MW-E2ED BoF

EDDY (End-to-End Diagnostic Discovery) concept and effort status
May 2, 2005

Chas DiFatta (chas@cmu.edu)
Mark Poepping (poepping@cmu.edu)
Outline

• Initiative vision and direction
• Concept
• Architecture
• Campus Department/Group Involvement
• Conclusion
• Next steps
Problem

Banès of the Distributed System Diagnostician

- No access to the diagnostic data
- Discovering valuable information in a sea of data
- Correlating different diagnostic data types
- Providing evidence for non-repudiation of a diagnosis
- Finding time to create tools to transfer diagnostic knowledge to less skilled organizations and/or individuals
State of Practice

• Network, application, system and security events separate, therefore extremely difficult to correlate

• Data represents only what has faulted

• No end-to-end accountability of transactions. I.e. email, web, VoIP, intrusion
Vision

Create an activity audit ledger/application that...

• Provides a means to study the behavior of faults and anomalies
• Explores the impact of an Internet with assured electronic communications and its influence on infrastructure, security, reliability, privacy and trust
• Assures the ‘default’ electronic interaction by creating a means of non-repudiation between two or more parties
Initial Direction

Enabling mechanism for investigating,

• Machine to machine interaction
• Taxonomic risk analysis of security anomalies
• Automated diagnostic practices, not just what has faulted but how the fault occurred
• Perceived anomalies verses actual faults
• Embedded system events
• High volume event driven systems
• Rapid tool development platform for diagnostic applications
## Effort Timeline

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Startup</td>
<td>Done</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Discovery</td>
<td>Done</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preliminary Design</td>
<td>Done</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pilot Design</td>
<td>Done</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pilot Implementation</td>
<td>Done</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pilot Verification</td>
<td>Done</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Findings and Redesign</td>
<td>Done</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDDY Implementation</td>
<td>Active</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDDY alpha/beta</td>
<td>Active</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDDY Distribution</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Major Milestones

- Advisory group formed
- CER conceived
- High level architecture finalized
- Pilot delivered
- EDDY backplane operational
- EDDY release
Outline

• Initiative vision and direction
• Concept
• Architecture
• Campus Department/Group Involvement
• Conclusion
• Next steps
EDDY: End-to-end Diagnostic DiscoverY

Goals of the effort,

- Enable the collection of a wide array of network, system, application, security, and environmental events
- Provide a feature rich event dissemination infrastructure that can scale
- Introduce an API that enables diagnostic tool developers to build the next generation or retrofit existing tools
Separate Event Domains

Distributed System Events

Diagnostic Tools
Separate Event Domains

Distributed System Events

Diagnostic Tools
Separate Event Domains

- Distributed System Events
- Environmental Application Security System Network
- Diagnostic Tools
Separate Event Domains

- Distributed System Events
- Diagnostic Tools
- Environmental
- Application
- Security
- System
- Network

Carnegie Mellon
Separate Event Domains

- Network
- Application/System
- Security
Separate Event Domains

Security
- Port Scan
- Denial of Service Attack

Network
- Network Transaction (Sendmail)
- Network Transaction (port 8080)
- Network Transaction (to router)

Application/System
- Sendmail Process Dies
- Sendmail Process Restarted
Combined Event Domains

- Network
- Application/System
- Security
EDDY Event Evolution

Security

Network

Application

System

Environmental
EDDY Event Evolution

Security

Network

Application

System

Environmental

Routing

Filtering

Archiving

Normalization

Transformation

Routing

Application

In-band

Anonymization

Filtering

Application

Out-of-band

Aggregation

Archiving

Database
EDDY Event Evolution

Security

Network

Application

System

Environmental

Normalization

Transformation

Routing

Filtering

Archiving

Application

Anonymization

In-band

Filtering

Application

Out-of-band

Aggregation

Archiving

Database

Tools

NMS

AMS

Alert
EDDY Event Evolution

Security
Network
Application
System
Environmental

Routing
Filtering
Archiving
Normalization
Transformation

Application
In-band
Anonymization
Filtering
Application
Out-of-band
Aggregation
Archiving
Database

