Scalable Data Management Alternatives to Support Data Mining Heterogeneous Logs for Computer Network Security

William Yurcik * James Barlow † Yuanyan Zhou ‡ Hrishikesh Raje § Yifan Li ¶
Xiaoxin Yin ‖ Mike Haberman ** Dora Cai †† Duane Searsmith ‡‡

Abstract

Today no commercial technology exists for fusing audit log data from heterogeneous sources into a single framework for assessing suspicious computer network behavior. Identifying data relationships and patterns from heterogeneous sources on an instrumented network provides situational awareness and decreases the impact of accidental or malicious system failures that may have large impacts if they would appear. In this paper we compare different data management alternatives for accessing heterogeneous data sources by a data mining application and provide an overview of implementations under development at NCSA. We conclude that at this time there is not a consensus best alternative but each approach has significant tradeoffs that should become more accurately quantifiable as data mining projects for computer network security are implemented

Keywords: intrusion detection, data management, data warehouse

1 Introduction

Computer network security is a continuously escalating battle. Protectors harden operating systems, networks, and applications while attackers find vulnerabilities in any of these areas. However, there are four fundamental asymmetries that favor attackers in this battle: (1) an Internet connection provides worldwide access for all attackers directly to any machine, (2) while protectors must patch all vulnerabilities, an attacker must only find and exploit one vulnerability (all it takes is one unpatched vulnerability to bring down a system), (3) all it takes is one attacker to find and develop exploits for unpatched vulnerabilities since exploits are shared between attackers, and (4) the element of surprise - new undocumented attacks (ϕ days) are being continuously developed and used at opportune times and situations designed to maximize attack effectiveness.

Once this battle had low stakes, but since 9/11 the stakes have risen to include consideration of cyber-terrorists – terrorists who target underlying computer-based systems controlling critical infrastructures such as stock markets, electric power grid, various transportation systems, etc. While a *major* attack on a computer system supporting a critical infrastructure has not yet occurred[1], the possibility of human casualties and economic disruption makes this an important topic for study.

One counter-terrorism strategy against attacks on networked computer systems is to devise better detection tools. While it is difficult (and likely impossible) to harden any networked system against all attacks, most attacks do not occur instantaneously but rather have a definite sequence of events (from reconnaissance to system changes upon compromise) that can be identified and tracked resulting in either the prevention of an attack in real-time or reacting to an attack sooner after it has occurred. Attackers realize this and try to obfuscate their trail from intrusion detection systems (IDSs) with time (both very fast and very slow), deception, levels of indirection, etc. and in many cases this has been successful. The application of data mining to this problem is a higher-level attempt to discover semantic connections between attacker traces in computer network data that intrusion detection systems miss.

Commonly understood prerequisite infrastructure to facilitate data mining includes mass storage, processing power, data mining software, and a database management system. However, a last additional, often ignored, prerequisite is a scalable data management strategy. There has been little work focusing specif-

*National Center for Supercomputing Applications (NCSA)
†National Center for Supercomputing Applications (NCSA)
‡Dept. of Computer Science, Univ. of Illinois - UC
§Dept. of Computer Science, Univ. of Illinois - UC
¶Dept. of Computer Science, Univ. of Illinois - UC
‖Dept. of Computer Science, Univ. of Illinois - UC
**National Center for Supercomputing Applications (NCSA)
††National Center for Supercomputing Applications (NCSA)
‡‡National Center for Supercomputing Applications (NCSA)

[1] Isolated minor attacks have occurred such as distributed denial-of-service attacks against multiple companies, website defacements, and spam attacks but the authors do not feel these attacks rise to the level of a major attack.

ically on data management for data mining, research assumes data is available a priori to a data mining application and instead focuses on developing more effective data mining algorithms.[2] However, practical experience shows that data management for data mining can be almost intractable especially for a high volume application domain such as computer network security.

Audit logs used to monitor computer network security scale with the size of the instrumented network but are typically in the GB range per day for medium size organizations. The scope of the data management problem becomes clear when considering multiple audit logs remotely scattered across a network. There are several data management alternatives that may offer possible solutions but all have current technical drawbacks that must first be overcome.

In this paper we focus specifically on *scalable* data management for data mining in the domain of computer network security by comparing schemes from the literature and sharing practical implementation experience. The remainder of this paper is organized as follows: Section 2 reviews previous work in data management for IDSs. Section 3 summarizes audit logs for providing computer network security. Section 4 provides an overview of implementations under development at NCSA. We end with a summary and conclusions in Section 5.

2 Literature Review

In terms of data management, we distinguish three different states: (1) data collection, (2) data retrieval, and (3) data processing. For the purposes of this paper, data collection refers to network-based or host-based sensors generating audit logs to record activity, data retrieval traditionally refers to the virtual movement of log data to serve as input to a data mining application but we also use this term to encompass the virtual movement of output from remote data mining applications back to an operator, and data processing refers to the use of data mining algorithms to discover computer network attacks within logs.

