System Monitoring and Network Intrusion Detection using DDS and CEP

Gerardo Pardo-Castellote, Ph.D. (RTI)
Joe Schlesselman (RTI)

OMG RTESS Workshop
Washington DC, July 14-16, 2008
Background

This presentation describes the results and architecture of a DoD-funded research effort to develop a common normalized information picture combining data from System Monitoring Tools and Intrusion Detection Systems.

The purpose of this work is to increase the speed and effectiveness of the technologies used to detect & counter network attacks.
Project Partners

- RTI (Prime) DDS
- Coral8 CEP
- SL Visualization
- Promia IDS Hardware
- ObjectSecurity Security Manager
Outline

- The problem of [Distributed] System Monitoring
  - System Monitoring
  - Intrusion Detection

- Available Technologies
  - Ganglia, Nagios
  - Snort, Nessus, Saint

- Practical solution using DDS and CEP
Middleware Technologies, such as DDS help build large distributed systems 100s or 1000s or computers with an order of magnitude more processes

- Functionally the system may be correct
- … but it may still fail operationally…

These large systems require monitoring. We focus on two aspects:

- Network Intrusion Detection
- System Monitoring
- Information Sharing/Normalization
- Information Processing

The integration of these technologies illustrates the essence of the open integration problem we are trying to address.
100’s of Available Tools

Tool Categories:
- Network Intrusion Detection Systems
  - Snort
  - OSSEC NDIS
- Vulnerability Scanners
  - Nessus, Nmap
  - Saint, GFI LanGuard
- Sniffers
  - Wireshark, Kismet,
  - Netcat, Metasplot, hping,
- OS fingerprinting, Service identification
  - Nmap, P0f
- System Monitoring
  - Ganglia
- Knowledge Repositories
  - IDS Rules
  - Vulnerability Databases (CVE)
- Analysis, Detection, Decision Making
  - General purpose (e.g. CEP)
  - Specialized (e.g. Snort rules)
- Monitoring, Visualization/HMI Tools
  - HP OpenView
  - SL RTView, Promia Asset Viewer

See: http://sectools.org/
The tool Integration problem

● Many open source and COTS tools
  – Complementary functionality
  – Overlapping functionality

● Each tool must combine:
  – Sensors, Rules, Alerts/Notifications, Visualization…

● “Each Tool and Isolated Island”
  – Rules/knowledge bound to the tool/sensor
  – Must learn specifics of each sensor
  – Can’t easily multiple sensors in a rule
  – Limited to what each sensor offers

● The Result: Cost, Complexity, Limited Power:
  – Cannot aggregate, correlate, extend
Each tool an isolated island
A better approach!

Shared Information Bus / Normalized Information Picture

- Snort
- nmap
- ossec
- nessus
- Ganglia
- SAINT
- Voltage

Visualization (3D Asset Viewer)

Visualization

Archive

Network Management

new sensors

COTS UTM

new processors

classified

IDS Detection Engine

General Analysis Engine

Custom Code

normalized sensor data inputs

normalized alarm/event outputs

Custom

classified

IDS Detection Engine

General Analysis Engine

Custom Code

new sensors

COTS UTM

custom CEP
A better approach with Standards & COTS

COTS System Sensors (Ganglia)

COTS IDS Sensors (Snort)

Custom Sensors

Host Sensor

Host Sensor

COTS Vulnerability Sensors (Nmap, Nessus, Saint)

Custom/Classified IDS Sensors

COTS Middleware (DDS Global Data Space)

Send updated signature

Vulnerability Signature Database

COTS Recording/Auditing

COTS HMI/ Dashboards

COTS Event Processing

Email
Infrastructure Technology Selection

- **DDS** was selected because it provides a Standard API and Network Protocol able to handle large volumes of real-time information and prioritize it by setting QoS policies.

- **CEP** was selected because it provides a familiar (SQL-like), powerful and extensible language able to process large amounts of streaming data, aggregate the information, correlate it, and uncover interesting events and threats.
Focus:
Network Intrusion & System Monitoring

- Network Intrusion Detection System (IDS)
  - Sensor: Monitor “raw” network packets
  - Process
    - Dissect packets, Identify conversations, Look inside
    - Look for: known attack patterns, scans, unusual activity
  - Output: Generate Alarms, Filter/drop packets
  - Example: Snort, OSSEC

- System Monitor
  - Sensor: Host sensors, look at CPU, memory, file system, network, …
  - Process: Detect unusual loads or changes
  - Output: Status, Alarms
  - Example: Ganglia

- System and Service Identification
  - Sensors: Host sensors, Network packet capture
  - Output: List of hosts, OS fingerprints, List of open services
  - Examples: nmap, P0f
IDS: Snort Overview

- Open source software (Linux, Windows)
  - Commercial support available
- Can operate as:
  - Sniffer / packet logger,
  - Inline Mode: Intrusion Prevention System (IPS)
    - Interfaces with IP tables to drop/reject packets/connections and log the alert
    - Can also modify bytes in the packets
  - Network intrusion detection system (NDIS)
    - Uses lib PCAP to monitor traffic & generate alerts
    - Can also operate from a saved PCAP file
    - Can match src/dst and do deep packet inspection
    - Can produce references to know attack databases (e.g. NIST’s CVE)
- Simple rule language for NDIS
  - Customizable/extensible by end user
  - Active community-based rules database
Snort deployment

Detect all intrusion events

Detect intrusion events that get thought the firewall
Snort Implementation

Outputs alerts to files, database, socket, ...

