CSE591: Security Policy Frameworks

Scott Stoller and Annie Liu

Project. Version: 4Mar2005a.

An initial proposal for your course project is due on March 10. Some ideas appear below. It is not a comprehensive list. Feel free to suggest other topics.

Half a page to a page is enough for the initial proposal. You may work in groups with one or more members. Each group should submit one proposal, with the names of all group members on it. The size of the project should be proportional to the size of the group.

Shortly after you submit the initial proposal, I will meet with each group to discuss it. You may submit a revised proposal after that. Of course, you are also welcome to discuss ideas with me before submitting the initial proposal. If you are undecided between two or three topics, you can briefly describe each of them in your initial proposal.

Each group will present its project in class, during the last week of class, so other groups can benefit from their experience. Printouts of final versions of all documents (i.e., revised versions of all previously submitted documents and all remaining documents) are due in class on the last day of class. Electronic submission of all project-related files (documents and code) is due by midnight on the same day and should be emailed to `stoller@cs.sunysb.edu`. On the next day, all groups that implemented something will demo their system.

This document contains active URLs, for your clicking convenience.

1 Security Policy for an Application Domain

Develop prototypical security policies for some application domain. Your project proposal should specify the application domain and sketch the aspects you will consider.

Project deliverables include:

AppDomain: Introduction to the application domain, including typical system architectures, and typical organizational and administrative structures. Be sure to indicate the relationship between the system architecture and the administrative structure: who administers each part of the system?

InformalPolicy: An informal (English) description of the security policy. "Informal" is not a synonym for "vague." Be precise. Organize the policy and your presentation of it, by dividing it into sections. Security policy administration is very important and is considered part of the security policy. This policy should be free from "implementation bias". In other words, your development of the desired policy should not be influenced by premature concerns about how it will be expressed in a particular policy framework or how it will be implemented.

PolicyDesign: Explanation and justification of the design of the policy, and description of alternatives that were considered. Discuss how well-known security design principles (least privilege, separation of privilege, separation of duties, accountability, etc.) were applied.

PolicyFramework: Description of the security policy framework used for the policy, and a justification of the choice. You should use the most suitable framework you can find or devise. It does not need to be one that we discussed in class. If you adopt an existing policy framework, just include a citation to a description of it, and explain any modifications or extensions that you made, by describing their syntax and semantics and algorithms needed to handle them.

If some aspects of the policy cannot be expressed in a satisfactory way in existing policy frameworks or reasonable extensions of them, then you should describe the difficulties, and if possible suggest how these aspects could be handled in the next generation of policy languages. For now, these aspects should be enforced by other parts of the system and should therefore be discussed in Interactions (described below) as well.

FormalPolicy: A formal specification of the security policy in the selected policy framework, accompanied by an informal description of any non-obvious aspects.

Interactions: Discussion of interactions between the formal security policy (and the access control engine) and other parts of the system. What external data and functions does the policy rely on? What assumptions about other parts of the application are needed to ensure that the overall security policy described in InformalPolicy is enforced?

Re-Use: Discussion of policy re-use. Is the policy structured so that it can easily be customized for use by different organizations in the same application domain?

PolicyFramework2: Evaluation of the suitability of the chosen policy framework for that application domain. Is it sufficiently expressive? If not, what are the limitations? Is it convenient? If not, how could the intentions of the policy writer be expressed more clearly and directly? (This overlaps with PolicyFramework, but the former item is written with foresight, and this item is written with hindsight.)

Implementation: An implementation of the security policy. The implementation can be centralized, even if actual deployment would require a distributed implementation. Similarly, use of real digital signatures and other authentication mechanisms is not required. You are free to use any available software. You should also describe testcases used to validate your implementation and the results of the testing.

UserManual: A brief user manual, describing how to start and use your implementation and run the testcases.

You are welcome to propose any application domain, and we will discuss its appropriateness. It should be an application domain that you know something about or want to learn about. Some ideas are: a university, a (specific) government agency, a consulting company, a financial institution, or a law firm. We will avoid multiple teams developing policies for the same application domain, so please try to suggest a first choice and a second choice, especially if you select application domains from this list.

