**April, 28th, 2014**

**Application and System Development**

Written By

**Lance Westberg**

**Table of Content**

**Introduction ……………………………………………………………………………….**p3

**Capability Maturity Model (CMMI) ……………………………………………………** p4

**The Internet ………………………………………………………………………………** p5

**Polymorphism .…………………………………………………………………………..** p6

**Inheritance..………………………………………………………………………………..**p7

**Encapsulation …………………………………………………………………………....**p7

**Abstraction ……………………………………………………………………………….**p8

**Product Development Lifecycle ……………………………………………………....**p8

**Initiation Phase …………………………………………………………………………..**p9

**Acquisition/development ……………………………………………………………….**p9

Threats, Attacks, Vulnerabilities **…………………………………………………** p10

**Security Layers ………………………………………………………………………….**p11

**Security Frames and Dimensions ……………………………………………………**p12

**Application Development and Threat Identification ……………………………….**p13

Identified Vulnerabilities for the Session Table **…………………………………**p13

**Testing Phase for Security ……………………………………………………………..**p15

**Management ……………………………………………………………………………...**p15

**Training ……………………………………………………………………………………**p15

**Developing the Application …………………………………………………………….**p15

Strong names and assemblies **…………………………………………………..**p16

**Permissions ………………………………………………………………………………**p16

**Security and Cryptography ……………………………………………………………..**p17

Restrict what your code can do **…………………………………………………..**p18

Identify code **………………………………………………………………………..**p18

Access Modifiers **…………………………………………………………………..**p18

### Permissions Classes …………………………………………………………………….p19

Assert, Deny, and PermitOnly Methods **………………………………………...**p20

Policy **………………………………………………………………………………...**p20

Implementation **…………………………………………………………………….**p20

**Operation/Maintenance ………………………………………………………………….**p21

Certification / Accreditation **………………………………………………………..**p21

**Conclusion ………………………………………………………………………………...**p21

**References ………………………………………………………………………………...**p24

**INTRODUCTION -** There are many models to choose from when designing an application as well as a few international standards that one can use in the development software. Each one falls in an out of favor at sometime, and most firms use a combination of techniques when developing software. Moreover, the CISSP does not distinguish between system and software development life-cycles yet. It follows the high-level classical life-cycle structure, which I covered a lot in the last paper. System development life-cycle has a focus of operations and is a model that the IT department would follow. Software development life-cycle has a focus on design and programming and is a model that software engineers and coders would follow. I wanted to focus in this paper on a more granular down to the coding level view. The best life-cycle one can use in software development is to put a repeatable and predictable process in place – easy to follow for business people and programmers. I wanted this life-cycle model to include security that was built in at every level and in every feature from the inside out.

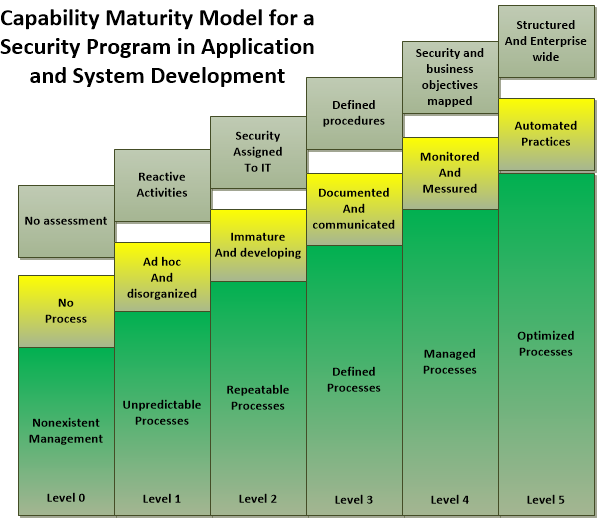
When one looks deeper into the subject of application and system development from a programmer point of view, programmers look at objects and roots levels at which a program will exists in. They look not just at techniques in building a secure application, but how the objects can be protected at the module level as they move through the systems and work within the environment/networks they operate.

The ISO/IEC 27034 is one of international standard I found that provides guidance to organizations looking to integrate security into their processes of application development; because building in security throughout the life-cycle of an application is now becoming a universally accepted practice.