Tools
NMS
AMS
Alert

API
API
API
API
API
EDDY Event Evolution

Security

Network

Application

System

Environmental

Routing

Filtering

Archiving

Normalization

Transformation

Visualization

API

Tools

NMS

AMS

Alert

Normalization

Transformation

Application In-band

Anonymization

Filtering

Application Out-of-band

Aggregation

Archiving

Database

Carnegie Mellon
EDDY Event Evolution

Security
- Routing
- Application in-band
- Anonymization
- Filtering

Network
- Application out-of-band
- Aggregation
- Archiving

Application
- Archiving
- Database

System
- Normalization
- Transformation
- Analysis

Environmental

Tools
- NMS
- AMS
- Alert

Carnegie Mellon
Combined Event Domains

- Port Scan
- Network Transaction (Sendmail)
- Sendmail Process Dies
- Network Transaction (port 8080)
- Sendmail Process Restarted
- Network Transaction (to router)
- Denial of Service Attack
- ALERT: sendmail-worm-ID:2353
Enterprise Implementation

- Departmental LANS
- Core Switch Fabric
- Wireless
- Internet
- Internet2 Abilene
- Remote Campuses

Edge Hosts
Backplane Hosts
Enterprise Implementation
EDDY Cluster Functionality

- Security
- Network
- Application
- System
- Environmental

Normalization → Transformation → Analysis

- Archive
- Database
- Application
- Bandwidth Abuse

Edge Nodes → Backplane Nodes
EDDY Cluster Functionality

- Security
- Network
- Application
- System
- Environmental

Normalization → Transformation → Analysis

- Archive
- Database
- Application
- Security Forensics

Edge Nodes → Backplane Nodes
The Scale Issue

• Scaleable store and forward
  – Project only what is needed to the next level
  – Select back to get data that you don’t have
  – Only cook data that you need

• Data lifecycle
The Scale Issue

Events 5k/sec

↓↓↓↓↓↓↓↓
The Scale Issue

Events 5k/sec

7 days Archiver
The Scale Issue

Events 5k/sec

7 days

Application or Database

Archiver
The Scale Issue

Events 5k/sec

Application or Database

Archiver

7 days

30 days
The Scale Issue

Events 5k/sec

7 days

30 days

365 days

Archiver

Application or Database

Application or Database
Diagnostic Data Lifecycle

Policy

- Anonymize/Filter
- Access
- Summarize
- Collection
- Scour
- Archive
Outline

• Initiative vision and direction
• Concept
• Architecture
• Campus Department/Group Involvement
• Conclusion
• Next steps
Solution

• **Import** a wide variety of event data easily

• **Disseminate** the events to elements in a distributed backplane that provides **core functionality** for diagnostics

• **Provide access** to the diagnostic data and a **platform** for rapid tool development
Diagnostic Backplane

• **Accommodates** a wide variety of event classes easily
• Enables most any device to **produce events**
• Supports **extensible classification** models
• **Event routing** via simple select/project functionality
Diagnostic Backplane Cont.

- **Edge hosts:**
  - Servers, clients, and embedded devices
  - Indirectly collecting flow and security data from switches, routers and security devices

- **Backplane hosts:**
  - Forward, manipulate and store event flows from edge hosts
  - Provide an API to query backplane for event information
  - Control and manage the backplane itself
Backplane Transport Channels

Backplane

Control

Query

Event

Transformation
Anonymization
Normalizing
Application
Analysis

Archive
DB
Directory

Display

Agent-manager
Backplane-manager

Control Agents

Storage Agents

Base Agents

console
console
console
Basic Agents

Functionality (XSLT or Java)

Filtering (XPath)  Copy (optional)

Authentication  Authentication

Transport  Transport

events
Basic Agents + Query

Queries → Transport (HTTP) → Authentication → Query (SOAP) → Functionality (Java) → Filtering (XPath) → Copy (optional) → Authentication → Transport → Transport

Events
Storage Agents

queries

Transport (HTTP)
Authentication
Query (SOAP)
Functionality (XSLT or Java)
Filtering (XPath)
Copy (optional)
Authentication
Transport

events
Agent Control

queries

Transport (HTTP)
Authentication
Query (SOAP)
Functionality (XSLT or Java)
Filtering (XPath)
Copy (optional)
Authentication
Transport
Transport

control

events
Base Agent Types

- **Normalization**: rapidly put external events into backplane via a raw CER. Small footprint, can be ported to embedded systems.
- **Transformation**: convert raw CERs into cooked (parsed into XML) and/or manipulate CERs
- **Anonymization**: anonymize specific fields of the CER
- **Application**: take out-of-band action
- **Analysis**: inject analysis CERs into backplane based on observed events
- **Display**: act and a filter/preprocessor for display consoles
Storage Agent Types

