Information Modeling for Intrusion Report Aggregation

Robert P. Goldman, Walter Heimerdinger, Steven A. Harp,
Christopher W. Geib, Vicraj Thomas, Robert L. Carter
Honeywell Laboratories
3660 Technology Drive
Minneapolis, MN 55418
goldman@htc.honeywell.com

Abstract

This paper describes the SCYLLARUS *approach to fusing reports from multiple intrusion detection systems (IDSes) to provide an overall approach to intrusion situation awareness. The overall view provided by* SCYLLARUS *centers around the site's* security goals, *aggregating large numbers of individual IDS reports based on their impact The overall view reduces information overload by aggregating multiple IDS reports in a top-down view; and by reducing false positives by weighing evidence provided by multiple IDSes and other information sources.*

Unlike previous efforts in this area, SCYLLARUS *is centered around its Intrusion Reference Model (IRM). The* SCYLLARUS *IRM contains both dynamic and static (configuration) information. A* **Network Entity/Relationship Database (NERD)***, providing information about the site's hardware and software; a* **Security Goal Database***, describing the site's objectives and security policy; and an* **Event Dictionary***, describing important events, both intrusions and benign; comprise the static portion of the IRM. The set of IDS reports; the events* SCYLLARUS *hypothesizes to explain them; and the resulting judgment of the state of site security goals comprise the dynamic part of the IRM.*

1. Introduction

We have developed the SCYLLARUS system, an architecture for integrating a number of individual IDSes into a unified intrusion detection approach. SCYLLARUS overcomes the limitations of both individual IDSes, and unstructured groups of IDSes. Instead of simply joining together multiple alert streams, SCYLLARUS provides a unified intrusion situation assessment. Critical to this unification is SCYLLARUS's Intrusion Reference Model (IRM), which contains information about the configuration of the site to be protected (including the IDSes), the site's security policies and objectives, and the phenomena of interest (intrusion events). In this paper we describe the SCYLLARUS approach, with particular attention to the role played by the IRM.

Over the past years, there has been a great deal of research in intrusion detection, the construction of systems designed to detect unauthorized use of computer systems. There are now a number of systems able to detect various classes of intrusions into individual hosts and computer networks. Some of these systems are still research prototypes, but several widely available, either as open source or commercial products.

These systems still don't provide system owners and administrators with comprehensive *intrusion awareness*. There are a number of drawbacks to existing IDSes. One of the most profound is that these systems are not designed to work together, as part of a suite of sensors. Instead, each program generates a separate stream of reports, and fusing them into a coherent view of the current situation is left as an exercise for the user. There has been some limited work on fusing together multiple IDS event streams (see Section 6), but it does not go nearly far enough.

System administrators must have multiple IDSes at their disposal because the various IDSes all have different "sweet spots" and blind spots. IDSes based on recognizing signatures of known exploits can have low false-alarm rates, but are limited to recognizing those exploits that were known at the time of their last update.[1] Furthermore, signature-based IDSes often provide only "late warning": they report when a system has been compromised, but typically don't

[1] Sometimes they can also recognize generalizations of known exploits.

provide warning that an attack is underway. On the other hand, anomaly-detecting IDSes can, at least in theory, provide early warning and detection of novel exploits. However, this additional sensitivity is purchased at the cost of high false alarm rates, often so high that system administrators are overwhelmed by alerts and disconnect or disregard the IDS.

A second split in IDS design is between host-based IDSes and network-based IDSes. Typically, host-based IDSes will generate alerts based on some event stream generated by the host's operating system (e.g., syslog, Solaris BSM log, Windows NT event log); network-based IDSes will typically use some form of packet-sniffer as their input. Again, these two approaches have blind spots and sweet spots. Only a network-based IDS will be able to detect exploits such as IP spoofing (now infamous for its use against Shimomura [13]), that take advantage of weaknesses in the IP protocol. On the other hand, network-based IDSes are blind to attacks that exploit weaknesses in host-based systems (such as buffer overflows), unless they can somehow be seen in input-output behavior visible in network traffic.

Rather than looking for the holy grail of a perfect intrusion detection system, our research centers around the development of an intrusion assessment framework that treats IDSes as *sensors*, and adds a knowledge-based sensor fusion, or evidence aggregation, component. We argue that this makes good engineering sense. As we explain above, each of the various IDS approaches has its strengths and weaknesses, considered as a sensor. An evidence aggregation component can also provide a number of useful features that are more properly shared among the set of IDSes, rather than being incorporated in each one separately.

In addition to partial coverage, current IDSes have a number of other weaknesses as tools for situation awareness. Current IDSes are not sensitive to an installation's mission (goals) and security policy. For example, for an internet stock trading firm, availability of its public web site is mission critical. On the other hand, for an Air Force base, a public web-site is a PR luxury that can readily be sacrificed in a crisis. Without some knowledge of an installation's mission and policies, no IDS can appropriately label its reports to highlight the important events and place less-important events in the background. For example, in general, reconnaissance events are less important than exploits; owners of externally-visible hosts must accept that they will be scanned regularly. On the other hand, to return to our hypothetical Air Force base in a national-security crisis, a scan of a host that is supposed to be stealthy may be more important than defacement of its external web server, which can readily be sacrificed. To properly understand intrusion-related events, we must be able to relate them to our objectives.

We have already alluded to trade-offs between different intrusion detection strategies. One of the key trade-offs is between sensitivity to true events and false positives. In general, with any sensor, we must pay in false positives for whatever we gain in sensitivity. One way to overcome this limitation is to assemble a suite of sensors. This can be a very efficient way to overcome the problem of false positives, as long as we can find sensors that fail relatively independently.

A shared framework like ours can also provide protection against *systematic* false positives. For example, the IDS `snort`, contains rules for detecting IP sweeps. We have found that, in sites where Norton AntiVirus™Corporate Edition is run, this rule will be tripped by the normal operation of Norton AntiVirus™. An automatic update server for Norton AntiVirus™ periodically scans the network, looking for hosts that are running the Norton AntiVirus™ update client. Such clients listen on UDP port 38293.[2]

The ordinary solution to this kind of problem is to edit the IDS rule to keep it from firing in this circumstance, for example by ignoring IP sweeps that hit only UDP port 38293. This approach is unsatisfactory for two reasons. First, in sites where multiple IDSes are run, this means that we must separately update the configuration of each individual IDS. Second, it allows a clever attacker free reign with traffic on this port.

