Secure Middleware for Robust and Efficient Interoperability over Disadvantaged Grids

Dr. Ramesh Bharadwaj
Center for High Assurance Computer Systems
Naval Research Laboratory
Washington DC 20375  USA
Tel: +1-202-767-7210
Fax: +1-202-404-7942
Email: ramesh@itd.nrl.navy.mil
<table>
<thead>
<tr>
<th>Report Documentation Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. REPORT DATE</td>
</tr>
<tr>
<td>01 DEC 2007</td>
</tr>
<tr>
<td>4. TITLE AND SUBTITLE</td>
</tr>
<tr>
<td>Secure Middleware for Robust and Efficient Interoperability over Disadvantaged Grids</td>
</tr>
<tr>
<td>6. AUTHOR(S)</td>
</tr>
<tr>
<td>Center for High Assurance Computer Systems Naval Research Laboratory Washington DC 20375 USA</td>
</tr>
<tr>
<td>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</td>
</tr>
<tr>
<td>5a. CONTRACT NUMBER</td>
</tr>
<tr>
<td>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</td>
</tr>
<tr>
<td>5b. GRANT NUMBER</td>
</tr>
<tr>
<td>5c. PROGRAM ELEMENT NUMBER</td>
</tr>
<tr>
<td>5d. PROJECT NUMBER</td>
</tr>
<tr>
<td>5e. TASK NUMBER</td>
</tr>
<tr>
<td>5f. WORK UNIT NUMBER</td>
</tr>
<tr>
<td>12. DISTRIBUTION/AVAILABILITY STATEMENT</td>
</tr>
<tr>
<td>Approved for public release, distribution unlimited.</td>
</tr>
<tr>
<td>13. SUPPLEMENTARY NOTES</td>
</tr>
<tr>
<td>14. ABSTRACT</td>
</tr>
<tr>
<td>15. SUBJECT TERMS</td>
</tr>
<tr>
<td>16. SECURITY CLASSIFICATION OF:</td>
</tr>
<tr>
<td>a. REPORT</td>
</tr>
<tr>
<td>unclassified</td>
</tr>
<tr>
<td>19a. NAME OF RESPONSIBLE PERSON</td>
</tr>
</tbody>
</table>

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18
Roadmap

1. Background and Motivation
2. Our Solution
3. Design Philosophy
4. Case Studies
5. Technical Approach
6. Major Accomplishments
7. Transition Plans
1. Background and Motivation

2. Our Solution

3. Design Philosophy

4. Case Studies

5. Technical Approach

6. Major Accomplishments

7. Transition Plans
Network-Centric Warfare Demands a Secure and Survivable Information Grid

Requirements for the Navy’s Command and Information Infrastructure are flexibility, modular system design, fast and easy configuration, and information assurance.

-- Committee on Network-Centric Naval Forces
The Navy’s Open Architecture: Requirements for Interoperability

“[The Open Architecture will …] substantially reduce shipboard computer maintenance by capitalizing on the fact that application components are not bound to computer locality but instead are free to migrate to available processors under Resource Management (RM) control.”

Open Architecture Computing Environment (NSWC Dahlgren)

Infrastructure must provide:
- Pool-of-computers architecture
- Applications not bound to computer locality but migrate to available processors
- Functionally distinct self-contained applications or components
- Components loosely coupled in space and time
- Applications built for portability and location transparent allocation and operation
How can we achieve this?

Software agents are computer programs with one or more of the following attributes:

- autonomy ("autonomous agents")
- mobility ("mobile agents")
- learned behavior ("learning agents")
- multiplicity ("multi-agent systems")
- distributed implementation
- cooperation and coordination
- "emergent" behavior
A Case for Distributed Agents: UAV Swarms
A Case for Distributed Middleware:
Intelligent Agent Security Module

- Real-time Intrusion Pattern Detection
- Proactive Attack Identification
- Cyberlab – Effectiveness Metrics

- Identify Attack Sources
- Forensic Analysis and Data Mining
- Correlation, Fusion, and Visualization
Threats to Interoperability

“A Network Enabled Battlespace is dangerous if content is not secured and guaranteed. [...] a major challenge is to ensure that data and communications, at rest and on the fly, are secure each time, every time.”