IDS data management literature can be grouped into three general approaches: (1) centralized, (2) distributed, and (3) hierarchical. A centralized IDS focuses on processing all data at central depository. A distributed IDS focuses on the simultaneous (parallel) processing of data at multiple remote sources such that processing is loosely coupled (autonomous) with minimum data dependency [23]. A hierarchical IDS aggregates data at each of multiple layers in one direction from the raw source data up to the human operator with maximum data dependency (the input of each layer is directly dependent upon output of its adjacent lower layer).

Centralized approaches has traditionally relied upon centralized database systems but the only requirement is for all the data to be accessible for processing at a single virtual location – data can also be stored in sequential or specialized file formats outside of a database system.

In [1], the data is processed in a centralized fashion. The transaction information is extracted from the web servers, after which it is packed into some format to be sent to a separate host for further analysis. The same method is used in [18], where raw audit data is collected by sensors and shipped to the modeling engine. In [17], correlated logs from separate system components are fed to a database where data mining techniques are applied. Both systems described in [26] and [22] perform centralized operations on the audit logs aggregated from distributed data sources. A centralized approach for network traffic analysis is proposed in [8], where all traffic information is archived to a central host.

There is a hybrid variation to the centralized approach using a single layer of aggregation. Data is preprocessed remotely at one lower layer of sensors before the results (e. g., alerts) are delivered to a central place (e g., a database), where further operations are carried out. If more than one layer is involved then this hybrid variation of the centralized approach becomes what we define later as a hierarchical approach. Quite a few systems take advantage of this hybrid variation we will refer to as the *centralized-aggregation* approach. In [3], local processing is done at distributed sensors in network-centric IP fusion systems and resultant output data is sent to a host for fusion. The fusion system introduced in [10] consists of the IDSs that produce the alerts, and a database as a central repository where the alerts are stored. In [25], alerts generated by heterogeneous sensors are fused at a central location to produce meta alerts. In similar papers, data correlation is completed at a separate host where alerts coming from individual sensors are assembled [19, 9, 21].

According to [4, 24], research interest in distributed and hierarchical approaches has increased because they share workload across different nodes and thus improve the data management scalability of intrusion detection systems. Next we introduce representative approaches for both or these approaches.

Distributed approaches typically use autonomous agents which can be characterized by processing inde-

[2] a similar realization about the often ignored but prerequisite capability of data management has also occurred in the sister area of sensor networks where sensors are small, low-power wireless devices sensing the environment [12]

pendent of any centralized coordination and peer communication without any intermediaries. [13] presents one such representative distributed architecture that contains autonomous agents and no central repository. Each agent collects certain information from sets of hosts and it communicates interests (specifications of data it wants) to other agents. Different interests are sent to different agents or agent groups. If an agent gets data that is required by another agent, it sends data to that agent through a tree-like structure. And each agent generates alerts based on the data it has gathered.

[16] proposes a distributed approach for intrusion detection based on mobile agents. In this approach attack patterns are sent to mobile agents through a language called EQL. The agents detect local attacks, and communicate with each other to detect attacks involving multiple hosts.

[2] introduces several distributed approaches based on a publish-subscribe architecture. Each agent generates alerts from a single host and posts the alerts (or other information) at a central database. Each agent then retrieves information from the central database and makes actions.

The focus of hierarchical approaches is to decrease large volumes of data by using aggregation at multiple layers. This allows an operator to abstract away large volumes of data noise in order to facilitate discovering or investigating suspicious activity in the remaining data. Examples include aggregating the following groups of data noise: normal activity, false positive alarms, or multiple alarms from the same attack [6]. In this approach, the view from each layer is only of its next lower layer – there is no direct global view or direct view of the original source data. Unlike distributed approaches in which agent processing is autonomous, some hierarchical approaches use agents that form a multi-layer or tree-like structures but the processing is tightly coupled with layered data dependency.

AAFID [4] is a typical hierarchical approach. The AAFID architecture contains three layers of entities: agents, transceivers and monitors. Each agent monitors a certain aspect of a host. A transceiver is the external interface of a host that controls agents, processes data from them, and responds to commands issued by its monitor. Monitors are high-level entities that control transceivers from multiple hosts.

There are several other examples of hierarchical approach to data management for IDSs. Both [15] and [20] uses three-layer structures. The lowest layer collects information from sensors and may generates alarms for single hosts. The middle layer does correlation analysis or aggregation to generate high-level information, which is sent to the highest layer. The SHOMAR system uses a different hierarchical approach [24]. SHOMAR contains the following components: ID services, IDS managers, certificate authority and capability manager. An ID service is a sensor. Some ID services and an IDS manager form an ID cluster. The IDS manager collects information from every ID service in the cluster, performs data aggregation, and sends aggregated data to its parent. An IDS manager can be an ID service of another IDS manager such that IDS managers typically form tree-like structures. The other two components, the certificate authority and capability manager, control access to resources in the system. [11] uses a similar hierarchical architecture for data management.

2.1 Tradeoff Discussion Data management decision-making depends on project requirements, available resources (equipment, expertise), and time frame. However, there are general tradeoffs with each approach that we discuss in this section.