Many security vulnerabilities in current information technology are the result of a piecemeal “bolt-on” approach to security. Many software projects relay on firewalls, anti-virus, encryption software or intrusion prevention systems…etc. This at one time might have been adequate to protect closed systems when there only were only few entrances and exits within the network system. These days you have systems that are being access from multiple devices, using software systems built with any number of development platforms from Android, Windows, Unix, to Java – each have their own strength and weakness when it comes to cyber-security. These day’s individual components within a software product are being hacked – object by object. Modern complex networks and services require that security be embedded in every component as a solution to the growing threat of network hacks and viruses.

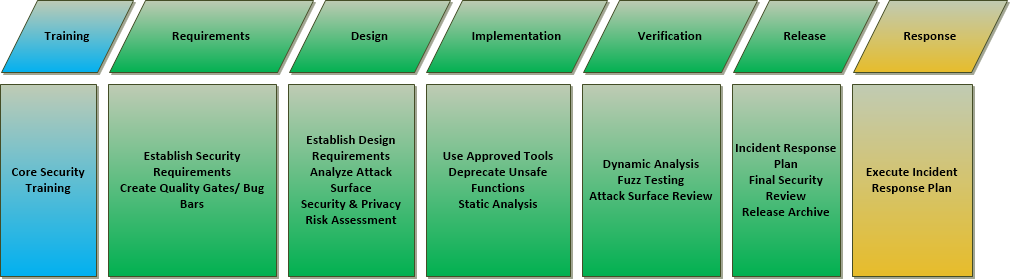
In addition, cloud computing also creates a new dynamic in that it allows millions of users to access a single instance of a software application, which amplifies the need for security down to the object level. Moreover, these new Cloud systems are built on virtual boxes housing multiple operating systems layered one-on-top of another. This will challenge risk managers because many systems lack transparency of design, especially when it comes to security. Cloud providers and organization’s that deliver software functionality as a service in the cloud will have to provide a secure environment, or the first major security breach or virus event could leave then hanging; as business after business leaves the cloud for a more secure company operated solution. The National Institute of Science and Technology (NIST) has defined three major delivery models cloud providers may focus on: software as a service (SaaS), platform as a service (PaaS, rapid application development), and infrastructure as a service (IaaS, basic operating system and storage).

**CAPABILITY MATURITY MODEL (CMMI) –** The (CMMI) model is a comprehensive integrated set of guidelines for developing products and software. It addresses the different phases of a software development life-cycle, including concept definition, requirements analysis, design, development, integration, installation, operations, and maintenance, and what should happen in each phase. It can be used to evaluate security engineering practices and identify ways to improve them. It can also be used by customers in the evaluation process of a software vendor. In the best of both worlds, both software vendors would use the model to help improve their processes and customers would use the model to assess the vendors’ practices.



* Initial Development process is ad hoc or even chaotic. The company does not use effective management procedures and plans. There is no assurance of consistency, and quality is unpredictable.
* Repeatable formal management structure, change control, and quality assurance are in place. The company can properly repeat processes throughout each project. The company does not have formal process models defined.
* Defined Formal procedures are in place that outline and define processes carried out in each project. The organization has a way to allow for quantitative process improvement.
* Managed the company has formal processes in place to collect and analyze quantitative data, and metrics are defined and fed into the process improvement program.
* Optimizing The company has budgeted and integrated plans for continuous process improvement.

**THE INTERNET -** As the Internet has grown, it has had a profound influence on the emergence of software components that are rapidly assembled into new business applications. The principles of service-oriented architecture (SOA), which views software as a combination of interoperable services, has led to mash-ups, or combinations of software components that can be altered and substituted at will. These components are often black boxes.A developer in today’s development environment may not know anything about the component other than its Application Programming Interface (API), and the events, methods and properties he can interact with to access the component. This can lead to a number of problems if the operating system in which the (API) lives in stops working or has been compromised. It can lead to non-secure software, because the developer did not know of the hidden vulnerabilities within the component leading to a devastating impact on his application and system. Microsoft and other big software makers have implemented a standard way to develop applications and system software over the last couple of years. The overview of this process is graphed below.



But this is more of a top down approach because application and system development of today is an increasingly complex monster. Yes most built software of today is built around the Object Oriented Programming platform (OOP). But what type of object is it, and who has access to it? Moreover, what controls are embedded within that object to help it protect itself from outside attacks? The OOP is based around the four pillars of polymorphism, inheritance, encapsulation, and abstraction. So if one understands how these core mythologies works it would go a long way in his/her understanding of how to protect them better as it runs within their application or network environment.

