Secure xADL

A Connector Centric-Approach to Architectural Access Control

Motivation

Highly networked and componentized modern software poses significant challenges to assuring its security. Traditional security research mostly focuses on formally assuring monolithic software and low-level supporting mechanisms. A more comprehensive perspective is needed to deal with the challenges of modern software. Traditional software architecture research provides such a perspective on decentralized and networked software, but lacks support for specific properties like securities.

We need a solution that can help us design, analyze, and execute software that is made up of components coming from and running at different locations with security as a major concern. Such a solution enables developers and administrators to design and maintain secure software, and provides them with the power to adopt security-aware design, identify potential problems, and effectively respond to attacks.

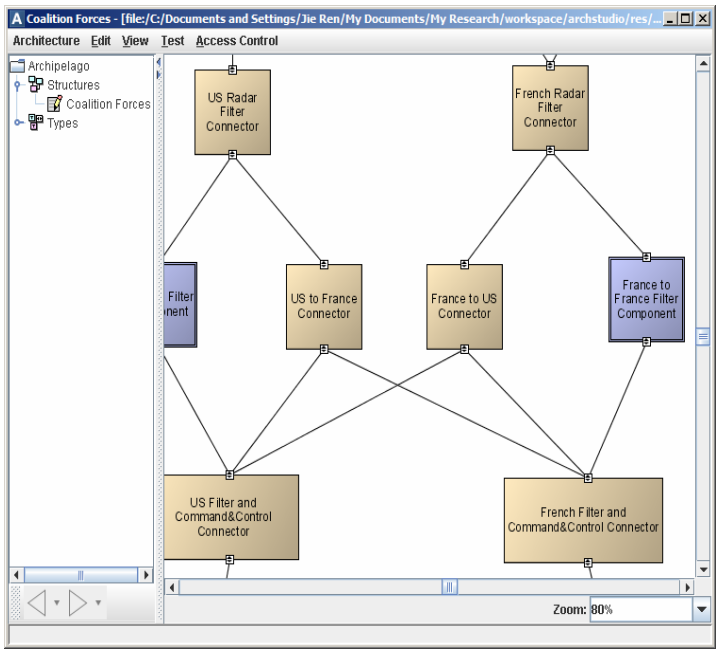

Figure 1: Coalition: two secure connectors connect partially trustworthy parties to exchange information

Insights

Traditional security research does not provide a high-level perspective on the software architecture of a system. Traditional software architecture research does not address security issues sufficiently. We believe that **a secure software architecture methodology advances state of art for secure software design and analysis.**

Connectors could be the center of this integration, connecting heterogeneous components to form a secure composite. We maintain that a **first class connector facilitates expressing, constructing, integrating, enforcing, reusing and evolving security properties among heterogeneous components.**

Approach

Our approach is to extend a widely-used base architecture description language, xADL, with constructs necessary for describing access control at the architecture level. Connectors are used to form secure composites. Our approach supports good security practices at the architecture level, and provides tools to support design, analysis, and execution of secure software.

Based on a unified access control model that integrates the classic, role-based, and trust management models, these security modeling constructs are identified: **subject, principal, resource, privilege, safeguard,** and **policy**. A new language, **Secure xADL**, is designed to incorporate these constructs. This is the first effort to model these security concepts directly in an architecture description language.

A **subject** is the user on whose behalf software executes. A subject can take multiple principals. Essentially, **principals** encapsulate the credentials a subject possess to acquire permissions.

A **resource** is an entity whose access should be protected. Traditionally such resources are *passive*, and they are accessed by active software components operating for different subjects. In a software architecture model, resources can also be *active*, such as components and connectors.

Permissions describes a possible operation on an object. **Privilege** describe what permissions a component is possessing depending on the executing subjects. We model two types of privileges, corresponding to the two types of resources. The first type handles passive resources, such as which subject has read/write access to which files. The second type handles active resources. These privileges include architecturally important privileges, such as instantiation and destruction of architectural constituents, connection of components with connectors, execution, and reading and writing of architecturally critical information. A corresponding notion is **safeguard**, which are permissions that are required to access the interfaces of the protected components and connectors.