A centralized approach using a database to support data mining for heterogeneous logs has these advantages:

- effective data sharing: combining log files with individually different formats into a common uniform format. Users can also access the database either locally or remotely.
- efficient data access: database systems utilize a variety of sophisticated techniques to store and retrieve data efficiently. Such techniques include indexing, clustering, parallel processing, and query optimization.
- high reliability and stability: after years of refinement, database systems have become robust and easy to manage
- improving scalability: database systems have been used in many data intensive applications. A recent research article has reported a 5TB database by Oracle [14]

A distributed approach to support data mining for heterogeneous logs has these advantages:

- processing at a level closest to the raw source data which facilitates specialized algorithms at a desirable levels of resolution
- superior fault tolerance
- dramatically reduced processing (per individual machine) and network communication requirements (no concentrated congestion point)

- scalability to easily monitor new logs by adding corresponding new agents

A hierarchical approach to support data mining for heterogeneous logs has these advantages:

- flexibility in both number of levels and processing at each level
- centralized management of different layers due to data dependency
- reduced requirements for combining different raw data formats, can be combined gradually by layers
- large centralized mass storage requirements shifted to reduced and manageable distributed storage requirements at each layer
- scalability with increased processing at individual layers and/or increasing the number of levels

Table 1 compares the primary resources necessary to implement the different approaches: processing power in cycles for executing complex algorithms on large data sets, network capacity in bandwidth for near-real-time data retrieval, and large mass storage capability for queuing source data before (data collection) and after it is processed. There is one clear tradeoff: the centralized approach requires the most resources and the distributed approach requires the least resources with the hierarchical approaches in between.

Table 2 summarizes selected secondary criteria for comparison assuming each approach is feasible given the necessary resources. There are four clear tradeoffs: the centralized approach has a single-point-of-failure while the distributed approach can tolerant sensor/agent failures with minimal disruption, a uniform and complex data format is necessary for schemes with relatively concentrated processing, query flexibility is highest in centralized schemes and low in distributed/hierarchical schemes, and in terms of privacy centralized approaches represent higher risk as a concentrated target and larger compromise impact while the distributed approach presents multiple lower value targets with each having lower compromise impacts.

3 Heterogeneous Computer Network Audit Logs

A computer network contains a variety of different infrastructure devices each of which may be instrumented to produce audit logs. Though it is possible to perform simplistic signature matching on streaming network traffic in real time, it is impossible to analyze this data in real-time, in a period of minutes the data size becomes unmanageable. As a result most network analysis tools log the data for offline analysis. Although the topic of computer network audit logs is broad, a topic onto its own, we feel a brief introduction to some of the different types of logs is important in order to better understand the data management issues surrounding using these logs for IDSs or data mining.

We have made a point emphasizing data management for *heterogeneous* audit logs. The fact that the audit logs are different is significant because it promotes multiple views for attack discovery, robustness against attack, interoperability, extensibility, and flexibility. However, heterogeneity also eliminates possible data management efficiencies that may be possible from uniform record formats, uniform configurations, and uniform control.

It is important to note that computer network sensors generate streaming data and not batch log files. However, processing streaming data for data mining is an open research question beyond the scope of this paper (although briefly mentioned in Section 4.3). We create batch log files by collecting streaming data over defined time periods.

One of the most common ways of collecting computer network data into logs is the use of the **tcpdump** utility. This utility captures packet headers passing through a network interface set in promiscuous mode and displays binary traffic in a number of human-readable formats.[3]

While TCPdump is a valuable tool, it focuses on the TCP/IP suite of protocols. There are a large variety of other utilities for "sniffing" raw packets of any protocol from monitor points on a network. Referred to as "**sniffers**", the most effective programs are Ethereal and the Sniffer from Network Associates although there are many others.[4] As networks increasingly employ "switch" technology, sniffers that rely on a shared medium network (Ethernet) are being moved from end systems to servers and routers. Sniffer logs are uniquely valuable in discerning low-level attacks such as abnormal traffic attacks (e.g. fragmentation) however their scope is limited by their monitoring position within a network.

NetFlow logs contain records of unidirectional flows between computer ports across an instrumentation point on a network. These records can be exported from routers or software such as ARGUS or NTOP.[5] NetFlows are a rich source of information for traffic

[3] similar tools to tcpdump are Ipgrab and Iplog

[4] the word "sniffer" is a registered trademark of Network Associates

[5] ARGUS http://www.qosient.com/argus/
NTOP http://www.ntop.org

analysis consisting of some or all of the following depending on version and configuration: IP address pairs (source/destination), port pairs (source/destination), protocol (TCP/UDP), packets per second, timestamps (start/end and/or time duration), and byte counts (Note: there are different versions of NetFlow software that allow configuring for varying attribute resolution including sampling).

Syslogs are an industry standard for capturing information about networked devices by encoded messages by level (e.g. warning, error, emergencies) and by facility (e.g. service areas such as printing, Email, network). Syslog also functions as a distributed error manager by forwarding log entries to other machines for processing. In addition to pattern-matching syslog entries for known attack signatures, other examples of suspicious activity requiring further investigation include critical events (system reboots), unsuccessful login attempts, new account creation (especially with special privileges), connections from the same external host to many internals hosts on the same port (port scan), or cessation of logging messages from a host (may indicate the logging process has been deleted or a Trojan logging process installed).

