Transactions of the Korean Nuclear Society Autumn Meeting
Busan, Korea, October 27-28, 2005

Preliminary Study on Secure Intranet Geographical Information System

KIM Hyun Tae, PARK Jee Won, KO Han Suk
Korea Atomic Energy Research Institute
htkim@kaeri.re.kr

1. Introduction

A Geographical Information System (GIS) is usually defined as an information system for capturing, checking, storing, retrieving, manipulating, analyzing, and displaying spatial and relevant non-spatial data. Here 'spatial' means 'geo-referenced to the earth.' It is estimated that about 80% of the data used in business and government are of spatial type.[1] The geo-referenced information on sensitive location is usually protected as the highest level of confidentiality by the most information system.

This paper discusses a commercial satellite imagery based secure Intranet GIS which runs the Microsoft .NET technology.

2. Methods and Tools

2.1 Intranet GIS

The GIS which takes account of location and shape of objects concerned along with traditional table-oriented attribute information becomes an integral part of the information infrastructure in many organizations. To bring together wide variety of information is the crucial problem of the GIS. The GIS is usually classified as static GIS and dynamic GIS. In case of buildings in the area of the nuclear site, most of their locations and shapes can be considered as spatially and temporally fixed for a span of 1 year or so. Therefore the GIS for the nuclear site can be regarded as the static GIS and the relevant spatial information can be extracted and exported into the folder of the web server in advance. For many spatial applications, it is sufficient and desirable to access geographic data in the form of simple non-topological features where there is no spatial relationship between features. This approach is suitable for developing integrated applications where geographic data is a vital component, but perhaps not the focus. Figure 1 shows our Intranet static GIS development environment.

Since commercially available satellites can take imagery nearly all area on the globe, geo-referenced satellite imagery can be used as basic and starting information for GIS. From the satellite imagery in .geotif format two kinds of raster images, full area and interested area, in the .jpg format were exported to an Intranet web site to be hyperlinked from the ASP.NET application. For GIS tool, ArcView 8.1 was used and the personal geographic database (geodatabase) was used as the geographic data format. In our static GIS, the personal geodatabase deals with vector data and raster data only. Vector data and extracted attribute data are stored in the personal geodatabase file. The raster data is located at the folder in the web server as shown in Figure 1. The personal geodatabase and GIS database which contains other relevant information was integrated with the UNION query. It is known that .mdb personal geodatabase of ArcView 8.1 can handle up to 250,000 geographic features without performance degradation. If the imagery of nuclear site has 1,000 features in average, then one .mdb personal geodatabase can store the spatial information of nearly 250 nuclear sites along with their vital attribute information.[2]

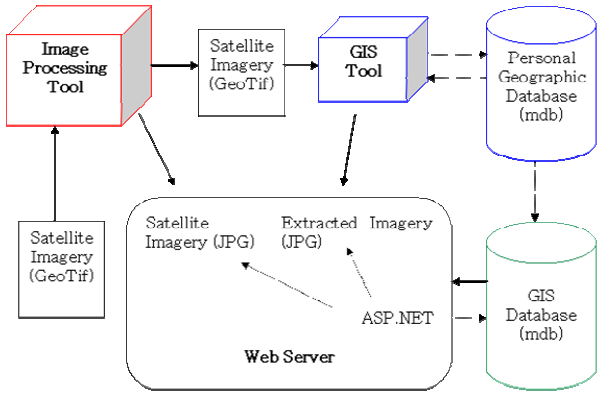

Figure 1. Diagram of Intranet GIS development

Related information stored in other databases such as SQL Server or Oracle not shown in Figure 1 is expected to be integrated in the future.