- **Archive**: repository of events indexed on the base correlation structure of their CER
- **Database**: repository of events indexed on a specific schema (can be very granular)
- **Directory**: provide a event location service
  - Where do I find this type of event?
  - What is the granularity of it?
Control Agent Types

- **Agent-manager**: operate and manage base and storage agents on each host
- **Backplane-manager**: operate and manage the host-configuration agents to build and operate a specific backplane topology
Display Agent Architecture

Event Transport Channel

Display Agent

Query/Response Channel

Display Console

Display Console

Display Console
Display Agent - Forensic
Display Agent - Forensic
Display Agent - Specialized

| IP Address | #I | #E | #P | #R | #O | #W | #S | #N | #I | #O | #S | #H | #S | #D | #E | #G | #E | #E | #E | #E | #E | #E | #E | #E | #E |
| 10.7.45.202 | 0 | 75 | 3 | 61 | 43 | 0.0 | 2.0 | 13 | 283 | - |
| 10.7.32.23 | 55 | 0 | 61 | 3 | 5 | 2.0 | 0.0 | 202 | 30 | - |
| 10.7.203.40 | 64 | 1 | 49 | 70 | 1 | 1.7 | 0.0 | 161 | 66 | - |
| 10.7.203.22 | 12 | 38 | 21 | 30 | 29 | 0.3 | 1.0 | 70 | 101 | - |
| 10.7.1.53 | 2 | 46 | 27 | 27 | 11 | 0.0 | 1.1 | 69 | 89 | - |
| 10.7.11.301 | 40 | 2 | 27 | 27 | 8 | 1.1 | 0.0 | 89 | 88 | 52435(1), 52437(1) |
| 10.7.11.135 | 32 | 3 | 48 | 43 | 246 | 1.0 | 0.1 | 160 | 143 | 2583(19), 5547(1), 55473(1) |
| 10.7.11.45 | 1 | 36 | 12 | 90 | 598 | 0.0 | 0.7 | 40 | 68 | - |
| 10.7.140.24 | 1 | 75 | 10 | 23 | 34 | 0.0 | 0.3 | 61 | 68 | 3137(4), 3134(4), 7244(3) |
| 10.7.1.39 | 2 | 24 | 27 | 30 | 4 | 0.0 | 0.6 | 88 | 101 | - |
| 10.7.10.45 | 25 | 0 | 17 | 3 | 24 | 0.7 | 0.0 | 55 | 10 | - |
| 10.7.1.156 | 13 | 14 | 14 | 12 | 22 | 0.3 | 0.3 | 43 | 40 | - |
| 10.7.1.73 | 0 | 22 | 7 | 16 | 5 | 0.0 | 0.6 | 22 | 54 | - |
| 10.7.1.63 | 0 | 22 | 7 | 16 | 4 | 0.0 | 0.6 | 22 | 53 | - |
| 10.7.2.69 | 0 | 22 | 7 | 16 | 4 | 0.0 | 0.6 | 22 | 53 | - |
| 10.7.26.342 | 0 | 22 | 7 | 16 | 4 | 0.0 | 0.6 | 22 | 53 | - |
| 10.7.23.2 | 10 | 10 | 17 | 16 | 1669 | 0.3 | 0.3 | 55 | 53 | 9850(715), 12607(20), 13772(10) |
| 10.7.22.200 | 6 | 0 | 16 | 5 | 11 | 22 | 0.0 | 0.4 | 35 | 36 | - |
| 10.7.32.15 | 16 | 0 | 31 | 6 | 32 | 0.4 | 0.0 | 36 | 10 | 4064(3) |
| 10.7.156.105 | 14 | 0 | 10 | 5 | 16 | 0.4 | 0.0 | 53 | 17 | - |
Common Agent Capabilities

• Every agent can forward, combine, split and filter event flows to other agents within the diagnostic backplane
• All transport channels (event, query, control) between agents are encrypted
• Mutual authentication based on certificates
• Initial design designed to scale to at least 5000 events/sec
• Can easily morph onto new agent types
Common Event Record (CER)

