Component Security Design Considerations for J2EE and .Net – An Architectural View Part 3

25

Gunnar Peterson

"When you are content to simply be yourself and don't compare or compete, everybody will respect you." -Lao Tzu

Introduction

In the previous two parts of this series [1, 2], we explored security design considerations for Sun's Java 2 Enterprise Edition (J2EE) and Microsoft's .Net platform for enterprise components. In this part we will examine which differentiators are available for those who are in the process of choosing between the two.

We will deal with the case of organizations running both types of software by discussing utilizing Web Services to provide interoperability between the two platforms and the security issues in these technologies.

Depending on your role, different comparison criteria can be more or less valuable to you from the perspective of using a given platform. The frameworks will be compared from the architectural viewpoint sine the architect should have a high degree of concern regarding security as well as the ability of the chosen framework to live up to design, development and deployment criteria.

This role will be defined differently in different organizations, but the point is to show which criteria are crucial from a cross-functional viewpoint. An effective security architect should be able to parse and consider the security issues and implications from many viewpoints.

Finally, if you are reading this and expecting some religious "good versus evil" comparison, you'll be disappointed. Surprisingly, the platforms are from many viewpoints, more similar than different. Each platform has its own nuances and implications and it is these subtle differences that we'll delve into.

Architect's View

Architect's Role

The architect is the chief advocate of the "ilities": scalability, stability, reliability, usability, interoperability and security. The architect is generally the individual charged with thinking horizontally across the organization. This role is an ideal spot from which to push for the holy grail of "defense in depth."

Process and Design Revisited

No matter which platform you choose, you are hosed from a security standpoint if you don't utilize software design and development process to the fullest extent possible. The architect is the logical starting point to address where security issues and requirements fit in the design and development process.

Frequently, security requirements are gathered at the end of a project life-cycle. Only after the functionality of a system has been coded the development team sets about figuring out how "secure" it. This is too late. As we have seen in the case of the ActiveX controls and Java Applets, designing security into the basic requirements from the beginning is essential to success.

The security team should be heavily involved during the *Inception* and *Elaboration* project phases to ensure that the requirements are gathered and described correctly and that the corresponding design handles the security issues.

Design Options Compared

Users

As an architect designing security implementations, one of the first things to work on is the storage and management of user information. J2EE and .Net allow for a variety of options regarding users. Both support LDAP and RDBMS as user stores while the .Net platform also adds its own Active Directory as a user solution. The architect may view the straightforward integration with Active Directory as a bonus for the .Net platform, since this may make for better reuse. Understanding the existing user stores in your organization and how well they integrate with either platform is important in order to create a manageable system.

Operating Systems

With all due respect to the Open Source ports of .Net, the price to pay for the integration with the Windows tools and technologies is that the system is also tied to the Windows OS and its security issues. The focus of this series is on the

middleware frameworks, not the supporting operating systems, but one of the limitations with .Net is that you inherit the security problems in the Windows operating systems.

Being tied to Windows (or any other single operating system) also limits the architect's choice regarding the scope of security management and monitoring tools that are available.

In a J2EE application (which can run under Windows), you will also inherit the security issues from the operating system, but you have the chance to pick which operating system can provide the best security for your organization. The advantage for J2EE is that the component framework does not require a specific operating system. However, if you organization is already running .Net servers this advantage is not so significant.

Note also, that for J2EE application portability across application server vendors is not ensured. Porting a Weblogic application from Windows to Linux will probably be easier than porting it from Weblogic on Windows to Websphere on Windows.

Programming Languages

Just as the .Net architecture necessitates the Windows operating system, J2EE requires that developers develop in the Java programming language. .Net developers are free to develop in a wide variety of languages, such as Visual Basic, C++, or the new C#.

While being locked into an operating system can be a security disadvantage, from an architectural viewpoint being locked into a language may not be.

The reason is that an operating system has a cascading effect across other enterprise design decisions. By being locked into a given operating system you are automatically limited with regard to which technologies you can use, e.g. for intrusion detection or as a firewall. The limitation of programming language is not nearly as restrictive for an architect, although a developer may feel differently.

For the architect the power is in the middleware framework itself, not in the language.

While the ability to support multiple languages is a significant productivity advantage for .Net, it adds complexity for the architect who must manage, standardize and design to accommodate security issues in each language.

As J2EE uses solely Java, the architect can provide a consistent, detailed set of requirements for secure utilization of Java in the environment. Then the architect is free to spin more brain cycles on middleware platform issues.

Developer Tools

Microsoft has historically been at the forefront at producing effective developer tools for middle of the road developers. The prime example of this is Visual Basic. As trendy as it is to bash VB, I am not going to knock it. Remember that architecturally speaking, the power is in the framework. With Visual Studio .Net, Microsoft continues their fine tradition at producing excellent development, debugging and deployment tools. Anything that facilitates higher quality code in your production system is an advantage to you, the security architect.

As you may expect, the tools story on the Java side is less consistent and varies from one vendor to another. Java tool vendors offer a variety of great tools ranging from products for relatively small scale systems up to enterprise environments that can model and generate Java code directly from UML models. However, there is no product in the Java space that can match Visual Studio.

Organizational and Integration Issues

In most organizations, security will not be the sole determining factor in selecting a middleware platform. If it were, Microsoft could very well be dominating the market with Visual Ada.

