

Scoring Alerts from Threat Detection Technologies

Robert C. Schrag^{*}, Masami Takikawa[†], Paul Goger[#], and James Eilbert[%]

^{†*}Information Extraction and Transport, Inc., {schrag, takikawa}@iet.com

^{*}1911 N Fort Myer Dr, Suite 600, Arlington, VA 22209 USA

[†]1600 SW Western Blvd, Suite 185, Corvallis, OR 97333 USA

[#]Metron, Inc., 19111 Freedom Dr, Suite 800, Reston, VA 20190 USA, goger@metsci.com

[%]Lockheed-Martin, Inc. Advanced Technology Laboratories, jeilbert@atl.lmco.com

3 Executive Campus, 6th Floor, Cherry Hill, NJ 08002 USA

Abstract

We describe methods to score alerts—hypotheses about suspected impending threat events that are issued, based on incrementally presented, time-stamped evidence, before the events occur. Our threat events (and thus alerts) have significant object-oriented structure. The alert scoring methods exploit related methods to score precision, recall, and F-value for structured threat hypotheses when such evidence is processed by threat detection technologies in a batch, forensic mode. We present a (deemed-impractical) idealized approach and derivative practical variants. The implemented approach is part of a performance evaluation laboratory (PE Lab) that we have applied during a multi-year, multi-contractor Government research program.

Introduction

Schrag and Takikawa (2006) describe methods to score structured hypotheses from threat detection technologies that fuse evidence from massive data streams. These methods generalize count-based metrics (precision, recall, F-value, area under curve) traditional in information retrieval to accommodate partial matching over structured case hypothesis objects with weighted attributes. They also include cost-based metrics that use count-based scores to pro-rate specified per-case costs of false-positive (F+) and false-negative (F-) threat reporting. Threat detection technologies may process evidence in either batch, forensic mode to tender threat event hypotheses retrospectively or in incremental, warning mode to tender threat event hypotheses prospectively (as alerts). This paper focuses on the cost-based metrics used to score alerts.

Our scoring methods are implemented in a performance evaluation laboratory (PE Lab) for threat detection technologies that also includes a massively parameterized dataset generator to support systematic exploration of threat detection performance boundaries, as summarized by Schrag (2006). The PE Lab is schematized in Figure 1, where square-cornered boxes represent artifacts, round-cornered boxes represent processes, and arrows represent flow of artifacts. The threat detection component—assumed to employ link discovery (LD) technology and

also referred to here as an LD component—is rendered 3-dimensionally to indicate its status outside of the PE Lab proper (as the technology under test).

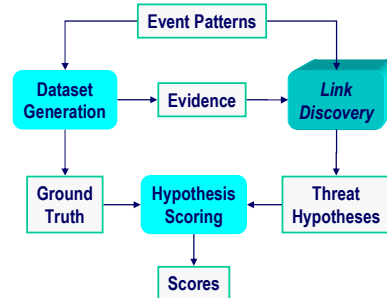


Figure 1: PE Lab schematic

Synthetic dataset generation creates evidence used to challenge LD and (synthetic) ground truth used in scoring LD's hypotheses. LD processes evidence to hypothesize threat phenomena. Generation uses simulation driven by discrete, stochastic event patterns that also are provided to LD. Hypothesis scoring compares technologies' output hypotheses to ground truth.

Counter-terrorism Threat Domain

The summary in this section follows that of Schrag and Takikawa (2006). It is included here for the sake of completeness.

Figure 2 exhibits some real-world motivation behind the abstract, artificial world challenge problem domain we have developed.

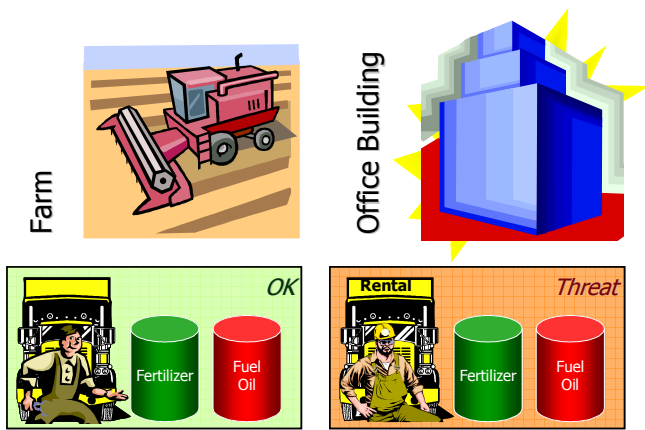


Figure 2: Real-world motivation for challenge problem

On the left-hand side of Figure 2, “Farmer Fred” buys fertilizer and fuel oil and transports these *via* truck to his farm. He applies the fertilizer using his tractor which (along with his truck) burns the fuel oil. (Fred is an honest, hard-working man.) On the right-hand side, “Demolition Dan” acquires the same resources but mixes them into a slurry that he transports (*via* rental truck) to the basement of an office building. (Dan is up to no good.)

