time and the consequences will be realized.

Thus we have a general architecture for creating and applying escalation and de-escalation with the bases for making architectural decisions (consequence, time, cost, and structure).

Conclusions

Escalation and de-escalation along with adaptation of sensors, analysis, decision-making, and response actions are called for at increasing levels of detail and certainty as the consequences of events and incidents rise.

The methodology described here is just that. A method for undertaking the process. But the application of the method is necessary on a case-by-case basis. The high cost of applying the method at increased granularity and certainty implies a need to understand consequences before undertaking the effort. Based on consequence and time, the identified structure can be realized within cost constraints to the level of granularity appropriate.

