A Framework for Cooperative Intrusion Detection

Deborah Frincke Don Tobin Jesse McConnell
Jamie Marconi Dean Polla

Center for Secure and Dependable Software
Department of Computer Science
University of Idaho
Moscow, ID 83844-1010

Abstract

The trend towards a strong interdependence among networks has serious security implications. Not only does the compromise of one network adversely affect resources needed by others, but the compromised network may be part of a multi-network attack targeting other systems. The task of

2 Framework for Cooperative Intrusion Detection

In this section we discuss principles for a cooperative intrusion detection system framework, where information sharing takes place both between distinct domains and within domains. Domains may have varying or even conflicting policies, and will not necessarily have the same mechanisms in place to identify system misuse.

We have found it useful to consider these principles with respect to the relation between participant pairs, and have identified the following primary relationships:

Peer: a relationship of equals between hosts, typically in different policy domains. Neither host controls the other, although they may send requests or information between themselves. Peers do not necessarily trust one another, and the level of trust is not necessarily identical.

Manager: a manager is a host that provides instructions regarding which data is to be collected, when warnings should be issued, etc. to a set of dependent hosts. Managers set a central policy for a group of hosts, generally within the same policy domain. Managers need not trust their subordinate hosts.

Subordinate/Managed Host: a host that receives some or all of its data collection and transmission policy from outside. Managed hosts may modify or add to this policy, and may themselves manage other hosts. Managed hosts must fully trust their managers, and will usually be within the same policy domain.

Slave Host: a host that receives all of its data collection and transmission policy from outside. Slave hosts may not modify or add to this policy, although they may themselves manage other hosts. Slave hosts must fully trust their managers, and will always be within the same policy domain.

Friend: a relationship of equals between hosts. Neither host controls the other, although they may send requests or information between themselves. Friends always trust one another, and the level of trust is identical. Friends should be within the same domain.

Symbiote: a relationship of interdependent hosts. Neither host controls the other, although they may send requests or information between themselves. Hosts with this relationship are expected to be 'identical' in terms of policies and security labels.

Since space does not permit a detailed discussion of the rules for each relationship, we have chosen to focus on the peer relationship. The peer relationship is appropriate between domains having different policies, or where domains do not fully trust one another, or a controlling relationship between them is undesirable.

We have identified the following principles as important in the development of a framework for cooperation between domains:

1. Local control over policy decisions must be maintained.
2. Data collection should be autonomous but cooperative.
3. Actions based on data obtained elsewhere should include a local trust factor, since incoming data may not be reliable.
4. The enforcement of policy and the identification of policy violations should be separate issues.
5. Transactions should be validated.
6. Structure information is shared both horizontally and vertically.
7. Data collectors should be layered, overlapping, and provide both data reduction and sanitization.
8. Audit tool management and visualization systems should be integrated.

Local Policy Control We believe that local control over policy decisions is one of the most important facets of cooperative intrusion detection systems. Cooperating domains may not fully trust one another and will be highly unlikely to grant any outsider the right to change their internal methods of misuse data collection or information flow. The local host should always determine whether a given interaction should take place based on its own local policies. Even for hierarchical relationships such as master/slave, the slaved host should have facilities for checking its own policies before reacting to the incoming message.

Autonomous, but cooperative data collection: Although the data collected and shared is determined locally, hosts and domains should share relevant information. This may involve collecting and sharing data which is irrelevant to the source host's own security policy but is needed by a peer or manager. However, even when hosts are obtaining data needed to identify policy violations for external domains, the decision regarding whether to collect and transmit this data is local, and should include the realization that the recipient will 'own' the data once transmitted and may decide to disseminate it further.

Data reliability: Actions based on data obtained elsewhere should include a local trust factor, since incoming data may not be reliable. For example, the source host may have been compromised and be unknowingly transmitting misleading data. Thus, the intrusion detection system using data obtained via the cooperative framework should employ a method which takes into account the local host's trust in the authenticity and the integrity of the data.