In the artificial world, capabilities (like farming and demolition) and resources (like fertilizer and fuel oil) are mapped to abstract elements that individuals can possess intrinsically or acquire. Infrastructure elements (like office buildings) are mapped to “targets” that support both legitimate/productive and destructive modes of use or “exploitation.” Non-threat and threat individuals (like Fred and Dan) each may belong to any of various groups whose members collaborate in sub-group teams towards different goals. Exploitations play out the general scheme of Figure 3.

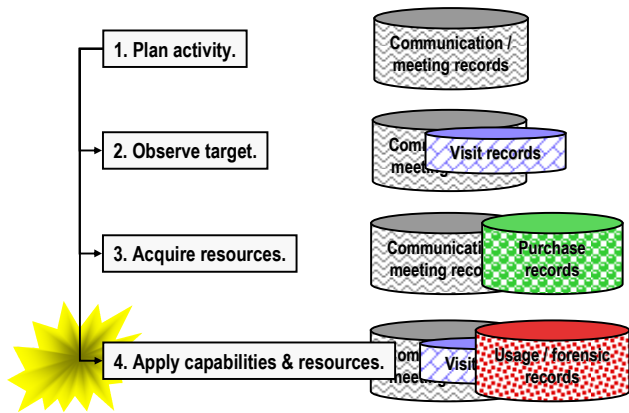


Figure 3: Generic exploitation scheme

The exploitation scheme on the left-hand side of Figure 3 unfolds through several levels of task decomposition (illustrated in Figure 4), bottoming out in transactions with record types indicated on the right-hand side of Figure 3. In a threat exploitation, the final, consummation phase—in which capabilities and resources are (destructively) applied to the target—defines the time by which alerting must

occur to be at all effective. Transactions appearing in incrementally presented, time-stamped evidence are the primary basis LD has for issuing alerts.

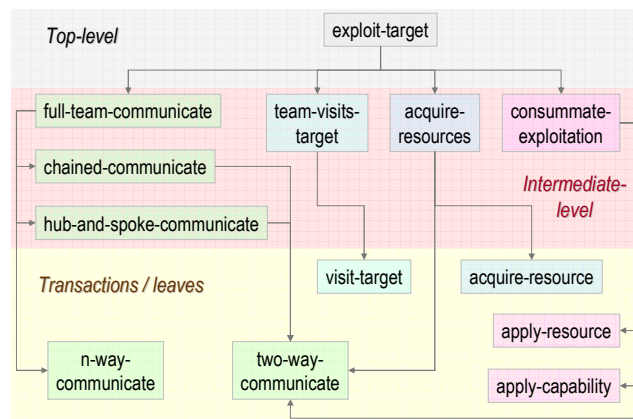


Figure 4: Invocation relationships among event generation patterns

The challenge to LD is to identify and report threat cases—top-level objects with attributes and values summarizing threat phenomena. The case types include threat actors (groups, individuals, and their aliases) and threat exploitation events. In the task to which we limit our attention here, LD should (if possible) issue an alert before the consummating attack. LD is given information about the underlying artificial world that is relatively complete and about events (including transactions) and actors that is only partial and may be corrupted.

Hypothesis Scoring Methods

Figure 5 depicts the generic scoring scheme that serves in retrospective hypothesis scoring and that we also exploit in alert scoring. We first compute scores for count-based metrics (precision, recall, and F-value), then for cost-based metrics (used in alert scoring).

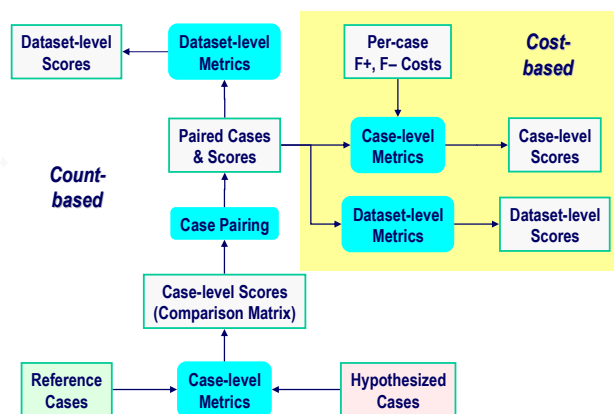


Figure 5: Generic scoring scheme

In Figure 5, the reference cases are summaries computed from ground truth, the hypothesized cases are from LD. Case objects have significant structure, and we want to