On the other hand, if we have a central repository of this kind of information, we get two corresponding advantages: First, we have only a single point of update. Instead of fixing each individual sensor, we reconfigure so that all of the reports corresponding to this class of false positives will be filtered out. Second, we can collect additional information, for example, logs of the Norton AntiVirus™ server, that allow us to distinguish between true Norton AntiVirus™ events, and transmissions from a clever attacker that are dis-

[2] We are not singling out `snort` for criticism. Our experience has been with `snort`, but virtually all network-based IDSes would exhibit this kind of behavior.

guised as Norton AntiVirus™ transmissions.

SCYLLARUS provides a knowledge-rich framework for IDS report aggregation and fusion. Central to SCYLLARUS is its Intrusion Reference Model (IRM), which provides a central repository of information about the site to be protected, about the security goals of that site, and the events of interest. The IRM also provides a central database for IDS reports, which may be filed by any IDS, using an API that we provide. In turn, those reports are examined by a Dynamic Evidence Aggregator, that fuses the reports into a coherent overall situation view, and determines which security goals are threatened by the events detected.

2. SCYLLARUS Architecture

In this section, we provide a brief capsule summary of the SCYLLARUS architecture. We explain the process of accepting IDS reports, computing a judgment of what events are plausible given the reports, and then identifying which of the site's security goals are threatened, based on the plausible events. This will give a brief understanding of the roles of the various subsystems of SCYLLARUS. We will go into greater depth about the contents of the Intrusion Reference Model in the following section, and explain how they support the various tasks.

The core of the SCYLLARUS architecture is the Intrusion Reference Model(IRM); see Figure 1. The other parts of SCYLLARUS read the contents of the IRM and post the results of their computations to it. The key information producers in SCYLLARUS are the Intrusion Detection Systems (IDSes); their output starts the whole process of situation assessment. The IDSes file their reports through the Intrusion Reporting API(IRAPI). The IRAPI ensures that the proper records are created in the reports database of the IRM.

The process of aggregating, correlating and fusing reports from multiple IDSes is a three-stage process, performed by the components of SCYLLARUS's Dynamic Evidence Aggregator (DEA). First, a component called the Cluster Preprocessor reads the IDS reports from the database, and then generates hypotheses that would explain the reports. These hypotheses are written to the Event Database, and then a second component, the Event Assessor, weighs the information for and against each hypothesis. It will judge some set of hypotheses to be plausible; those hypotheses will be so marked in the database. The final stage of the process is to determine which of the installation's security goals are threatened by the plausible events. This task is performed by facilities native to the Intrusion Reference Model.

The Dynamic Evidence Aggregator component reads the IDS reports from the Report Database; it is the job of this component to compute the overall judgment of what events are plausible based on the reports. This judgment is computed in a two-step process. First, the Cluster Preprocessor (CP) consults the Report Database to identify the set of reports. From these reports, the CP generates a set of hypothesized events, that will be recorded in the Event Database. In general, in response to an IDS report, the CP will generate at least one event, that corresponds to the event the IDS has reported.

To return to our earlier example, consider a case where `snort` has reported an `IPSWEEP` occurring. The CP would create a hypothetical `IPSWEEP` event. However, based on knowledge about the ways other events can be perceived, the CP will also hypothesize that there may have been a Norton AntiVirus™ event.

The CP may also attach other reports to the hypothesized events. For example, if there were also a network anomaly detector IDS active at this site, it might generate an `ANOMALOUS-TCP-TRAFFIC` report on port UDP port 38293, with targets the appropriate IP addresses. If the times of occurrence lined up appropriately, the CP would connect this event with the earlier `IPSWEEP` event. We do this because the anomaly detector's report provides corroboration for the `snort` report. To look at it another way, the Norton AntiVirus™ event would *explain away* the anomaly detector's report. The reports can be presented to a security officer together, helping to alleviate the problem of information overload. We will have more to say about the phenomenon of explaining away later.

We may summarize the function of the Cluster Preprocessor as follows: to assemble the set of hypotheses suggested by the IDS reports, and to marshal the evidence for these hypotheses. If the Cluster Preprocessor *proposes* event hypotheses, it is the job of the Event Assessor to *dispose* of them. The Assessor will weigh the evidence for the various hypotheses against each other, and determine which are plausible. To do this, the Assessor retrieves the set of events from the Event Database and examines their

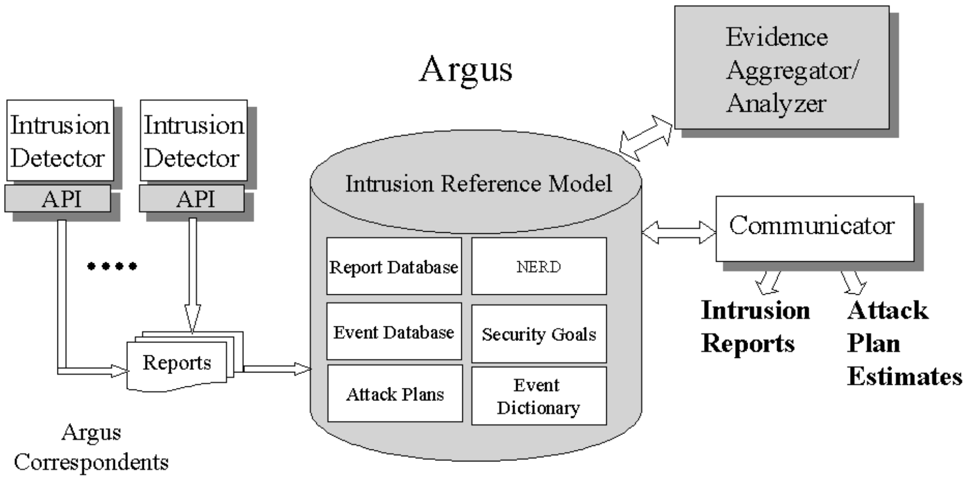

Figure 1. The SCYLLARUS **architecture.**

interrelationships and the evidence (in the form of IDS reports) that support them. The set of events and their evidence make up a directed graph that we may interpret as a *causal probabilistic network* [9]. Based on this interpretation, we may compute the plausibility of the various hypotheses. "Plausibility" is a qualitative analog of normal probability theory [7]. The database entries of the event hypotheses deemed plausible are marked to indicate this fact in the Event Database.