-- Battlespace Information 2003

Interoperability goals:
• reduce total ownership costs
• quick and easy system upgrade and reconfiguration
• lower impact of COTS upgrades
• reduce compatibility problems

THREATS
• COTS flaws
• Insiders
• Nation States
• Hackers
• User mistakes
• Trojan horses
“Information Operations That Protect and Defend Information and Information Systems by Ensuring Their Availability, Integrity, Authentication, Confidentiality, and Non-repudiation. This Includes Providing for Restoration of Information Systems by Incorporating Protection, Detection, and Reaction Capabilities.”

Joint Doctrine for Information Operations
Joint Pub 3-13, Oct 9, 1998
IA Is An Enabler

- **We Count** on Information Superiority to Improve Combat Effectiveness
  - *Full Spectrum Dominance*
  - *Network Centric Warfare*

- **IA Enables** Information Superiority in a Network-Centric Paradigm
  - *Global Secure, Interoperable Network*
  - *State-of-the Art Protection for Information Infrastructure*
1. Background and Motivation
2. Our Solution
3. Design Philosophy
4. Case Studies
5. Technical Approach
6. Major Accomplishments
7. Transition Plans
Solution: Secure and Reconfigurable Middleware

Distributed middleware researchers\(^1\) identify the following challenges:

- Programming Abstractions
- Naming and Resource Discovery
- Adaptive Data Fusion
- Adaptive Distributed Plumbing
- Failure Semantics
- Runtime Mechanisms
- System Evaluation

... but miss the most important\(^2\) ones:

- Trustworthiness
- Security
- Robustness
- System Survivability


Secure Infrastructure for Networked Systems (SINS)

- Uses software agents technology
- Addresses security, performance, and robustness (survivability addressed in a related NRL 6.2 project)
- Builds security into agent middleware

What can we prove about agents in the SINS architecture?

- Completeness and Consistency of Agent Behavior
- Mechanical proofs of safety properties and agent compliance with local security policies
- Determination of emergent behavior of a community of agents
Security Agents Enforce a Consistent Security Policy

Never issue a CFF if forceCode == <friendly>

Security Agents act as mini-firewalls between an application and the OS resources.

- intrusion detection
- application monitoring
- survivability
- infrastructure monitoring
1. Background and Motivation
2. Our Solution
3. Design Philosophy
4. Case Studies
5. Technical Approach
6. Major Accomplishments
7. Transition Plans
Design Tradeoffs

Security and Survivability

Functionality

Security Agents enabled application

Usability and Efficiency

Distribution

**Security and Survivability must be considered in the context of applications.**
Based on a Dual-Layer Approach

Services

Spatially distributed objects

References:


Secure Infrastructure for Networked Systems (SINS)

Domain Engineering: Identification and Design of SOL Components


Secure Agent Development Process

- Secure Agent Requirements
- Standard Decomposition
- Agent Design
- Agent Implementation
- Agent Deployment
1. Background and Motivation
2. Our Solution
3. Design Philosophy
4. Case Studies
5. Technical Approach
6. Major Accomplishments
7. Transition Plans
Case Studies

Next-Generation agent-based Command and Control Systems:

- Integrated Marine Multi-Agent Command and Control System (IMMCCS): Agent-based C2 system
- Real-time Execution Decision Support (REDS): Decision Support System which uses agents for information access and dissemination

Current agent-based systems cannot guarantee:
- **Integrity**: System safety and information assurance are not considered
- **Performance**: The distributed object model is inefficient
- **Robustness**: Agents are brittle, hard to create, deploy, and debug
Case Study: IMMACCS

System Integrity

if Radar.forceCode == \textless\textit{not friendly}\textgreater\ && Radar.status == \textit{ACTIVE}
then
  CallForFire.target = \text{name (Radar)}
  CallForFire.controlMethod = \text{WHEN READY}
endif

Safety Property
Never issue a Call For Fire if forceCode == \textless\textit{friendly}\textgreater

Integrity factors
- information leaks
- user mistakes
- malicious attacks
Case Study: IMMACCS

Performance factors
- replication of data
- bandwidth of links
- reliability of links

Distributed Objects
Evaluating agent behavior
Completeness and consistency of emergent agent behavior

Domain A

Information Flow

Domain B

Evaluating agent behavior
Completeness and consistency of emergent agent behavior