Policy enforcement and identification: The enforcement of policy and the identification of policy violations should be separate issues. Hosts may be identifying policy violations for another domain than their own, but they should not be responsible for enforcing those policies.

Validated Transactions: It will be necessary to perform authentication of some kind between cooperating hosts. This authentication might involve digitally signed messages, for instance.

Structure of sharing: Information sharing may be both horizontal and vertical. For example, a manager and its subordinates may perform vertical sharing, where subordinates transmit data "upwards" to the manager and managers transmit commands "downwards." The trust relationships between participants are likely to be strong in the downward direction (subordinates trust their manager) and weaker in the opposite direction (managers may not fully trust their subordinates). Between peers, sharing should be horizontal, due to the more collaborative nature of their relationship.

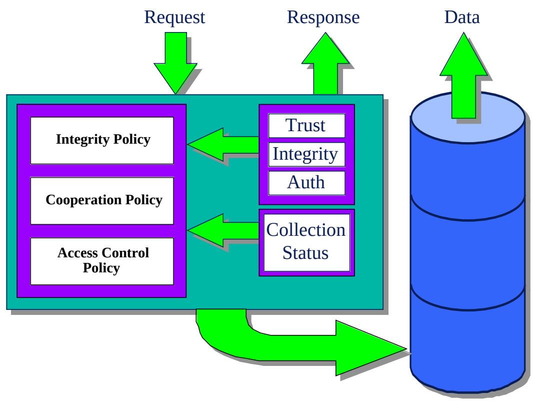
Data collectors: Data collectors should be overlapping and provide both data reduction and sanitization. Overlapping data collectors are needed whenever one data collector might be subverted or become unavailable. Data reduction is important to reduce the extraneous data transmitted between sharing partners. Data sanitization is needed to eliminate host and network specific attributes of the data. This is important since such data might cause a security risk to the transmitting host.

Integrated audit tool management and visualization systems: As data collection systems become more complex and handle more and more systems, manual configuration of audit tools becomes impossible. Further, examination of voluminous textual information for potential violations is difficult; humans are far better at processing and identifying oddities in graphical forms while systems produce faster initial response. Thus, any data management system should include methods for managing sets of audit tools and providing graphical views of examining such data.

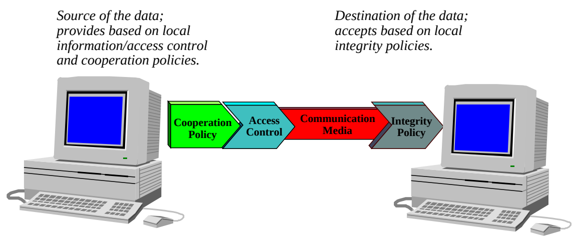
2.1 Policy Sharing Issues

When managing data between cooperating domains, it does not appear to be realistic to require an 'overall' policy encompassing all participants. Instead, each host should let its local policy and local ratings govern its interactions. This permits domains to cooperate even when they disagree on some policy issues. We divide policy into three components: access control, integrity, and cooperation, where requests are handled as in Figure 1(A). Policies are checked using local assessment of integrity, trust relationships, and the status of ongoing data collection activities. These policies interact. One possible interaction is shown in Figure 1(B): the transmitting host relies on its cooperation and access control policies to determine which data it will send, and the receiving host relies on its integrity policy to determine how (and whether) the data will be used. Other combinations are possible as well.