The final step of Evidence Aggregation is the step of determining the *impact* of the plausible hypotheses. To this end, each of the security goals in the IRM has, associated with it, a *concept* (class) corresponding to the set of events that will compromise this security goal. For example, the goal of maintaining root authentication (`maintain-root-authentication`) on the host `kubrick` would be compromised by any user-to-root privilege escalation taking place on that host, or by successful password guessing on the root account. When an event that meets this description is instantiated and then marked as plausible, it is automatically categorized as a goal-attacker for the appropriate security goal, and the corresponding security goal is reclassified as (potentially) compromised.

Goals may also be *indirectly* compromised, since security goals can be hierarchically composed. For example, the goal of `network-nondisclosure` represents a desire to maintain the confidentiality of the IP addresses that make up the network. The successful maintenance of this goal depends on the successful maintenance of nondisclosure goals for the individual hosts. Accordingly, when one or more of these goals is classified as compromised, the overarching goal of `network-nondisclosure` is also marked as compromised.

Figure 2 shows an example of the Dynamic Evidence Aggregator process in SCYLLARUS. This figure shows the compromised security goals (ovals), the events that SCYLLARUS believes compromised them (rectangles with single outlines) and the reports (rectangles with double outlines) that provided evidence for the events. The arrows are drawn in directions that are intended to capture causal influence, corresponding to the interpretation of the drawing as a causal probabilistic network (see above):

- from events to the security goals they compromise;
- from events to the reports that provide evidence for them (the events cause the IDS to issue a report;
- (dashed) from events to other events that are manifestations of the underlying events.

This figure, taken from a display tool for our internal use, shows how SCYLLARUS has concluded that the goal of maintaining user authentication on the host `kubrick` was compromised. The user authentication goal is represented by the bottom-most oval node in the figure, labeled as `KUBRICK-PROTECT-USER-AUTHENTICATION`. The figure shows that the goal was compromised in two ways, both directly and indirectly.

The direct compromise in this scenario occurred when an attacker used IP spoofing to masquerade as a user logged

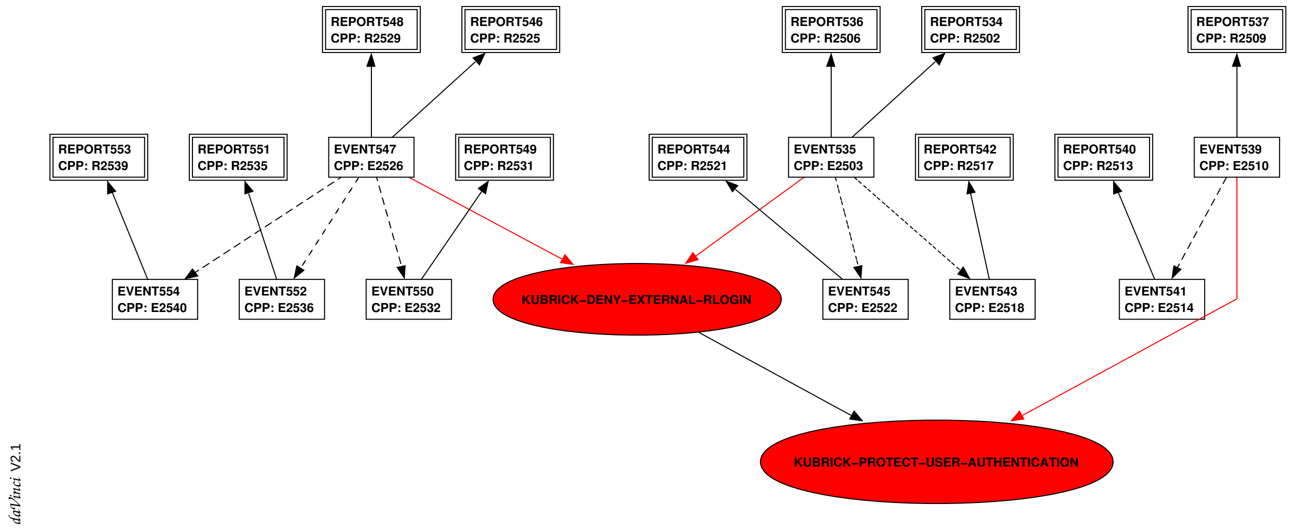

Figure 2. A figure that shows how the goal of maintaining user account authentication has been violated on the host `kubrick`**.**

into a trusted host. The attacker took advantage of this privilege to introduce an `.rhosts` file into a legitimate users' account. This part of the compromise is shown in the rightmost part of Figure 2. The box labeled `EVENT539` corresponds to the event of writing the `.rhosts` file. There are two pieces of evidence for this event. The first was a very specific report from a signature-based IDS, illustrated by the report box in the upper right corner of the figure, labeled `REPORT537`. SCYLLARUS also received corroboration, in the form of a more vague report shown as the report box labeled `REPORT540`, immediately below and to the left of `REPORT537`. `REPORT537` came from an anomaly-detector, and described an unusual file writing event, notated as `EVENT541`, shown immediately below and to the right of `REPORT537`. SCYLLARUS knows that writing a `.rhosts` file is a kind of file-writing event, leading it to recognize that the anomaly event could be a manifestation of `EVENT539`. This relationship is denoted in our figure by the dashed line between the nodes for `EVENT539` and `EVENT541`.