Workstation logs are standard utilities that keep login/logout entries on a workstation's local hard disk (in addition to centrally maintained syslogs). Some application software also maintain access logs. One such example is the use of network-based "license servers". Workstation logs are a standard function provided on many operating systems but they are possible to disable. Each workstation also maintains a log of mail transactions originating from that workstation.

ARP cache at subnet routers and switches contain cached tables of recent conversions from IP addresses to physical hardware addresses for lookup efficiency. The entries are of two types: dynamic entries that are added/removed automatically over time and static entries which remain in the cache until the computer is restarted. Each dynamic ARP cache entry has a potential lifetime of between 2-10 minutes (depending on operating system settings, new entries are time-stamped) and a log of all entries can be created over a specified time period. The ARP cache is useful to determine static IP addresses; to identify unregistered/unknown, misregistered (including malicious spoofing), and misconfigured devices attached to a network; identifying what IP address(es) a particular hardware address is using; to debug if a particular device has connectivity; tracking unsuccessful connection attempts to devices that either are not currently on the network or do not exist; and lastly "arp cache poisoning" attacks against arp itself (the insertion of fabricated data). This log capability is becoming more important with the growth of wireless access points into networks.

Nameserver DNS cache contains mappings between fully-qualified hostnames and corresponding IP addresses (and corresponding name server hostnames and nameserver IP addresses) based on recent requests to other name servers. The amount of time a name server retains cache data is controlled by the time-to-live (TTL) for the data. These logs can be created via periodic snap shots of the cache timed shorter than the TTL to capture data before it expires. Host tables (. *rhosts* and *hosts.equiv*) that map IP addresses to hostnames also provides recent hostname-to-IP address mapping information. DNS cache is most effective in detecting IP/URL spoof attacks and malicious sniffing (by identifying machines performing high volumes of DNS queries with automated scripts).

Dial-up server logs maintain system accounting records on who makes ingoing/outgoing network connections to help identify suspicious activity ingoing or outgoing at this access choke point.

Kerberos logs contain all instances of the use of the kerberos authentication system in a network - Kerberos tickets requested. This information can be used to generate "login" graphs and determine who was logged into a particular workstation at a particular time.

SNMP logs , referred to as management information bases (MIBs), are databases of managed objects that store information about a wide variety of network device attributes. The SNMP (Simple Network Management Protocol) operator application involves monitoring network devices via polls to network device agents for specified MIB information or traps from network device agents notifying the operator of an event.

Routing table logs (e.g., inter-domain BGP, intra-domain OSPF or RIP) provide information about routing-based attacks ranging from: (1) individual misbehaving routers that drop/misroute packets or inject disruptively large routing tables, to (2) the systemic network-wide advertisement of false routing information or the instability caused by the propagation of worms. Global, local, or peer routing tables provide different vantage points for analysis.[6]

Firewall logs are important in a recursive way, to maintain the effectiveness of its internal rule set. A rule set exactly specifies what traffic to permit/block – typically growing in number of rules beyond human comprehension. A firewall is a computer or group of computers that interfaces between an internal network/computer

[6] one example of routing table logging is the Routeviews Project for BGP http://www.routeviews.org

and the Internet to enforce an organizational access control policy by processing packets/connections based on the rule set.[7] Firewalls can be used to monitor both normal activity (types of services requested and used, common external IP addresses accessing internal services, common access time patterns) and suspicious activity (probes to ports with no authorized services, external-to-internal flows with source internal IP addresses, outbound connections from uncharacteristic internal machines, and modification/disabling of the firewall rule set). Border routers with the ability to add static routes can serve a firewall function (NCSA does this)

Intrusion Detection System logs contain alerts indicating specific attacks. Generally, while a firewall has a proactive preventative focus, an IDS has a passive reactive focus. IDSs can be categorized in two ways: by sensor placement (network versus host) and by technique (signature versus anomaly), with all combinations producing logs. Real-time IDSs have been plagued by large log size, high false positive rates, and mimicry but incremental improvements are increasing their effectiveness for post-mortem forensics.

Mail logs maintain a log of completed transactions (as well as a queue of pending mail) including sender and recipient address, subject title, date and time of transmission, and size of file.[8] Common tests include: total length of time spent receiving and sending Email and the number of Emails by an entity (organization/group/individual) for period of time (day/week/month), stratify Email by time (work hours/off-hours) and common addresses, stratify size and type of file attachments, internal and external Email, and identifying dormant accounts.

Web server logs provide feedback on performance. Weblogs provide detailed records of requests to the webserver and statistical information about network traffic. The web log record attributes include: the source IP address from which a request was generated, whether a requests is satisfied, a userid determined by the HTTP authentication, a status code, and the size of the object returned with each satisfied request.

Dynamic Host Configuration Protocol (DHCP) server logs can be used to track unique IP address assignments to devices as they join/leave a network. DHCP servers manage two databases: (1) an Address Pool database for holding IP addresses and other network configurations and (2) a Binding database for mappings between Ethernet addresses and an entry in the Address Pool. Though best known for assigning dynamic IP addresses, DHCP can also assign a dedicated static address for a device that re-joins. On a network that uses DHCP with dynamic addresses, maintaining a log is absolutely necessary to be able to forensically associate dynamically changing IP addresses to specific devices/interfaces.