- Accommodates a wide variety of event classes easily (network, system, application, security)
- Enables high correlation between events through time, location, type and/or extensible tags
- Can be lightweight to conserve space but can be transformed onto a highly descriptive structure
- Highly flexible structure that morph to accommodate new correlation schemes
Event Progression

1. External Event Record
2. Normalizing Agent
3. CER: Raw
4. Transformation Agent
5. CER: Cooked
6. Parsed Payload
7. Analysis Agent
8. CER: Analyzed
9. Analysis Payload
Common Event Record

Type Raw – no parsing of event payload

**Event Descriptor**

**Base Information**
- **Version** - version of CER
- **typeID** – event type (NetFlow, /var/log/messages, MS security event, etc.)
- **eventID** – identifier unique across the backplane
- **occurredStamp** – time of the event
- **eventHostname** – where the event occurred
- **eventHostAddress** – address where the event occurred
- **eventType** – network, system, security, application or environmental
- **normalizerHostname** – host where the normalization agent was run
- **normalizerAddress** – address of the host where the normalization was run
- **warningLevelType** – emergency, alert, critical, error, warning, notice, informational, debug
- **correlationDescriptor** – highly flexible structure to aid correlation (one for every major event type)
- **userTag** – tag:value pairs defined at the setup of backplane to give unique meaning to events

**Raw Event Data**
Common Event Record

Type Cooked – raw event payload is parsed into XML

- Event Descriptor
- Parsed Event Data

XML Structure:
- Schema of raw event data
- Can be highly granular
- Defined by transformation agent
Common Event Record

Type Analyzed – high order diagnostic event

Event Descriptor | Analyzed Event Data

Diagnosis of observed events
• DiagnosisID – specific name of diagnosis
• Hypothesis – what it thought happened
• EventPointers – pointers to all the events that contributed to the hypothesis
Common Event Record Examples

- **Raw**
  - Network: Cisco NetFlow version 9 in payload
  - Security: Snort or MS security event
  - Application: /var/log/smtpd or MS application event
  - System: /var/log/dmesg or MS system event
  - Environmental: temperature

- **Cooked**
  - XML representation of raw events
  - Specific fields of the XML representation of raw events

- **Analyzed**
  - Diagnosis of DoS attack based on raw and or cooked events
Rapid Enabling of Diagnostic Applications

- Enable the forensic process
- Feeding NMS to enhance their functionality
- New visualizations to represent real-time and historical events
- Feeding research with an enormous set of data
EDDY Enabled Devices

- Workstation and servers
- Network devices (routers and switches)
- Security devices (firewalls and IDS)
- Embedded EDDY
  - Environmental devices (premises control/monitoring)
  - Transportation (automotive, etc.)
  - Robotics
Outline

• Initiative vision and direction
• Concept
• Architecture
• Campus Department/Group Involvement
• Conclusion
• Next steps
What EDDY is

- Consolidates events into a simple framework to enable correlation
- Event dissemination environment
- Diagnostic tool platform that leverages and enhances existing tools while enabling the next generation
What EDDY is not

• A system/network/application/security management platform
• The analysis engine, it enables the analysis to happen with domain expertise
Unleashing the Genie

Exposing an unprecedented wealth of diagnostic information for

- Enabling new and enhancing existing diagnostic and security applications
- Visualizing events
- Security forensics
- Researchers through the establishment of a diagnostic observatory
- Modeling new policy configurations to assess their impact on daily operations
- Analysis, validation and troubleshooting of distributed composite applications
Next Generation

- Network, application, system and security events combined
- Data represents discrete events that make up successful or failed service delivery
- True end-to-end accountability of transactions
- Auditing the behavior of an electronic transaction to establish an event profile
Seeding the Environment

• **EDDY** as an enabling technology provides,
  – Event dissemination and correlation infrastructure
    • Gives researchers access to event data (anonymized) on the security, application and network domains
  – A development platform for diagnostic research in the areas of
    – Applications and Middleware
    – Networking
    – Security
Outline

• Initiative vision and direction
• Concept
• Architecture
• Campus Department/Group Involvement
• Conclusion
• Next steps
Enabling Campus Members

• Funding for extended research
  – A platform to discover new diagnostic application methods
  – Exposing a “petri-dish” for researchers to gain access to security, system, application, environmental and network events

