Security Risks in Systems of Distributed Objects, Components, and Services

David Chizmadia
Promia, Inc
dchizmadia @ promia.com
(410) 694-0322

Topics

- Elements of a Distributed Objects, Components, and Services Systems
- Security Threats Within a Distributed Objects, Components, and Services System
Elements of Distributed Object, Component, and Service (DOCS) Systems

Security Boundaries in DOCS Systems
Topics

• Elements of a Distributed Objects, Components, and Services Systems

• Security Threats Within a Distributed Objects, Components, and Services System

Security Boundary Terms

1 ⇒ User Interface
2 ⇒ Container Interface
3 ⇒ Client Invocation Interface
4 ⇒ Transport Interface
5 ⇒ Network
6 ⇒ Server Invocation Interface
7 ⇒ Inter-Component Interface
8 ⇒ Delegation Interface
9 ⇒ Persistence Interface
10 ⇒ Storage Interface
User Interface

- Primary entry point for human users
- Usually consists of:
  - commodity hardware (Intel, Mac, Palm)
  - commodity OS (Windows, MacOS, Linux, PalmOS)
  - a web browser
- Environment is generally uncontrolled and probably has minimal physical security

User Interface Threats

- User Impersonation
- User exceeding assigned authorization
Container Interface

• Containers primarily hold software components
  – term is used here to also refer to browser ability to run applets
• Container provides the services needed by components to interact with external world
  – events/notification, transactions, persistence, and security
• Container manages component lifecycle and execution state

Container Interface Threats

• Undesired Use of an Object Implementation
• Unauthorized access to a subset of response data
• Request/Response Repudiation
• Disclosure of “Eyes-only” data
Client Invocation Interface

- Invocations transferred by communications middleware (CORBA, Web Services, RMI)
- Target of invocation (server) is identified by a reference to hide details of server location and implementation from client
- Communications middleware handles resolution of server location and translation of data when client and server use any combination of different hardware, OS, and programming language

Client Invocation Interface Threats

- Spoofing of the Server Reference
- Unprotected, Security-Unaware Containers or Components
- Unwanted Revelation of Client Host Existence
- Inappropriate protection of request integrity and confidentiality
- Client Authentication Domain Different From Server Authentication Domain
- Client Authorization Domain Different From Server Authorization Domain
Transport Interface

• Communication middleware still relies on standard data transport abstractions to transfer data - it just hides details from components
• Most kinds of communications middleware are evolving to allow invocations to occur over a variety of data transports
  – e.g., TCP/IP, HTTP, WAP, etc

Transport Interface Threats

• Compromised communications middleware masquerading as a different client or server
• Disclosure of Request Contents
• Modification/Destruction of Request Contents
• Flooding attacks against specific ports
Network

• The network effects the actual transfer of data between computers
• There are numerous network media
• Most data transfers will traverse one or more administrative boundaries that are protected by firewalls or virtual private networks

Network Threats

• Network Eavesdropping
  – Disclosure of request contents
• Message Tampering
  – Modification or destruction of request contents
• Inability to cross network boundaries (e.g., firewalls)
• Flooding attacks against specific nodes or subnets
Server Invocation Interface

• Accepts client invocation requests
• Resolves target of request to a component within a container
• Converts request into a language-specific call up to the component
• Converts response from component into an invocation reply that is returned to the client

Server Invocation Interface Threats

• Spoofing of the Client Identity
• Unwanted Revelation of Server Host Existence
• Inappropriate Protection of Response Integrity and Confidentiality
• Client Authentication Domain Different From Server Authentication Domain
• Client Authorization Domain Different From Server Authorization Domain
• Flooding Attacks Against Specific Components, Containers, or Servers
Inter-Component Interface

- Goal of Container design pattern is to allow creation of components with common, well-defined functions that can be combined to meet specific application requirements
- Components are designed to present and use well-defined interfaces to and from other components and the Container
- In most current systems a Container runs as a single process under a single identity

Inter-Component Interface Threats

- Components may attempt to modify each other
- Security architecture may warrant components with independent identity and authorization
- Malicious component could masquerade (e.g., present same inter-component interfaces) as another component
Delegation Interface

- In DOC systems, the component a client invokes may invoke other components in other containers, possibly on other server hosts, to process the request.
- Some security policies may require a client to allow the other components to be invoked with some, or all, of the originating client’s authorizations.
  - i.e., the client must “delegate” authorization to the component.
- In emerging environment, the Server Middleware may be a different technology than the Client Middleware.
  - e.g., Server may accept Web Services invocations, while Client may make CORBA invocations.

Delegation Interface Threats

- The delegation model is incompatible with the security policy of the originating client.
- The delegation model is incompatible with the security policy of the downstream component.
- The originating client, intermediate component, and target component are in different Authentication Domains.
- The originating client, intermediate component, and target component are in different Authorization Domain.
- The security information received by the Server Middleware is incomplete or incompatible with the security information required to invoke the next component with the Client Middleware.
Persistence Interface

- Interface between the Container and an external storage system - usually a database
- Makes it appear that a component instance has been saved
- Ultimately only saves the data from the component instance

Persistence Interface Threats

- Saved component instance state could be accessed by unauthorized users
- Saved component instance state could be modified by unauthorized users
Storage Interface

• Actually saves the data that storage middleware extracts from a component instance
• Is tightly bound to the storage mechanism (filesystem, database)
• Must interact with any security mechanisms found in the storage mechanism

Storage Interface Threats

• Unauthorized access to component state by authorized storage system administrators
• Unauthorized access to component state by unauthorized users of the storage system
• Unexpected interaction between storage middleware and storage system security policy or mechanism
• Storage System, Persistence System, and Container/Component are in Different Authentication Domains
• Storage System, Persistence System, and Container/Component are in Different Authorization Domains
Other Threats to Consider

- Bypass of Security Controls
- Lack or loss of accountability
- Misconfiguration of the system
- Vulnerabilities with no countermeasures

Conclusion

- Multiple levels of threat have to be addressed
- No current technology counters all of the threats
- These threats are starting point for
  - Analyzing risk in a specific system
  - Comparing security models of different DOCS technologies

Questions?