**POLYMORPHISM** - polymorphism derived from the Greek meaning “many shapes.” It refers to polymorphic functions which can be applied to arguments of different types, but which behave differently depending on the type of the argument to which they are applied. There is different type of polymorphism – ad hoc, Parametric, Subtype polymorphism, which is beyond the scope of this paper but put in simple terms means – “The ability of objects of one type to have one and the same interface, but different implementation of this interface.” Moreover, Polymorphism is declaring a uniform interface that isn't type aware, leaving implementation details to concrete types that implement the interface.

**INHERITANCE** – Inheritance enables you to inherit all of the basic functionality of an object that was already created. Example: You create a new word template to use throughout you writing project. It includes all the margins, line spacing, and font types you like. You don’t want to have to redo this form or template each time you create a new page for your document. So you built a template and just inherit it and all its properties each time you need a new page. Example below is how you would call your template in C# and Java. Each time to want to create a new page you just invoke your template. In word you would just click the file menu and click new page, but in the background this is what the program is doing as you invoke your template was called for example: “MyForm1.”

**public class frmMain : MyForm1**

Public and Private are special key words within C# inheritance that let you lock down your code. In this instance I am using “public,” so everyone can access this template object. Every object in object oriented programming has three basic components or things you can access which are: Methods, Events and Properties. You can lock down each or exclude each one, and even stop other objects from inheriting them. Programmers like using public, because it less hassle in coding. They will even create maintains channels for easier access when doing updates or changes if you let them. So if you don’t put in policies and procedures to control this type of coding. Programmers can and offend do forget to remove them before going live which it can be a backdoor way into your application if a hacker discovers them.

**ENCAPSULATION** **-** The process of hiding data within an object is called encapsulation. Encapsulation is one of the cornerstones of OOP. In the spirit of encapsulation which has key words one can use to restrict access - called access specifiers. Using an access specifier like private makes it much more difficult for someone to advertently change the value of a class property. This protection for the class properties exists because the private access specifier limits the scope of the properties to the class in which they are defined. Making the properties invisible to the outside world! Now consider the other alternative: making all the properties and methods public. If you do that, you just threw away all the benefits that encapsulation brings to the party. Not only can a hacker derive the class, he could gain access to the properties and methods of the base class. This would allow him/her to pass in variables to change password, or gain root access to the system.

**ABSTRACTION** **-** A class is a template used to describe an object. As such, a class is an abstraction or simplification of some object you observe in the real world. You can break a class down into two basic components: 1. those properties that describe the object, and 2. those methods, or actions, that you wish to associate with the object. Code reuse is one of the main advantages of object oriented programming (OOP). The more generic the object is hence “abstract,” the easier it is to reuse the object. In a sense, therefore, you might want to define your classes as the minimal abstraction necessary to describe an object in a way that fulfills your needs yet is generic enough to be reused throughout the application.

**PRODUCT DEVELOPMENT LIFECYCLE -** Every organization has its own customized version of the product development lifecycle. So I will just outline the basic components of each **SDLC**: Initiation - Acquisition/Development – Implementation – Operation/Maintenance – Disposal, but wait - where to put security and the risk associated from threats to your applications and systems. Moreover, risks that are not identified early in the requirements analysis phases can lead to a host of vulnerabilities that are only exposed later during the operational life of the product. It’s not only beneficial to identify key risk early and built for them. In the long run could save you and the company if your betting on customers staying with you after a major security breach happens.

**The SDLC basics**:

• Initiation Need for a new system is defined

• Acquisition/development New system is either created or purchased

• Implementation New system is installed into production environment

• Operation/maintenance System is used and cared for

• Disposal System is removed from production environment

**INITIATION PHASE** – The perception of a need within an organization, and how that new software product might help the enterprise. Since this phase establishes the business need for the product and a strategy of how to accomplish it. It defines the business case for proceeding to the next stage of the lifecycle. In this phase, initial risk assessment and cost-benefit analysis are performed on the basis of a preliminary list of customer requirements. The phase can also include security categorization, and risk assessment.