Scanning logs for defensive purposes are used to perform risk management by tracking vulnerabilities and notifying system administrators about potential exploits and patches that need to be installed. However, scanners are also the reconnaissance tool of choice for attackers to identify target IP addresses, servers, operating systems, and ports. Currently the four port scanning tools that stand out are nmap, nessus, SAINT, and the ISS scanner.

Other useful proprietary logs too specific to describe here include **operating system kernel logs**, **router traps,** and **application software logs**. Figure 1 shows a distribution of security relevant attributes across a selection of logs (although the individual attributes are too small to be readable). Some of the attributes may initially appear redundant since they are contained in multiple logs but this overlap is vital to enable event correlation between logs.

Figure 2 is a logical diagram of a centralized approach to the data management problem specific to the computer network security domain. Data in multiple types of logs may be retrieved by a centralized data mining application over the network. The time dimension is implicitly shown with multiple logs under each log type (NIDS is used as an example), with each log containing data from a defined time period.

4 NCSA SIFT Data Management Efforts

In August 2002, NCSA embarked on a data mining for computer network security project named SIFT (Security Incident Fusion Tools). The goal of the project is to fuse together many of the computer network audit logs we have identified for security event knowledge discovery.

As part of this effort we have already developed two breakthrough frameworks for visualizing individual logs across an entire IP address space (Class B address space = 65,000 computers with each computer having 65,000 different ports). Although we have organizational mass storage resources for data collection and high performance local/grid cluster resources for data processing, we have faced scalability challenges with data retrieval for fusing heterogeneous logs – moving data from data collection to data processing.

To appreciate the data retrieval for data mining challenge we face, here are approximate figures for just some of our log volumes:

[7] This includes two general classes of firewalls: (1) Internet/intranet firewalls (many variants) and (2) host-based firewalls

[8] An organization should have a clear Email security policy addressing privacy issues before data mining this content.

- NetFlow Logs - 200M - 4GB/day
- Syslogs - 250 MB/day
- IDS logs - 1 GB/day

These log sizes come with a caveat that they vary widely in size over time and are unique to our special NCSA environment. The point is a comparison of the log sizes relative to *current* data retrieval capabilities (acknowledging data retrieval capabilities are temporal and will change over time as technology evolves).

After surveying related work on data management for data mining and not finding a consensus solution, we embarked on four complementary efforts (see Figure 3). These four efforts map roughly to the approaches we described in the literature review: (1) a centralized database approach, (2) a centralized-aggregation middleware approach (to access log files at their original location), (3) a centralized-aggregation intelligent database approach (using DataSpace servers), and (4) a distributed agent approach.

4.1 Centralized Data/Centralized Processing

Figure 4 gives an overview of an architecture that employs a centralized database for unifying all the log activities. It includes a model to correlate data from multiple sources as it is collected, and to store only the information relevant to the behavioral models in a database format. A log from a sensor corresponds to a table in the activity database, and an event is stored as a row in the corresponding table. Correlations between logs are expressed using relations. For example, suppose a system call event from operating system is $<$ *Time-Stamp, ProcessID, API-Name* $>$, and a network TCP packet event is $<$ *Time-Stamp, ProcessID, Packet-Size, TCP-Flag, Src, Dst* $>$. They are correlated to each other if they have the same *ProcessID* and similar *Time-Stamp* .

The primary advantage of this organization is that it can easily interface to the activity miner and generate detection models. The database organization also has the advantage of supporting various queries. For example, when an anomaly is detected from mining the system call activities, the IDS system can query the activity database to see if there is any abnormal behavior in the network activities in the corresponding time range or by the corresponding processes. The IDS system can also query any aggregated information such as: the number of times a system call is made, or the average network traffic, or the highest busty network traffic, or the time range most logins happen etc. Similar to other centralized system, this approach provides a simple architecture, which is easy to implement and manage comparing to the other approaches discussed in followed subsections.

However, scalability is one of the primary challenges in this architecture. If the system is running for a long time or on a large-scale network, the database size may increase to Giga or Tera-bytes. To reduce the disk and memory footprint of the activity database, it is necessary to compress old events and archive or remove obsolete events. Moreover, stream-based data mining techniques are necessary to analyze only recent activities to generate model and detect intrusions. Time and space overheads are another problem with this approach. Since everything needs to be logged into this centralized activity database, making it a bottleneck in heavy loads.

To address these challenges, the activity log database may require special DBMS management. Most existing database servers such as IBM DB2, Microsoft SQL Server and Oracle are tailored for on-line transaction processing applications or decision support systems, and thereby may not be suitable for managing activity logs. Activity logs consist of time sequence data. Each activity's contribution to the model or the rule decays with time (last year's activities are less important than this year's activities). Therefore, old activities can be archived to secondary storage to reduce the database size and improve mining performance.