credit LD when hypothesized cases approximately match reference cases. Match quality is determined by case comparison. When a hypothesized case object's existence has been inferred from lower-level evidence, we can decide which reference case to pair the hypothesized one with only by comparing the hypothesized with all relevant reference cases—on the basis of their attribute values. We store comparison results for the candidate pairs in a matrix. With inexact matching, it also can be ambiguous which of the one-to-one mappings admitted by the candidate pairs should be selected, so we use an algorithm that optimizes dataset-level scores. Given these pairs, we compute scores for count-based metrics based on the precision, recall, and F-value metrics traditional in information retrieval. Using the scores for count-based metrics and specified per-case costs of F+ and F- reporting, we additionally compute scores for cost-based metrics. In alert scoring, we use the “assessed cost” metric to discount these costs based on a hypothesized case's quality, as follows.

- To the extent (corresponding to the factor $(1 - \mathcal{P})$) that the hypothesis includes content beyond that of its paired reference case, we assess the associated F+ cost.
- To the extent (corresponding to the factor $(1 - \mathcal{R})$) that the hypothesis lacks content included in its paired reference case, we assess the associated F- cost.

The remainder of this section summarizes the concepts introduced above that are important in alert scoring. Schrag and Takikawa (2006) provide a similar summary and additional details. We include this information here for the sake of completeness.

Figure 6 illustrates traditional precision and recall (which presume exact matching between hypothesized and reference items).

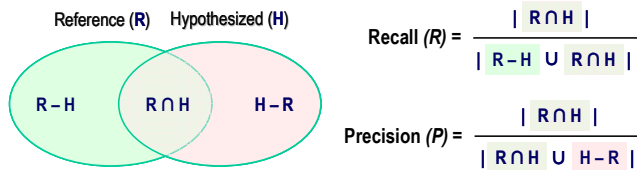


Figure 6: Traditional precision and recall

Traditionally, recall R is the number of valid hypotheses divided by the number of detection targets (the required number of valid hypotheses). Precision P is the number of valid hypotheses divided by the number of all hypotheses.

Because a single metric to summarize the values of recall and precision is frequently useful, traditionally an appeal is made to $F\text{-value} = 2PR / (P + R)$ —the harmonic mean of precision and recall. (When both precision and recall are zero, we define F-value as zero.)

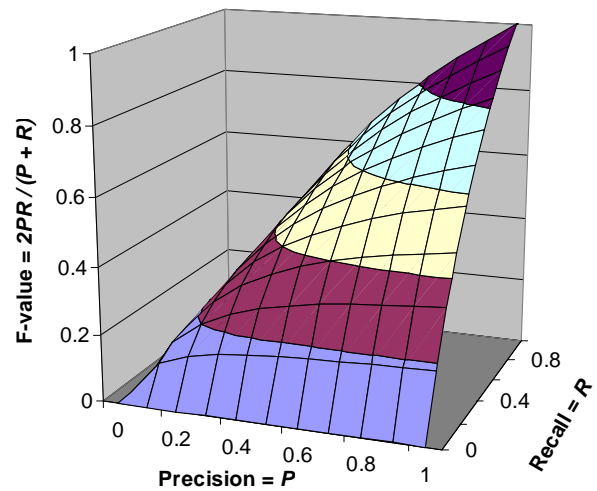


Figure 7: The F-value surface

As shown in Figure 7, F-value (also known as “F-score” or “F-measure”) has the same extremes as a simple average of precision and recall but discounts differences between them more steeply (not as steeply as $\min(P, R)$).

To accommodate inexact matching over structured case objects, we define object-oriented versions of precision, recall, and F-value, as illustrated in Figure 8.

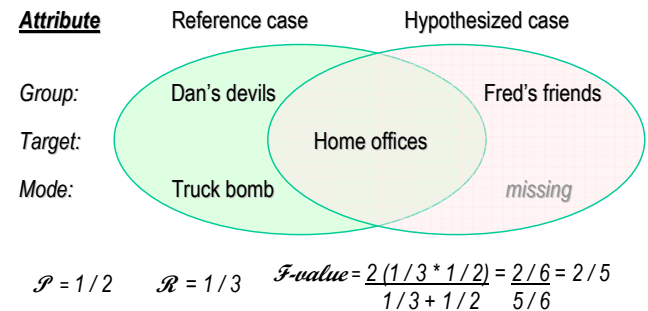


Figure 8: Object-oriented count-based metrics

Of the three attribute values in the reference case of Figure 8, the hypothesized case agrees only for the target attribute, so the object-oriented recall score \mathcal{R} is 1/3. Of the two attributes included in the hypothesis, only one agrees with the reference, so the object-oriented precision score \mathcal{P} is 1/2. The corresponding object-oriented F-value ($F\text{-value}$) is 2/5, as shown.