Security is a strategic concern, but other issues such as existing staff skill set and integration will probably dominate the discussion of which platform to choose. Understanding which security model makes sense in your organization can add value to this discussion, however.

Since one of the main roles of middleware is to facilitate the integration of legacy and back-end systems it is worth considering the security implications of the integration strategies inherent in J2EE and .Net. J2EE provides the Java Connector Architecture (JCA) as a means of legacy integration. Microsoft offers a variety of technologies for legacy integration, such as BizTalk.

Each type of integration project has many variables. Hence, it is important to consider the middleware security implications for J2EE or .Net in the context of the target system and its limitations. Some issues to consider when choosing J2EE or .Net as the middleware platform for integrating to a legacy system such as SAP or Siebel are:

- Is it possible or preferable to use Single Sign-On?
- Can the context from the session be passed to the legacy system?
- Are roles supported?
- Which protocols are available for communication?
- Can the applications exchange keys?

- How are errors handled? Can error message be propagated up to the middleware layer?

Management and Monitoring

Since .Net is designed as a vertical slice, it benefits from great management tools from all parts of the .Net and Windows platforms. The security team can leverage management tools from contiguous platform elements such as Active Directory to aid administration on .Net applications. In order to establish more pervasive security it is beneficial to supplement the standard .Net administration tools with technologies that are not part of the .Net stack. In particular, utilizing a 3rd party monitoring or managed security monitoring system such as Counterpane [3] in conjunction with standard .Net tools enables ease of administration and provides a balanced view of the system.

In J2EE, the application server vendors each have a unique implementation regarding administration. The *Java Management Extension* (JMX) standard is attempting to bring all of these implementations under the same roof, but right now there is no common solution. Third party products can be used to integrate the systems and provide functionality for security management and monitoring. Since J2EE was designed as an ecosystem for multiple vendors it is possible to come up with a variety of approaches such as best of breed or lowest cost for management and monitoring. However, achieving this in practice places a greater burden on the architect.

Despite being limited to the Windows operating system with regard to management and monitoring functionality, the .Net environment offers a fairly complete set of tools that can be supplemented as needed. J2EE management provides a greater breadth of functionality, but is basically a series of toolkits that need to be configured and integrated to work together harmoniously.

Interoperability

From the "why can't we all just get along?" department we have Web Services. Both Scott McNealy and Bill Gates agree this is the way systems should communicate in the future. History tells us that this may be the last time these two agree on anything for a while, so let's make the most of it.

Web Services with SOAP

"Keep your friends close, but keep your enemies closer."

-Michael Corleone in
"The Godfather" by Mario Puzo

Many security pundits deride Web Services and SOAP because of their ability to be a "firewall friendly" protocol. The reality is that today organizations already traverse the firewall using

RMI-IIOP (J2EE or CORBA), DCOM (COM/COM+), or proprietary messaging systems. Given that Web Services simplify this equation by offering a consistent, standard way to communicate for distributed applications, Web Services actually are an incremental improvement for secure development versus the current situation.

There are numerous issues to consider regarding securing your Web Service. A full discussion of this is out of the scope of a component security article. Some of the details to examine are covered in [4].

In the case of interoperable Web Services where each Web Service is running on a different type of server, for example a J2EE Web Service communicating with a .Net Web Service, one of the main security advantages is to leverage the XML document that defines the data payload between the services. There are two main things to consider using the XML document to add layers of security to your Web Services application.

- The XML payload can be used to propagate important security attributes such as roles and policies in a format that is understandable across middleware frameworks.
- Utilize the Web Service XML schema to embed logic and hierarchies that enforce security rules, such as relationships and data access permissions.

Note also that Web Services are an important integration strategy for interoperability across disparate J2EE application servers.

Conclusion

Given an architecture-centric software development methodology and an energetic security team both J2EE and .Net enable the security team to play a significant role in shaping the overall security of the component development.

Unless you are starting an organization from scratch many of the design decisions will be dictated to you by the current system. It is vital to understand which platform offers better security within the context of your organization's systems, skills and processes. The fundamental key to success is for the security team to participate shoulder to shoulder with the developers and architects throughout the development life-cycle.

References

[1] *Component Security Design Considerations for J2EE and .Net - An Architectural View, Part 1*, Gunnar Peterson, Information Security Bulletin, Vol 7, iss. 5 (May 2002), pp 29-34.

[2] *Component Security Design Considerations for J2EE and .Net - An Architectural View, Part 2*, Gunnar Peterson, Information Security Bulletin, Vol 7, iss. 6 (June 2002), pp 17-20.

[3] www.counterpane.com

[4] http://www-106.ibm.com/developerworks/webservices/library/ws-secmap/

Gunnar Peterson is a Software Architect. He designs secure, stable, and scalable solutions for complex problem spaces. Over his ten year career, he has been dedicated to design and development of distributed middleware Object-Oriented and Component systems for clients ranging from large enterprises to start ups. Currently, Gunnar is CTO of Arctec Group. Arctec Group's primary focus is "Strategies for Industrial Strength, Integrated Architectures." He is shown here on the trail of the wily cutthroat trout deep in the Rocky Mountains. He can be reached at gunnar@arctecgroup.net.