**ACQUISITION/DEVELOPMENT –** In this phase of development you produce a detailed market, customer and financial assessment enabling a business decision to be made, including commitment of resources and management. Design a conversion study of older systems, and statement of need along with functional requirements analysis, and risk/cost analysis. In addition, hopefully you had a change management system in place for a history of things you might be able to do better in the future. Because in this phase most of the security is created in the product; it is also when most of the vulnerabilities are created so having a record of the past might help you not repeat the same mistakes in this future application?

Now since most modern applications are built with the object-oriented-programming design methodologies. If you kept your change management records up to date. They can help you create standard protocols that all developers should follow, with policy guidelines on maintenance, security, and the most likely place errors have happened in the past. You can then expand on these records creating a vertical and horizontal picture of your applications and system network architecture.

The reason for this is that with most modern designs of today, programmers are building in systems to seek out other like minded systems to connect with automatically. Example: your Wi-Fi network, other applications and databases, it will try to access and by default start uploading person information to someone’s cloud network. Take for example depending on the version of windows you install. Windows will automatically create a user with super user rights by default, unless you change that default after installing. Hackers and virus writers knew this and took advantage of that fact to create viruses that would automatically check for these defaults. So having in place a good change management policy on installations and configurations within your applications and system network will help you spot these types installation flaws.

**Threats, Attacks, Vulnerabilities -** Security issues that need to be addressed in order to prevent both intentional threats as well as accidental threats before your new application goes live.

You can start by focusing on three essential questions:

1. What kind of protection is needed against what threats and vulnerabilities?

Example:

* Destruction of information and/or other resources
* Corruption or modification of information
* Theft, removal or loss of information and/or other resources
* Disclosure of information
* Interruption of services

1. What are the distinct types of network equipment and facility groupings that need to be protected?

Example:

* Types of networking and server equipment being installed?
* What types of Database Servers, and security built within?
* Defense in depth, DMZ zones, and firewalls.

1. What are the distinct types of network activities that need to be protected?

Example:

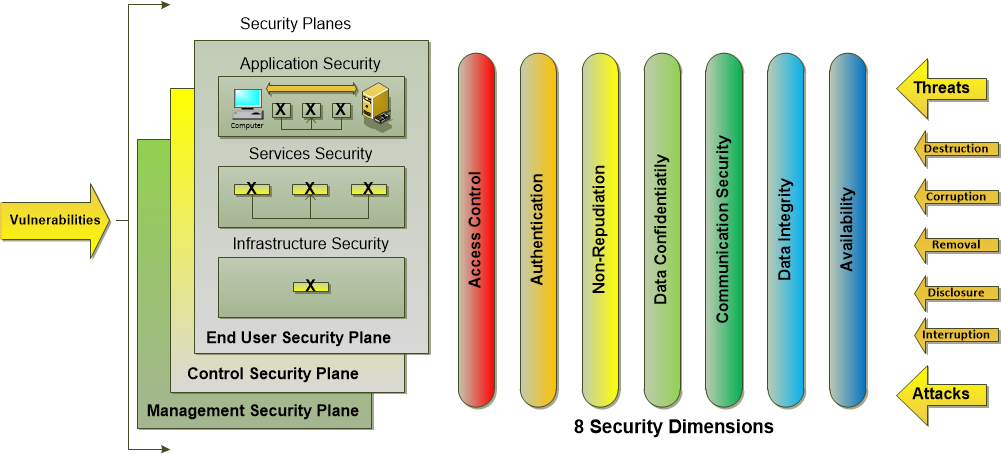
* T1 lines, DSL lines, VPN connections?
* Access points, of network input/outputs.
* Fiber-Optics?
* Cellar and Wi-Fi networks

The answers to these questions can only come after analyzing your applications development and security needs using a systematic approach of checklists. The graphic below illustrates and looks at the 8 security dimensions, and the end user security, control, and managed security. It was originally developed to help network operators understand what is needed to design, implement, and maintain a secure network. The framework is flexible enough so that it can be adapted to the ever-changing world of cyber-security and regulations - because as more and more identity attacks happen the government and public will start demanding change. The framework, depicted can be used to divide complex end-to-end network security logically into manageable system architectural components. This separation allows for a systematic approach to end-to-end security analysis to plan new security solutions as well as assess the security of the existing networks.

**SECURITY LAYERS -** The framework defines three security layers, which describe a hierarchy of network equipment and facility groupings. The infrastructure layer includes the basic buildingblocks used to create the network, services, and application systems.