A **policy** ties all above mentioned concepts together. It specifies what privileges a subject should have to access resources protected by safeguards. It is the foundation for making access control decisions. We adopt the eXtensible Access Control Markup Language (**XACML**) as the basis for our architectural security policy modeling.

When components and connectors are making security decisions, the decisions might be based on entities other than the decision maker itself. More specifically, the ***context*** of the decision making, such as the neighboring entities, the type, the containing sub-architecture, and the global architecture, can play an important role in such decision makings.

Secure Connectors play a key role in our approach. They decide what subjects the connected components are executing for, inspect whether components have sufficient privileges to communicate through the connectors, and have potentials to provide secure interaction between insecure components. Connectors can be composite connectors, where a composite connector combines several connectors together into a large connector to achieve a composite policy. During execution, connectors can decide whether instantiation, connection and routing should be allowed.

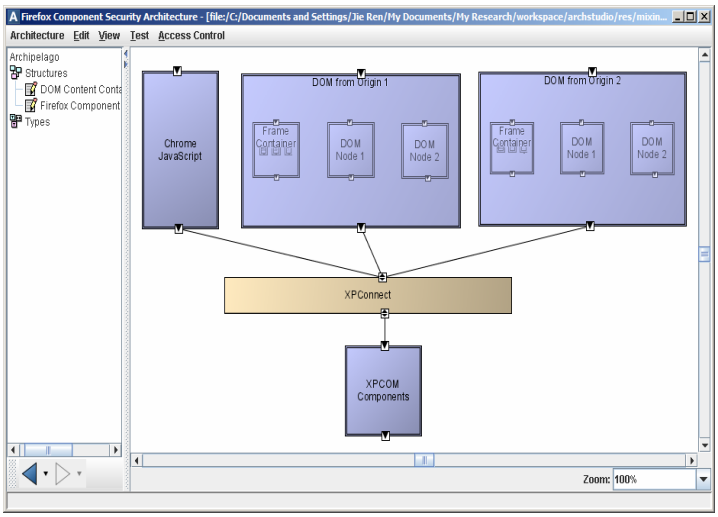

Figure 2, Firefox component security architecture described in Secure xADL. XPConnect is the central connector for JavaScript security.

```
<connectorType>
  <security>
    <principal>NATO</principal>
    <policies>
      <PolicySet PolicySetId =
        "InstantiateConnectorType"
        PolicyCombiningAlgId =
        "deny-overrides">
        <Policy RuleCombiningAlgId
          = "deny-overrides">
          <Rule Effect="Deny">
            <SubjectMatch
              MatchId="string-equal">
              <AttributeValue>
                SecureManagedSystem
              <AttributeDesignator>
                subject-id
            <AnyResource />
            <ActionMatch
              MatchId="string-equal">
              <AttributeValue>
                urn:xadl:action:AddBrick
              <AttributeDesignator>
                action-id
            <Condition FunctionId="not">
              <Apply
                FunctionId="string-is-in">
                <AttributeValue>NATO
                <AttributeDesignator>
                 principal
```

Figure 3, Secure xADL description of Coalition. The policy will not create any connector that does not possess a NATO principal.

Tool Support

Tool support is included as part of our base architecture development environment, ArchStudio. The tools include an editor to describe a secure software architecture written in Secure xADL, a checker to decide whether one interface has sufficient privileges to access another, and an execution engine to execute secure architectural operations for event-based software architectures.

Contact Information

Jie Ren
Professor Richard Taylor
Institute for Software Research
University of California
Irvine, California 92697-3425
{jie, taylor}@ics.uci.edu
+1-949-824-{2776, 6429}
Fax 949-824-1715

To learn more about the Secure xADL language, please visit the website:
http://www.isr.uci.edu/projects/xarchuci/
http://www.isr.uci.edu/projects/archstudio/

This material is based upon work sponsored by the National Science Foundation under grant number 0326105 and by the Intel Corporation. The content of the work should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of either organization.