Model-based Intrusion Assessment in Common Lisp

Robert P. Goldman
SIFT, LLC
rpgoldman@sift.info

Steven A. Harp
Adventium Labs
sharp@adventiumlabs.org

Publication note This is a revised, updated, and shortened version of a paper that was presented at the International Lisp Conference, Cambridge, MA, USA, March 2009.

Introduction

We describe the Scyllarus system, which performs Intrusion Detection System (IDS) fusion, using Bayes nets and qualitative probability.[1] IDSes are systems that sense intrusions in computer networks and hosts. IDS fusion is the problem of fusing reports from multiple IDSes scattered around a computer network we wish to defend, into a coherent overall picture of network status. Scyllarus treats the problem of IDS fusion as an abduction problem, formalized using Bayes nets and Knowledge-based Model Construction (KBMC). Because of the coarseness of the data available, Scyllarus uses a qualitative framework, based on System-Z+. Qualitative Bayes nets allow Scyllarus to exploit the strengths of probabilistic reasoning, without excessive knowledge acquisition and without committing to a misleading level of accuracy in its conclusions. The Scyllarus system gave excellent results on a medium-sized corporate network, where it was in continuous use for approximately four years, and was validated in a DARPA-funded assessment. Under US Federal government funding, we are now working to adapt Scyllarus to analyze detection reports from sensors monitoring very high speed (10 - 100 Gb/second) networks in a project called "SMITE."

Intrusion detection

The function of Scyllarus is to take reports from multiple intrusion detection algorithms and fuse them into a coherent picture of the state of the defended network (together with some information about the environment in which that network operates). To perform this task, Scyllarus uses Bayesian (probabilistic) reasoning, primarily to answer two (interrelated) questions:

1. Is the notification (are the notifications) that Scyllarus has received from the algorithms likely to reflect a false positive?

2. Is there a benign explanation that can explain away the notification or notifications that Scyllarus has received? For example, a flood of SMTP messages with duplicated content from a particular host might be a sign that that host has been compromised and turned into a spam bot. However, it's also possible that the host is a bona fide mailing list server, and it's just sending out the day's digest messages.

Because the domain does not afford us access to good statistics, we do not use conventional Bayesian reasoning. Instead, we use a qualitative abstraction of probabilistic reasoning, very similar to the big-O scheme familiar to computer scientists, *System-Z+* (Goldszmidt and Pearl 1996).

Existing IDSes are not designed to work together, as part of a suite of sensors. Instead, each program generates a separate, and often voluminous, stream of reports, and fusing them into a coherent view of the current situation is left as an exercise for the user. 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).

Data reduction is a primary goal of Scyllarus. IDS owners regularly either ignore or partially disable them, unable to absorb the massive stream of reports. To get a sense of the gravity of this problem, see Figure 1, which shows how Scyllarus was able to winnow the flow of reports in a small corporate network.

Often the most damning weakness of an IDS is a high false positive rate. In general, with any sensor, one must pay in false positives for whatever is gained 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.

[1] The Scyllarus system is named after the Mantis shrimp, an animal that detects its prey with one of the world's most complex retinas. It is also one of the most formidable animals, for its weight, smashing its prey with heavily calcified clubs. "Mantis shrimp can break through aquarium glass with a single strike from this weapon."(Wikipedia 2008)

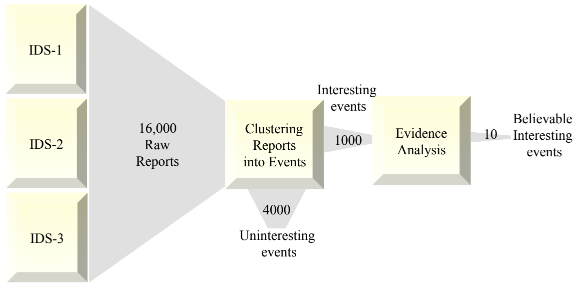

Figure 1: Scyllarus workload reduction.

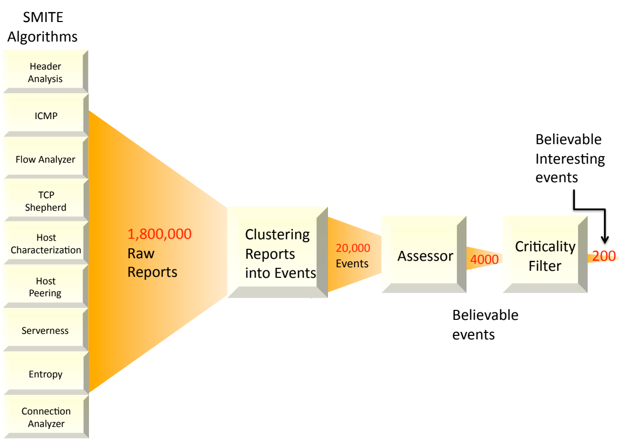

Figure 2: Preliminary results in information filtering for SMITE.

