Open Source and Commercial Applications in a Java-based SELinux Cross Domain Solution

2 March 2006
Doc Rev 1.0.2

Boyd Fletcher
boyd@spawar.navy.mil
USJFCOM J9/SPAWAR

Spencer Shimko
sshimko@tresys.com
Tresys Technology

Daniel LaPrade
daniel.laprade@je.jfcom.mil
USJFCOM J9/EG&G

Brian Raymond
brian.raymond@je.jfcom.mil
USJFCOM J9/Dataline

Scott Thomas
scott@tridsys.com
Trident Systems

Wayne Franklin
wayne.franklin@tridsys.com
Trident Systems

DISTRIBUTION STATEMENT A.
Approved for public release; distribution is unlimited.
Disclaimer

• We are not endorsing products, we are endorsing open standards.
  – The products chosen for CDCIE were those products that best suited our needs at the time the project started and they may be replaced in the future.

• Any material quoted with attribution from this brief must be approved by JFCOM before being used.
Project Overview

Develop a standards based, non-proprietary, secure, scalable collaborative information environment (CIE) to enable cost-effective multinational information sharing (MNIS) in both single and cross domain environments.

- **CDCIE 2.2 - Cross Domain Portal and Portal Applications**
  - Provide a portal and suite of commonly used portal applications that are classification labeling aware

- **CDCIE 2.2 - Cross Domain Document Management System**
  - Provide an easy to use system for securely sharing documents with versioning and subscription support
  - Provide a method to automate much of the Reviewer/Releaser process

- **CDCIE 2.1 & 2.3 - Cross Domain Collaborative Tool**
  - Provide a secure and scalable collaboration tool for DOD that solves the tactical chat, cross domain, full function (minus video) collaboration requirements

- **CDCIE 2.2 - Security Enhanced Office Automation Suite**
  - Provide a method to safely redact documents for release to lower classification levels & external entities.
  - Improve the Reviewer/Release process

- **Cross Domain Guards**
Why we are building CDCIE

• Near-Term
  – Provide an integrated solution to identified MNIS problems:
    • Support DJC2 Baseline Requirements Document
    • Support COCOM cross domain information sharing requirements
    • GRIFFIN & CENTRIXS information sharing requirements
    • OIF Information Sharing Lessons Learned
  – Solve the Tactical Chat problem
  – Increase the efficiency of the Reviewer/Releaser process
  – Force policy and mindset changes in DoD

• Future
  – Work toward GIG/NCES vision
  – Promote next generation standards and develop new ones where they are lacking.

• Maximize Benefits of Open Source & Open Standards
  – Stimulate industry globally
  – Enable Coalition partners the ability to roll their own interoperable solution
  – Reduce the cost of collaboration in DoD
What is a Cross Domain Solution

• A device that acts as a trusted boundary between two or more networks of different security contexts (classifications)

• Typically has some additional goals such as:
  – Only permits data of a certain type to transit the boundary
  – Prevents inadvertent disclosure of information through the use of filters
    • Filters can include clean/dirty word scanning, schema validation, fixed message format validation, data skewing/transliteration, classification label checking, etc...
  – Prevents unauthorized users from transmitting data that crosses the domain boundary
How does SE Linux help CDS Developers?

• Type Enforcement is a better CDS foundation
  – Provides mandatory access control
  – Reduces trust placed in guard applications
    • Security and accreditation burden shifted to architecture
    • Architecture defined and enforced in policy
  – Implemented in a powerful mainstream operating system
    • Including enterprise level support (e.g., RHEL 4.1)

• Type Enforcement simplifies CDS development
  – Ensures that data flows between applications only in the prescribed manner
  – Guard applications can be narrowly focused
    • Inspection/Filtering, logging, etc.
  – Simplified architecture by reducing security components to small discrete modules
    • Reduces development time/cost
    • Eases certification and accreditation
CDCIE Chat High Level Architecture

High

- Buddyspace Client
- InfoWorkSpace Client
- XMPP Collaboration Gateway 1.1

DSG

XMPP

Low

- Buddyspace Client
- InfoWorkSpace Client
- XMPP Collaboration Gateway 1.1
- XMPP
Collaboration Gateway 1.1
Process Interconnections

