Probabilistic Alert Correlation*

Alfonso Valdes and Keith Skinner

SRI International
{valdes, skinner}@sdl.sri.com

Abstract. With the growing deployment of host and network intrusion detection systems, managing reports from these systems becomes critically important. We present a probabilistic approach to alert correlation, extending ideas from multisensor data fusion. Features used for alert correlation are based on alert content that anticipates evolving IETF standards. The probabilistic approach provides a unified mathematical framework for correlating alerts that match closely but not perfectly, where the minimum degree of match required to fuse alerts is controlled by a single configurable parameter. Only features in common are considered in the fusion algorithm. For each feature we define an appropriate similarity function. The overall similarity is weighted by a specifiable expectation of similarity. In addition, a minimum similarity may be specified for some or all features. Features in this set must match at least as well as the minimum similarity specification in order to combine alerts, regardless of the goodness of match on the feature set as a whole. Our approach correlates attacks over time, correlates reports from heterogeneous sensors, and correlates multiple attack steps.

Keywords: Network security, sensor correlation, alert management, adaptive systems.

1 Introduction

In response to attacks and potential attacks against enterprise networks, administrators are increasingly deploying intrusion detection systems (IDSs). These systems monitor hosts, networks, critical files, and so forth, using a variety of signature and probabilistic techniques [1,2]. The use of such systems has given rise to another difficulty, namely, correlating a potentially large number of alerts from heterogeneous sensors. To the degree that this has been addressed in current systems, heuristic techniques have been used. In this paper, we describe a probabilistic approach to this critical problem.

The intrusion detection community is actively developing standards for the content of alert messages [5]. Systems adhering to these evolving standards forward attack descriptions and supporting diagnostic data to an alert management

* This research is sponsored by DARPA under contract numbers F30602-99-C-0149 and N66001-00-C-8058. The views herein are those of the author(s) and do not necessarily reflect the views of the supporting agency.

W. Lee, L. Mé, and A. Wespi (Eds.): RAID 2001, LNCS 2212, pp. 54–68, 2001.

interface (AMI), which may be remotely located and consolidating the reports of numerous sensors. In anticipation of these standards, we are evolving systems for alert correlation and ranking.

In [3] we introduced sensor coupling and an initial approach to probabilistic sensor correlation. At the time, we demonstrated the utility of the coupled sensors approach for false alarm reduction and improved detection performance. We also introduced a correlation approach based on feature-specific similarity functions. The correlation example cited at that time consisted of a single sensor examining a probe that was distributed in time. Since then, we have introduced some important innovations:

- Capability to comprehend heterogeneous sensors.
- Introduction of the probabilistic minimum match criterion, which prevents spurious matches on less important features from causing the system to correlate alerts. By specifying minimum match of unity, this capability causes the system to function as a rule-based correlation system. Explicitly tolerating slightly imperfect matches provides an important generalization of rule-based correlation methods.
- Hierarchy of sensor correlation: thread, incident, and correlated reports from multistep attack scenarios.

We have also gained considerable experimental experience in both live and simulated environments.

The remainder of this paper is organized as follows. We introduce our approach to alert fusion, using our defined alert template to report alerts from EMERALD and third-party sensors. We introduce notions of feature overlap, similarity, expectation of similarity, and minimum similarity. We show how these are used to achieve a hierarchy of alert correlation defined as inferred thread (within sensor), security incident (between sensor) and correlated attack report (between sensor and attack step). We then present preliminary results from the alerts generated by various EMERALD and third-party monitors [1,2].

2 Sensor Correlation and Alert Fusion

In [3], we introduced probabilistic methods for sensor correlation. Specifically, we considered two closely coupled sensors, a TCP misuse monitor and an asset availability monitor. Both are based on Bayes inference [4], and the former sensor is aware of the state of the latter. Sensor coupling in this fashion is easily expressed in Bayes formalisms; specifically, the coupling is achieved by dynamically modifying priors in the TCP session monitor. The approach demonstrated two important capabilities:

- Failed accesses to services that appear to be invalid are viewed with greater suspicion than other failed accesses. This greatly increases sensitivity against certain stealth probe attacks.

– Certain types of failed access to services that appear "down" are tolerated, so that when a service fails (for malicious or accidental reasons) we do not generate a large number of false alarms.

Having demonstrated the utility of sensor fusion for improved sensitivity and false alarm reduction, we have further explored the general problem of sensor correlation. In particular, we examine the situation of multiple heterogeneous sensors on diverse platforms, considering realistic problems of late reports and clock drift.