• The services layer focuses on services that end users receive from networks.

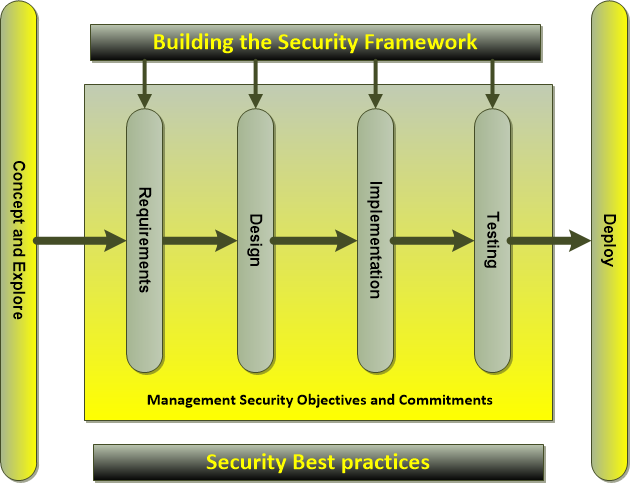
• The applications layer consists of network-based applications accessed by end users.

****

**SECURITY FRAMES AND DIMENSIONS –** Abovethe framework image outlines an application one must consider; the network the application will live in, management, control, and end-user activities. That also encompasses the eight dimensions:

* Access control protects against unauthorized use of resources.
* Authentication confirms the identities of each entity using the resource.
* Non-repudiation proves the origin of the data or identifies the cause of an event or action.
* Data confidentiality or data security ensures that data are not disclosed to unauthorized entities.
* Communication security allows information to flow only between authorized end points.
* Data integrity ensures the accuracy of data so they cannot be modified, deleted, created, or replicated\without authorization and provides an indication of unauthorized attempts to change data.
* Availability ensures that there is no denial of authorized access to network elements, stored information, information flows, services, and applications.
* Privacy provides for the protection of information that could be derived from the observation of network activities. (Note that privacy can be viewed as a goal realized by combining other security dimensions rather than a separate dimension.

Also it can help if you project manager is a trained programmer with a cyber security background.

****

To ensure that security objectives and commitments are integrated into product realization processes as you design you’re application. This gives a graphical presentation of the keys factors when integrating security into your overall development life-cycle.

In addition, it is essential that the product be deployed in a secure manner; otherwise many of the security controls integrated in the product may be easily circumvented. As I outline earlier know your vertical and horizontal applications and the network architecture they will live in, because the term software as a service (SAAS), can mean many things. Many applications of today have features that will automatically try to interconnect with other software products as its’ being installed, or it could have conflicts with a software product which will not be known till the application is turned on. Things can happen on the backend with regards to defaults, and networks protocols. Installing the application is one thing, how it’s configured to work within the environment and plays with others is another.

Example: One of the newest features of Google Android Kit-Kat 4.4 operating system for example; takes away your ability to save information to the micro-SD card. It will also automatically start saving your person information to the cloud the moment you turn on your phone or tablet. You have to root hack the device in order to save information to the SD card, but that also voids the warranty. Moreover, if you turn-off those default software setting options. If you download any new apps to your phone or tablet, some apps will automatically turn those features back-on as part of the installation process.

**APPLICATION DEVELOPMENT AND THREAT IDENTIFICATION**

|  |  |  |  |
| --- | --- | --- | --- |
|  | End User | Controls | Management |
| Identified Assets | Network Interfaces  User Authentication  Deep Screened Protocols  User Authentication  Traffic Passing Through | Reset Switch  Session Table  VLAN’s  Routing Protocol  Rules Update Traffic  State Information | Management GUI  Policies  Logs  Alarms  Configuration  Administrative Acct |

Identified Vulnerabilities for the Session Table

|  |  |
| --- | --- |
| **Dimension** | **Vulnerabilities** |
| Access Control | Unauthorized modification to the “sessions” can be made if a process person gains access to the firewall’s address space where the session table is stored. This can lead to possible root control of the server, and applications. |
| Authentication | “Any” process can modify the session table without any authentication |
| Non-repudiation | There is no definite record/proof of who made the changes to the sessions table and database. |
| Data Confidentiality | The contents of the session tale can be seen by an unauthorized entity |
| Communication Security | None, as the sessions table, which is difficult to detect |
| Data Integrity | Modification to sessions table, which is difficult to detect |
| Availability | Attacks like SYN flooding can quickly fill the session table, leaving no space for new sessions. |
| Privacy | Critical information about the sessions tables can be deduced by examining the sessions log. |