• Enterprise diagnostics
  – Within CMU Computer Services
  – Other federated applications
Dragnet (use case for scale)

- Real-time security analysis using network flow records across campus core
Dragnet (use case for scale)

- Real-time security analysis using network flow records across campus core
Dragnet (use case for scale)

- Real-time security analysis using network flow records across campus core
Intelligent Workplace – School of Architecture (use case for CER)

• Capturing events from all aspects of a physical environment

- Motion Sensors
- Heating Sensors
- Lighting Sensors

Collection Engine

Workspace Analysis
Intelligent Workplace – School of Architecture (use case for CER)

- Capturing events from all aspects of a physical environment
Year Two Goals

Mature the Common Event Record

- Solicit input on completeness of version 1.0
- Must be able to morph to new CER formats and providing backward compatibility
- Address scaling issues with respect to the record size and consider other data representation formats
- Include second order events such as measurement and performance
- Incorporate a mechanism for more granular correlation of events
Year Two Goals Cont.

Scale the diagnostic backplane

- Adopt a real Authz/Authn methodology
  - We use certificates at this time, but management is an issue
  - Shibboleth non-web version ready

- Provide an event anonymization
  - Specific agent devoted to policy based functionality

- Transport method evolution
  - Removed the dependency of SCP
  - Add real-time flow capability

- Migration from Python or offload compute intensive areas
  - Now Java

- Management and Configuration
  - Centralized configuration
  - Keep the configuration work on the clients hands free
Year Two Goals Cont.

Add Applications...

Domain specific

• Work with middleware application, network, system, security groups to build focused apps based on what we’ve learned from scenario writing process

• Discuss performance/measurement with external groups

Mature and establish a base application with GUI interface for forensics and reporting

• Reporting – feed appellations like cricket and crystal reports

• Forensics – need a client GUI interface that is ported to Linux, Mac and Windows
Year Two Goals Cont.

Add more applications...

Build simple but high value tools that extract information from the archive and not the DB

- Summaries of events
- Top event hosts
- For retrieving data that is not sent to the DB

Version 1 of the Event API

- Acquiring a real-time event flow from any node
- Simple data locator service (where can I find this data)
- Querying data repositories directly but be conscious of future capabilities where agents may mine data over multiple repositories
Status

• Development
  – Core developers driving to core release 5/05

• Campus Adopters – initial use cases
  – CS/Cylab – security research, real time flow events from commodity Internet
    • Dragnet – network flow event security analysis
  – Architecture – environmental monitoring and control
    • Environmental event data from many ultra small devices and embedded systems
  – Computing Services ISAM/Security Office
    • Consolidation of application log files, fault analysis
    • Conduit for reporting and high level event consumption
Status Cont.

- **Outreach**
  - Involving others in the development process
  - Expand to other use cases external to CMU

- **Funding**
  - Sponsored by the National Science Foundation under the NSF Middleware Initiative - Grant No. ANI-0330626
  - Expanding the effort by increasing funding to
    - Mature base technology
    - Spawn effort for diagnostic application development
    - Enable multi-subsystem correlation
    - Experiment with extending research data flow analysis into multi-campus; federating/automating some diagnostic data sharing
  - Soliciting development partners in both industry and government
Enabling other Efforts and Tools

Diagnostic assistance is provided through the system in several ways:

- Existing diagnostic tools have been or can be fitted with EDDY normalizers and translators to join into the backplane and make their data available to other applications or to specific help desk/service personnel.
- Applications can be fitted with similar EDDY normalizers to inject their error logs and diagnostic information into the Backplane.
- Existing diagnostic tools can be enriched through access to additional diagnostic data through tapping into other sources of information within the backplane.
- New diagnostic consoles can be developed and assembled from components that access and analyze the rich resources on the backplane.
- Applications can utilize diagnostic data at lower levels of the protocol stack and present better information to users about problems in access or performance.
- The diagnostic capabilities can be positioned to provide audit mechanisms as well.

This material is based upon work supported by the National Science Foundation under Grant No. 0330626, Carnegie Mellon University, and Internet2. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.
Discussion

Special thanks to the following to make the effort possible...

- Jim Gargani (CMU) – lead developer/design - core
- Kevin Miller (Duke) – design - core
- Tom Neuendorffer (CMU) – design/developer – visualization
- Walter Wong (CMU) – developer/design - core