Case pairing determines which detector-hypothesized case objects to pair with which ground-truth, reference case objects—since this may not be obvious, as illustrated in Figure 9.

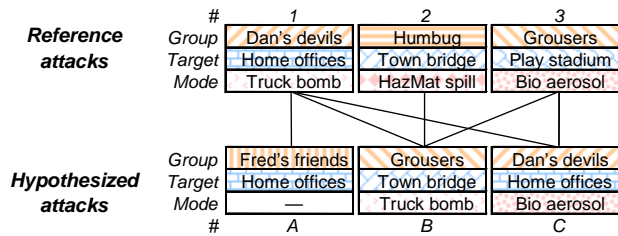


Figure 9: Case pairing issue

In Figure 9, we have three reference and three hypothesized attack cases. (Reference Case 1 and Hypothesized Case A correspond to the pairing of Figure 8.) Links appear in the bipartite graph between reference and hypothesized cases wherever these share one or more attributes. Figure 10 illustrates how we perform one-to-one case pairing using a matrix over all possible pairings.

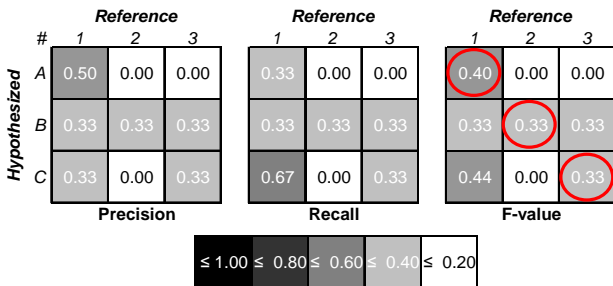


Figure 10: Case pairing matrix and metrics

In Figure 10, we compute per-pair object-oriented precision, recall, and F-value (as in Figure 8). Then we use an optimization algorithm to select (red-circled) pairs leading to the greatest average object-oriented F-value. Figure 11 presents the corresponding matched version of the bipartite case graph introduced in Figure 9.

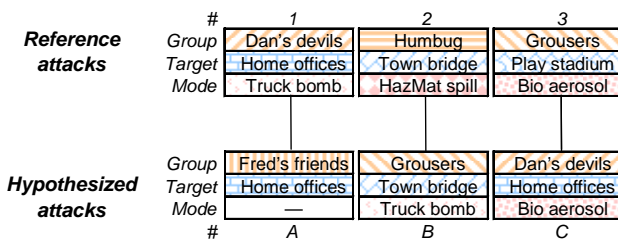


Figure 11: Selected case pairs

The case pairing process additionally requires pairs to exhibit adequate match quality. A minimum F-value threshold (Schrag and Takikawa, 2006) and any per-case type logical conditions (or “gates”) must be satisfied to admit a pair to the comparison matrix. For an event hypothesis (even an alert) to be considered for pairing with a given reference case, a specified gate requires that the interval bounded by the hypothesized exploitation event’s start and end times must overlap with the corresponding interval for the reference case.

Related Work

Related work appears to be limited. Others have evaluated event prediction where exact hypothesized-to-reference case matching is appropriate. Weiss and Hirsh (1998) specialize precision to discount temporally close false-positive hypotheses. Létourneau, Famili, and Matwin (1999) apply a nonmonotonic timeliness function to determine rewards and penalties for true-positive and false-positive predictions of appropriate aircraft component replacement times. Different metrics are certainly appropriate in different contexts, and we believe the accommodation of inexactly matching hypothesized and reference cases and attendant case pairing entail issues for structured threat alert scoring that others have not addressed.

The overall hypothesis evaluation framework of Mahoney *et al.* (2000) is close in spirit to our own. They suggest some overall strategies for comparing hypothesized vs. reference situation histories and note the requirement for timeliness, without directly addressing threat event prediction.

Alert Scoring

Methods for scoring alerts must address requirements corresponding to the following practical questions.

Representation: How should alerts be represented?

Scoring basis: On what basis should alerts be scored?

Case pairing: How should alerts be paired with reference cases for scoring?

Case comparison: How should the quality of match between an alert and a paired reference case be reflected in scoring?

Timeliness: How should an alert’s timeliness (or tardiness) with respect to a paired reference case be rewarded (or penalized)?