Schedule	
Mar 10	initial project proposal due
Mar 10-16	meetings to discuss initial proposal
Mar 17	revised project proposal due
Apr 14	AppDomain, InformalPolicy, and PolicyDesign due
Apr 21	PolicyFramework due
Apr 28	FormalPolicy due
May 3,5	in-class presentations
May 5	printout of all documents due in class
May 5	electronic submission of everything due at midnight
May 6	demos

2 Security Policy Infrastructure

Develop a system that supports trust management and is compatible, as much as possible, with some existing system or standard. A significant part of this project is designing the trust management system to fit smoothly with and maximize backward-compatibility with the system or standard you are extending.

Some possible projects of this kind are described below, as examples. There are many other possibilities. You are encouraged to explore and think about the options before settling on a project. Your project proposal should briefly describe the architecture of your system and a sample application.

One possible project is a trust management system that uses OASIS's eXtensible Access Control Markup Language (XACML). Details on XACML, and links to implementations of XACML, are available at www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

The XACML standard defines an XML-based language for access-control policies. It also defines various XML-based interfaces, including an interface for authorization requests, which applications use to ask an authorization server, called a Policy Decision Point (PDP), whether an action is permitted. The project would involve extending the XACML policy language to support trust management, and implementing a Policy Decision Point that evaluates policies in that language. The design should discuss to what extent this can be achieved by backward-compatible extensions to XACML, as opposed to completely replacing the existing XACML policy language. The XACML standard does not specify whether the application and PDP run in the same or different processes, on the same or different machines, etc. It does provide a Security Assertion Markup Language (http://xml.coverpages.org/saml.html) profile for XACML that standardizes a remote communication mechanism for systems that require one. Delegation is being considered by the technical committee developing XACML (search for "delegation" on the web page mentioned above) but is not currently part of the XACML standard. You could use an extended version of XACML with SAML to add trust management functionality to a web-based system.

Another possibility is to add trust management functionality to IBM Tivoli Access Manager, an enterprise-level access-control solution that integrates with Websphere and other popular products from IBM. We have access to this software through the IBM Academic Initiative. I downloaded IBM Tivoli Access Manager Web Security for Solaris and for Windows 2000/2003 but have not installed them yet. Tivoli Access Manager can be extended by adding plug-in External Authorization Service (EAS) modules. It may be possible to implement trust management this way. There are some important issues that you should consider before deciding to pursue this project. For example, how would an EAS module request foreign credentials from other entities? Can this be done through an extension of an existing interface, or is a new interface needed? Also, how would an EAS module obtain information about objects (users, patient records, etc.)? In other words, how would such external functions be implemented in a way that integrates smoothly with the rest of the system? For general information about IBM Tivoli Access Manager, see http://www-306.ibm.com/software/tivoli/products/access-mgr-e-bus/. Voluminous documentation is available. For example, see the 678-page redbook *Enterprise Security Architecture Using IBM Tivoli Security Solutions* at http://www.redbooks.ibm.com/abstracts/sg246014.html?Open, especially Section 5.5 (Interfaces).

Another possibility is to extend the J2EE security model, which currently supports a simple form of role-based access control, to support trust management, and implement the extended model in some open-source J2EE framework, such as JBoss http://www.jboss.org.

The following projects might also be feasible, but I have some reservations about them, so you would need to discuss them with me before selecting them.

You could try to extend SPKI/SDSI (http://www.ietf.org/rfc/rfc2693.txt?number=2693), which already supports a limited form of trust management, to support a richer trust management language. It is unclear to me whether this can be done in a reasonably backward-compatible way. Also, I don't know what systems are based on SPKI/SDSI are available for you to build on.

You could try to extend the Open Group's authorization interface, aznAPI, to support trust management. For this to be an interesting project, you would need to find some system that uses aznAPI and that you can extend to take advantage of your extensions. I am not aware of any, but I didn't look. aznAPI is described at http://www.opengroup.org/onlinepubs/009609199/index.htm.