Agent1

Agent2

Agent3

if Munitions.ECR < TargetSize
then ratings = ratings - 10

if Munitions.CEP > Munitions.ECR
then ratings = ratings - 5

Robustness factors
• compositionality
• code safety
• modularity
• dynamic reconfigurability
1. Background and Motivation
2. Our Solution
3. Design Philosophy
4. Case Studies
5. Technical Approach
6. Major Accomplishments
7. Transition Plans
# Three-Pronged Approach

## System Integrity
- Authentication and authorization
- Confidentiality and integrity of transmitted information
- Security Protocols for fast/easy configuration
- Safety and Security Policy Enforcement

## Performance
- Dynamically determined agent routing patterns
- Flexible event handling and propagation
- Highly-efficient transmission of relevant information

## Robustness
- Secure Operations Language (SOL)
- Agent Creation Framework
- Assurance of agent behavior
Proposed SINS Architecture

Decrypt Agent
Decrypt Agent
Decrypt Agent
Decrypt Agent
... 

Public Key Infrastructure/Trust Management

MOPED (Model checker for SPKI/SDSI)

Security Policy Definition Language

Local Security Policy
1. Disk Access Allowed
2. Not to exceed 5 MB

Schneider Automata
Checking Consistency of Emergent Agent Behavior

if Munitions.ECR < TargetSize
then ratings = ratings - 10

if Munitions.CEP < Munitions.ECR
then ratings = ratings - 5

if Munitions.CEP < Munitions.ECR
then ratings = ratings - 10

Salsa: NRL Patented Theorem
Proving Technology

module intel_agent
functions
target_size = 20;
type definitions
ratings : integer range [-20,100];
monitored variables
CEP, ECR : integer;
controlled variables
rating: ratings;
definitions
var rating initially 100 :=
if
[] ECR < target_size -> rating -10
[] CEP < ECR -> rating -5
[] CEP < ECR -> rating -10
fi
end module // intel_agent

Inconsistency!!
Salsa: An Automatic Invariant Checker

Agent description

Parser
Term Rewriter
Verification Condition Generator
Formula Reducer

Disjointness
Coverage
Application
Properties

Salsa contains 30,000+ lines of source code (previous ONR 6.2 work)

UNSATISFIABILITY CHECKER integrates two important decision procedures: a BDD algorithm and an integer linear constraint solver.

The UNSATISFIABILITY CHECKER

boolean
enumerated types
integers

description is valid
description invalid + counterexample
1. Background and Motivation
2. Our Solution
3. Design Philosophy
4. Case Studies
5. Technical Approach
6. Major Accomplishments
7. Transition Plans
Additional Publications


FY 2003 Milestones

1. SOL (Secure Operations Language)
   - Design and implementation of SOL compiler for distributed agent implementation over SSL (Secure Sockets Layer) network connections [Bha03b, KIB03].
   - Development of techniques to ensure that SOL agents are composable, consistent, safe, secure, and verifiable. References [Bha02] and [Bha03a] provide details.

2. Agent monitoring and coordination
   - Design of Inter-Agent Protocol (designated the Agent Control Protocol, or ACP) and a secondary protocol (Module Transfer Protocol, or MCP) for inter-agent communication and distributed agent deployment [TB03].

3. Determining emergent properties of multi-agent systems
   - Implementation of translators SOL2SAL and SAL2SOL as interim solution for using formal verification tool Salsa (implemented in previously funded ONR 6.2 project).
## Overall Project Milestones

<table>
<thead>
<tr>
<th>ublisher</th>
<th>FY03</th>
<th>FY04</th>
<th>FY05</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secure Operations Language (SOL)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Making SOL composable, consistent, safe, secure, verifiable</td>
<td>♦</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Formal proofs of application properties</td>
<td>0</td>
<td>0</td>
<td>♦</td>
</tr>
<tr>
<td>Secure Infrastructure for Networked Systems (SINS)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Prototype Implementation</td>
<td>♦</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Requirements Elicitation and Design</td>
<td>0</td>
<td>♦</td>
<td></td>
</tr>
<tr>
<td>- Demonstration System</td>
<td>0</td>
<td>♦</td>
<td></td>
</tr>
<tr>
<td>Agent monitoring and coordination</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Monitoring architecture over physically distributed domains</td>
<td>♦</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Selecting security protocols to enforce/maintain consistency</td>
<td>0</td>
<td>♦</td>
<td></td>
</tr>
<tr>
<td>- Establishing the consistency of agent behavior and data</td>
<td></td>
<td>♦</td>
<td></td>
</tr>
<tr>
<td>- Establishing that agents enforce a consistent security policy</td>
<td></td>
<td>♦</td>
<td></td>
</tr>
<tr>
<td>- Obtaining a situational awareness picture for agents</td>
<td>♦</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security Agents:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Establishing trust in security agents</td>
<td>♦</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Development of application-specific security agents:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Intrusion detection</td>
<td>♦</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Survivability and adaptability</td>
<td>♦</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Key:**
- ♦ Milestone
- ○ Ongoing Activity
Operational Payoff:
**Secure and Efficient C² for Combat Systems**