SMITE project

The Scyllarus project was begun at Honeywell in 1999 and has been intermittently active since then. Current development is being done in the context of the SMITE system, a BBN project funded by DARPA's[2] Scalable Network Monitoring (SNM) program. The SNM program's goal is to develop new approaches to network-based monitoring that deliver performance capabilities orders of magnitude better than conventional approaches, regardless of the network's size and computational burden. BBN's approach deploys pipelined systems as data collectors on networks with multi-gigabit speeds. Special-purpose algorithms are being developed that are able to detect intrusion-relevant events while keeping up with the network flow. The events are aggregated and fused by Scyllarus. Current work on Scyllarus aims at optimizing it to be able to keep up with the flow of events from the hardware-based SMITE sensors, expected to cover 2 to 3 orders of magnitude more traffic. Preliminary results are shown in Figure 2.

[2] DARPA is the U.S. Defense Advanced Research Projects Agency.

Scenario of Use

In this section, we provide a brief scenario of report fusion, to make the earlier discussion more concrete. Note that this example was constructed for expository purposes; it does not correspond precisely to any actual computation. Consider a single sensor reporting that a server in the defended network has opened one or more new ports to external connections (Figure 3). It is possible that the port has been opened by some malware installed on the machine, and that the server is compromised.

The malign explanation is not the only possible one, however. Figure 3 shows that there is an alternative, benign explanation. It is possible that what has really happened is that a new service has been installed on this host ("Legit Svc Added") — that would account for new ports being opened. However, if a new service was legitimately added, we would also expect to see a change in the system's (overt) configuration, but we are not seeing that. On balance, the "compromised host" explanation is considered possible, but not especially likely.

Figure 4 shows how the situation might evolve with the arrival of more evidence for intrusion. Here we see that not only is the host in question accepting connections to a new port, but we have also seen that it is initiating a lot of connections outward, "Initiates Conns," which we infer from reports from two sensors. Typically, we would not expect a server to be initiating outward connections.[3] Intuitively, the pattern of inference is as follows: the new legitimate service explanation would account for the newly opened ports, but would not account for the connection initiation. However, a compromised host (perhaps a host that has been added to a botnet) would explain both symptoms.

Scyllarus Architecture

The architecture of Scyllarus, divided into four modules, is depicted in Figure 5. The first is the input module, made up of the Sensor converters (or "verters") and the Report concentrator. The verters take reports from IDSes and other sensors, translate them into Scyllarus-specific data strucures, and hand them off to the report concentrator for eventual storage and analysis. The second is the *Cluster Preprocessor* (CP), which assembles together sets of reports that could correspond to a single underlying event or process. The CP collects reports that could tend to either reinforce or disconfirm particular hypotheses. The CP builds structures that are similar to belief networks (Pearl 1988). The third component, and the last of the active components, the *Event Assessor* (EA) applies the logic of System-Z+ to evaluate competing explanations (e.g., mailserver versus spam bot) for the reports. The final core component of Scyllarus is the *Intrusion Reference Model* (IRM), a knowledge base describing the environment in which Scyllarus operates, and which supports the processing done by the CP and EA.

[3] with some exceptions such as DNS queries, SMTP transfers if a mail server, etc.

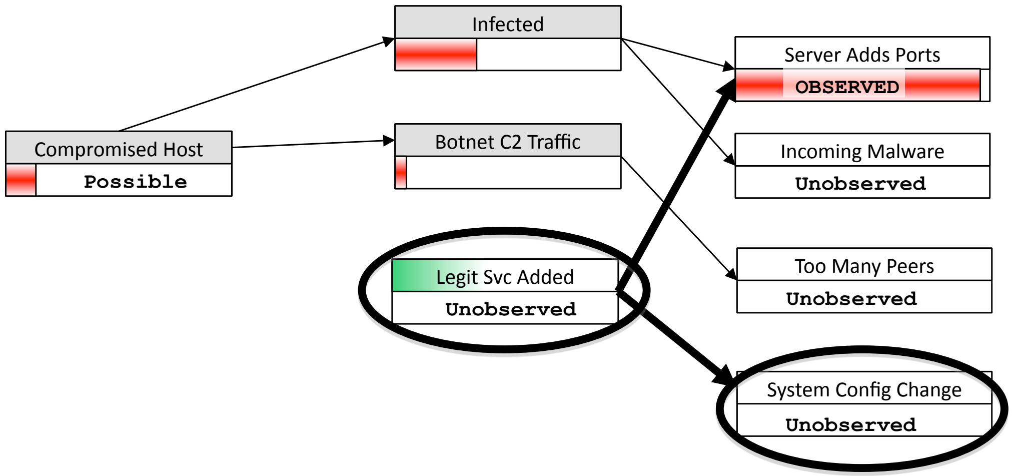

Figure 3: Alternative explanations and the observations they might cause.