The *access control policy* is used to determine whether a subject (in the context of the evaluation domain) has sufficient clearance to perform a given operation on a particular object. The *integrity policy* is used to determine whether information should be allowed to



A. Handling Information Requests


Source of the data; provides based on local information/access control and cooperation policies.
Destination of the data; accepts based on local integrity policies.
Cooperation Policy
Access Control
Communication Media
Integrity Policy

B. Peer Information E change Policies with Weakened Bell-LaPadula and Biba

Figure 1: Information Exchange patterns

be modified or added to a system. The *cooperation policy* determines whether data which the access control policy permits to be shared will in fact be shared. Providing a separate cooperation policy allows the choice to share to be handled separately from the legality of sharing. The cooperation policy is somewhat reminiscent of a discretionary access policy, since decisions to cooperate are based on the relationship between hosts and should be made on an individual basis. However, cooperation requires expenditure of resources to collect, store, and transmit data, and to manage incoming requests, and sharing of data opens the risk of releasing data which may increase a participant's own vulnerability. Thus, we believe that the cooperation policy should be developed separately. Sample policies may base cooperation on a reciprocal agreement, the difficulty of obtaining the desired information, or on the likely benefit to an individual host. If one host relies on another for some critical functionality, then it is to the benefit of the first host to improve the second's security. If the second host does not provide any critical functionality to the first, then there may be no benefit to collaboration.

2.2 Data Filtering

Our filtering model is drawn from existing technologies such as filtering routers and from sensitivity downgrading issues [Smi92]. Filtering is needed for data reduction and to protect sensitive internal resources. We consider filtering to be a useful mechanism for enforcing aspects of the access control policy as well as an important performance measure. Filtering includes both *data reduction* and *data sanitization*. Data reduction benefits the 'requesting' host, since it reduces the transmission of extraneous information. Although the source host must perform additional data processing, communication costs should be reduced for it as well. Data sanitization benefits the source host by removing potentially sensitive information which might induce or expose a vulnerability.

Filtering may be performed during transmission, data collection, and/or storage. Data transmission filtering is done with respect to the specific target, can be host specific, and has a fine granularity. Collection/storage filtering is done with respect to a general policy regarding what will be stored, is less specific, and will tend to result in either a generally lower level of sensitivity of data being transmitted (since the data will be highly sanitized) or a reduced amount of information being transmitted (since fewer hosts will be allowed to read the data).

3 Hummingbird System

The HMMR protocol has been designed to address the requirements needed to permit network sites to share security relevant data while still retaining local control of data gathering activities, deciding locally how much trust to place in data received from outside sites, and determining how much data to share with outside sites. An early version is described in [FEA96]. Small-scale models of this protocol have been developed [NCD97][ILL97][Auc94][Eva95] and combined in a prototype Hummingbird System. This system is being used as a testbed for exploring practical issues in sharing data between sites, such as how much reliance to place on offsite intruder alerts and misuse data, how data may be shared safely, which collected data makes a quantifiable contribution to intruder identification, and so on.

Hummingbird is formulated as a distributed system for managing misuse data. Hummingbird employs a set of *Hummer* agents, each assigned to a single host or host set. Each Hummer interacts with others in the system through *manager*, *subordinate*, and *peer* relationships. Managers may transmit commands to subordinates; such commands include requirements to gather/stop gathering data, to forward/stop forwarding data, and the like. Peers may send requests for data forwarding/gathering/receiving to peers; the peer decides whether to honor such requests. Subordinates may send requests to their managers as well. Configuration is performed using the interface shown in Figure 3. Kerberos [NT94] is used for authentication and encryption of communication between hummers and for authentication between Hummers and their associated database. Individual Hummers share data based on their own locally controlled and set policies regarding trust, information flow, and cooperation. Simple filtering is also performed by each host.

The prototype Hummers use POSTGRES, an SQL-compliant database, for storing misuse data. By using an SQL database, it is possible to easily obtain subsets of data to: generate reports, generate information for an intrusion detection system, and data for the visualization system (Section 3.2), and supply data to the feature selection routines (under development). The POSTGRES database in the prototype system can supply this information either in text format or as an HTML table.

