

# IMPLEMENTING APPLICATION SECURITY

---

THOUGHTS AT RANDOM

PRINTED: FRIDAY, AUGUST 29, 2003

---

© PREPARED BY HILBERT COMPUTING, INC.,  
2003



---

# IMPLEMENTING APPLICATION SECURITY

THOUGHTS AT RANDOM

---

## TABLE OF CONTENTS

<b>OVERVIEW.....</b>	<b>2</b>
<b>AN ABSTRACT VIEW OF SECURITY.....</b>	<b>3</b>
<b>Authentication.....</b>	<b>3</b>
<b>Authorization.....</b>	<b>4</b>
<b>GREATEST RISKS AND COSTS.....</b>	<b>5</b>
<b>Strategic Security Services.....</b>	<b>5</b>
<b>Application Integration.....</b>	<b>5</b>
<b>MITIGATING COSTS AND RISKS.....</b>	<b>7</b>
<b>CONCLUSION.....</b>	<b>9</b>

---

# IMPLEMENTING APPLICATION SECURITY

THOUGHTS AT RANDOM

---

## OVERVIEW

There is a need for Sprint to adopt a process for integrating application-level security within all of its applications. There is a sense of urgency for two reasons. The longer it takes to make a decision on an implementation, the greater the migration costs. The second reason is that many development projects will simply omit application security until the security decisions have been finalized.

Security is a complex issue for several reasons. The technology itself is complex. The needs of each application may be different. The requirements for the operational groups may drive a different decision than the needs of the applications groups. The requirements for the end-user, namely for security to be as unobtrusive as possible, are in direct conflict with the needs of the organizations responsible for enterprise security. Security by its nature is intrusive when requesting credentials.

This document wades through these issues with the goal of creating an approach to getting application security implemented in as short a time frame as possible while minimizing risk and accommodating the needs of a diverse group of stakeholders.

---

# IMPLEMENTING APPLICATION SECURITY

THOUGHTS AT RANDOM

---

## AN ABSTRACT VIEW OF SECURITY

There are two primary components of application security: authentication and authorization. While there is obviously overlap in the two, the two are independent enough to be considered separately. In fact, it can be possible to implement authentication and authorization using different vendors and different technologies.

The key to managing the complexity of application security is the same as managing complexity in all object-oriented systems. The application view must be kept as abstract as possible. In this case, the application systems view security as calls to authentication and authorization services with no understanding of the concrete implementation of those services.

Inserting an abstraction layer enables a migration from the current collection of point solutions to a common architecture and ultimately a common security store. A collection of point solutions makes it difficult, if not impossible, to establish a single signon solution. Management of the security datastore also becomes more costly since there are multiple techniques required to manage permissions and passwords for multiple applications. The collection of point solutions is also less secure since employees leaving the company or transferring to a different department require that multiple security stores be updated to revoke or alter permissions. As the number of security stores increase, the likelihood of missing a few needed changes increases.

## AUTHENTICATION

Authentication answers the simple question, “*Are you who you say you are?*”, although finding that answer may be quite complex. If we look at this from the highest level, an application is presented with a set of credentials and the application needs to decide if this is sufficient to adequately ensure identity.

Applications with no security requirements may require no credentials. Applications with some security requirements may be satisfied with userid/password credentials that have been validated within some recent period of time (the “freshness”). As applications increase their security needs, greater demands may be placed on the credentials with respect to time or contents. For example, applications that have personalized, but not sensitive data, may require that the credentials have been authenticated within a couple of

---

# IMPLEMENTING APPLICATION SECURITY

THOUGHTS AT RANDOM

---

hours<sup>1</sup>. Applications with more sensitive content may require that the user has been challenged for explicit verification of credentials within a shorter time frame. Highly sensitive information may require stronger credentials such as a digital certificate, validation via SecureID or biometric data.

There also may be a desire to request additional security based on the way in which the application is accessed. Remote access may require additional credentials such as a certificate or network access credentials.

From an application developer perspective, it is desirable to push as much work into the authentication services as possible. Most Unix distributions have implemented PAM (Pluggable Authentication Modules) that allow the operational staff to replace authentication mechanisms without changing the applications that exploit. The Java2 security model implements the same model as PAM for all platforms.

Fortunately, there are multiple authentication services on the market that can satisfy these requirements.

## AUTHORIZATION

Authorization is the process of determining what can be done by the software once the identity of the user has been established by the authentication process. The concept is familiar, so not a lot of commentary is required.

The key to success here is similar to that of authentication, but it is even more critical since authorization tends to pervade the application code more than authentication, which tends to be a one-shot and isolated piece of code. Authorization (aka “permissions”) that is granted to an authenticated identity **absolutely must** be handled externally to the application as a part of an authorization service. That is, the developer should not code what permissions are granted given certain credentials. Explicit logic that is coded to say “if you are a member of administrators, then you can do this” is not a supportable or scalable security architecture. The code must turn the logic around to state “to execute this piece of code, you must have this permission”. The way in which the permissions are granted is a function of the security services and is external to the application.

---

<sup>1</sup>With some security services, credentials can be automatically refreshed without requiring user interaction. Basically, this is a guard against leaving the workstation unattended and an impostor taking over the interaction.

---

# IMPLEMENTING APPLICATION SECURITY

THOUGHTS AT RANDOM

---

## GREATEST RISKS AND COSTS

Given the short refresher on the security concepts, some discussion is warranted on what can go wrong and what can be done to mitigate those concerns.