Our correlation algorithm considers features as reported in a standard alert template. For each matchable feature, we have defined an appropriate similarity function that returns a value between 0 and 1. New alerts are compared to a list of existing meta alerts. The overall similarity is the weighted average of the feature similarities, with expectation of similarity as the normalizing weights. If the minimum match criterion fails for any feature, the match is rejected regardless of the overall similarity. The new alert is correlated with the most similar meta alert, assuming the similarity is above a specified minimum match threshold. Otherwise, the new alert starts a new meta alert thread.

2.1 Meta Alerts and the Alert Template

We have defined an enhanced alert template that enriches standards such as IETF/IDWG [5] with respect to content while avoiding issues of language specifics. Our template includes the concept of alert thread, which is present in all EMERALD monitors and alert management components. Alert reports are part of the same thread if they are from the same sensor and refer to the same attack, possibly separated in time. Where provided, such as is the case with EMERALD monitors, the thread mechanism causes new reports for an existing thread to be treated as updates, dramatically reducing clutter on a display interface that comprehends threads. For this reason, we identified the capability to reliably infer thread when the sensor does not provide one as a very desirable capability of alert correlation.

We include an "anomaly" field in addition to the "confidence" field used in the evolving IETF standard. We also include arrays to describe in greater detail the target(s) of an attack (for example, the specific ports scanned). Finally, we include fields describing the sensor type and placement, enabling us to identify each unique instantiation of any sensor in the monitored domain.

This template is presently used by a diversity of sensors employing both signature and probabilistic techniques. This includes sensors that we have developed under the EMERALD program, as well as prototype wrappers for ISS RealSecure and Checkpoint Firewall-1. Our early experiments indicate that diverse sensors will be able to fill this template with content that will be more useful to correlation engines than is currently available.

2.2 Feature Similarity

Our probabilistic alert fusion approach considers feature overlap, feature similarity, minimum similarity, and expectation of similarity. We maintain a list of "meta alerts" that are possibly composed of several alerts, potentially from heterogeneous sensors. For two alerts (typically a new alert and a meta alert), we begin by identifying features they have in common (feature overlap). Such features include the source of the attack, the target (hosts and ports), the class of the attack, and time information. With each feature, we have a similarity function that returns a number between 0 and 1, with 1 corresponding to a perfect match. Similarity is a feature-specific function that considers such issues as

- How well do two lists overlap (for example, list of targeted ports)?
- Is one observed value contained in the other (for example, is the target port of a denial-of-service (DOS) attack one of the ports that was the target of a recent probe)?
- If two source addresses are different, are they likely to be from the same subnet?

For attack class similarity, we maintain a matrix of similarity between attack classes, with values of unity along the diagonal and off-diagonal values that heuristically express similarity between the corresponding attack classes. We prefer to consider attack classes rather than attack signatures, which are much more specific and numerous but may be erroneously or incompletely reported. For example, in our demonstration environment, we run a variant of mscan that probes certain sensitive ports, that is, it is of the attack class "portsweep". Our host sensors have a specific signature for this attack and call it "mscan". The Bayes sensor trades specificity for generalization capability and has no "mscan" model, but successfully detects this attack as a "portsweep". These reports are considered similar ($S = 1$) with respect to attack class.

Not all sensors produce all possible identifying features. For example, a host sensor provides process ID, while a network sensor does not. Features not common to both alerts are not considered for the overall similarity match.

The meta alert itself supports the threading concept, so we can visualize composing meta alerts from meta alerts.

2.3 Situation-Specific Similarity Expectation

An important innovation we introduce is expectation of similarity. As with similarity, this is also between 0 and 1, and expresses our prior expectations that the feature should match if the two alerts are related, considering the specifics of each. For example, two probes from the same target might scan the same set of ports on different parts of our subnet (so expectation of matching target IP address is low). Also, some attacks such as SYN FLOOD spoof the source address, so we would allow a match with an earlier probe of the same target even if the source does not match (expectation of match for source IP is low).

We now give some examples of how expectation of similarity depends on the situation, that is, the features in the meta alert and the new alert.

If an alert from a sensor has a thread identifier that matches the list of sensor/thread identifiers for some meta alert, the alert is considered a match and fusion is done immediately. In other words, the individual sensor's determination that an alert is an update of or otherwise related to one of its own alerts overrides other considerations of alert similarity.

If the meta alert has received reports from host sensors on different hosts, we do not expect the target host feature to match. If at least one report from a network sensor has contributed to the meta alert and a host sensor alert is received, the expectation of similarity is that the target address of the latter is contained in the target list of the former.

In determining whether an exploit can be plausibly considered the next stage of an attack for which a probe was observed, we expect the target of the exploit (the features host and port) to be contained in the target host and port list of the meta alert.