Scyllarus input processing

The Scyllarus architecture was developed to flexibly accommodate reports from a diverse and changing set of sensors (primarily IDSes). One may plug arbitrary sets of translators into Scyllarus. These "verters" are small translator programs that translate IDS reports, which do not come in standardized formats,[4] into a standard Scyllarus input report. The verters must be written anew for each IDS, but the effort is not too substantial. For the SMITE project, we have the advantage of a standard report format (implemented as a reporting library to be compiled into each of the sensors) negotiated between the sensor and correlation teams, so that we need only a single SMITE verter.

After the reports have been translated into Scyllarus format, they pass from the verters to the Report Concentrator. The Report Concentrator receives incoming reports from IDSes and buffers the reports, ensuring that the system remains responsive while not losing data The Report Concentrator provides a real-time feed of reports to subscribers, the most important of which are the event database and the Cluster Preprocessor.

Cluster Preprocessor

The Cluster Preprocessor reads raw reports posted by the IDSs, and using background information provided by the IRM, a model of the protected network and a key for intrepreting IDS messages, produces clusters of IDS reports to be evaluated as *events* explaining the reports. A single IDS report may give rise to one or more such clusters.

The Cluster Preprocessor follows a simple processing loop:

1. Read the next IDS report from a socket stream connected to the Scyllarus Report Concentrator.
2. Match the IDS-provided report type to one or more intrepretations known to Scyllarus. Each provides a hypothetical *event* purporting to explain the report. Scyllarus has models for various common network and host-based IDSs.
3. Search through already hypothesized events for ones that would explain each intrepretation of the new report. Criteria for consistency vary from one type of event to another, and are specified in the IRM as a set of "event test" objects to be satisified. Some basic criteria include:
   - occurrence within an acceptable temporal window
   - directed at the same target host and/or port
   - apparently originating from the same source
   - sharing a common user or login session
4. Propose new events as needed when existing ones are inconsistent with the new report.
5. Assemble new (or recently modified) events into other larger-scale events. This allows Scyllarus to consider multi-step attacks. Further model-based tests for consistency are applied to this clustering.
6. Submit new or modified events and their supporting reports for evaluation.

Identifying Independent Subsets of Events The Cluster Preprocessor is driven entirely by incoming reports. It spools events that need likelihood evaluation to the EA, but does not halt clustering to wait for assessment to complete, since this evaluation time may be relatively long–extracting the most likely interpretations from a very large ATMS network may take seconds.

Instead, the EA runs in a separate thread and evaluates independent clusters of events as its processing budget al-

[4]Even when we found sensors that complied with some standard, such as IDMEF, the standard wasn't helpful, because it did not specify semantics sufficiently to allow us to simply accept the reports.