### STRATEGIC SECURITY SERVICES

Since there are multiple approaches to application security and multiple vendors offering solutions, there is a requirement to pick the vendor to provide security. Since security is an essential part of the IT infrastructure, this is clearly a strategic decision.

The nightmare scenario is that a product is selected, and applications developers have spent thousands of hours coding to a product that doesn't meet the needs of the IT infrastructure. The product may fail in at least one of the following areas:

- The product fails to meet the requirements for application security.
- The product causes operational problems or has stability problems.
- The product is functional, but fails to scale adequately
- The production is functional, scalable, stable and meets the demands of the applications developers, but is too costly to administer.

To state this risk another way, there are multiple stakeholders, two of which are operations and applications, that are exposed to the risk of an ineffective choice for the product that provides security services.

### APPLICATION INTEGRATION

Another area of concern is the time and cost associated with integrating application security for those applications that do not have security. A second similar issue is how to migrate applications that have proprietary security into the strategic product for security so that all applications can participate in single signon and so applications can leverage a

---

# IMPLEMENTING APPLICATION SECURITY

THOUGHTS AT RANDOM

---

single security store and thus lower administrative costs of maintaining authentication credentials and associated authorizations.

When looking and migration and integration techniques, the topic of declarative vs. programmatic security may arise, so this topic deserves some commentary. In most cases, both would be considered object-level security, although the implementation really drives the granularity of the security.

As the name suggests, programmatic security is coded within the application to specify the permission requirements that must be present within the credentials of the authenticated user before the code will be allowed to execute. This is the authorization implementation that has been around since the 1960's.

Declarative security is more recent. In an effort to make security more easily implemented, there is a trend toward security being "declared" instead of programmed<sup>2</sup>. In this technique, there is a way of identifying a functional piece of code for which a set of authorizations are required before it is allowed to execute. If a piece of code can be identified outside of the code itself, the security requirements can also be described outside the code itself. My understanding of SiteMinder™ is that it is declarative in nature. It uses a URL to determine a functional piece of code. Required permissions are indicated within the policy and no programmatic security is required within the application.

The main advantage of declarative security is that it can be implemented more easily and changed more easily since the requirements can be described externally to the application and no programming is required. The problem with SiteMinder™ is that it uses a URL to determine the unit of code to be secured. This does not necessarily reflect the actual code being executed and may not provide a granular enough to represent the smaller units of code within an object the require protection.

---

<sup>2</sup> Declarative security is similar to declarative languages. SQL is a declarative language and merely states what is desired in a result set. IMS programming, on the other hand, requires the developer to explicitly code the migration through the data structures.

## MITIGATING COSTS AND RISKS

There are some things that can happen that will significantly reduce the exposure to Sprint in the implementation of application security.

- **Convert a strategic decision into a tactical decision.** The fear for applications developers is to expend effort coding to a security programming interface that will change. The solution to this concern is to create an abstraction within the application that enables the actual implementation of authorization and authentication to be loaded dynamically and changed dynamically at a future time. This empowers applications developers to begin implementing application security without concern for what products are used to implement the security services.

This technique is documented for authorization in the separate white paper entitled “*Pluggable Security. Techniques for Object-Level Security in Java*”. This is not a theoretical concept. It is in production for another customer using LDAP-based security and is in production at Sprint in the Hilbert Frameworks Log Monitor application where the security implementation is an “allow everything” algorithm.

- **Migrate proprietary security to pluggable security.** Some applications at Sprint have already implemented an application-specific security model. If that model uses the industry-accepted practices of authenticated subjects and authorizations as noted in this paper, they could convert to the pluggable technique discussed above to implement access to their custom security store.

When the strategic security store is available, migration can occur without an coding change. While this doesn't necessarily lessen the cost of migration, it can enable migration to begin before the strategic decision for security services is made.

- **Decouple operational requirements from application requirements.** Using the same pluggable security technique, the applications are coded to an abstraction layer that does not involve a concrete implementation of security services. This eliminates much of the need to reconcile application requirements with operational requirements.
- **Move to declarative security where possible.** If applications are developed within an framework, then there is a good possibility that security can be implemented at the framework level. If the framework is such that it can externally identify functional pieces of code and those pieces of code are

---

# IMPLEMENTING APPLICATION SECURITY

THOUGHTS AT RANDOM

---

assembled externally to the application, then there is a good chance that security can be implemented in a declarative fashion and eliminate much of the cost of implementing security. This is certainly the most desirable for new applications development. For example, the Jakarta Struts framework for servlets has a convenient place to enable declarative security.

This declarative technique has been used by Hilbert Computing and is in production at another customer location. Security was added to all pre-existing servlet applications with zero additional code except the addition of a web page for logon and password. The total effort for all applications was less than a day.

## CONCLUSION

No one debates the need for application security. The decision for which security services to use is a critical strategic decision. The risks of that decision can be significantly reduced using the techniques discussed in this paper.

This was meant to stimulate thoughts and conversations on the implementation of application security at Sprint. If there are questions, I can be reached at:

### **Author Information**

*Gary Murphy  
Hilbert Computing, Inc.  
13632 S. Sycamore Dr.  
Olathe, KS 66062*

*e-mail: glm@hilbertinc.com  
voice: (913) 780-5051*