Some sensors, particularly those that maintain a degree of state, report start and end times for an attack, while others can only timestamp a given alert. The former deal with time intervals, while the latter do not. Similarity in time comprehends overlap of the time intervals in the alerts considered for correlation, as well as the notion of precedence. We do not penalize time similarity too far from unity if the time difference is plausibly due to clock drift.

Deciding whether the attacker is similar is somewhat more involved. In the case of an exact match of originating IP address, similarity is perfect. We assign high similarity if the subnet appears to match. In this way, a meta alert may potentially contain a list of attacker addresses. At this point, we consider similarity based on containment. In addition, if an attacker compromises a host within our network (as inferred by a successful outcome for an attack of the root compromise class), that host is added to the list of attacker hosts for the meta alert in question. Finally, for attack classes where the attacker's address is likely to be spoofed (for example, the Neptune attack), similarity expectation with respect to attacker address is assigned a low value.

2.4 Minimum Similarity

Our correlation component implements not just expectation of similarity (which effectively acts as a weight vector on the features used for similarity matching) but also enforces situation-specific minimum similarity. Certain features can be required to match exactly (minimum similarity for these is unity) or approximately (minimum similarity is less than unity, but strictly positive) for an alert to be considered as a candidate for fusion with another. Minimum expectation thus expresses necessary but not sufficient conditions for correlation.

The overall similarity between two alerts is zero if any overlapping feature matches at a value less than the minimum similarity for the feature (features for which no minimum similarity is specified are treated as having a minimum

similarity of 0). Otherwise, overall similarity is the weighted average of the similarities of the overlapping features, using the respective expectations of similarity as weights.

By appropriate settings of similarity expectation and minimum similarity, the correlation component achieves the following hierarchy of correlation. The system is composable in that we can deploy multiple instances to obtain correlation at different stages in the hierarchy. For example, we can infer threads (within sensor correlation) and then correlate threaded alerts from heterogeneous sensors into security incidents.

Synthetic Threads: For sensors that do not employ the thread concept, the correlation synthesizes threads by enforcing high minimum expectation similarity on the sensor itself (the thread must come from a single sensor) and the attack class, as well as source and target (IP and ports). We have wrapped the alert messages from a leading commercial sensor and observed that this facility reliably reconstructs threads.

In particular, by placing an aggregator component topologically close to an IDS, the pair is made robust against attacks that cause the IDS itself to flood, as described in a recent NIPC advisory [6].

Security Incidents: By suppressing minimum expectation of similarity on the sensor identifier, and relaxing expectation of similarity for this feature, we can fuse reports of the same incident from several heterogeneous sensors into a single incident report. In this case, we enforce a moderately high expectation of similarity on the attack class. This is not unity because different sensors may report a different attack class for the same attack. We construct a table of distances between attack classes that expresses which ones are acceptably close. For security incident correlation, we enforce minimum expectations on the source and target of the attack. Using this technique, we have been able to fuse alert reports from commercial and EMERALD sensors into security incident reports.

Correlated Attack Reports: By relaxing the minimum expectation of similarity on the attack class, we are able to reconstruct various steps in a multistage attack. Each stage in an attack may itself be a correlated security incident as described above. In this fashion, it is possible to recognize a staged attack composed of, for example, a probe followed by an exploit to gain access to an internal machine, and then using that machine to launch an attack against a more critical asset.

2.5 Feature Fusion

When the system decides to fuse two alerts, based on aggregate similarity across common features, the fused feature set is a superset of the features of the two alerts. Feature values in fused alerts are typically lists, so alert fusion involves list merging. For example, suppose a probe of certain ports on some range of the protected network matches in terms of the port list with an existing probe that originated from the same attacker subnet, but the target hosts in the prior alert were to a different range of our network. The attacker address list has the new

attacker address appended, and the lists of target hosts are merged. The port list matches and is thus unchanged.

Two important features are the sensor and thread identifiers of all the component alerts, so that the operator is always able to examine in detail the alerts that contribute to the meta alert report.

One additional feature is the priority of the meta alert, supported by our template and provided by EMERALD sensors. We are developing a component that estimates criticality based on the assets affected, the type of attack, the likelihood of attack success, and an administrative preference. The aggregator maintains the high water mark for this field. We are investigating approaches whereby the contributing threads are permitted to update their priority downward, computing meta alert priority as the maximum across thread priorities at any given time. This approach would permit downward revision of the meta alert priority.