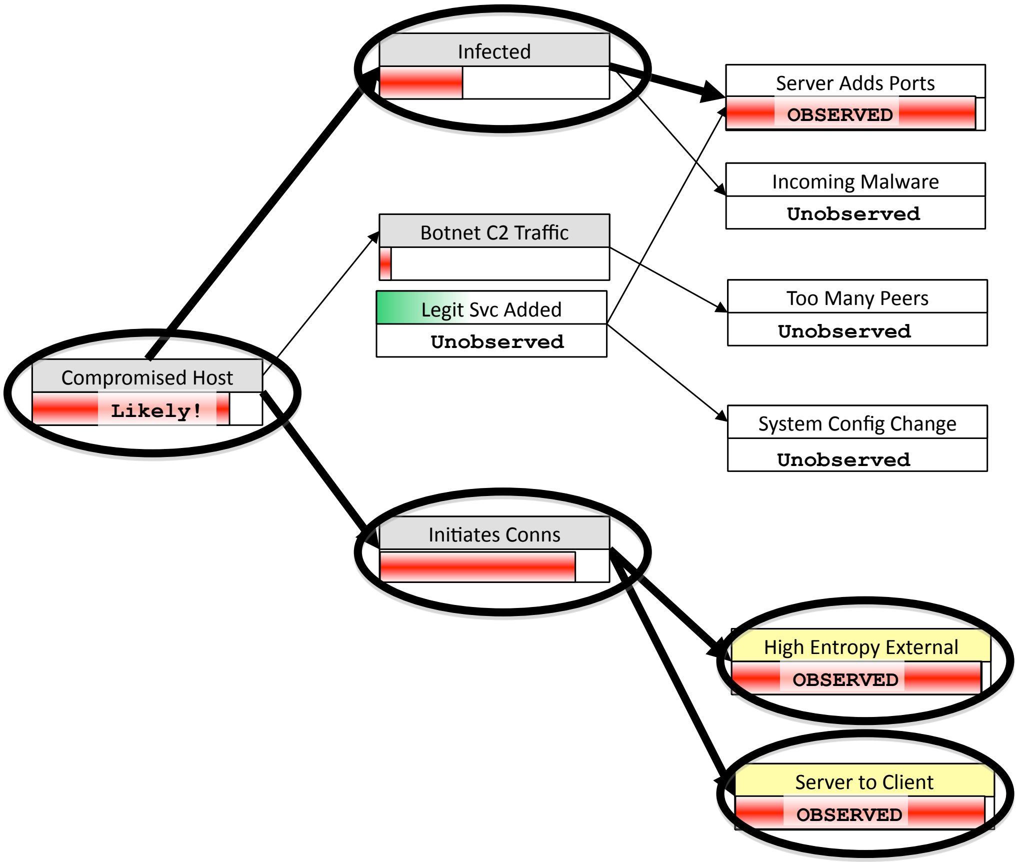

Figure 4: More evidence of intrusion arrives.

lows. The fundamental independence criterion is that the set of events implicitly defines a directed acyclic graph. Events are linked to other events and to reports according to the following relationships:

Supporters $E \rightarrow R$. This is a relationship between an event and an IDS report that provides direct evidence for it. For example, a certain network IDS rule that is triggered by a sequence of bytes commonly found in propagation of the Peacomm trojan could support an event hypthesizing the malware infection of the target host with Peacomm.

Components $E(whole) \rightarrow E(component)$. This is a relationship between events and other events that might be component parts of them. For example, one component of a DNS cache poisoning attack is the sending of a flood of DNS queries.

Manifestations $E(underlying) \rightarrow E(manifestation)$. This is a relationship between an event and other events that might occur because the first event is occurring. For example, a worm's propagation might manifest as repeated content transmission from the attacking host.

Specializes $E(specific) \rightarrow E(more\ general)$. Different IDS algorithms operate at different levels of resolution. This link is a relationship between one proposed event and a more specific proposed event (e.g. induced by a more precise type of IDS) that could be identical.

The graph is defined by the closure under the above four links (and their inverses) of the set of events given to the assessor. Typically, this graph will have many connected components that are not connected with each other. The connected components are the independent subsets that the EA operates on separately.

Event Assessor

The Event Assessor uses qualitative probabilistic/Bayesian reasoning to assess the likelihood of various event hypotheses. In the current Scyllarus architecture, the EA is invoked by the Scyllarus Cluster Preprocessor. The EA accepts as input clustered event hypotheses, together with their supporting reports. The EA builds qualitative probabilistic inference networks corresponding to the clustered reports and events. It uses these networks to compute posterior surprise levels (qualitative likelihoods) for the event hypotheses. These surprise levels are recorded in the event structures, and may be written into the IRM database for persistent storage.