After identifying and prioritizing the threats, it is necessary to select appropriate security controls for mitigation. The eight security dimensions represent classes of actions that can be taken, or technologies that can be deployed to counter the threats and potential attacks present at each security layer. The architecture and design phase provides the foundation for the security considerations in the implementation phase. Controls introduced during the design phase are significantly cheaper to implement and maintain than those included during implementation. The key is to come up with standard protocols for the components you build with a security infrastructure that is built in, not bolted on for example.

* Authorization based on privilege management and compartmentalization.
* Access control by ensuring a secure default configuration, and validating user input. This means keeping programmer from building in nest eggs, and maintenance hooks for easy access to the core operations code of your application. Keeping secrets in implementation is extremely difficult and poses a security risk. So security by obscurity should be prevented in the design and implementation phases.
* Security considerations for web application development such as end-to-end session encryption and secure session management embedded.
* Data confidentiality using encryption and segmentation; ensuring that the operating system utilizes files systems with access controls.
* Non-repudiation or accountability can be implemented to some extent by code signing and code authorization. Logging should be part of the software design for accountability during and after development. Track Code reversions, and maintains updates are tracked recorded and handled correctly by change management protocols.
* Data integrity for stored software and data should be implemented using encryption algorithms and message integrity code mechanisms. Software design consideration should allow for the integrity of log files to be checked
* Availability using static code analysis tools for standards compliance and mitigating against common vulnerabilities like buffer overflows. Also, in code design, one must ensure that the code fails securely. In other words, if the system fails in any way, the security of the system should be maintained and the system should move into protective mode. Because a “system” does not just refer to a computer system. It is a general term that can represent a computer, network, or piece of software. Life-cycle approaches need to be applied to each type of system.
* Privacy by ensuring that control/signaling message headers are encrypted end to end such that the network activity cannot be observed for extracting information about the origin/destination of the transmission.

**TESTING PHASE FOR SECURITY -** Typically, functional testing involves verification of the documented implementation requirements. However, for security it is important to include strategic testing specific to configuration/protocol vulnerabilities, security policy compliance, and secure code testing. Moreover, don’t forget the environment that your application will live within. What will be the operating system and configuration on the network server? Is it a Window, Unix, or Linux servers..etc? Every server has some services turned on in order to install, and each will have different configuration options once installed, from memory management options, to security setting, and port assignment…etc?

**MANAGEMENT -** During the entire product/solution lifecycle, management should actively support security through clear direction, demonstrated commitment, dedicating resources, explicit assignment of team members, approval of roles and responsibilities, and acknowledgment of security responsibilities overall. This has to be a long term commitment not a project based do it once and let’s forget about it.

**TRAINING -** For an effective product/solution lifecycle, it is essential that resources are provided and security related roles are defined. In addition, the entire organization should be aware of the assigned team and their security responsibilities. Security awareness, education, and training activities should be suitable and applicable to each person’s defined role, responsibilities, and skills.

**DEVELOPING THE APPLICATION -** In developing your application using strong names can help ensure your application from outside attacks. A strong name consists of the assembly's identity—it’s a simple text name, version number, and culture information (if provided)—plus a public key and a digital signature. It is generated from an assembly file (the file that contains the assembly manifest, which in turn contains the names and hashes of all the files that make up the assembly), using the corresponding private key. An assembly is a set of classes that are designed to do certain things, for example; print functions, input/output of the keyboard functions, how your monitor works etc. Strong names guarantee name uniqueness by relying on unique key pairs. No one can generate the same assembly name that you can, because an assembly generated with one private key has a different name than an assembly generated with another private key.

* Strong names protect the version lineage of an assembly. A strong name can ensure that no one can produce a subsequent version of your assembly. Users can be sure that a version of the assembly they are loading comes from the same publisher that created the version the application was built with.
* Strong names provide a strong integrity check. Passing objects between the .NET Framework security checks guarantees that the contents of the assembly have not been changed since it was built. Note, however, that strong names in and of themselves do not imply a level of trust like that provided, for example, by a digital signature and supporting certificate. Java as well as a similar structure built into its language.