4.2 Distributed Data/Centralized Processing

Leaving raw data in its native format and original location allows data to retain the highest resolution with all its attributes without summarization or other lossy compression to shrink size. This is especially important for computer network security which requires a flexibility to ask hard to anticipate "what if" questions and depend on rare events that may be lost via summarization or compression.

Additional advantages with not moving raw source data from where it was initially created and leaving it in its native format include: (1) eliminates intermediate redundant storage and data consistency problems, (2) data that is not needed for processing does not need to be moved or reformatted, and (3) the responsibilities for data collection and retrieval can be decoupled for organization or technical efficiencies. The disadvantages of keeping raw source data in its original location are: (1) decoupling collection and retrieval means that when data is eventually retrieved for data mining, detailed knowledge of storage logistics is required, (2) each new high-level query will require a new, potentially huge, data retrieval (caching does may not help with such large log sizes) and (3) there is less fault-tolerance due to lack of redundant intermediate storage and thus more

dependence on system backups.

4.2.1 Middleware Mediation When trying to incorporate large amounts of data, from multiple sources, for a processing task like data mining, it can be very difficult to get the data you need when you need it. One approach NCSA is taking uses "middleware" mediation as a means to access source data remaining at its point of origination [5]. We use the term middleware to describe a software layer in between applications and network resources, specifically storage. Conceptually, middleware acts a centralized manager mediating and disassociating data processing from data collection – facilitating uniform access to heterogeneous logs based on attributes rather than filenames or physical locations. This approach benefits data management in the following ways:

abstracting the data

Using a middleware approach to data access allows data collection details to be abstracted from the data processing task (data mining application) such that high-level requests are transparently translated to low-level file access operations. As shown in Figure 5, the application does not have direct access to the data types. To retrieve data, an operator or application makes a high-level query to the middleware program that then determines what data is required and where the data is located. The data mining application or user does not know where the source data is stored, how it is stored, or in what format it is stored.

modularization

The middleware approach can also create a modular method to data access such that audit logs in different forms can be flexibly added and removed. This is possible since there are a small number of ways to store source data on the data collection machines, whether it is flat files, databases, or some other method. This also enables the use of reusable components for accessing new data sources.

To be more specific on our efforts at NCSA, Figure 5 shows a high-level request from either the user or data mining application sent to middleware. The middleware then mediates the request in one of three ways: (1) the middleware locates and then retrieves the exact log file(s) based on filename schema, directory structure, and desired time period (log files are created per log type and time period), (2) the middleware redirects the query to a central database corresponding to the log type selected (each log type may have a different database), or (3) the query is sent to an intelligent database server, in our case a DataSpace server hosting a specific log, which accepts complex queries for file attributes across multiple log files via communications between servers (explained in more detail in the next section).

A similar middleware approach focusing on network performance is described in [7]. This paper presents the Cichlid visualization tool that contains: (1) applications specific code for sensors and (2) an application independent visualization engine. The internal data representation is not attached to a particular application – user requests go through middleware to find and retrieve the corresponding data set.

4.2.2 DataSpace Servers The underlying data management architecture should (A) extract only that data which is currently requested for mining, (B) minimize the time to deliver the data, (c) retrieve data transparent from the physical aspects of storage. As we have documented throughput the paper, it may not be advisable to store all data on a central server given the large volumes. This lead to a search for an efficient way to retrieve portions of a data set irrespective of size.

NCSA has decided to implement DataSpace servers - an infrastructure for creating a data web (see Figure 6). Each DataSpace server keeps metadata about its logs allowing Internet communication between servers. This functionality represents logs as database tables by splitting text into columns and tuples as defined by the metadata. Also of interest are: (1) universal correlation keys as primary indices for active querying, (2) the predictive scoring and update protocol (PSUP) for event-driven real-time scoring and model development, and (3) the DataSpace transfer protocol (DSTP) for data queries over the web (the semantics of the protocol involve the rules for querying files based on their metadata descriptors).

To summarize, DataSpace inherently supports distributed data mining in the following ways:[9]

- DataSpaces supports distributed data by supplying a metadata standard, a federation mechanism, and easy to setup server code (Java, C++) that can be deployed near the data. A client can readily retrieve a table view that consists of columns joined from multiple DataSpace servers.

- DataSpace servers can interface with flat files, databases, and other formats. This ability to serve binary/text data reduces the large overhead of a uniform database record format. Server code is also extensible to new data format handlers.

- DataSpaces supports specialized socket layers (parallel sockets, etc.) to facilitate high throughput data transfers.

[9] DataSpace – A Web for Data http://www.dataspaceweb.net/

- DataSpace also supports publishing logs from a large number of sources with desired data privacy preserved with access control lists.

4.3 Distributed Data/Distributed Processing

Given the amount of data that can be generated by processes such as network packet traces, attempting to manage data using a centralized approach may be computationally infeasible.Even for a small organization, the size of audit logs can become quite large. For large ISPs, where data collected can be hundreds of gigabytes a day, moving source data becomes infeasible. The major issue with centralized data collection is that as data is being collected, the collection point can easily become overwhelmed with respect to the required network bandwidth needed to move the data, as well as processor cycles, and I/O requirements of the host to move the data through the TCP/IP stack and onto the storage device. With enough data sources, it is unlikely that a single host will be able to keep up. Centralized processing may work if the node acting as the sole processing point acts upon aggregated or generalized data of reduced size.