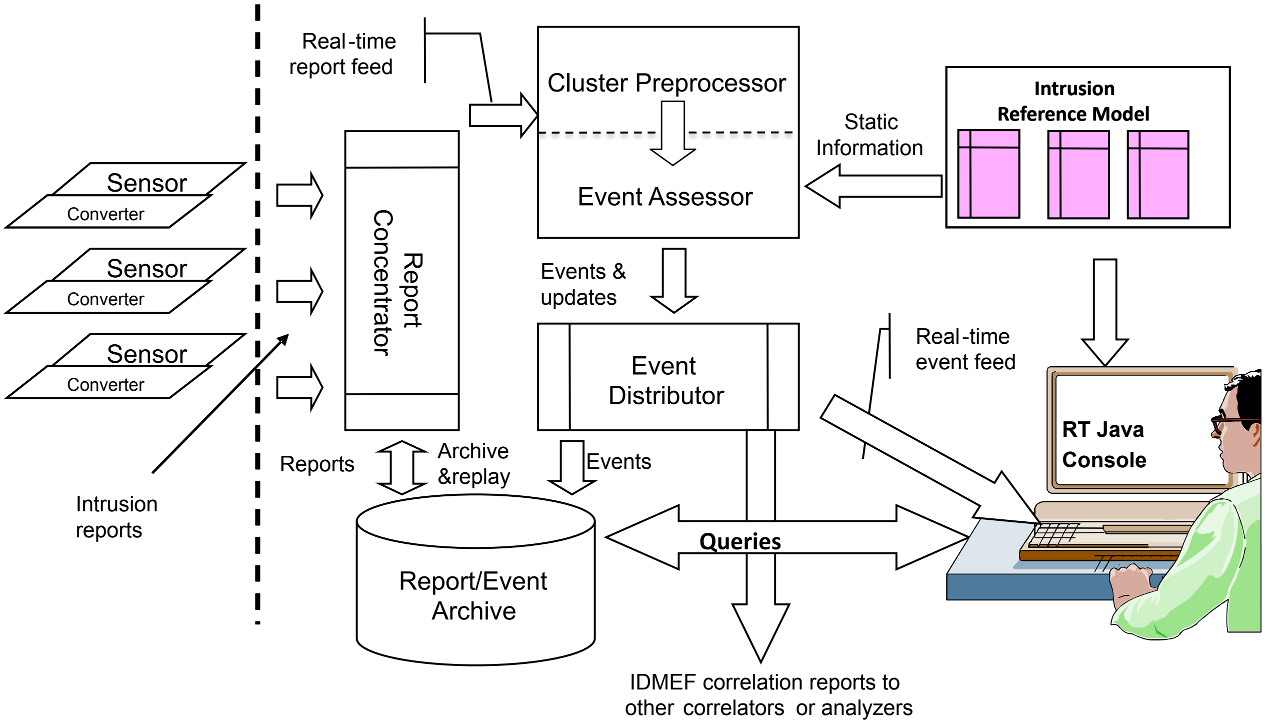

Figure 5: Scyllarus architecture

The EA must perform four primary computational tasks:

1. Identify independent sub-graphs in the network defined by the event and report structures and the links between them. This is done with depth-first search.
2. Build Bayes networks and identify evidence interpretations in these networks. The Bayes nets are implemented as ATMS dependency networks. The ATMS computes interpretations corresponding to the Bayes networks using its labeling algorithms.
3. Extract the set of most likely interpretations from an ATMS network. This is done using search algorithms. We search for interpretations of minimal cost. The solution used is primarily one of depth-first iterative deepening, although some special cases are handled differently.
4. Extract surprise levels from the most likely interpretations. Currently, we simply differentiate between three classes of events: plausible events, that appear in some of the most likely interpretations, unlikely or implausible events, that do not appear in any of the most likely interpretations and likely events, which are plausible events and, additionally, whose negation never appears in a likely interpretation. That is, for a plausible event, E, it is also possible that not(E) is plausible. An event E is likely if E is plausible and not(E) is implausible. Extracting surprise levels may simply be done by examining the interpretations generated in step 3.

Underlying Theory: System-Z+ Qualitative Probability

We have taken an approach, based on qualitative probabilities, that shares the basic structure of normal probability theory but abstracts the actual probabilities used. We did this primarily to simplify knowledge acquisition and make it as simple as possible to incorporate new IDSes into the Scyllarus architecture. This approach may also permit cheaper computations than the normal probability calculus, but that remains to be seen.

Our approach is based on System-Z+, developed by Moisés Goldszmidt and Judea Pearl (1996). In System-Z+, events are given a natural number rank, κ, that corresponds to their degree of surprise (e.g., a rank of one is more surprising than zero). The semantics of this scheme comes from a set of probability distributions in which the probabilities are polynomials in some infinitesimal ϵ. In this scheme, the κ rank corresponds to the exponent of the leading term of the polynomial. The scheme is similar to the "big-O" notation used for evaluating computational complexity in computer science.

In practical terms, the effect of this semantics is to give System-Z+ a qualitative flavor by providing a "ladder" of events of qualitatively different orders of likelihood. Of course, we sacrifice exactness in doing so; we lose the ability to talk about events being slightly more or less likely. However, this sacrifice of exactness is not an issue in the Scyllarus intrusion detection application.

The EA must combine the judgments of a wide variety of intrusion detection systems (and potentially other relevant information sources), that use widely varying sources of information and algorithms. Further, in general we will not have access to the internals of these sensors. In such an environment, it is not realistic to expect good models of the response of these sensors; in particular, exact measures of P(sensor response—event) are not available. There have been some attempts to investigate sensor response (e.g., the studies conducted by Lincoln Labs (Lippmann et al. 2000)), but the results seem heavily dependent on the context in which the sensors are deployed.

The issue of prior probabilities also militates against the use of exact probabilities. In order to use an exact Bayesian method, we would need not only the detection probability, $P(sensor\ response|event)$, and the false alarm probability, $P(sensor\ response|\neg event)$, but also $P(event)$, a measure of the prior probabilities of the events that interest us, in this case the attacks and the benign events that can cause false positives. Even in the most constrained environments, the probabilities of the various attacks, are unlikely to be available to us, and the Scyllarus system is designed for application across a wide variety of enterprises. Further, the probability distributions for benign events are likely to be of odd forms (e.g., one's own network-mapping software runs at particular times of the day). So our solution must tolerate vague measures of likelihood.