2.2 IBuySpy

The prototype of the secure Intranet GIS to be used by this project is the IBuySpy Portal application, a free, open-source software. The IBuySpy Portal application is shipped with six modules designed to cover the most common content types – (announcements, links, images, discussions, html/text, XML) as well as a number of modules for administering the portal site.[3] The IBuySpy Portal application is developed with Microsoft ASP.NET and Visual Basic .NET technology. The copyright license applied is the BSD (Berkeley Software Distribution) which permits free use of the software on condition to retain the copyright notice in all copies or derivative works. Until DotNetNuke 3.0, the portal only offered a single security solution: the forms-based security that was included with the core release. The forms-based security worked well, but it

limited the ability to implement DotNetNuke in way that tightly integrates with other security mechanisms.

2.3 Security

Authentication is the process of positively identifying the clients of application; clients might include end-users, services, processes or computers. Authenticated clients are referred to as principals.

The authorization process governs which resources and operations the authenticated client is allowed to access.[4]

In an environment that allows for dynamic downloads and execution and even remote execution, security is the most important issue. Many traditional security models attach security to users and their groups (or roles). Additionally, the .NET Framework also provides security on code and this is referred to as code access security which enforces its security policy based on where code coming from rather than who the user is. This model makes sense in today's environment because so much code is installed over the Internet.

The Common Language Runtime (CLR) of the .NET Framework has its own secure execution model that transcends the underlying operating system's security model. The CLR implements a code access security model in which privileges are granted to code, not users. Upon loading a new assembly, the CLR gathers evidence about the code's origins. The security policy accepts evidence as input and produces a permission set as output. A permission is a right to perform some trusted operation. The determination of which code is assigned which permission is called policy. The Common Language Infrastructure (CLI) categorizes code into two broad families: verifiable code and non-verifiable code. Verifiable code can be mathematically proven to adhere to the type-safe execution model that the CLI encourages. .NET compliant languages such as Visual Basic .NET and Visual C# .NET produce verifiable code by default.[5]

To customize the DotNetNuke for our needs, additional ASP.NET pages and server controls might be added to the DotNetNuke.

ASP.NET pages and server controls might be subject to intense probing by attackers who are intent on compromising our Intranet GIS's security. These attacks are often ultimately aimed at back-end systems and data stores. Security risks can come from anywhere. Here are 10 basic guidelines to follow:[6]

1. Don't trust user input
2. Protect against buffer overruns
3. Prevent cross-site scripting
4. Don't require sa permissions
5. Watch home grown crypto code
6. Reduce your attack profile
7. Employ the principle of least privilege
8. Pay attention to failure modes
9. Impersonation in fragile
10. Write applications that non-admins can use

3. Conclusion

Our prototype software for secure Intranet GIS is the free, open-source DotNetNuke based on Microsoft ASP.NET and Visual Basic .NET technology. Secure web pages and controls conforming to basic 10 guidelines are required to add additional functionality.

REFERENCES

1. A. Poucet, S. Contini, F. Bellezza, Geographical information systems in nuclear safeguards. In: Bhupendra Jasani and Gotthard Stein (eds), Commercial Satellite Imagery, Springer-Praxis, 2002, pp. 211-236.
2. KIM Hyun Tae, A Study of Geographic Database Application integrated with Nuclear Site Image Information Database, 45th Annual Meeting of Institute of Nuclear Materials Management, Renaissance Orlando Resort at Sea World, Orlando, Florida, USA, July 18-22, 2004.
3. Shaun Walker, et al., Professional DotNetNuke ASP.NET Portals, Wiley Publishing, Inc., 2005, pp. 2, 13, 202.
4. Patterns & Practices, Building Secure Microsoft ASP.NET Applications, Microsoft Press, 2003, Chapter 5 Intranet Security.
5. http://msdn.microsoft.com/msdnmag/issues/02/09/SecurityinNET/default.aspx, Don Box, The Security Infrastructure of the CLR Provides Evidence, Policy, Permissions, and Enforcement Services, in September 2002 issue of MSDN Magazine.
6. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secmod/html/secmod02.asp, J.D. Meier, et al., *Improving Web Application Security*, Microsoft Press, 2003, Chapter 10 Building Secure ASP.NET Pages and Controls.