Supersession: When later alerts are designated to supersede earlier ones (because LD has changed its opinion about the impending event) or when earlier-tendered alerts have been retracted, which one(s) should be scored, how?

Submission: Should LD submit alerts spontaneously or only when solicited (*e.g.*, at designated intervals)?

In the following two subsections, we describe, first, a (deemed-impractical) idealized approach and, second, some derivative practical variants, including the one currently implemented in PE Lab.

Idealized Alert Scoring

Our idealized approach addresses the alert scoring requirements as follows.

Representation: Represent alerts just like retrospectively tendered hypotheses and just like reference cases—as objects with attributes.

Scoring basis: Use cost-based scoring—specifically, assessed cost, in which alerts that are paired with reference cases incur F+ and F− costs discounted according to

object-oriented precision and recall, respectively. Alerts that are not paired with reference cases (or that are late but supersede a paired alert, as explained under *Timeliness and supersession* below) incur the full F+ cost. Ideally, compute a cost at every simulation time point and sum these costs to obtain LD’s alerting score for the full dataset. Refer to the F+ cost, F− cost, and attribute weights provided in the alert scoring specification.

Case pairing: At each simulation time point, compare every active alert with every reference case that has begun. An alert is considered to be “active” if LD has tendered it and has neither superseded nor retracted it. At each time point, select pairs for scoring using one-to-one case pairing. (Any alert that remains unpaired—because it didn’t satisfy the gating temporal condition against any reference case or because other alerts matched better to the reference cases—incur the full F+ cost.) This heavy usage of case pairing—the most computationally expensive element of our hypothesis scoring approach—makes the idealized approach impractical for datasets with many attacks.

Case comparison: Compute match quality using the same object-oriented count-based metrics we use for respectively hypothesized cases.

Timeliness and supersession: An alert’s timeliness is rewarded in that LD will incur lower costs by issuing the warning about a given attack earlier. Because we are summing costs over all simulation time points, LD will receive a better cost-based score.

To be considered timely with respect to a given reference case, an alert must have a time of tendering falling within the “viable” (simulation) time window beginning with the reference case’s start time and ending when the first resource or capability is applied (signifying the threat event’s consummation in an attack). See Figure 12, in which alert A_1 and superseding alert A_2 tendered at times t_1 and t_2 , respectively, have object-oriented recall scores \mathcal{R}_1 and \mathcal{R}_2 , respectively. At each time point during the reference case’s viable window, the F− component of assessed cost (light-shaded area at the top of Figure 12) discounts the specified (full) cost of F− reporting by an amount (dark-shaded area at the bottom) corresponding to the prevailing object-oriented recall.

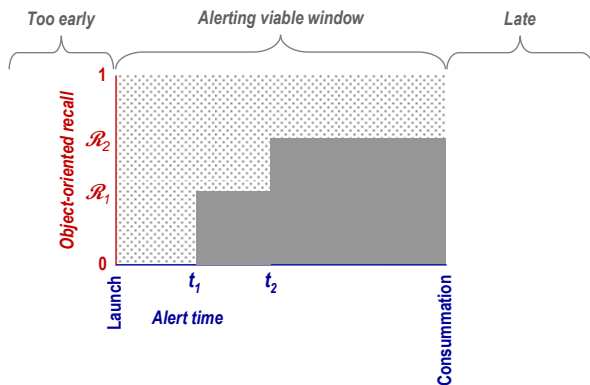


Figure 12: Ideal alert scoring example

The F+ component of assessed cost similarly discounts the specified (full) cost of F+ reporting by an amount corresponding to the prevailing object-oriented precision. This amounts to viewing an active alert’s F+ content as having cost that is constant as time progresses. While such an assumption seems appropriate for F− content, some of a hypothesis’ individual elements may in reality accommodate an analyst’s verification. While this verification activity might incur some cost, such cost also might be limited to a bounded time period and thus might not necessarily be treated as constant over an alert’s full time interval of “active” status. We leave a more detailed examination of time-dependent F+ costs to potential future work.

Figure 12 illustrates the basic scheme for timely alerts. Early alerts must be spurious (so receive the full F+ cost), because LD couldn’t have detected the reference event before it started. An alert that is late may nonetheless be associated with a reference case, in one of the following ways.

- It may be paired with the reference case (presuming their temporal intervals overlap). While we expect such an alert to afford no opportunity to thwart an attack, it may—if it precedes any other report of the attack in evidence—afford benefit toward mitigating a slow-acting (*e.g.*, biological) attack. We choose not to treat such an alert as F+. At worst, its correct content may be “old news” that does not provoke significant cost at the recipient’s end. Its incorrect content similarly may usually be adjudicated against old news reports on the reference case without much cost.
- It may not have been paired with any reference case and yet have been designated by LD to supersede another alert that has been paired with a reference case. We appeal to the same rationale as above in ignoring these reports.