As we explained above, the goal `KUBRICK-PROTECT-USER-AUTHENTICATION` was compromised in two ways. One of the sub-goals of maintaining user authentication is to deny all remote logins from hosts outside our site. This subgoal, labeled `KUBRICK-DENY-EXTERNAL-RLOGIN`, is represented by the second shaded oval, immediately above and to the left of `KUBRICK-PROTECT-USER-AUTHENTICATION`. The arrow from the former to the latter represents the fact that compromising the former threatens the latter. The left-hand side of Figure 2 shows the traces of two external rlogins performed by the attacker, after s/he introduced the `.rhosts` file. Those were recognized by SCYLLARUS as violating the goal of not permitting rlogins from outside the network. The upshot is that our goal of maintaining control of who logs into our site has been compromised.

Now the interested user can "drill down" to find the details, inspecting the hypothesized events and the reports that provide evidence for them. Tables 1 and 2 show the sort of information that is available. These two tables show information computed by the Cluster Preprocessor (CP). The unique identifiers (e.g., `E2510` and `R2513` in Table 1) were computed by the CP and correspond to the numbers after the `CPP:` in Figure 2. The event described in Table 1 is the writing of the `.rhosts` file that directly compromises the goal of protecting user authentication. The report providing support, `R2513`, is the one shown at the upper right hand corner of Figure 2. The description of `E2510` refers to `E2514` as a manifestation. This is the event shown at the

```
Event E2510
CORRUPT-RHOST-EVENT
Start time: 10:20:13 2000/10/31
End time  10:20:13 2000/10/31
Status  SUCCEEDED
Sources  PIDs=(1776)
UIDs=(1234)
unames=(ROCKY)

Targets  Hostname=KUBRICK
IP-Addrs=((129 168 2 60))
Services=(LOGIN)

Related Files  (/home/rocky/.rhosts)

1 supporting report follows.

Possible Manifestations
10:20:13 2000/10/31 : E2514
UNUSUAL-MOD-OF-CRITICAL-FILE-EVENT
Report R2513
CORRUPT-RHOST-EVENT
  submitted by IDS: USTAT-60
Start time: 10:20:13 2000/10/31
End time  -
Report time  10:20:13 2000/10/31
Status  SUCCEEDED
Anomaly Score  NIL
Sources  PIDs=(1776)
UIDs=(1234)
unames=(ROCKY)

Targets  Hostname=KUBRICK
IP-Addrs=((129 168 2 60))
Services=(LOGIN)

Related Files  (/home/rocky/.rhosts)
```

Table 1. Detailed data about the event SCYLLARUS hypothesizes in response to USTAT-60's report of a security policy violation: the creation of a `.rhosts` file in a user's home directory. Note the cross-reference to another hypothesized event, E2514, see Table 2.

```
Event E2514
UNUSUAL-MOD-OF-CRITICAL-FILE-EVENT
Start time: 10:20:13 2000/10/31
End time  10:20:13 2000/10/31
Status  SUCCEEDED
Sources  PIDs=(1776)
UIDs=(1234)
unames=(ROCKY)

Targets  Hostname=KUBRICK
IP-Addrs=((129 168 2 60))
Services=(LOGIN)

Related Files  (/home/rocky/.rhosts)

1 supporting report follows.

Possible Manifestation Of
10:20:13 2000/10/31 : E2510
    CORRUPT-RHOST-EVENT
Report R2509
UNUSUAL-MOD-OF-CRITICAL-FILE-EVENT
    submitted by IDS: UANOM-60
Start time: 10:20:13 2000/10/31
End time  -
Report time  10:20:19 2000/10/31
Status  SUCCEEDED
Anomaly Score  30
Sources  PIDs=(1776)
UIDs=(1234)
unames=(ROCKY)

Targets  Hostname=KUBRICK
IP-Addrs=((129 168 2 60))
Services=(LOGIN)

Related Files  (/home/rocky/.rhosts)
```

Table 2. This table shows the SCYLLARUS CP's response to a report from an anomaly detector, UANOM-60. Note that the CP recognizes that the anomaly event it has hypothesized may be a manifestation of the event of the `.rhosts` file being written (see Table 1).

end of the dashed arc from E2510.

The most important thing to notice here is the way our system provides an *understandable, goal-based* summary of a large number of IDS reports. Note that we do *not* claim to have a good graphical user interface! However, we *do* claim to have captured the important relationships: reports providing evidence for hypothesized events; multiple reports corroborating each other by providing evidence for the same event; and the way events can be understood as compromising particular security goals. Note that this diagram provides a compact summary of 11 IDS reports. This is the information that is needed in order to provide any user interface that will usefully summarize the large quantities of information provided by intrusion detection systems and other system security assessment tools.

3. CLASSIC overview

We have implemented the SCYLLARUS Intrusion Reference Model (IRM) using the CLASSIC object-oriented database system, developed at Bell Laboratories [2, 3] The CLASSIC system grows out of Artificial Intelligence work on frame-based, or "description logic" systems.[3] For the purposes of the SCYLLARUS system, CLASSIC provides several advantages: rapid prototyping; metadata; clear handling of multiple inheritance and automatic classification. By and large, any description logic system would provide these advantages. Among the description logic systems, CLASSIC seemed the most mature, reliable, and best documented. We use the version of CLASSIC implemented in Common Lisp; there are also versions written in C and C++.

Like most object-oriented databases (OODBs), CLASSIC provides storage and retrieval of structured objects, or *individuals*. Also as in conventional OODBs, these individuals have *roles*, properties, that may be filled by primitive objects or other individuals. Finally, these individuals are instances of *concepts* (classes), from which they may inherit role-fillers, role constraints, etc. CLASSIC permits *multiple inheritance*; individuals may be direct instances of more than one parent concept. For example, a particular individual representing a host in a site, may be both a HOST, INTERNAL (because it is under our control), a DNS-HOST (it runs the DNS service), and an HTTP-HOST (it runs the HTTP service). Note that multiple inheritance is only permitted when the multiple parent concepts are consistent; CLASSIC has no facilities for overriding inheritance.