Finally, in this domain, as with most practical applications of probabilistic updating, the effect of the evidence will *usually* overwhelm the effect of the prior likelihoods(e.g., (Pradhan et al. 1996)). So inexactitude in the quantities specified will not matter to our final conclusions.

As far as computation is concerned, we may apply the normal operation of probability theory: conditionalization, Bayes' law, etc. However, the arithmetic operations we use must change. Rather than multiplying probabilities, we add degrees of surprise. Rather than adding probabilities, we use min. Goldszmidt and Pearl (1996, p. 59) provide the following substitutions in their paper:

$P(\omega) = \sum_{\phi \in \omega} P(\phi)$	$\kappa(\omega) = \min_{\phi \in \omega} \kappa(\phi)$
$P(\omega) + P(\neg\omega) = 1$	$\kappa(\omega) = 0 \vee \kappa(\neg\omega) = 0$
$P(\omega\|\phi) = P(\omega \wedge \phi)/P(\phi)$	$\kappa(\omega\|\phi) = \kappa(\omega \wedge \phi) - \kappa(\phi)$

Instead of the probability of an event being the sum of the probabilities of the primitive outcomes that make up that event, the degree of surprise of an event is the minimum of the degrees of surprise of the primitive outcomes that make it up. Instead of having the probabilities of mutually exclusive and exhaustive events sum to one, at least one of a set of mutually exclusive and exhaustive events must be unsurprising. Finally, we have an analog of Bayes' law in which the normalizing operation consists of subtraction rather than division.

We used Bayesian networks to help us in modeling and solving the correlation problem. Bayesian networks are ways of graphically capturing probabilistic reasoning. They are useful in expert systems because they simplify knowledge acquisition and, by capturing (conditional) independences, simplify computation (Pearl, 1988). In particular, in the domain of intrusion detection, Bayes nets help us capture several important patterns or probabilistic reasoning:

- Reasoning based on evidence merging;
- "Explaining away" reports by alternative explanations. E.g., if a benign event accounts for a number of reports, those reports will be explained away, and no longer provide support for more alarming hypotheses.
- Abstraction reasoning that employs the subclass/superclass relationships in the event dictionary.
- Part/whole reasoning, to recognize complex composite events.
- Distinguishing between judgments that are based on different sensor bases and those that use the same sensor. This helps us distinguish between cases when two sensors provide support for each other and when we simply have redundant reports (e.g., two network intrusion detection systems using exactly the same algorithm that see the same traffic, at two different points).

A Bayesian network is a directed, acyclic graph (DAG) depicting a set of random variables. Edges between nodes in the DAG represent causal influences. Using a Bayesian network, we can capture a joint distribution factorized into unconditional probabilities for root nodes and conditional probability tables for non-root nodes. The conditional probability tables contain probability distributions for the child nodes, conditioned on all the values of their parents.

There are a number of efficient algorithms for finding the posterior distributions of Bayesian networks, conditional on observations of some of the random variables. These algorithms may readily be adapted to provide posterior κ rankings instead of probabilities.

System-Z+ and the ATMS

The Scyllarus Event Assessor (EA) does System-Z+ Bayes net inference by representing the Bayes nets in a Assumption-based Truth Maintenance System (ATMS) with weighted assumptions, and finding minimum cost environments for the ATMS networks. We adopted the ATMS approach simply because the ATMS code was readily available, and we expected later to replace the ATMS with a special-purpose System-Z+ Bayes net evaluator. However, with the exception of some pathological cases, which we handle specially, System-Z+ inference has never been a bottleneck in Scyllarus.

An ATMS (deKleer 1986) is a propositional logic database with data dependencies or *justifications*, that record the derivation of the *literals* from distinguished *assumptions*. ATMSes can be used to encode Bayes networks (Charniak and Goldman 1988; Provan 1989). Each value assignment to a random variable in the Bayes net is represented by a literal. Each conditional or unconditional probability in the Bayes net is represented by an assumption. For example, in a Bayes net with the (boolean) nodes A, B, C and edges

$A \rightarrow C, B \rightarrow C$, there will be literals $l_A, l_B, l_C, l_{\overline{A}}, l_{\overline{B}}, l_{\overline{C}}$ and assumptions:

$$\begin{gathered} a_A, a_{\overline{A}}, \\ a_B, a_{\overline{B}}, \\ a_{C|AB}, a_{C|\overline{A}B}, a_{C|A\overline{B}}, a_{C|\overline{AB}}, \\ a_{\overline{C}|AB}, a_{\overline{C}|\overline{A}B}, a_{\overline{C}|A\overline{B}}, a_{\overline{C}|\overline{AB}} \end{gathered}$$

There will also be justifications representing the probabilistic entailments. For example, $\langle a_A \rightarrow l_A \rangle$ and $\langle a_{\overline{A}} \rightarrow l_{\overline{A}} \rangle$ illustrate the representation of root nodes and unconditional priors in the Bayes net. Posterior probabilities can be computed by collecting the set of ATMS *environments* consistent with the observation set and normalizing the prior probabilities accordingly. Computing System-Z+ κ values may be done in a similar way, with some differences to account for the differences in the calculi.