By similar rationale, after a reference event’s consummation, we ignore the F+ content of any viable alert that has been paired with it—even when that alert has not been superseded or retracted by LD before consummation time.

Submission: LD may submit alerts at any (simulation) time, with any frequency it chooses. As it advances incrementally through time-stamped evidence, it must examine only the events that are reported (with respect to simulation time) either at or before its current processing time. Finer incremental processing time intervals may incur greater overall processing time but also may afford more opportunity to detect impending threat events and issue alerts sooner. Coarser intervals may not pick up evidence regarding threat events until after they have been consummated—and so miss the opportunity for alerting.

Requiring alerts at fixed simulation-time frequencies would entail similar issues. Requiring alerts at specified simulation times (*e.g.*, near threat event consummations) might engender gaming by LD.

Practical Alert Scoring

Our practical approach addresses the alert scoring requirements as follows.

Practical case pairing and supersession: We finesse alert case pairing by requiring that LD designate, for each threat event hypotheses that it tenders retrospectively, any of its previously tendered alerts that it believes corresponds to the same case. We score the full set of retrospectively tendered hypotheses first, resulting in a set of case pairs mapping the reference cases to retrospective hypotheses. Then, for alerts, we appeal to the same mapping, so that a retrospective hypothesis and the alerts that LD has designated as belonging to it are paired with the same reference case. Thus, we reuse an existing case pairing, rather than re-pairing afresh at each simulation time point—and we avoid the expensive computation that case pairing incurs.

For each reported alert, LD may designate supersession by another alert or by a retrospective hypothesis, in a supersession *chain*, as illustrated in Figure 13.

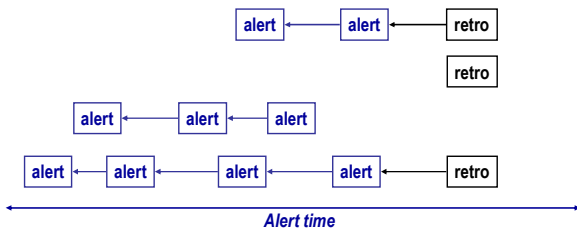


Figure 13: Alerts and retrospective hypotheses, with supersession relationships

To simplify scoring, LD may designate exactly one earlier alert to directly supersede or be superseded by exactly one later alert or retrospective hypothesis—no branching is allowed in supersession chains. A retrospective hypothesis need not supersede any alert, and an alert need not be ultimately superseded by a retrospective hypothesis. We score an alert against a reference case only when the alert is ultimately superseded by a retrospective hypothesis to which the reference case has been paired. See Figure 14.

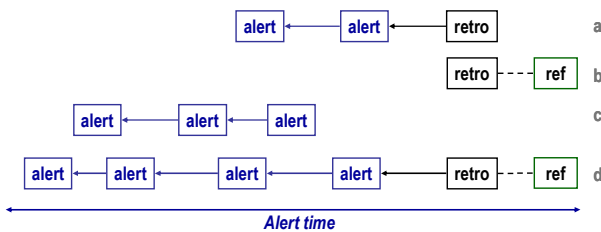


Figure 14: Alerts, retrospective hypotheses, and reference cases, with supersession and pairing relationships

In Figure 14, two retrospectively hypothesized cases have been paired with reference cases. Only one of the paired retrospective cases (in Row d) heads an alert supersession chain, so only the alerts in this chain will have the opportunity to match a reference case. Chains

(like those in Row c) not headed by retrospective hypotheses remain unpaired, so are assessed the full F+ cost. Chains (like those in Row a) headed by a retrospective hypotheses that is not paired with a reference case also are assessed the full F+ cost (because the pairing—or, here, non-pairing—against retrospectively hypothesized cases prevails).

Switching to this practical approach to case pairing and supersession would suffice to make the (otherwise-) idealized approach practical. Our approach actually implemented in the PE Lab also differs from the idealized approach in the following additional ways that we comment on (with benefit of hindsight).

Implemented cost basis: Rather than scoring at every time point, we score each tendered alert at most once, as explained below. Compared to the idealized method, the implemented method returns an overall cost that is lower by roughly the factor of a threat exploitation event's duration (in time points).

Implemented timeliness: We discount the F- component of the latest timely alert's cost-based score to reward timeliness. As discussed further below, to be most consistent with the idealized approach, we also should (but do not in our current implementation) discount the F+ cost component to *penalize* timeliness. (Per the idealized approach, we should consider an alert's F+ spurious, undesirable content to have a deleterious effect that persists until consummation.)