The features presently considered in the probabilistic correlator component include sensor identification (identifier, location, name), alert thread, incident class, source and target IP lists, target TCP/UDP port lists, source user id, target user id, and time. Computations are only over features that overlap in the alert to be merged and the candidate meta alert into which it is to be merged. Incident signature is used as well, but with a low expectation of similarity as these vary widely across heterogeneous sensors.

If present, a thread identifier from the reporting sensor overrides other match criteria. A new alert that matches the sensor and thread of an existing meta alert is considered an update of the earlier alert.

The correlator first tries to infer a thread by looking for an exact match in sensor identification and incident class and signature. Note that alerts that are inferred to be from the same thread may be separated in time. The system attempts to infer threads even in incident and scenario operational modes.

Next the system checks that all overlapping features match at least at their minimum similarity value. Setting minimum expectation for some features to unity (not normally recommended) causes the system to behave like a heuristic system that requires exact matches on these features. Given that this criterion passes, we compute the overall similarity between the two alerts as follows:

$$SIM(X,Y) = \frac{\sum_j E_j SIM(X_j, Y_j)}{\sum_j E_j}$$

$$\begin{aligned} X &= \text{Candidate meta alert for matching} \\ Y &= \text{New alert} \\ j &= \text{Index over the alert features} \\ E_j &= \text{Expectation of similarity for feature } j \\ X_j, Y_j &= \text{Values for feature j in alerts } X \text{ and } Y, \\ &\quad \text{respectively (may be list valued)} \end{aligned}$$

Incident class similarity is based on a notion of proximity, which at present is the result of our judgement. The proximity of class A to B reflects how reasonably an attack currently of incident class A may progress to class B. Note that this is not symmetric; we more strongly expect an exploit to follow a probe than the other way around. The incident classes are from the EMERALD 602 message format. Note that some "default" classes such as "invalid" and "action logged" are reasonably proximal to most other classes. This is because the IETF standard does not require a common ontology, and reports from heterogeneous sensors for the same incident may not reliably represent this field. As such, we do not want to reject potential matches based on this field alone.

For operational modes other than thread level aggregation, we do not recommend a high minimum similarity value for this field.

Table 1. Incident Class Similarity Matrix

	INVALID	PRIVILEGE_VIOLATION	USER_SUBVERSION	DENIAL_OF_SERVICE	PROBE	ACCESS_VIOLATION	INTEGRITY_VIOLATION	SYSTEM_ENV_CORRUPTION	USER_ENV_CORRUPTION	ASSET_DISTRESS	SUSPICIOUS_USAGE	CONNECTION_VIOLATION	BINARY_SUBVERSION	ACTION_LOGGED
INVALID	1	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.6
PRIVILEGE_VIOLATION	0.3	1	0.6	0.3	0.6	0.6	0.6	0.6	0.4	0.3	0.4	0.1	0.5	0.6
USER_SUBVERSION	0.3	0.6	1	0.3	0.6	0.5	0.5	0.4	0.6	0.3	0.4	0.1	0.5	0.6
DENIAL_OF_SERVICE	0.3	0.3	0.3	1	0.6	0.3	0.3	0.4	0.3	0.5	0.4	0.1	0.5	0.6
PROBE	0.3	0.2	0.2	0.3	1	0.7	0.3	0.3	0.3	0.3	0.4	0.8	0.3	0.6
ACCESS_VIOLATION	0.3	0.6	0.3	0.5	0.6	1	0.6	0.6	0.3	0.3	0.4	0.1	0.5	0.6
INTEGRITY VIOLATION	0.3	0.5	0.3	0.5	0.6	0.8	1	0.6	0.5	0.3	0.4	0.1	0.5	0.6
SYSTEM_ENV_CORRUPTION	0.3	0.5	0.3	0.5	0.6	0.6	0.6	1	0.6	0.3	0.4	0.1	0.5	0.6
USER_ENV_CORRUPTION	0.3	0.5	0.5	0.3	0.6	0.6	0.6	0.6	1	0.3	0.4	0.1	0.5	0.6
ASSET_DISTRESS	0.3	0.3	0.3	0.6	0.3	0.3	0.3	0.3	0.3	1	0.4	0.4	0.3	0.6
SUSPICIOUS_USAGE	0.3	0.3	0.5	0.3	0.5	0.6	0.5	0.6	0.5	0.3	1	0.1	0.3	0.6
CONNECTION_VIOLATION	0.3	0.1	0.1	0.3	0.8	0.3	0.3	0.3	0.3	0.5	0.4	1	0.3	0.6
BINARY_SUBVERSION	0.3	0.3	0.3	0.3	0.3	0.6	0.6	0.6	0.5	0.3	0.4	0.1	1	0.6
ACTION_LOGGED	0.3	0.3	0.3	0.3	0.6	0.5	0.3	0.3	0.3	0.3	0.4	0.3	0.3	1

For two alerts that are extremely close in time, it is possible that the alerts may not be in time order. In this case, incident class similarity is the greater of $SIM(X, Y)$ and $SIM(Y, X)$. Mathematically, the similarity computation for incident class can comprehend a discrete call (the alert is from one of the above

classes) or a call that is a probability distribution over the above classes (as might result from a meta alert in which the contributing sensors do not agree on the class).