Audit tools run either individually under each Hummer's control or else as a suite through the Audit Tool Manager (Section 3.1). These audit tools collect security relevant data from system log files, by running system utilities, or by running specialized auditing processes. Each Hummer's configuration includes information about the audit tools it is running, so that if a Hummer is stopped and restarted, it can regain control over the tools or start them up again if necessary. Alert tools are a special case of audit tools, since they only examine a Hummer's own log files. These alert tools perform simple security assessments by looking

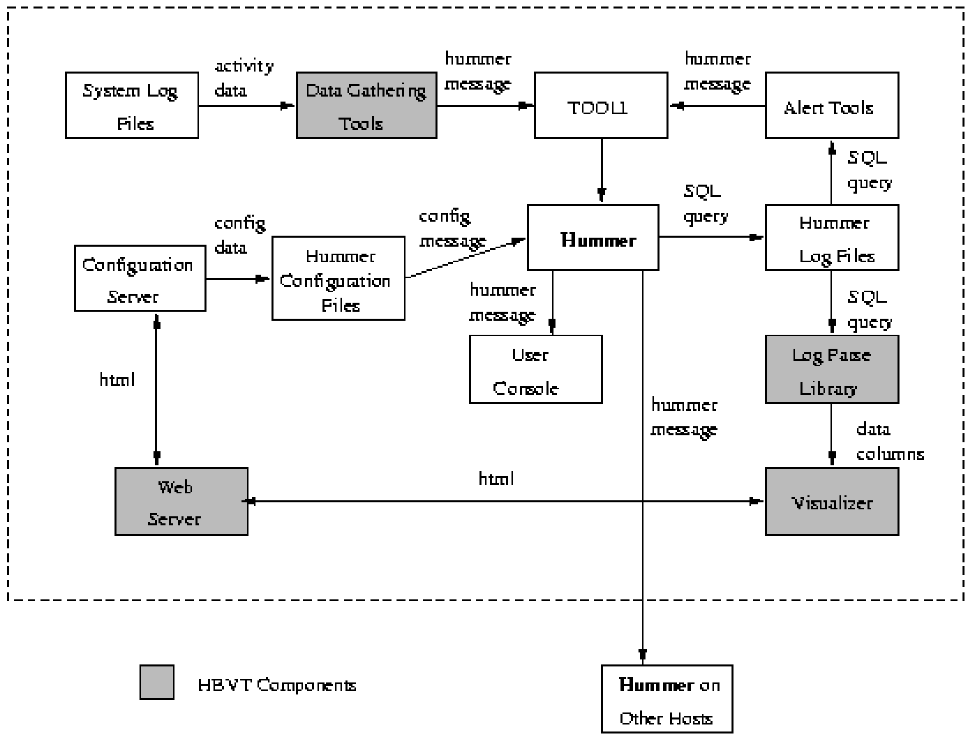
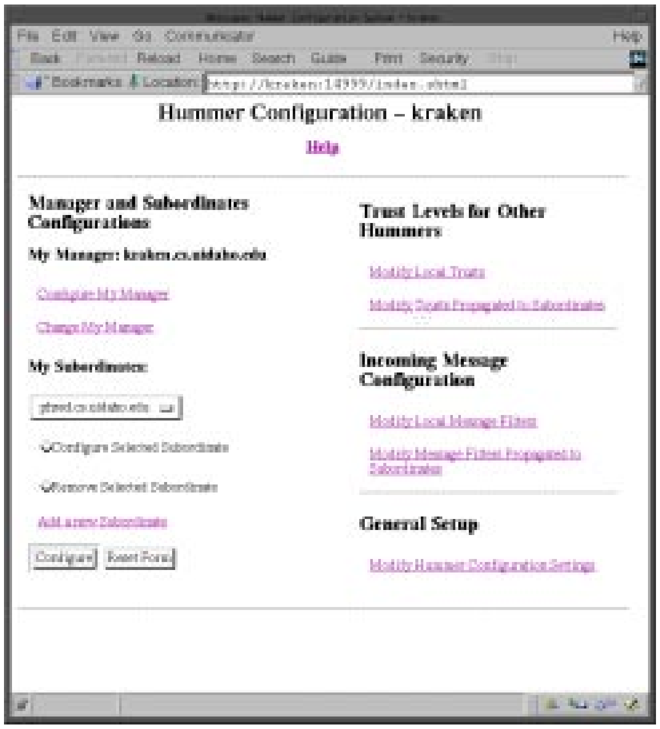


Figure 2: Hummingbird component interaction. Not shown: audit management and feature selection tools.