The timeliness reward function (provided in the scoring specification for alerts) is called when the alert reporting time falls during the viable window and returns a value in the range [0, 1], as illustrated in Figure 15. We discuss its use under **Implemented supersession**, below.

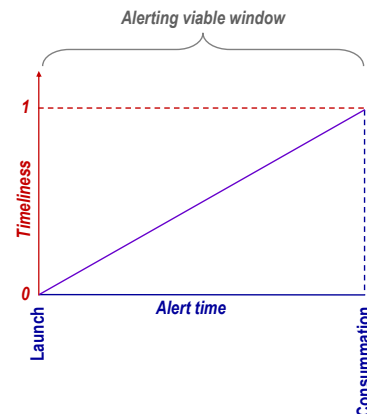


Figure 15: Timeliness for an alert paired with a given reference case

Implemented supersession: With respect to the reference case, a paired retrospective hypothesis' supersession chain—which may be empty (as in Row b of Figure 14) or contain alerts (as in Row d)—will correspond to one of the eight situations illustrated in Figure 16.

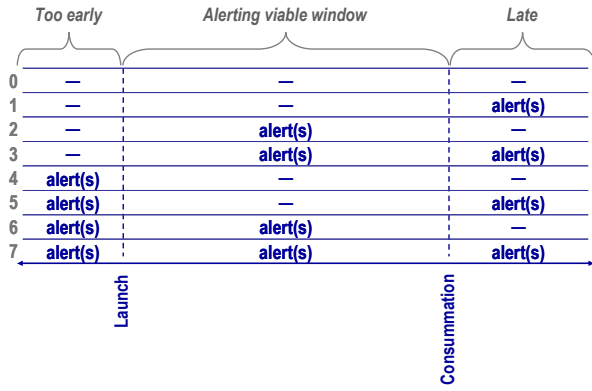


Figure 16: Possible relationships of alerts to reference cases

Our implementation handles the situations illustrated in Figure 16 as follows.

- If the retrospective hypothesis supersedes no alert (Situation 0, corresponding to Row 0 in Figure 14), assess the full prospective F- cost. No alert was issued for this reference case.
- If there exist alerts superseded by a retrospectively hypothesized case and all are late (Situation 1), assess no cost. As in the idealized approach, we consider these alerts to pertain to “old news.”
- If at least one alert is too early (Situations 4–7), assess the full prospective F+ cost. The implementation takes the (strongly conservative) view that any alert designated to supersede (perhaps indirectly) an early, spurious alert must itself be spurious. We have omitted this view from our idealized approach. To relax it in the implementation, and thus make the implementation more consistent with the idealized approach, we would address Situations 6 and 7 by assessing the full F+ cost for early alerts (in the “Too early” column on the left-hand side of Figure 16) but score the timely alerts (“Alerting viable window” column) and late alerts (“Late” column) the same way as in Situations 2 and 3.
- If the earliest alert is timely (Situations 2 and 3), score the latest timely alert, as follows.
 1. Determine object-oriented recall \mathcal{R} and object-oriented precision \mathcal{P} by comparing this alert to the reference case.
 2. Determine the alert’s timeliness, t .
 3. Assess for this alert the specified F- cost multiplied by the factor $1 - \mathcal{R} * (1 - t)$, as illustrated in Figure 17. An alert with perfect timeliness and perfect recall will incur no F- cost. Otherwise, F- cost will be discounted by the fraction of the graph’s area corresponding to the lower-right, solid-shaded area. Considering just the scored alert (ignoring earlier timely alerts for this reference case), the linear function depicted in Figure 15 effectively integrates the constant per-time point F- costs from our idealized approach.

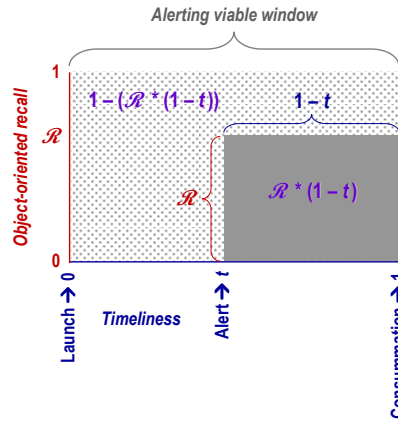


Figure 17: F- cost discounting for the latest timely alert

4. Assess for this alert the prospective F+ cost discounted by the factor $1 - \mathcal{P}$. This step of our implementation thus includes no consideration of timeliness. Per our discussion of alert F+ costs in the idealized approach, the implementation should assess a penalty proportional to the alert’s earliness—*i.e.*, multiply the specified F+ cost by the factor $1 - (\mathcal{P} * t)$, as illustrated in Figure 18. LD can avoid this penalty by alerting either late or with no F+ content. With the penalty, considering just the scored alert, the linear timeliness function effectively integrates our idealized constant per-time point F+ costs. The implemented approach thus assesses a timeliness penalty that is excessive, in that it does not penalize less F+ content reported later.