Most other features are potentially list-valued. For lists, the notion of similarity generally expresses the fraction of the smaller list that is contained in the larger. For source IP addresses, similarity also attempts to express the notion that the addresses in question may come from the same subnet.

Time similarity is a step function that drops to 0.5 after one hour. A close match in time is expected only when the system operates in "incident" mode. Thread and scenario aggregation may be over time intervals of days.

3 Results

3.1 Live Data

The following is an example of alert correlation over time, in this case correlating alerts that are components of a stealthy port sweep. The following is an example of one of the contributing alerts. In the interest of space, we do not include all the content that is in the alert and meta alert templates, but limit ourselves to the fields needed to illustrate the result.

```
Thread ID 69156 Class= portsweep   BEL(class)= 0.994 BEL(attack)=
1.000
2001-06-15 17:34:35 from xx.yyy.148.33 ports 1064 to 1066
duration= 0.000
dest IP aaa.bbb.30.117
3 dest ports: 12345{2} 27374{3} 139
```

This is a probe for 3 vulnerable ports on a single IP address in the protected network, and is detected by the system described in [2]. The above is just a single step in a probe that apparently transpired over several days, and resulted in the following correlated meta alert.

```
Meta Alert Thread 248
Source IPs source_IParray: xx.yyy.148.33 xx.yyy.148.47

Target IPs target_IParray: aaa.bbb.30.117 aaa.bbb.6.232
aaa.bbb.8.31 aaa.bbb.1.166 aaa.bbb.7.118 aaa.bbb.28.83
aaa.bbb.19.121 aaa.bbb.21.130 aaa.bbb.6.194 aaa.bbb.1.114
aaa.bbb.16.150

From 2001-06-15 17:34:35 to 2001-06-21 09:19:57
correlated_alert_priority -1

Ports target_TCP_portarray: 12345{4} 27374{4} 139{3}
```

```
Number of threads: 10
Threads: 69156 71090 76696 84793 86412 87214 119525
124933 125331 126201
Fused: PORT_SCAN
```

We note that we have correlated events from two source addresses that were judged to be sufficiently similar. The attack is quite stealthy, consisting of a small number of attempted connections to single target hosts over a period of days. The list of thread identifiers permits the administrator to examine any of the stages in the attack. In this case, each attack stage is considered a portsweep; if the stages consisted of different attack classes, these would be listed under "Attack steps". Over the three week time period containing this attack, the IDS sensor processed over 200,000 sessions and generated 4439 alerts. The probabilistic correlation system produced 604 meta alerts.

3.2 Experimentation Environment

The experimentation environment simulates an electronic commerce site, and provides Web and mail services over the Internet. The protected network is behind a firewall, and is instrumented with several host, network, and protocol monitors. Two machines are visible through the firewall, and two auxiliary machines are used for network monitors and alert management. The firewall blocks all services other than Web, FTP, and mail. We have simulated traffic that appears to come from a number of sources, and an attacker who executes certain well-known attacks. The background traffic includes legitimate anonymous and nonanonymous FTP use; the latter triggers false alarms in some signature sensors. There are also low-priority nuisance attacks, as well as low-level failed accesses to blocked ports, which trigger firewall log messages.

The attack begins with an MSCAN probe to the two machines visible through the firewall. EMERALD eBayes and eXpert-Net sensors the two machines detect this and set the attack code for mscan in their alert report. The Bayes network sensor detects the attack as of the class "portsweep". The target port lists for these alerts match, as does the attacker address. Similarity with respect to the target address depends on the order in which the alerts arrive (the order at which the alerts arrive is not deterministic as the sensors are distributed). Until the network sensor's alert is received, we do not expect target addresses to match for distinct sensors, since a host sensor can typically report only its own address as a target. We expect the network sensor's target list to contain the targets seen by host sensors (tolerating imperfect matches due to arrival order), and we expect the target address of subsequent host alerts to be contained in the meta alert's target host list. Therefore, regardless of arrival order, these alerts are acceptably similar with respect to target host.

The attacker next uses a CGI exploit to obtain the password file from the Web server. This exploit is detected by the host sensor. The next stage of the attack is a buffer overflow to gain access on the host providing mail service, detected by an EMERALD host sensor as well as by ISS.