CLASSIC has a powerful notation for describing its concepts and individuals.The language allows us to specify that a concept inherits from one or more parent concepts. For example, the concept of an INTERNAL-HOST, which is used in recognizing hosts that are important for security goals, is defined as:

```
(AND INTERNAL HOST)
```

That is, INTERNAL-HOST inherits from both the INTERNAL and HOST concepts. Individuals, instances of concepts, will have roles, and the CLASSIC description language lets us specify constraints on those roles. We may specify constraints on the cardinality of those roles and the type of object that may fill them. For example, the SOFTWARE-VERSION concept is defined as follows:

```
(AND CLASSIC-THING
    (ALL major-version INTEGER)
    (ALL minor-version INTEGER)
    (ALL patchlevel INTEGER)
    (ALL build INTEGER)
    (AT-MOST 1 minor-version)
    (AT-MOST 1 patchlevel)
    (AT-LEAST 1 major-version)
    (AT-MOST 1 major-version))
```

This indicates that the any values of major-version, minor-version and patchlevel must be integers. It also indicates that the major-version role is mandatory, but that the minor-version and patchlevel roles are optional. Note that there may be an unbounded number of values for a particular role. For example, there is no *a priori* limit on the number of nodes that are members of a NETWORK. The language also allows us to constrain roles to be filled by particular entities. For example, the ETHERNET_100 concept is defined as follows:

```
(AND ETHERNET-LINK
     (FILLS has-medium Twisted-pair)
     (FILLS how-switched Hub)
     (FILLS link-speed 100))
```

The CLASSIC concept description language is used in three different ways. First, it is used to define how a

[3] For those not familiar with such systems, a paper written for the SIGMOD conference provides the best introduction to CLASSIC [2].

concept is to be *classified* (recognized). For example, if we have an individual link between two nodes that is an ETHERNET-LINK, and the database contains information about its has-medium, how-switched and link-speed roles, that is consistent with the above definition, then that individual will automatically be classified as an ETHERNET_100 link. Likewise, when we learn that a security goal has a plausible event in its goal-attacker role, then that security goal will be reclassified as a COMPROMISED-SECURITY-GOAL.

The description language may also be used to impose additional constraints on concepts, constraints that are not used in classification. For example, the concept ETHERNET-LINK has the following constraints:

```
(AND LINK
     (AT-LEAST 1 has-medium)
     (AT-LEAST 1 link-speed)
     (ALL has-medium LINK-MEDIUM)
     (ALL link-speed INTEGER)
```

Notice that these are necessary conditions for an ETHERNET-LINK, but they are not sufficient. We would not want the database to conclude that individual was an ETHERNET-LINK, just because it met this description.

Finally, this language is used to define individuals as well as concepts. For example, in setting up the NERD for our test scenario, we create the following instance:

```
(and WORKSTATION
     (FILLS runs BSD-LOGIN)
     (FILLS runs PS)
     (FILLS ip-address IP-ADDR-192-168-1-78)
     (FILLS manufacturer "Sun")
     (FILLS memory 128)
     (FILLS storage 13600) ...)
```

One also uses the description language to define queries. One defines a new concept that corresponds to the set of individuals one wishes to find, and then asks for instances of that new concept. For example:

```
(AND COMPROMISED-SECURITY-GOAL
     (FILLS subject KUBRICK))
```

would find all of the security goals that have been compromised that pertain to the host kubrick.

CLASSIC also provides some facilities for rules. These rules are indexed with particular concepts. They fire whenever an individual is newly (re)classified as an instance of the corresponding concept. Such rules may be used to flesh out individual descriptions, impose consistency constraints, or overcome limitations in the classification process.

4. The Intrusion Reference Model

At the highest level, the Intrusion Reference Model is divided between repositories of static and dynamic information. This distinction is not a very crisply-defined one; it is a pragmatic division between information that changes rapidly (e.g., the set of IRM reports) and relatively slowly (e.g., network structure). The static components of the IRM are as follows:

- The Network Entity/Relationship Database (NERD)
- The Security Goal database
- The Event dictionary
- The Attack Plan library

The last of these, the Attack Plan library, we will not discuss here. We will cover this in a forthcoming paper that describes our work in attacker plan recognition. The dynamic components of the IRM are:

- The Report Database
- The Event Database
- Security Goal Status database

A high-level overview of the IRM structure is given in Figure 3.

4.1. Static components of the IRM

4.1.1 The NERD

The Network Entity/Relationship Database provides the central representation for the configuration of hardware and software installed at a given site. This provides the framework for building a set of security goals (since those goals have to do with the use and protection of the site's hardware and software). The NERD also provides a way to correlate