Collaboration Gateway
RHEL AS 4.1 w/ SE Linux Enabled

- database
- chat server
- app server
- admin tool
- Logging Subsystem

Guard Process1
- antivirus

Guard Process2
- antivirus

LDAP Server

Chat Client

Data Sync Guard

Copyright 2005 Trident Systems
Java

- Java is used for most of the filters, application logging subsystem, guard interface, and applications developed for CDCIE
- JVM security manager complements SELinux access control
- Java was surprisingly well-behaved from the perspective of SE Linux policies
  - No extraneous access required
  - Memory protection permissions may be needed (execmem, execmod, etc)
- Well understood security model
- Low-level Linux IPC mechanisms can be used through Java To UniX (JTuX) library
  - Helpful in leveraging fine-grained access controls offered by SELinux on IPC
- Java's strong typing, resistance to buffer overflow attacks, stack smashing, automatic memory handling, and lack of pointers yields safer and more secure code
- Single Java executable
  - Separate domains required creative use of file labeling and entrypoints
- Disadvantages
  - Possibility of improperly handled exceptions
Closed Source Software

- Software used:
  - Jabber Inc’s XCP Server, others
    - Supports XMPP (Jabber) protocol
    - Allowed developers to focus on other tasks rather than reimplementing support for Jabber

- Close Source Software & SELinux
  - Required most “loosely” written policy
    - SELinux policy still based on least privilege
  - Some extraneous access can be denied without impacting functionality
    - Reduced impact of programming flaws
      - Without access to source code
Open Source Software

• Software used:
  – JBoss, Log4j w/ Simple Socket Server, ClamAV, PostgreSQL, Linux

• This combination provided
  – Secure and flexible operating system (RHEL)
  – Flexible application level Logging (Log4j w/ Simple Socket Server)
    • A separate Simple Socket Server used for each application that is logging data.
  – Ability to do low latency high performance virus scanning using ClamAV with its Socket based interface
  – Leveraging this huge code base let the developers focus on other tasks (e.g. developing the trusted applications)
  – Existing policies could be modified and used (e.g., clamav and postgresql)
Open Source Software and SELinux

- Ability to fix flaws exposed through SELinux policy development
  - Changes contributed back to community
- Existing policies could be utilized
- Policy capable of describing access to complex filesystem layout used by JBoss
  - Deployment of applications requires write access to certain portions of the directory tree
  - This write access was confined at the “lowest” level in the tree
SE Linux Lessons Learned

- Modularity is key if using Type Enforcement for securing a filter pipeline
- Commercial applications are not always well behaved.
  - Required most “loosely” written policy
  - SELinux policy ultimately still grants this application access based on least privilege
  - Some extraneous access can be denied without impacting functionality
  - Reduced impact of programming flaws without access to the source code
  - Example: Jabber XCP – a XMPP Chat server.
- In a mixed environment of commercial and open source software, Java, and SELinux
  - SELinux can be used to enforce least privilege in individual applications
  - SELinux can be useful in exposing flaws in applications
  - SELinux can be used to deny extraneous access due to flaws in closed source applications
    - Prevent exploits from being leveraged or propagating to other parts of the system
  - The Java-based filters can focus on correctly processing the data
  - The collection of applications facilitates cross-domain chat while reducing development time and complexity (and costs!)
General Lessons Learned

• Avoid using the same instance of an internal server as this opens the possibility of creating untrusted paths through the device
• Leverage role separation for controlling access to the different functions in the guard.
• Leverage all aspects of Linux security when building cross domains solutions
  – SE Linux
  – Bind internal only servers (like AntiVirus) to only use Loopback addresses.
  – Use NSA, DISA & CIS security lockdowns.
    • Run the DISA Security Readiness Review (SRR) scripts
  – Script Everything
  – Use custom kickstart (ks.cfg) and customized installers to minimize amount of user level configuration.
    • We used Perl with Newt to create the custom installers
  – Use udev to control user level access to removable media (used for archiving logs and uploading antivirus definitions)
  – Use IP Tables to control inbound and outbound connections
• Insulate your users from command line – write GUI admin tools
  – We used JWM to provide users a familiar Windows like look-n-feel
  – We used Java Swing and Perl/Tk based applications
Future Ideas for the Community

• Trusted PostgreSQL – an SE Linux enhanced version of Postgres Database. This database will implement Role-Based Access Control (RBAC), Mandatory Access Control (MAC), and Label Security.
  – They are very few trusted databases on the market. They are expensive and/or do not support Java applications.
  – In order to achieve many of the goals of NCES and the GIG we need the ability to bind the database security (RBAC, MAC, LS) to the o/s capabilities.