Although telnet through the firewall is blocked, the attacker may telnet (using a password obtained from the earlier exploit) from the compromised internal host to the critical host, and he now does so. On that host, he uses another overflow attack to obtain root access. He is now free to change Web content.

Figure 1 shows the EMERALD alert console for this scenario, containing alert reports for the above attack steps as well as alerts due to false alarms and nuisance attacks. It is not very informative, except to show that more than 1000 alerts are received for this scenario, which lasts approximately 15 minutes, and the critical attack is effectively buried.

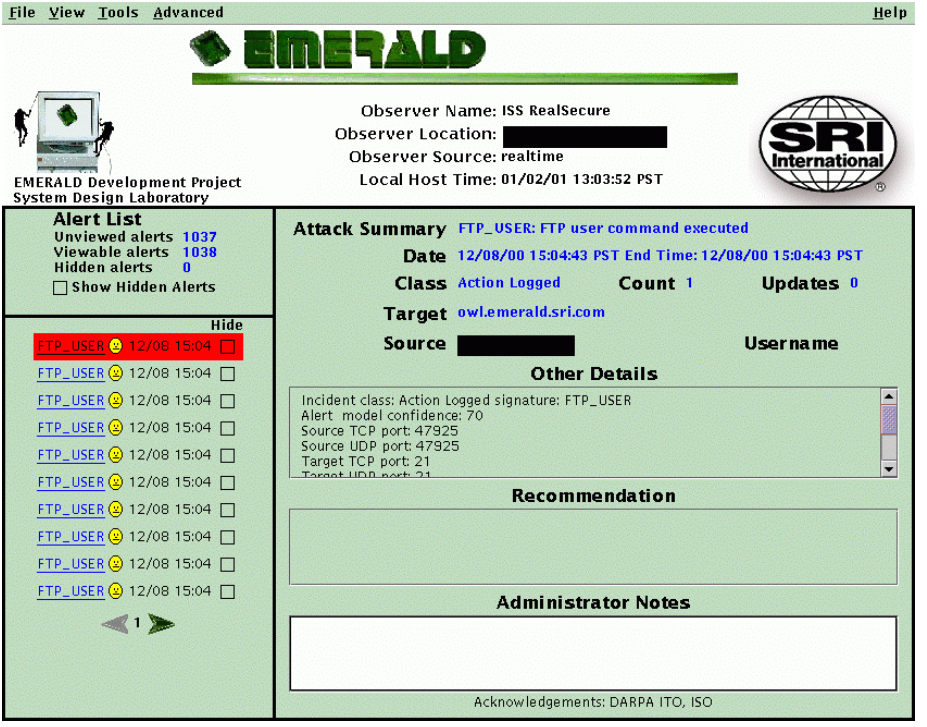

Fig. 1. Raw Sensor Alerts for the Simulated Attack Scenario

The majority of the alerts are from nonanonymous FTP write operations, which are allowed but trigger alerts from a commercial sensor.

Figure 2 shows the result of inferring synthetic threads for sensors that do not support the threading concept. For alert thread inference, we set the minimum match for sensor identifier fields to unity, and require a high match for attacker and target parameters as well; we do not require a minimum similarity in the time field to allow inference of threads that occur slowly over time. Most of the

alerts are threaded into sessions according to originating IP address, and the total number of alerts reported is reduced to about sixty. We have highlighted an alert, which is one of the nuisance TCP sweep attacks in the background traffic.