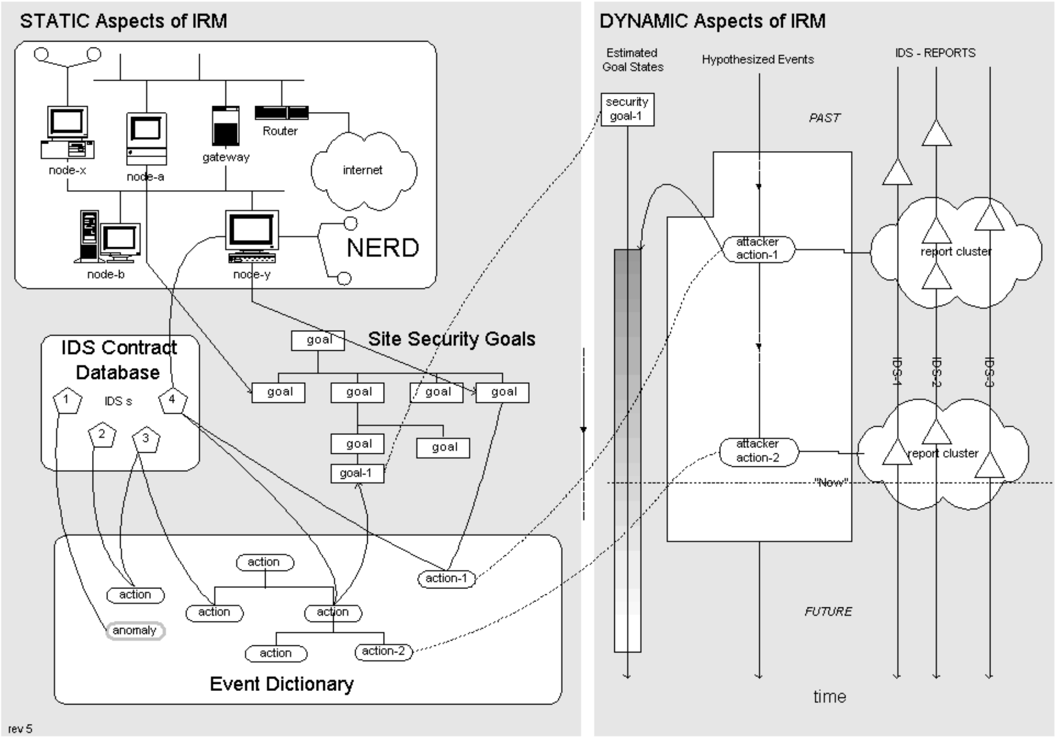

Figure 3. The structure of the SCYLLARUS **IRM.**

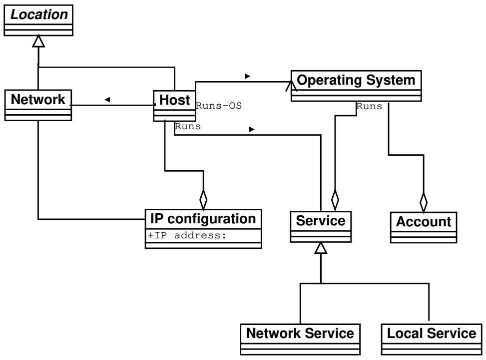

Figure 4. A UML diagram of part of the NERD schema.

the different IDS reports by allowing SCYLLARUS to reason about *where* events are happening.

The central concepts, or classes, of the NERD are those of network, host, operating system, and service. A rough UML diagram of these classes is shown in Figure 4. The key facts to notice are that hosts and networks are both locations, and that we reason about the way a host belongs to a network based on its IP configuration(s). Hosts run operating systems and operating systems run services. Services are a concept that subsumes both local services, those provided to users of the machine itself (e.g., `ps`, `tex`, etc.) and network services, provided to remote users (e.g., `http`, `ftp`, etc.). Naturally, there are many details that we cannot discuss here. For example, operating system has a rich structure of subtypes, and attributes that permit one to specify versions, etc. We have written a description of the NERD schema that we will make publicly available.

4.1.2 The Security Goal Database

The security goal database contains security goal objects that describe the uses to which we want to put the equipment (hardware and software) described in the NERD. Security goals are structured objects that refer to objects in the NERD, and that contain specifications of the kinds of event that will compromise them.

Security goals are structured in two important ways. First, they can refer to particular NERD entities as their subjects. For example, we can have a goal to `PROTECT-ROOT-AUTHENTICATION` whose subject is the host `kubrick`:

```
(AND PROTECT-ROOT-AUTHENTICATION
     (FILLS subject KUBRICK))
```

The second kind of security goal structure is based on goal-subgoal decomposition. For many high-level, mission-related goals, simple specifications like the one above, are insufficiently expressive and elegant. For example, if (one aspect of) the mission of a particular site is to provide spare parts ordering to a particular military unit, it will be difficult, if not impossible, to capture this directly in terms of hardware and software entities. Instead, it will be better to decompose this high-level goal into a set of subgoals such as protecting authentication of accounts with access to the database, protecting database host availability, etc. Note that this analysis approach is effectively the inverse of the attack trees analysis technique promoted by Schneier [12, Chapter 21].

Two aspects of this structure are worth mention. First, using multiple inheritance makes our representation of security goals easier. We can categorize security goals on a number of dimensions. For example, we use the four high-level categories of authentication, integrity, nondisclosure and neutrality.[4] We may also simultaneously categorize goals in terms of their role in high-level goals, or the kinds of subject they pertain to (single hosts, networks, etc.). A second important issue is that we may be able to use CLASSIC's rule facilities to design a "meta-policy" that we can use to make security goal design more efficient. For example, we could use a rule to capture the meta-policy that we don't want any internal DNS server to do zone transfers to external hosts. We have experimented with this technique a little, but not yet applied it to the test scenario.

4.1.3 The Event Dictionary

The Event Dictionary of the IRM performs three important roles in the SCYLLARUS system. First, the event dictionary provides the *lingua franca* for communications between all the IDSes. Second, the event dictionary contains representation of benign events that may trick IDSes into generating false positives. Finally, the event dictionary contains concepts that represent classes of events that have significant impacts on particular security goals.

[4] Our site should not be used as a means to attack others.

Although it may seem obvious, we know of few efforts to establish a shared vocabulary for multiple IDSes. There have been a number of efforts to provide message protocols to permit IDSes to publish their results; for example CISL and IETF IDMF (see Section 6). However, these efforts have mostly been limited to providing a shared *syntax*, rather than a shared vocabulary. For the vocabulary, both CISL and IDMF have adopted existing intrusion and vulnerability taxonomies (CVE[5] and Bugtraq[6], among others). However, they have done little to ensure that these vocabularies have clear semantics to permit results to be fused.

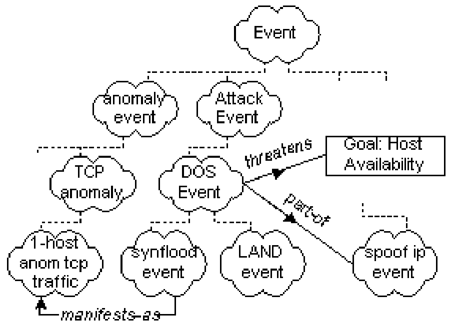

Figure 5. A small portion of the SCYLLARUS Event Dictionary.

The SCYLLARUS event dictionary is an attempt to provide an unambiguous framework for specifying events of interest. We expect that achieving a true intercommunication will require efforts on the part of both IDS developers and aggregators like ourselves. IDS developers will have to commit to correct use of known vocabularies. Aggregator developers will have to provide semantic "glue" that will overcome ambiguities in the vocabularies adopted (especially if these are *ad hoc* exploit vocabularies like Bugtraq's). Our approach to IDS aggregation centers around the development of a structured dictionary of events using CLASSIC. A part of our current event dictionary/taxonomy

[5] `cve.mitre.org`
[6] `www.securityfocus.com/forums/bugtraq/intro.html`

is shown in Figure 5.