Intrusion Reference Model

The process of knowledge-based model construction is driven largely by extensive models of existing IDSes. The task of putting the various sorts of IDS reports on a common semantic footing has proved more challenging than expected. There is little consistency in terminology between (or even within) IDSes and often quite different principles of detection are employed, making nominally similar messages less than fully comparable.

An extensive ontology for expressing the IRM has evolved over several versions of Scyllarus to become the foundation of our approach to this problem. All of the IRM concepts are expressed in this modular ontology, maintained in the Protégé (Noy et al. 2001) tool. Part of this ontology is used to model the protected computers and network, while other parts are devoted to the characterstics of the defenses. In particular, an IDS ontology module exists in the IRM for each sort of IDS supported. These models pertain both to hypothesis formation and evaluation, so we will discuss them briefly here.

Each type of IDS is capable of emitting a range of different reports describing some aspect of a possible intrusion it has observed. Commonly these different messages will be derived from different discrete elements such as rules or detection modules within the IDS. Some have a repertoire of just a few messages (e.g. firewalls) while others have thousands (e.g. Snort). Scyllarus maintains an explicit model of each message generating element.

Each message-generating element in a supported type of IDS has one or more models in the IDS specific ontology module. These models, which we call *report signatures*, are causal interpretations of the report in terms of the ontology. A signature is a template that describes a possible attack or other event that may give rise to the given sort of report. It uses several extensive hierarchies of IRM concepts to do so. The first, is a taxonomy of *operations*. Operations are elementary actions on the protected system that may be undertaken for good or ill, such as reading a file, starting a process, or executing a step in a protocol. Scyllarus has an a-kind-of hierarchy listing hundreds of operations. A different IRM taxonomy models the possible intentions of a causal agent. The *intent* may be specified in a report signature to cast a benign or malevolent interpretation of the operation. The intent classification also provides a rough measure of the seriousness of the event. Modeled intentions vary from specific sorts of denial of service, the seizing of privileges, as well as administrative (wholesome) intentions such as "achieve file-server archival backup to tape."

Many signatures also cite specific victim software or systems, vulnerabilities (Scyllarus incorporates CVE and other classification schemes), or certain "malware" (malicious programs) that are implicated. An important part of the signature model is the qualitative false positive rate of generating element, and the presumed rarity of the interpretation. Various other details of the representation are omitted here for brevity.

Experience

Scyllarus was used to monitor an operational network of over 500 workstations and servers using three different types of network intrusion detector and two different types of host intrusion detectors located at various points in the network over a period of 4 years, at which time the sensors were relocated to a small test network with limited access to network traffic.[5] Over the period of its use, Scyllarus proved itself to be a substantial advance in the state of the art for IDS fusion.

Scyllarus routinely handled quiet day traffic of 10,000 – 20,000 IDS reports per day. On more "exciting" days, the traffic was considerably heavier; e.g., on the day of the release of the Code Red worm, Scyllarus received more than 1,000,000 reports. Our current efforts aim at networks where upwards of a million sensor reports might be expected on a normal day.

We tested Scyllarus with controlled exploits on our network and the system has responded appropriately. We were also able to detect an episode of penetration testing conducted without warning by an independent security team.

In 2003, the ability of Scyllarus attacks was demonstrated in an evaluation conducted as part of the DARPA Cyber Panel program (Haines et al. 2003). In this evaluation, a number of network attacks were launched by a dedicated Red Team in a simulated warfare planning environment.

Scyllarus addresses the information overload faced by IDS users. See Figure 1 for representative data on Scyllarus's report filtering on a small corporate network. Our current efforts are aimed at extending this performance to ultra high-speed networks, and the preliminary results, as seen in Figure 2, show great promise.

Related Work

SecurityFocus has developed the Attack Registry and Intelligence Service (ARIS) (ARIS 2003). The ARIS extractor collects IDS reports from four different IDSes, formats them in XML, and presents them in an incident console. However, it makes no attempts to fuse the reports or weigh the evidence for and against them.