With the distributed data paradigm, keeping data at the collection points requires no additional resources other than those already in place to collect the data. Distributed processing also enables real-time analysis at the lowest level closest to the source that would not be possible if data is moved away from its site of origination or centralized processing of logs.

Distributed processing of data at collection points is another logical choice. A variation of distributed processing is aggregating data and then moving a data summary to another node responsible for answering data queries. The primary disadvantage of a distributed approach is the administrative overhead necessary to coordinate tasks on various hosts.

For a distributed processing model, a framework is needed where one can specify how and when data is to be processed providing a base set of features that all logging processes require. It also allows both ad-hoc and canned/general processing on data as it arrives. The actual processing of data can be defined statically or dynamically – processing depends on predefined paths/actions and also on the contents of the data packets. For example, if a certain signature is matched, the data flow path can be switched and other processing agents can be alerted as well. In this manner, independent agents that manage a particular data source can work together to accomplish overall goals.

At NCSA as we collect NetFlow records we store the data locally in audit logs for forensic purposes and also process each flow record as it arrives and either send it to another location for further processing or generate an Email or web page based on profiles or signature matches. Using the same framework at all data collection and processing points helps to mitigate software maintenance problems.

Used in conjunction with the distributed collection paradigm, data is kept where it was generated (e.g. host-based logs) or first collected (e.g. syslogs). Data can then be processed in real-time at these points and meta events can either be sent to some centralized location that is collecting events from various data generators or it can be sent to another collector to affect how it will process subsequent incoming data.

5 Conclusions

In this paper we outline different data management approaches for data mining audit logs. The core problem is scalability since individual logs, and thus also the union of multiple logs, can be huge in size depending on the scope of a particular network.

Different organizations approach this same data management problem by leveraging the unique resources available to them such as mass storage, processing power, software developers, network bandwidth, and database capacity. Specifically in this paper we present the multiple approaches NCSA is currently developing for data management of its heterogeneous network logs for data mining. As exemplified in our use of multiple approaches, at this time there is no clear consensus on the best approach for large networks. However, future plans include quantitative measurements for the different data management approaches we are implementing.

6 Acknowledgments

The following members of the NCSA SIFT team made significant indirect contributions to this paper (alphabetically): Loretta Auvil, Ratna Bearavolu, Randy Butler, Ruth Aydt, David Clutter, Kiran Lakkaraju, Doru Marcusiu, Bobby Rariden, David Tcheng, and Michael Welge.

References

[1] M. Almgren and U. Lindqvist, *Application-Integrated Data Collection for Security Monitoring* , Recent Advances in Intrusion Detection (RAID), (2001).

[2] T. Bass, *The Federation of Critical Infrastructure Information via Publish and Subscribe Enabled Multisensor Data Fusion* , 5th Itnl. Conference on Information Fusion, (July 2002), pp. 1076-1083.

[3] T. Bass, *Multisensor Data Fusion for Next Generation Distributed Intrusion Detection Systems* , IRIS Nat. Symp. on Sensor & Data Fusion, Johns Hopkins University/Applied Physics Lab., (May 1999).

[4] J. Balasubramaniyan et al., *An Architecture for Intrusion Detection using Autonomous Agents,* Annual Computer Security Applications Conf., (1998).

[5] P. Bernstein, *Middleware: A Model for Distributed Services* , Comm. of the ACM 39, 2(February 1996), pp. 86-97.

[6] E. Bloedorn, et al., *Data Mining for Network Intrusion Detection: How to Get Started* , AFCEA Fed. Database Colloq. and Exposition, (2001).

[7] J. Brown, A. McGregor, and H-W. Braun, *Network Performance Visualization: Insight Through Animation,* Passive/Active Measurement Wksp., (2000).

[8] V. Corey, et al., *Network Forensics Analysis* , IEEE Internet Computing, (Nov/Dec 2002), pp. 60-66.

[9] F. Cuppens and A. Miege, *Alert Correlation in a Cooperative Intrusion Detection Framework* , IEEE Symp. on Security & Privacy, (2002).

[10] O. Dain, and R. Cunningham, *Fusing a Heterogeneous Alert Stream into Scenarios*, within Appl. of Data Mining in Comp. Sec. Kluwer, (2002), pp. 103-122.

[11] H. Debar and A. Wespi, *Aggregation and Correlation of Intrusion-Detection Alerts* , Recent Advances in Intrusion Detection (RAID), (2001).

[12] D. Ganeson, D. Estrin, and J. Heidemann, *DIMENSIONS: Why Do We Need a New Data Handling Architecture for Sensor Networks?* HotNets-I, Princeton University, (2002).

[13] R. Gopalakrishna and E. Spafford, *Framework for Distributed Intrusion Detection using Interest Driven Cooperating Agents*, Recent Advances in Intrusion Detection (RAID), 2001.

[14] A. Joch. *Cracking the Code of Life* , Oracle Magazine, (Jan/Feb 2003), pp. 36-42.