File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: http://kraken:14999/index.shtml

Hummer Configuration – kraken

Help

Manager and Subordinates Configurations

My Manager: kraken.cs.uidaho.edu

Configure My Manager

Change My Manager

My Subordinates:

ptred.cs.uidaho.edu

Configure Selected Subordinate

Remove Selected Subordinate

Add a new Subordinate

Configure Reset Form

Trust Levels for Other Hummers

Modify Local Trusts

Modify Trusts Propagated to Subordinates

Incoming Message Configuration

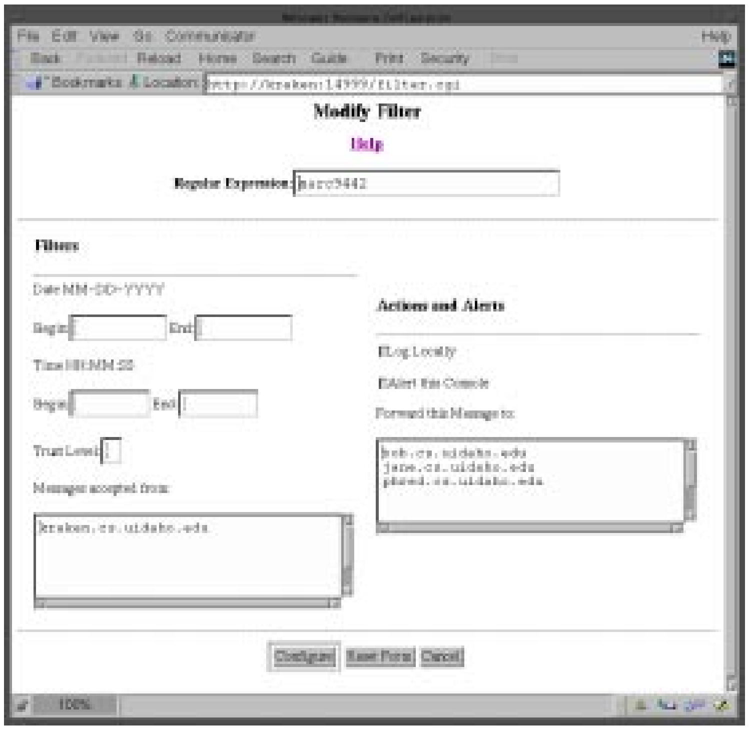
Modify Local Message Filters

Modify Message Filters Propagated to Subordinates

General Setup

Modify Hummer Configuration Settings

Figure 3: Main Configuration Window for Individual Hummer: set peers, managers, subordinates; assign behavior for handling alerts and alarm situations; handle data filtering; etc


File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location:

http://kraken:14999/filter.cgi

Modify Filter

Help

Regular Expression: marc9442

Filters

Date MM-DD-YYYY

Begin: End:

Time HH:MM:SS

Begin: End:

Trust Level: 1

Messages accepted from:

kraken.cs.uidaho.edu

Actions and Alerts

Log Locally

Alert this Console

Forward this Message to:

bob.cs.uidaho.edu
jane.cs.uidaho.edu
phoed.cs.uidaho.edu

Configure Reset Form Cancel
100%

Figure 4: Filter Configuration

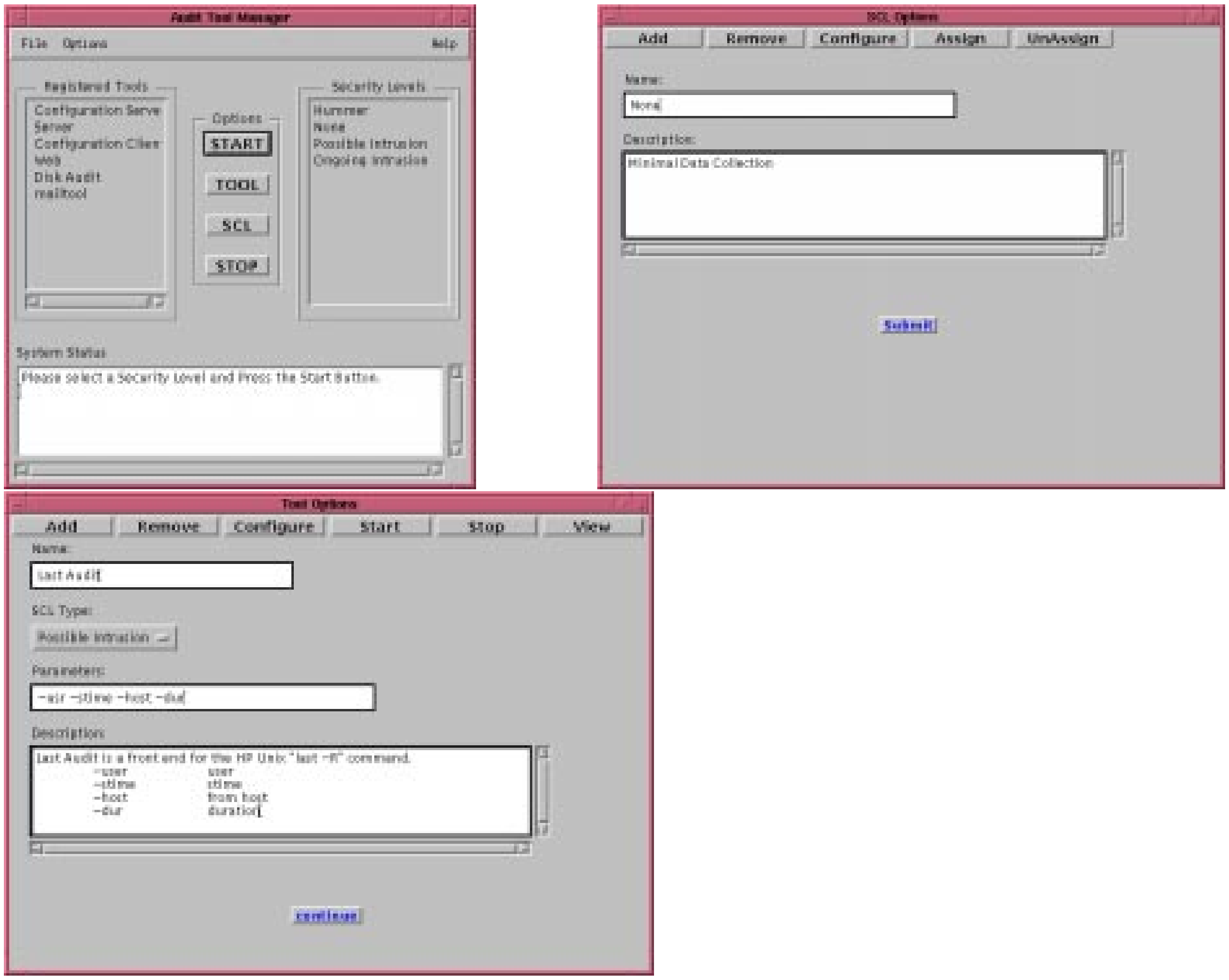
for indications that an intrusion may be in process. Reports from the data collected by a Hummer may be obtained either directly from the POSTGRES database or through the visualization tool (Section 3.2).

3.1 Audit Tool Manager

The audit tool manager, Atom, is a front end for managing Hummingbird audit tools based on the perceived level of threat to the network (Figure 5). It implements a graphical user interface that allows the user to manage audit tools on a network or a single host. Atom is integrated with the Hummingbird system to improve the management of audit tools and increase specialized data collection.