MetaSTAT is a fusion system that is built on a set of STAT-based IDSes (Vigna, Kemmerer, and Blix 2001). STAT is a

[5]Walt Heimerdinger, personal communication

signature-based IDS that detects events by matching against extended finite-state event models. MetaSTAT uses finite-state models of across-sensor events to consume at a higher level the events generated by lower-level sensors. MetaSTAT does not attempt to judge the plausibility of different events.

EMERALD/eBayes (Valdes and Skinner 2001) fusion is the most similar to Scyllarus. The eBayes sensors are Bayes net-based, and the correlation approach allows "upstream" sensors to adjust the priors on "downstream" sensors. eBayes fusion is limited to clustering together alerts that meet a similarity criterion; they do not have models of high-level events as in the Scyllarus IRM.

Prelude Correlator (Vandoorselaere 2008) is part of the open source Prelude IDS information system, and allows users to analyze reports sent to Prelude from compatible IDSs. Users provide rules written in Lua (Ierusalimschy, de Figueiredo, and Celes 2006), a scripting language inspired by Scheme and Icon. Its function is closest to the Scyllarus clustering preprocessor, but knowledge resides in stateful rules instead of an ontology of attacks.

A commercial product, Arcsight Enterprise Security Manager (ArcSight 2008), also ties correlated IDS reports to an installation's security goals and vulnerability information.

Conclusions

The Scyllarus system is an AI application that handles high speed online fusion and filtering of network intrusion detection data. Its reasoning involves a combination of expert domain knowledge and reasoning under uncertainty. Its design reflects two key strategies. First, it manages the domain knowlege about the peformance of individual intrusion detection systems and the defended network using a generic ontology, allowing reports from a wide variety of intrusion detection systems to be described in a common framework. Second it simplifies the reasoning under uncertainty problem through a qualitative probablilty scheme, System-Z+, which demands little in the way of subjective estimation from experts configuring Scyllarus, but provides adequate resolution for this fusion problem. Current testing is providing encouraging results on the ability of Scyllarus to scale to monitoring future high-speed networks.

Acknowledgments

This material is based on work funded by the DARPA Scalable Network Monitoring program under contract to SPAWAR Systems Center. Approved for Public Release, Distribution Unlimited.

References

ArcSight. 2008. Arcsight enterprise security manager. `http://www.arcsight.com/product_info_esm.htm`.

2003. Attack registry & intelligence service. `http://aris.securityfocus.com/AboutAris.asp`. ARIS analyzer Data Sheet.

Charniak, E., and Goldman, R. P. 1988. A logic for semantic interpretation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 87–94.

deKleer, J. 1986. An assumption-based TMS. *Artificial Intelligence* 28:127–162.

Goldszmidt, M. G., and Pearl, J. 1996. Qualitative probabilities for default reasoning, belief revision and causal modeling. *Artificial Intelligence* 84(1–2):57–112.

Haines, J.; Ryder, D. K.; Tinnel, L.; and Taylor, S. 2003. Validation of sensor alert correlators. *IEEE Security and Privacy* 1(1):46–56.

Ierusalimschy, R.; de Figueiredo, L. H.; and Celes, W. 2006. *Lua 5.1 Reference Manual*. Lua.org.

Lee, W.; Mè, L.; and Wespi, A., eds. 2001. *Recent Advances in Intrusion Detection (RAID 2001)*, number 2212 in LNCS. Springer-Verlag.

Lippmann, R.; Haines, J. W.; Fried, D. J.; Korba, J.; and Das, K. 2000. Analysis and results of the 1999 DARPA off-line intrusion detection evaluation. In *RAID '00: Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection*, 162–182. London, UK: Springer-Verlag.

Noy, N. F.; Sintek, M.; Decker, S.; Crubezy, M.; Fergerson, R. W.; and Musen, M. A. 2001. Creating semantic web contents with protege-2000. *IEEE Intelligent Systems* 16(2):60–71.

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

Pradhan, M.; Henrion, M.; Provan, G.; Favero, B. D.; and Huang, K. 1996. The sensitivity of belief networks to imprecise probabilities: an experimental investigation. *Artificial Intelligence* 85(1–2):363–397.

Provan, G. 1989. An Analysis of ATMS-based Techniques for Computing Dempster-Shafer Belief Functions. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, 1115–1120. Morgan Kaufmann Publishers, Inc.

Valdes, A., and Skinner, K. 2001. Probabilistic alert correlation. In Lee et al. (2001).

Vandoorselaere, Y. 2008. Prelude correlator. `https://trac.prelude-ids.org/wiki/PreludeCorrelator`. Prelude Correlator online documentation.

Vigna, G.; Kemmerer, R. A.; and Blix, P. 2001. Designing a Web of Highly-Configurable Intrusion Detection Sensors. In Lee et al. (2001), 69–84.

2008. Mantis shrimp. Wikipedia entry.