The rich structure of the event dictionary, using multiple inheritance, is critical to our approach. The fact that our dictionary is not simply a list of specific exploits makes it easier to combine together reports of different IDSes. First, we permit IDSes to file reports of varying levels of specificity. For example, consider what might happen when an attacker successfully executes a `sadmindex` buffer overflow on a Solaris™system, escalating her privileges to root level. A signature-based IDS, such as USTAT [8], would detect the exact exploit used and file a report in those terms. On the other hand, a policy-monitoring IDS, that is watching only for inappropriate root shells, would detect only that some user-to-root exploit has occurred. In order to be able to correctly aggregate these reports, we need the inheritance information in the event dictionary.

The event dictionary's structure also helps us combine together signature- and anomaly-based IDSes. To do this, we use the *manifestation* relationship. Anomaly detectors don't report particular exploits; instead they report that some anomalous event has occurred. To return to our `sadmindex` example, an anomaly detector like that developed by Cigital [6, 5], might generate a report indicating that there has been an anomalous event in the `sadmind` process. Information about manifestation relationships in the IRM allows the Cluster Preprocessor to recognize that a `sadmindex` exploit may be manifested as an anomaly in the `sadmind` process.

The Event Dictionary also contains information about benign events, that could be mistakenly identified as intrusions. Consider the Norton AntiVirus™ update event that could be misclassified as an IP sweep (see Section 1). The Event Dictionary contains the information that an IP sweep report actually might be the detection of a Norton AntiVirus™ update, when the port swept is the appropriate one.

Note that none of the information we have discussed here is specific to a particular IDS. Instead the structure of the event taxonomy, and the relationships therein, are properties of the events being described. This means that this information appropriately resides in a shared component like the IRM, rather than scattered piecemeal in each IDS' configuration database, for elementary software engineering reasons.

Finally, the Event Dictionary contains entries for concepts that correspond to the classes of events that will compromise particular security goals. For example, for the security goal of maintaining root authentication on the host `kubrick`, the corresponding `GOAL-ATTACKER` concept is the concept of a `USER-TO-ROOT` event whose `target` is `kubrick`. In CLASSIC notation:

```
(AND USER-TO-ROOT (FILLS target KUBRICK))
```

This is a particularly simple example: because the security vocabulary is so focused on root compromises, we have a preexisting concept readily available. On the other hand, the `GOAL-ATTACKER` concept for user authentication goals is much more complex, subsuming not only out-and-out `REMOTE-TO-LOCAL` attack events, but also such policy violations as the writing of a `.rhosts` file on a machine running the `rlogind`, etc.

These `GOAL-ATTACKER` concepts might be thought of as cached queries for rule triggers. When we tell CLASSIC about a plausible event that is subsumed by one of these concepts, CLASSIC will automatically classify the event as being an instance of the appropriate `GOAL-ATTACKER` concept. In turn, this will cause CLASSIC to reclassify the corresponding goal as (potentially) compromised.

Since the security goal database has a taxonomy of its own, we do not need to create these `GOAL-ATTACKER` concepts by hand. Instead, we can use CLASSIC's rule engine to automatically generate the corresponding concept. For example, we have a rule associated with the security goal concept `PROTECT-ROOT-AUTHENTICATION`, that is triggered whenever a new instance of that concept is created. For a new `PROTECT-ROOT-AUTHENTICATION` whose subject is a particular host (e.g., `kubrick`), CLASSIC will automatically create a new `GOAL-ATTACKER` concept whose target is bound to the same host.

4.2. Dynamic components of the IRM

The Report Database and Event Database comprise the dynamic part of the IRM. The Report Database is the shared repository of all of the IDSreports. The Event Database contains the event objects hypothesized by the Cluster Preprocessor (CP) as possible reasons for the reports.

The Report Database contains instances of the CLASSIC `REPORT` concept. These may be filed by any IDS, using our Intrusion Reporting API. Reports are periodically loaded into the Report Database by IDSes, triggering the operation of the SCYLLARUS CP.

The Event Database is the place where the CP places the events it has hypothesized. These events are all instances of the CLASSIC concept, EVENT, and are also instances of the more specific sub-concepts of EVENT that we discussed above in Section 4.1.3. These event objects also contain information about the target of the event, the time when the event occurred, etc. As described above, the Event Assessor component will weigh the evidence for and against the various event hypotheses, and mark some of them as plausible. CLASSIC will then identify those plausible events that are GOAL−ATTACKERs for particular security goals, and reclassify those goals as (potentially) compromised.

5. Status of SCYLLARUS

Our current implementation of SCYLLARUS is a proof-of-concept prototype. The components described here have been fully implemented, but not thoroughly tested. We have all the components of the IRM described above, the Cluster Preprocessor and Event Assessor. We have also developed a version of the Intrusion Reporting API.

The SCYLLARUS prototype has been tested on a multi-stage attack scenario we developed with assistance from Giovanni Vigna and Dick Kemmerer of R.A. Kemmerer Associates, Richard Lippman and Josh Haines of MIT Lincoln Laboratories. Although the Intrusion Reporting API has been implemented, the tests were conducted with simulated IDS reports. We generated those IDS reports based on conversations with our scenario advisors, and based on our experience in analyzing the 1999 Intrusion Assessment experiments. All the examples in this paper, with the exception of the Norton AntiVirus™ example, have been tested as part of this scenario.

6. Related Work

Although sophisticated IDSes have not been around many years, there have already been a number of efforts to make them interoperate. This interest is driven by practical considerations, including:

- individual detectors have blind spots–using multiple IDSes can cover these
- reliability can be improved with the right mixture of IDSes
- the scope of the network to be protected can be too large for a single IDS

The advantages of cooperation are contingent on a model of interaction and some shared language. Both advantages and difficulties of this endeavor were well illustrated in the results of the 1999 Lincoln Labs IDS evaluations. Our analysis of this data showed that combining judgements of participants could yield superior decisions. Yet despite careful preparations in the staging of the experiments, valuable information from detectors was effectively lost to higher level correlators.