• Security Enhanced Java #1 – Create a modified version of the Sun JDK 5.0 JVM’s Security Manager so that its security services leverage the SE Linux policy framework.
  – This will allow the JVM to use SE Linux policies to enforce which systems calls are allowed. Currently we can use SE Linux to secure a JVM but this security lockdown is for the entire JVM process not to just the apps within the JVM. This is not a very elegant solution.

• Security Enhanced Java #2 – Implementation of the Mandatory Access Control in the JVM by leveraging “JSR-121: Application Isolation API Specification” and binding of the islets to the operating system’s MAC layer.

• Security Enhanced Java #3 – Implementation of object level labeling, label security, the development of an API to manipulate operating system security labels.

• Security Enhanced Java #4 – Binding of the JVM class loader to the Trusted Platform Module (TPM). This capability will provide a potentially much higher level assurance in the execution of java applications on both guard and regular systems.
Questions?
Backup Slides
Cross Domain Guard

- Cross domain XML guard is the BAE Systems (formerly DigitalNet) Data Sync Guard (DSG)
  - Supports TCP/IP Socket connections for fast low-latency data movement
  - Data movement within guard is via shared memory. Data regrading does not involve file system access.
  - Supports W3.org XML Schema Validation
  - Schema updates can be done directly on guard and do not require interaction with vendor.
    - NOTE: Schema updates on productions devices would not normally be allowed by policy
  - Supports IC Metadata Standard for Information Security Markings
  - Supports Unicode (UTF-8) compliant Clean and Dirty Word Search
  - Supports normalization (identity transformation) of XML messages
  - Lower cost compared to existing GOTS guards
    - Less than $100K per instance, installed with training

- Hardware:
  - XTS-400 is based on a 2.8 Ghz Intel Xeon based server
  - 3U Rack Mounted and Tower configurations available
  - Supports up to 8 standard connections at different system-high single-level networks

- Operating System:
  - EAL 5+ certified STOP/OS 6.1E
  - Has a Red Hat Linux 8 compatible API for developing applications

- Software:
  - DSG 2.0 software
  - Java based API for application development on guard interface
  - Apache Xerces 2.6 XML Parser (C/C++ Version)
    - Supports W3.org XML Schema Validation
    - Adding RelaxNG and Schematron support in CY06
Fielding and Schedule

• Version 1.0
  – Original CIE based on proprietary software
  – Proprietary. Temporary solution
  – Based on work from MC02 experiment

• Version 1.2
  – Fielded Portal and Document Management to Multinational Forces Iraq (MNF-I) in Feb/Mar 05
  – Standards based solution based on eXo and Xythos

• Version 2.1
  – Installing Portal and Document Management at JFCOM as enterprise solution in Aug/Sep 05
  – Cross Domain Chat started CT&E at Ft. Huachuca in Sep 05

• Versions 2.2 & 2.3
  – Portal, Cross Domain Collaboration, and Document Management to be delivered for CT&E in FY06
<table>
<thead>
<tr>
<th>Feature</th>
<th>CG 1.0</th>
<th>CG 1.1</th>
<th>CG 2.0</th>
<th>CG 2.1</th>
<th>CG 2.0</th>
<th>CG 2.1</th>
<th>CG 2.0</th>
<th>CG 2.1</th>
<th>CG 2.0</th>
<th>CG 2.1</th>
<th>CG 2.0</th>
<th>CG 2.1</th>
<th>CG 2.0</th>
<th>CG 2.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Whiteboarding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Audio</td>
<td>CG 2.0</td>
<td>SDG 1.0</td>
<td>CG 2.1</td>
<td>SDG 1.1</td>
<td>CG 2.1</td>
<td>SDG 1.1</td>
<td>CG 2.1</td>
<td>SDG 1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application Casting</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Document Transfer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PFMG 1.0</td>
<td>PFMG 1.1</td>
<td>PFMG 1.0</td>
<td>PFMG 1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Portlet Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Sync Guard</td>
<td>DSG 2.0</td>
<td>DSG 2.1</td>
<td>DSG 3.0</td>
<td>DSG 4.0</td>
<td>DSG 3.1</td>
<td>DSG 4.1</td>
<td>DSG 3.1</td>
<td>DSG 4.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>