You could try to extend the Java Authentication and Authorization Service (JAAS) to support trust management. JAAS is described at http://java.sun.com/products/jaas/. Section 5.3 of the white paper on JAAS at http://java.sun.com/security/jaas/doc/acsac.html mentions the possibility of a policy language that, like Keynote or SPKI, supports delegation. However, JAAS is fairly limited, supporting authentication and simple identity-based access control, and does not seem to be designed to be extensible, so I am uncertain whether extending it to support trust management is feasible.

Please keep in mind that, while I will be happy to help as much as I can, it is infeasible for me to be an expert on all of the technologies and systems that different groups might use in their projects.

Project deliverables include:

Design: A design document, describing and justifying the design of your system. The design should include a discussion of security policy administration.

Implementation: The implementation. The code should be well documented.

TestPlan: A test plan, describing testcases used to validate your implementation, and the results of testing.

UserManual: A user's manual, including instructions for installing, compiling, and running your system.

SampleApp: A sample application, with documentation of how to run it. The sample application does not need to be elaborate. It needs to demonstrate the features of your system.

Schedule	
Mar 10	initial project proposal due
Mar 10-16	meetings to discuss initial proposal
Mar 17	revised project proposal due
Mar 31	Design due
Apr 14	status report due
Apr 28	TestPlan and UserManual due
May 3,5	in-class presentations
May 5	printout of all documents due in class
May 5	electronic submission of everything due at midnight
May 6	demos

The status report due on Apr 14 should indicate which parts of your system and sample application have been implemented; if there have been any changes to the design, please submit a revised design document, too. The user manual and test plan due on Apr 28 are not expected to be final versions but should be mostly finished.

The system should use authentication (e.g., Kerberos), digital signatures, and secure communication (e.g., SSL/TLS), as appropriate for a secure distributed system. The sample application should be distributed, i.e., involve processes running on multiple machines.

You may use any available software as part of your system. You may implement it in any combination of programming languages.

3 Other Projects

If you are interested in working on any other topic, you are welcome to discuss it with me. Send me email suggesting some times when you are available, and I will try to choose one of them.

Grading

Each submission listed in the Schedule tables, with the exception of the initial and revised project proposals, should include a work breakdown, briefly describing the contributions of each group member to the project since the previous submission. This information, and all other available information, will be taken into account at the end of the semester when determining each group member's score for the project. All members of a group will receive the same score for the project if they all contributed equally to the group's effort. We expect this to be the case for most teams.

Constraint Logic Programming Systems

If your project involves a rule-based security policy language, you might want to use an existing constraint logic programming (CLP) system in your implementation. A web search will quickly turn up several of them. Two things to look for are support for external functions (usually called a foreign language interface) and constraints. I investigated a bit and suggest that, if you need to use a CLP system, you consider XSB or Eclipse. If you find any other CLP systems that look promising, please let me know, and I will share the information with the class.

XSB, available at xsb.sourceforge.net. XSB has a foreign language interface for calling C and C++ functions. XSB can also interface with Java. XSB has a database interface based on ODBC. XSB supports Constraint Handling Rules (CHR), which is a mechanism for extending logic programming systems to handle constraints. CHR is designed to make CLP systems open-ended: adding support for a constraint domain ideally just requires writing a few high-level constraint-handling rules, which, roughly speaking, are re-write rules that describe how to simplify those constraints. XSB is a mature and powerful system, developed mainly at Stony Brook. It uses tabling for efficient top-down evaluation. Support for CHR has been added relatively recently, largely by Tom Schrijvers at Katholieke Universiteit Leuven in Belgium. For more on CHR in XSB, see volume 2 of the XSB manual, available from the above URL. For even more details, see the publications at www.cs.kuleuven.ac.be/˜toms/Research/CHR/.

Eclipse (not the Java IDE) is available at http://www.icparc.ic.ac.uk/eclipse/ . Eclipse has existing support for numerous (more numerous than XSB) constraint domains, some built-in and some implemented using Constraint Handling Rules. There seems to be good documentation for Eclipse (and XSB), but if you get stuck on something with Eclipse, we don't have local experts on it, as far as I know. I never used Eclipse, so I don't know how responsive the developers are. Also, Eclipse uses top-down evaluation without tabling, so it is less efficient than XSB for some classes of programs.