Operational Payoff:
Secure and Efficient C² for Combat Systems

**Operational Payoff:**
Secure and Efficient C² for Combat Systems.
Multi-Security Levels: One Role for Security Agents

Security Agents ensure secure dissemination of information across domains

Domain A
- Security agents make decisions
- Sanitize information
- Optional process (e.g., remove source, fuzz image)
- Enforce organization or application-specific release policy

Domain B
- Security agents make decisions
- Receive policy server
- Enforce flow direction
- Enforce authentication, integrity, labeling, ..., policy

Flow controller
1. Background and Motivation
2. Our Solution
3. Design Philosophy
4. Case Studies
5. Technical Approach
6. Major Accomplishments
7. Transition Plans
Transition Opportunities

- Navy’s Open Architecture Computing Environment
  - Aegis-equipped cruisers and destroyers
  - SSDS-equipped carriers and large deck amphibs
  - Submarines
  - DD(X) land attack destroyer
  - Littoral Combat Ship (LCS)
- UAV Swarms
- Distributed Sensor Networks
Open Architecture Characteristics

Designers have identified the following requirements:

- Portability
- Location transparency
- Loosely coupled components
  - Time and space
- Preservation of data integrity across threads, processes, computers, networks

NRL Secure Agents Middleware will provide these characteristics.
Open Architecture Challenges Addressed by SINS

We have identified the following additional challenges:

• Security
  – Malicious users
  – Malicious code
  – Confidentiality

• Impact of COTS upgrades on applications
  – Immature standards
  – 30 year lifetime
  – Vendor-specific changes

• Difficulty of finding (COTS) middleware talent
• Complexity of (COTS) middleware

How to design applications with the desired characteristics?
Agents for UAV Swarms
Sensor networks collect and transfer information critical to provide a complete, accurate and trusted situational awareness picture.

If this information cannot be trusted, it cannot be utilized.

Sensor networks are thus critical components.

Their security is critical!
Sensor Network Characteristics

Sensor Attributes
- Power Constrained
- Limited Memory
- Limited Processor Capability
- Expendable

Communication Capabilities
- Wireless Interface
- Limited Bandwidth
- Limited Range

Networking
- Ad Hoc
- Self-Organizing
- Randomly Failing Nodes
- Dynamic Routing

Security Threats
- Denial of Service (e.g., Jamming)
- Compromise (Sensor, Network)
- Injection of False Data
- Spoofing

Mote (tiny, wireless) Sensor
**Project Description & Technical Approach**

Design and advanced prototype development of secure distributed middleware for efficient, reconfigurable, and scalable system interoperability, using the novel concept of “security agents,” i.e., mini-firewalls, to ensure system integrity, efficiency and robustness. Target applications are information network situational awareness, networked C^2 for combat applications, the Open Architecture, and Unmanned Aerial Vehicle (UAV) swarms.

**Project Objectives**

Ensure secure, efficient, and robust distributed system interoperability. Additionally, reduce total ownership costs, allow quick and easy system upgrade and reconfiguration, lower the impact of COTS upgrades, and reduce compatibility problems.

**Project Payoff/Impact on Naval Needs**

- Networked systems that are provably secure and intrusion tolerant
- Networked systems that are flexible, reconfigurable, and survivable
- New ways of tackling complexity, the Achilles heel of system vulnerabilities
- Introduces a novel notion of security agents – software that polices malevolent foreign code

**Project Start/Milestones/Funding**

<table>
<thead>
<tr>
<th>Task</th>
<th>FY 03</th>
<th>FY04</th>
<th>FY05</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1: Secure Operations Language</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task 2: Secure Infrastructure for Networked Systems</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task 3: Application-Specific Security Agent Development</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task 4: Monitoring, Coordination, and Experimentation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$600K $600K $600K
END