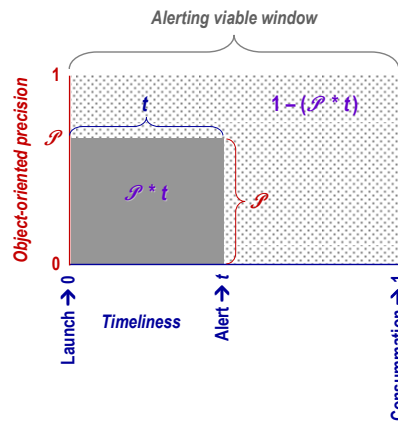


Figure 18: F+ cost discounting for the latest timely alert

Given the implementation’s scoring of only the latest timely alert, it is to LD’s advantage to supersede an alert only when it expects lower overall cost—because of better object-oriented precision and/or recall. In hindsight, this causes our evaluation to conflate two separate processes of threat decision aiding—threat detection (LD) and alert presentation management—illustrated in Figure 19. (Note that in the foregoing we have—perhaps somewhat loosely—used the term “alert” to refer to any prospectively

tendered threat event hypothesis, independent of a decision whether to present it to a user.)

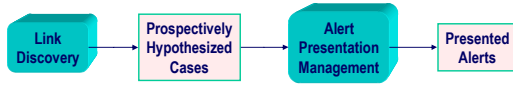


Figure 19: Threat decision aiding processes and products

The idealized approach scores LD’s hypotheses independent of its alert presentation decisions. So would our (as-yet-unimplemented) practical departure with respect to just case pairing and supersession (the approach variant with which we opened this section).

Alternative Approach

In the Hats simulator by Morrison *et al.* (2005), an agent with threat detection capability acts as a player playing a game. (The PE lab’s dataset generator uses an artificial world abstraction style inspired by that of Hats.) Rather than submitting prospective event hypotheses (“alerts,” in our terminology), players interdict suspected threat actors before they can inflict any damage. Performance is determined by the game’s final score (accounting for any damage, false arrest, and information costs incurred). Thus, Hats-based performance evaluation holistically considers both the processes included in Figure 20, rather than evaluating threat detection (LD) separately. Our contrasting orientation has led us to invent the present alert scoring methods.

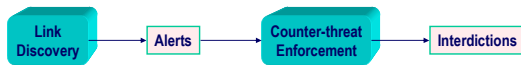


Figure 20: Automated threat management processes and products

Conclusion

Threat detection research programs provide unique opportunities to develop and address challenge problems and associated evaluation metrics. It has been our privilege to work with members of the research community in defining and refining PE Lab alert scoring methods. We expect that the developed methods may be applied with benefit in other alerting performance evaluation contexts.

Acknowledgements

We thank our colleagues who have contributed to the PE Lab’s alert scoring design, notably (with apologies to anyone we overlook) Chris Boner and Dan Hunter. We also thank anonymous reviewers for their helpful comments.

References

Létourneau, S., Famili, F., and Matwin, S. 1999. Data Mining for Prediction of Aircraft Component Replacement. *IEEE Intelligent Systems* special issue on data mining (December), 59–66.

Mahoney, S. Laskey, K., Wright, E., and Ng, K. 2000. Measuring Performance for Situation Assessment. In *Proceedings of the MSS National Symposium on Sensor and Data Fusion*, San Antonio, Texas.

Morrison, C., Cohen, P., King, G., Moody, J., and Hannon, A. 2005. Simulating Terrorist Threat in the Hats Simulator. In *Proceedings of the First International Conference on Intelligence Analysis*: MITRE Corp.

Schrag, R. 2006. A Performance Evaluation Laboratory for Automated Threat Detection Technologies. Performance Metrics for Intelligent Systems Workshop. National Institute of Standards and Technology.

Schrag, R. and Takikawa, M. 2006. Scoring Hypotheses from Threat Detection Technologies. AAI Fall Symposium on Capturing and Using Patterns for Evidence Detection.

Weiss, G., and Hirsh, H. 1998. Learning to Predict Rare Events in Event Sequences. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, 359–363. Menlo Park, CA: AAI Press.