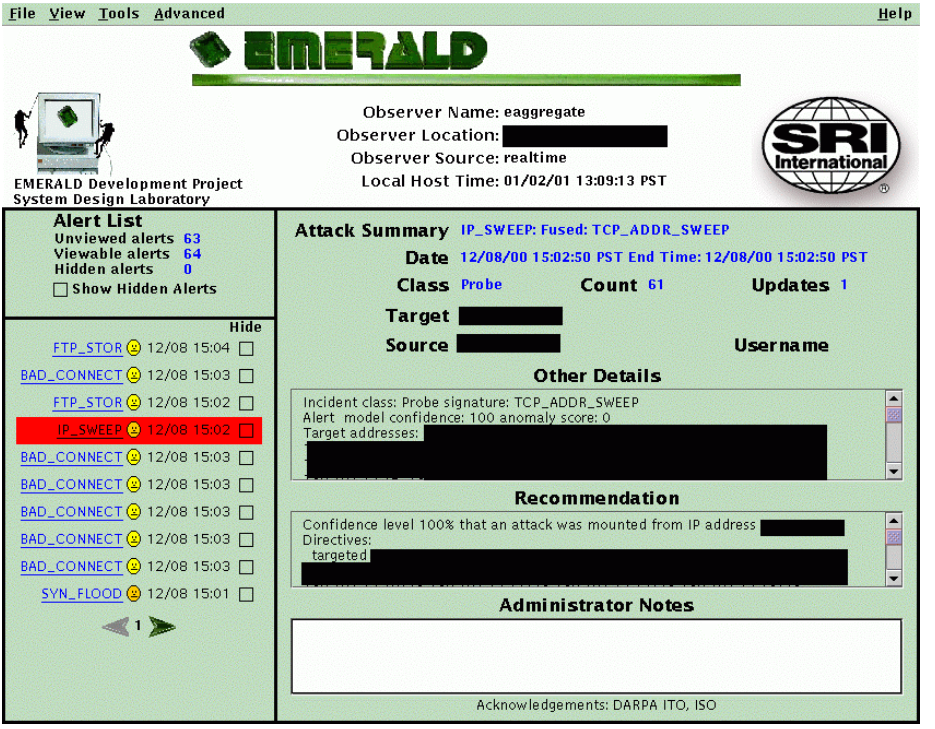

Fig. 2. Alert Fusion via Synthetic Thread Inference

For attack incident matching, we must look across sensors to determine if several alerts refer to the same security incident. We suppress minimum similarity for sensors, set a moderately high similarity for attacker and target identifiers, and set a moderate minimum for time (to allow for clock drift and sensor response time differences). We also require a match in attack class, making allowances for sensors that report different attack classes for a given attack sequence. Figure 3 shows the result of incident matching for the attack scenario. Some of the critical steps in the attack reported by several sensors now result in a single alert. For example, the highlighted buffer overflow attack is reported by an EMERALD sensor as well as by ISS. These are fused into a single correlated incident report. The two mscan reports from EMERALD host sensors and the portscan report from the EMERALD Bayes TCP sensor are also fused into a single incident.

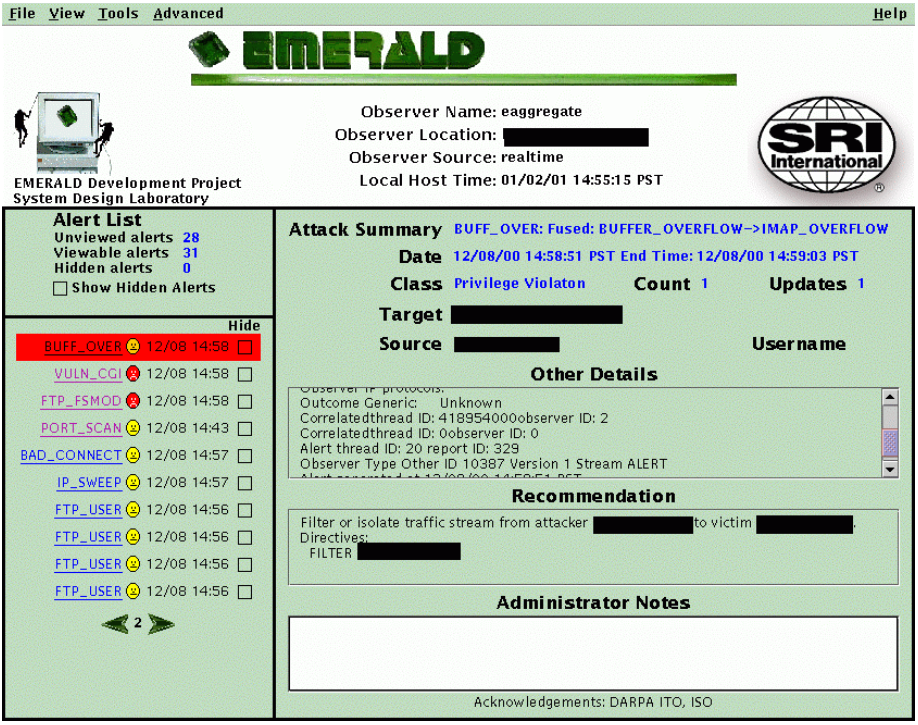

Fig. 3. Security Incidents

The final step in correlation is an attempt to reconstruct the attack scenario. We now relax attack class similarity to consider what attack steps might follow the present step. For example, it is reasonable to follow a probe with an exploit to a target "contained" in the probe. Also, if the meta alert includes a step indicating a likely root compromise, the host in question is added to the attacker's assets (that is, it is both a victim and now a potential attacker as well). In Figure 4, the correlated attack report correctly lists as attacker assets his originating host, an internal host compromised at an intermediate step, and the spoofed address of the SYN flood stage. In this fashion, we have successfully combined both the external and internal threads of the attack scenario.