Maximizing the information transmission is one of the goals of related work on IDS interoperability. Several of the proposed frameworks, some still under development, have informed our own research.

Among the most extensive of the proposals for interoperability is the Common Intrusion Detection Framework (CIDF) [4]. CIDF contains a high-level model consisting of event generators, analyzers, databases, and responders. CIDF specifies a Common Intrusion Specification Language (CISL) that is used to communicate between the components. The CISL syntax employs nested S-expressions with a fairly rich vocabulary to form messages describing attacks. The language includes nouns describing the subjects and objects, and verbs, such as "delete" and "open session". It also has modifiers that define attack attributes, e.g. when, outcome, by-means-of. Conjunctions allow CISL clauses to be logically combined. While quite powerful, some IDS authors have found CISL to be unwieldy, and to date its practical applications have been limited. It has, however, been influential in shaping other efforts.

The Intruder Detection and Isolation Protocol [11] is an infrastructure for integrating IDSes and automated response components. IDIP has been tested with a variety of IDSes, boundary controllers, and host-based responders. It provides a discovery coordinator API to allow components access to services including data management, situation display, access to network management and response policy management. IDIP uses CISL as the attack description language. The emphasis in IDIP is on data management and secure communications between diverse components.

EMERALD is a framework for multiple IDSes developed at SRI International [10]. Architecturally, it consists of a set of independent *monitors*, each with analysis and response capabilities. A monitor includes both signature detection and statistical profiling engines, and possiby other components conforming to a common API. A monitor has responsibility for its local corner of the network. These can be hierarchically composed to scale to an enterprise level. Within a monitor, there are "target specific resource objects". These contain all analysis-target-specific information needed to configure a single service monitor, e.g. FTP. Thus a network with many similar nodes or subnets could configure EMERALD by making small changes to the same template set of resources. Like the IRM, this could ease configuration significantly; however, it falls well short of the general information model we have proposed.

The IETF Intrusion Detection Working Group has a draft standard Intrusion Detection Message Exchange Format (IDMEF) [1]. Like CISL, IDMEF attempts to facilitate information sharing between IDS and response systems. The IDMEF syntax is based on XML. While less flexible than CISL, it is arguably simpler to use, and developers should benefit from the recent growth in XML tools.

Our work employs reports consistent with the proposed IDMEF specification. We have made available to the DARPA research community a C/C++ API for generating such reports.

The cited frameworks are limited to the description of specific incidents. In our IRM, we attempt to providing a more general model of the protected network, both at a physical and functional level. This model is key to enabling more sophisticated reasoning.

7. Conclusions

The SCYLLARUS system provides an architecture for integrating multiple Intrusion Detection Systems into a framework for overall network security situation assessment. Unlike other approaches to IDS fusion, our approach makes use of a rich knowledge base, the Intrusion Reference Model, to provide a comprehensive overview, reducing security officer information overload and filtering false positives.

Our future work will move in two directions. First, we will expand the existing SCYLLARUS prototype and validate it on larger and more realistic situations. In parallel, we will attack research problems that would provide obstacles to successful fielding of SCYLLARUS. Where the former is concerned, we must reengineer aspects of the system to make it cleaner and more elegant, must expand our coverage and conduct experiments "live" with real IDS reports. We must verify that our algorithms scale successfully to large installations, and develop approaches to have multiple SCYLLARUS systems work together.

8. Acknowledgments

Support for this work was provided by DARPA/AFRL through contract number F30602-99-C-0177. Thanks to Giovanni Vigna and Dick Kemmerer of R.A. Kemmerer Associates for their advice and help in designing the Intrusion Reference Model NERD. Thanks to our anonymous reviewers for many suggestions and corrections.

References

[1] "Intrusion Detection Message Exchange Format," `http://search.ietf.org/-internet-drafts/-draft-ietf-idwg-idmef-xml-01.txt`.

[2] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick, "CLASSIC: A Structural Data Model for Objects," in *ACM SIGMOD International Conference on the Management of Data*, pp. 58–67, 1989.

[3] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida, "Living with CLASSIC: When and How to use a KL-ONE-like Language," in *Principles of Semantic Networks*, J. F. Sowa, editor, chapter 14, pp. 401–456, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1991.

[4] R. Feiertag, C. Kahn, P. Porras, S. Schnackenberg, S. Staniford, and B. Tung, "A Common Intrusion Detection Language (CISL)," Available at: `http://www.gidos.org/drafts/language.txt`.

[5] A. K. Ghosh, A. Schwartzbard, and M. Schatz, "Using Program Behavior Profiles for Intrusion Detection," in *SANS Workshop on the State of the Art and Future Directions of Intrusion Detection and Response*, February 1999.

[6] A. K. Ghosh, J. Wanken, and F. Charron, "Detecting Anomalous and Unknown Intrusions Against Programs," in *Proceedings of Annual Computer Security Applications Conference (ACSAC'98)*, December 1998.

[7] M. Goldszmidt and J. Pearl, "Rank-based systems: A simple approach to belief revision, belief update and reasoning about evidence and actions," in *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA, October 1992.

[8] K. Ilgun, R. A. Kemmerer, and P. A. Porras, "State Transition Analysis: A Rule-based Intrusion Detection Approach," *IEEE Transactions on Software Engineering*, vol. 21, no. 3, , mar 1995.

[9] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1988.

[10] P. Porras and P. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," in *National Information Security Conference*, 1997.

[11] D. Schnackenberg, K. Djahandari, and D. Sterne, "Infrastructure for intrusion detection and response," in *DARPA Information Survivability Conference and Exposition(DISCEX-2000)*, 2000.

[12] B. Schneier, *Secrets & Lies*, John Wiley & Sons, 2000.

[13] T. Shimomura and J. Markoff, *Takedown : The Pursuit and Capture of Kevin Mitnick, America's Most Wanted Computer Outlaw — By the Man Who Did It*, Hyperion, 1996.