Each audit tool must be registered with Atom. Registration information includes the name of the audit tool, a security classification level, parameters of the tool, textual description, tool classification type, directory where the tool is located, and the default execution of the tool. The default execution is the command used to start the audit tool on the host.

The primary purposes of Atom is to increase and ease the task of performing specialized data collection. This is accomplished by assigning a security classification level (SCL) to a suite of audit tools. The framework for a SCL is a descriptive classification level name determined by the user and assigned to one or more audit tools. Typical SCLs for a network may be *none, possible intrusion,* and *ongoing intrusion.* Using this as a scenario, the classi-



Please select a Security Level and Press the Start Button.

None

Minimal Data Collection

last Audit

-usr -stime -host -dur

```
last Audit is a front end for the HP Unix "last -R" command.
        -user           user
        -stime          stime
        -host           from host
        -dur            duration
```

Figure 5: Audit Tool Management: main and tool options interface.

fication level *none* is considered to be the default, or normal level of data collection on the network. When potential security risks develop, the system administrator can switch to the *possible intrusion* or *ongoing intrusion* SCLs, increasing the data collection by reconfiguring the operating set of audit tools.

One of Atom's more important capabilities is the ability to start or stop a suite of data collection tools on a host or network with the click of a single button. These suites of audit tools allow Hummingbird to be tailored to collect information important in the detection and/or prevention of specific network-based attacks, or to increase the level of auditing when an attack is thought to be underway. Suites of tools which are often used together may be developed and managed under a single name. This also permits easy "upgrading" of system auditing when the perceived level of system threat increases, and an easy way to decrease system auditing when the perceivedeasy choice if desired.

In addition to tool suites, Atom provides several advantages. First, all the tools are now managed by one central manager, making it easy to identify the security mechanisms available to perform intrusion detection or auditing. Since each tool is registered with the manager and has a tool classification type, the user can easily identify and launch tools with particular characteristics, such as login monitoring tools. The parameters and the default execution command of the audit tool are registered providing efficient information about the tool without searching the system for it. In addition, scripts that are developed and used with Hummingbird can be registered and managed by Atom. Finally and most importantly, it creates a more secure and integrated system with Hummingbird for handling audit data.

3.2 Visualization

Visualization is a useful part of any intrusion detection system, since presenting information graphically to the system security officer can greatly ease the task of identifying intrusions by making use of human ability to more rapidly perform pattern matching with graphical data than with raw text. Figure 2 shows a block diagram of the dataflow and interfaces between components in the Hummingbird System, including visualization. We have developed a simple visualization tool which allows subsets of data obtained from the POSTGRES database to be viewed as a histogram. Details about the audit records used to generate the histogram may be obtained by clicking on individual entries. We are continuing to examine alternative forms of data visualization.

4 Related Systems

Earlier studies indicate that detection of certain attacks required data from multiple hosts on a network, and that networks under centralized control could effectively combine data sources to minimize damage. University of California, Davis' Distributed Intrusion Detection (DIDS) system[SB$^+$91] addressed system attacks across a network. Attacks such as doorknob, chaining, and loopback could be detected when data from hosts within a given network was combined[1]. Without data combination, threshold-based intrusion identification schemes are easily subverted by reducing the volume of attacks directed at a particular host to avoid detection. DIDS combined data from hosts within a network under centralized control, but clever attackers could still subvert DIDS by reducing the volume of attacks for a given network. Data sharing as proposed in this project should prove to be a useful way to detect such attacks.

A smaller-scale prototype system for Hummingbird [FEA96] permitted different trust levels between "neighborhoods of networks", and these networks were not assumed to fall

[1]Doorknob attack is considered to be an attempt to break into a system by testing one or more passwords—i.e., rattling the door—on a system. Chaining refers to an intruder's attempt to hide his/her origin by logging in through a sequence of hosts, and a loopback attack is one in which an intruder combines chaining with a visit to an earlier host in the sequence (often with a different login name).

under any central jurisdiction. However, this prototype did not explicitly distinguish between policy decisions based on information flow, trust, or cooperation levels.