It is worth pointing out that the EMERALD sensors in this experiment provide response directives which, if followed, would stop the attack in progress.

4 Summary

We have adapted and extended notions from the field of multisensor data fusion for alert correlation. The extensions are principally in the area of generalizing feature similarity functions to comprehend observables in the intrusion detection

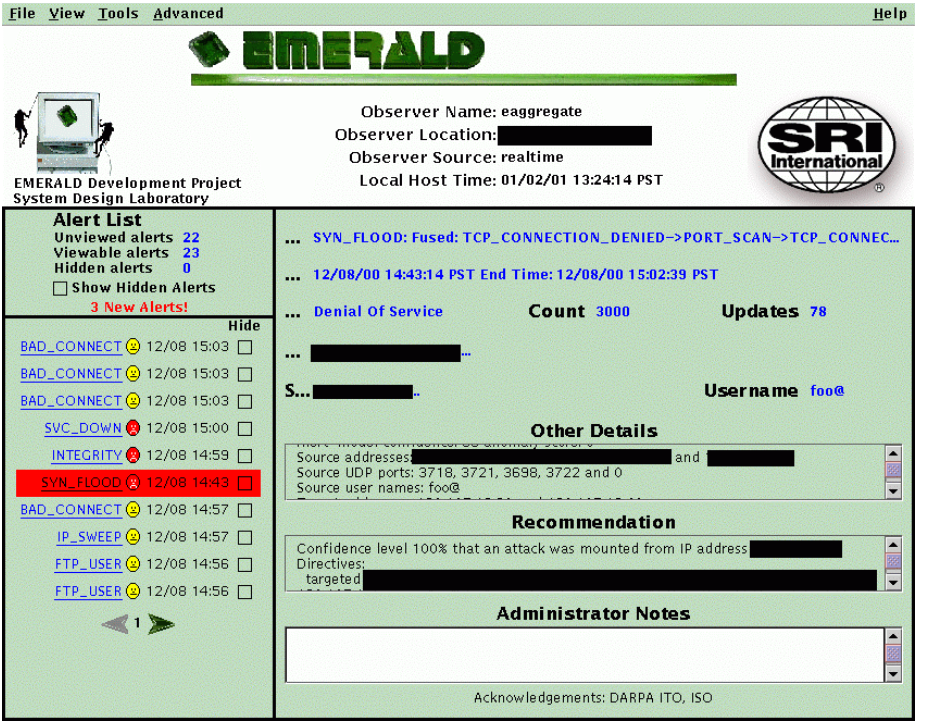

Fig. 4. Correlated Attack Scenario

domain. The approach has the ability to fuse alerts for which the match is good but not perfect. The method considers appropriate fields in alert reports as features for a multivariate matching algorithm. Only features that overlap are considered for the overall similarity calculation. For each matchable feature, we have defined an appropriate similarity function with range 0 (mismatch) to 1 (perfect match). Depending on the situation, we incorporate expectation of match values (which are used to compute a weighted average of similarity over the overlapping features), as well as a minimum match specification that unconditionally rejects a match if any feature fails to match at the minimum specified value. For each new alert, we compute similarity to existing meta alerts, and merge the new alert with the best matching meta alert, as long as the match passes a threshold value. We realize a reduction of one-half to two-thirds in alert volume in a live environment, and approach a fiftyfold reduction in alert volume in a simulated attack scenario.

References

1. Porras, P. and Neumann, P. "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances", National Information Security Conference, 1997. http://www.sdl.sri.com/emerald/emerald-niss97.html
2. Valdes, A. and Skinner, S. "Adaptive, Model-based Monitoring for Cyber Attack Detection", Recent Advances in Intrusion Detection (RAID 2000), Toulouse, France, October 2000. http://www.raid-symposium.org/raid2000/program.html
3. Valdes, A. and Skinner, S. "Blue Sensors, Sensor Correlation, and Alert Fusion", Recent Advances in Intrusion Detection (RAID 2000), Toulouse, France, October 2000. http://www.raid-symposium.org/raid2000/program.html
4. Pearl, J. "Probabilistic Reasoning in Intelligent Systems", Morgan-Kaufmann (1988).
5. Erlinger, M. and Stanniford, S. "Intrusion Detection Interchange Format", http://www.ietf.org/html.charters/idwg-charter.html
6. National Infrastructure Protection Center advisory 01-004, http://www.nipc.gov/warnings/assessments/2001/01-004.htm, March 2001.