[15] C. Kruegel, T. Toth, and C. Kerer, *Decentralized Event Correlation for Intrusion Detection* , Intl. Conf. on Info. Security and Cryptology, (2001).

[16] C. Kruegel and T. Toth, *Applying Mobile Agent Technology to Intrusion Detection* , ICSE Workshop on Software Engineering & Mobility, (2001).

[17] Z. Li, et al., *The Case of Centralized Logging Database for Intrusion Detection* , submitted to Hot Topics in Operating Systems (HOTOS), (2003).

[18] W. Lee et al., *A Data Mining and CIDF Based Approach for Detecting Novel and Distributed Intrusions* , RAID, (2000).

[19] P. Ning, Y. Cui, *An Intrusion Alert Correlator Based on Prerequisites of Intrusions* , Technical Report, TR-2002-01, North Carolina State University, Department of Computer Science, (Jan. 2002).

[20] P. Petrov, et al., *A Hierarchical Collective Agents Network for Real-Time Sensor Fusion and Decision Support* , AAI/KDD/UAI Joint Wksp. on Real-Time Decision Support & Diagnosis Systems, (2002).

[21] P. Porras, M Fong, and A. Valdes, *A Mission-Impact-Based Approach to INFOSEC Alarm Correlation* , Recent Adv. in Intrusion Detection, (2002).

[22] C. Silvestro, *Intrusion Detection Systems and Log Correlation.* Masters Thesis, Politecnico di Milano, (2002).

[23] E. Spafford, D. Zamboni, *Data Collection Mechanisms for Intrusion Detection Systems* , Purdue University, CERIAS Tech. Report 2000-8, (2000).

[24] J. Undercoffer, F. Perich, and C. Nicholas, *SHOMAR: An Open Architecture for Distributed Intrusion Detection Services* , Tech. Report, University of Maryland Baltimore County, (Sept. 2002).

[25] A. Valdes, K. Skinner, *An Approach to Sensor Correlation* , RAID, (2001).

[26] G. Vigna, B. Cassell, and D. Fayram, *An Intrusion Detection System for Aglets* , International Conference on Mobile Agents (MA), (2002).

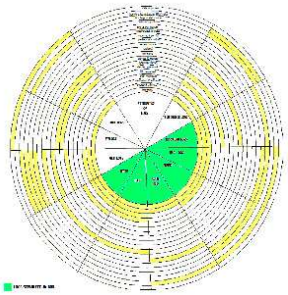

Figure 1: Attribute Distribution Across Audit Logs

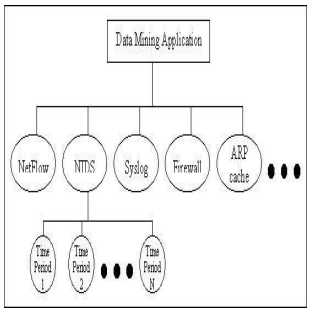

Figure 2: Management of Computer Network Logs

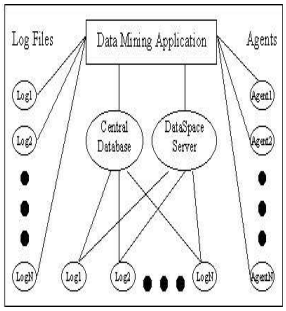

Figure 3: Parallel NCSA SIFT Data Management Efforts.

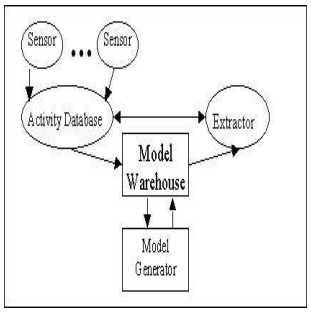

Figure 4: SIFT Central Database Architecture

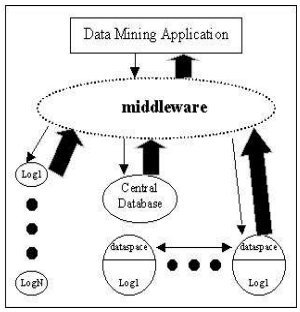

Figure 5: SIFT Middleware Architectures

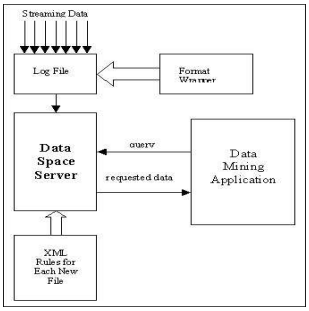

Figure 6: SIFT DataSpace Server Architecture

APPROACHES	processing power	network bandwith	mass storage
Centralized	high	high	high
Centralized - Aggregation	medium	medium	medium
Distributed	low/machine	high	low
Hierarchical	medium	low	medium

Table 1: Primary Comparison Criteria

APPROACHES	fault tolerance	data format	query flexibility	privacy: target/ scope
Centralized	low	complex	high	concentrated /global
Centralized - Aggregation	low-medium	medium-complex	medium	multiple-concentrated /limited-global
Distributed	medium-high	simple	low	multiple /limited
Hierarchical	low-medium	simple	low	multiple /global

Table 2: Secondary Comparison Criteria