Applies rules to detect "intrusion" events

Analyzes, modifies, and takes actions like: Alert, Log, Drop, Reject.

Selected Output Mode
(Log files, console, sockets...)

Output Plug-ins

Detection Engine

Preprocessors

Decoder

Packet Capture Module
(Based on LibPcap or WinPcap)

Network Traffic

Detection Plug-ins

References

Rules file

Reads/Applies
Snort rules

The rule matches if all elements match: protocol, src, destination, … content, etc.

- `<direction>` can be “-“ or “<“
- “content:” used for deep-packet inspection. There is a mini language:
  - Can specify offsets and depth,
  - Can relate to other matches in same packet
  - Can use PERL regular expressions
- Can also do matches in the packet (IP/TCP,UDP) headers

If a rule matches the action is taken:

- Alert, log, pass,

Rules are applied in sequence
Snort Actions

- alert – generate alert and log packet
- log  - log packet
- pass – ignore packet
- activate – alert and turn on dynamic rule
- dynamic – inactive rule until activated then log

Inline-mode only:
- drop – drop packet and log
- sdrop – drop packet without logging
- reject – drop packet, log, and send reset/ICMP port unreachable to sender

Custom actions can also be used to bind to specific output plugins:
```plaintext
Ruletype dds_alert {
    type alert
    output dds_write: domain=36 topic=“SnortAlert”
}
```
Normalized IDS Alert information

```cpp
struct NormalizedAlert {
    SensorDetails sensor;
    NodeInfo source;
    NodeInfo target;
    ProtocolInfo protocolInfo;
    Timestamp alarmTimestamp;
    string alertMsg;
    short priority;
    sequence<ExtraValue> extra;
};

struct SensorDetails {
    string name;
    string version;
    long agentId;
    long type;
    NodeInfo node;
};

struct NodeInfo {
    NetworkAddr networkAddr;
    unsigned short port;
    PhysicalAddress phyAddr;
    string hostName;
};

struct ProtocolInfo {
    string type;
    UdpInfo udpInfo;
    TcpInfo tcpInfo;
    IpInfo ipInfo;
    IcmpInfo icmpInfo;
    EthernetInfo ethInfo;
};
```
System Monitoring: Ganglia Overview

- Open source software (Linux, Windows)
  - Originally developed at UC Berkeley
- Distributed Monitoring system for Clusters/Grids
  - Has scaled to clusters with 2000 nodes
- Many built-in metrics
  - CPU, Network, IO, Memory, Disk
  - Can be extended with user-defined metrics
Ganglia Deployment (data collection)

- Host-based (gmond daemon on each host)
  - monitor changes in host state
  - announce relevant changes
  - listen to the state of other ganglia nodes
  - answer requests for an XML description of the cluster state.

- Outputs:
  - Metrics to multicast UDP
    - Sent at configured period
    - ... or when value changes beyond configurable threshold
  - Responses to requests received via TCP

![Diagram of Ganglia Deployment](image)

- Multicast
  - GMOND
  - GMOND
  - GMOND
  - GMOND
  - GMETRIC
  - APP
  - GMOND
  - GMOND
  - GMOND

(standard metrics)  (custom metrics)
Ganglia Implementation & Deployment
Normalized System Monitoring Information

struct StandardMetrics {
    DoubleMetric disk_total;
    LongMetric cpu_speed;
    LongMetric swap_total;
    StringMetric os_name;
    FloatMetric cpu_user;
    FloatMetric cpu_system;
    FloatMetric bytes_out;
    FloatMetric pckts_in;
    ...
};

struct ClusterInfo {
    string name;
    Timestamp localtime;
    string owner;
    string latlong;
    string url;
};

struct NormalizedSysMon {
    SensorDetails sensorDetails;
    ClusterInfo clusterInfo;
    StandardMetrics stdMetrics;
};
CEP Overview

Capabilities:
- Filtering
- Pattern Detection
- Aggregation, transformation
- Time-Correlation
... Across multiple streams

Real-Time Data Feeds

CEP Engine

Event Storage

Composite Events

Alerts

Actions
CEP – Example 1

activity: ftp request

tool: Snort

activity: disk free

tool: Ganglia

Rule:
If free disk < 10% and ftp request
Then send DDS topic alert

RTView

DDS Topic

DDS Topic

DDS Topic

CEP

RTI Recorder
activity: ssh

tool: Snort

event: CPU load

tool: Ganglia

Rule:
If CPU > 50% and ssh request
Then send DDS topic alert
Potential

- Information normalization would allow rules to be written independent of sensors
- Use of CEP would
  - allow rules to apply across sensors
  - enable much more sophisticated rules that mix historical data, time-windows, causality, etc.
- Use of DDS would
  - enable prioritization and other QoS
  - provide unified API to access all data with highest performance
  - allow segmentation and parallelization of processing
  - enable easy visualization using general purpose tools
  - Provide many out of the box services: recording, durability… across all sensors with no additional work
Shapes traffic = UDP/IPv4 DDS traffic created by RTI DDS Shapes demo application
Generic Traffic = legitimate and illegitimate random TCP/IPv4 network traffic
Legitimate = valid, well-formed packets of various protocols created using network traffic stimulator
Illegitimate = invalid, non-well-formed but not malicious packets of various protocols
Rogue = packets that violate one or more security policies, whether or not malicious
Future objectives

- Integrate additional sensors of each class & refine the Normalized data format
- Gather IDS data in real scenarios
- Develop more sophisticated CEP rules based on IDS data
- Develop CONOPS v 1.0