In contrast, the EMERALD system (SRI International), is intended to address intrusions within very large networks [PN97]. EMERALD also considers separate administrative domains. Although these domains are assumed to lie within a single corporate structure, EMERALD includes features for handling different levels of trust between the domains from the standpoint of a centralized system: individual monitors are deployed in a distributed fashion, but still contribute to a high-level event-analysis system. EMERALD appears to scale well to very large domains.

Another network-based intrusion detection system, GrIDS, uses graphing techniques to detect coordinated network attacks [SC+96]. This is done by representing the data collected from hosts and networks as activity graphs. These graphs are then analyzed to detect potential violations. Graph reduction techniques are used to reduce the volume of data represented in the graphs. If these reduction techniques are successful, they should permit scaling of GrIDS to large-scale systems.

5 Ongoing Efforts

At present, Hummingbird implementations run under HP/UX, Solaris, and Linux. An NT port is underway and expected to be complete by August 1998. The underlying model is expected to require some additions. Experiments are being conducted to determine whether the overall policy governing data sharing should be augmented by an *availability policy* to indicate requirements for timely responses to requests for data, etc. An availability policy could be used to refine cooperation policies that fluctuate depending on a host's ability and willingness to provide the information requested at the desired rate. We are presently developing both feature selection tools and machine learning tools which will be integrated with Hummingbird (initial completion expected Fall 1998) to provide a means for assessing the effectiveness and efficiency of our system in identify intrusions.

We have deliberately excluded a "true" intrusion identification component from our discussion of our framework for cooperative detection, preferring instead to concentrate on the issues involved in data sharing. However, the issue of trust in decision making warrants further examination. In the context of data sharing, we identify three facets of trust in a message: authentication (Who sent the message?), reliability (Is the data accurate?), and integrity (Is the received message what was sent?) If newly arrived data is sent from an untrusted source, then it may not be safe to use it in making decisions. Also, if the data purports to be from a trusted source but this cannot be authenticated, then it may be wiser to ignore it. Considerable work remains to be done in the development of intrusion detection systems that handle data with varying degrees of reliability.

References

[Auc94] D. Aucutt. Hummingbird Communication Protocol. Technical report, University of Idaho, 1994.

[Dep97] Wheeling Police Department. Neighborhood watch program. Technical report, `http://nsn.nslsilus.org/wgkhome/WHLPOLIC/neighbor.html`, 1997.

[Eva95] J. Evans. Hummingbird Trust System. Technical report, University of Idaho, 1995.

[FEA96] D. Frincke, J. Evans, and D. Aucutt. Hierarchical management of misuse reports. *Journal Computing Informatics*, 1996.

[ILL97] J. Ingalls, T. Lawrence, and E. Lustig. Hummingbird Team B. Technical report, University of Idaho, 1997.

[Isr94] United States Information Service Israel. Israel-jordan peace treaty (26 oct 94). Technical report, `http://www.usis-israel.org.il/publish/peace/annex3.htm`, 1994.

[NCD97] P. Neisen, C. Coltrin, and J. Dowding. Hummingbird Team A. Technical report, University of Idaho, 1997.

[NT94] B. Neuman and T. Ts'o. Kerberos: An Authentication Service for Computer Networks. *IEEE Communications*, 32(9):33 – 38, Sept 1994. `http://nii.isi.edu/publications/kerberos-neuman-tso.html`.

[PN97] P. Porras and P. Neumann. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. *Proceedings of the 1997 National Computer Security Systems Conference*, pages 719–729, 1997.

[SB$^+$91] S. Snapp, J. Brentano, et al. DIDS (Distributed Intrusion Detection System)– Motivation, Architecture and An Early Prototype. *Proceedings of the 1991 National Computer Security Conference*, 1991.

[SC$^+$96] S. Staniford-Chen, S. Cheung, et al. GrIDS–A Graph-Based Intrusion Detection System for Large Netnln,