To minimize the security risks posed by malicious code, the runtime does not allow assemblies granted only partial trust to access strongly named assemblies. This restriction dramatically reduces the opportunity for malicious code to attack your system

Generally, the code most likely to be malicious is that which is loaded from remote locations over which you have little or no control (such as over the Internet). Under the default security policy of the .NET Framework, all code run from the local machine has full trust, whereas code loaded from remote locations has only partial trust. Stopping partially trusted code from accessing strongly named assemblies means that partially trusted code has no opportunity to use the features of the assembly for malicious purposes, and cannot probe and explore the assembly to find exploitable holes.

**PERMISSIONS -** The examples I will show are all done in C#, but can easily be converted to Java since C# and Java share over 70% of the same functionally and design, just the names are changed but carry out the same functionally. Each language has assemblies that are custom designed for security of classes and how to give permissions to objects, for example: permissions to access a method within a class that will let you write to a drive on the computer. The following example demonstrates how to use the IsGranted method to determine if the assembly has write permission to the directory C:\Data.

using System.Security; //.Net Security Class

using System.Security.Permissions; //.Net Security Class control write access.

namespace Apress.VisualCSharpRecipes.Chapter11

{

class MyWritePremission

{

// Define a variable to indicate whether the assembly has write

// access to the C:\Data folder.

private bool canWrite = false;

public MyWritePremission()

{

// Create and configure a FileIOPermission object that represents

// write access to the C:\Data folder.

FileIOPermission fileIOPerm =

new FileIOPermission(FileIOPermissionAccess.Write, @"C:\Data");

// Test if the current assembly has the specified permission.

canWrite = SecurityManager.IsGranted(fileIOPerm);

}

}

}

**SECURITY AND CRYPTOGRAPHY -** In this section I will look closely at object level programming and how to design an application with a structure that is not only secure, but is also functional. I have already outlined key elements like using strong names and assemblies. The different ways you can secure objects within an OOP design. Assemblies are made up of many objects that can interact on multiple levels within your code, from the root kernel of an operating system to the web interface. Programmers and application architects need to find middle ground between functionality of a program, security requirement, and user’s needs. An application’s routine interaction, global and local variables, input received from other programs, output fed to different applications and user’s interactions with the software can all have an effect making it very hard to predict the “what ifs.”

In .NET Framework code access security (CAS) is a resource constraint model designed to restrict the types of system resource that code can access and the types of privileged operation that the code can perform. These restrictions are independent of the user who calls the code or the user account under which the code runs. In java there is a simpler system call sandboxing, which will place outside code from accessing different parts of a system. But software architecture is a moving target, because as hardware and network structures change and improve, so too must application and system development designs.

 Restrict what your code can do **-** For example, if you develop an assembly that performs files I/O. You can use code access security to restrict your code's access to specific files or directories. This reduces the opportunities for an attacker to coerce your code to access arbitrary files. For example, you may only want your assembly to be called by other code developed by your organization. One way to do this is to use the public key component of an assembly's strong name to apply this kind of restriction. This helps prevent malicious code from calling your code.

 Identify code - To successfully administer code access security policy and restrict what code can do, the code must be identifiable. Code access security uses evidence such as an assembly's strong name or URL, or its computed hash to identify code (assemblies). Moreover, within the class itself you can use key words, like “Public,” and “Private,” to restrict access down to the methods themselves. In .Net these are called "Access Modifiers," which can also be used to restrict how other classes can access the class within an assembly and from outside of the assembly. The access modifier is an optional part of the member declaration that specifies what other parts of the program has access to the member.

Access Moditiers

**Example of how it is declared in code**:

Fields

Access Modifier Type Identifiers;

Methods

**AccessModifier** (private, public, protected, internal, protected internal)

Return Type MethodName()

{

....

}

“Private,” members are only accessible from within the class in which they are declared. “Public,” is pretty much what it means full access. The “protected,” access level is like the private access level, except that it allows classes derived from the class to access the member. “Internal,” access levels are visible to all the classes in the assembly, but not to classes outside the assembly. A protected internal member of a public class is visible to members of classes in the same assembly or to members of classes derived from that class. It is not visible to classes in other assemblies that are not derived from the class. Evidence is used by the .NET Framework security system to identify assemblies. Code access security policy uses evidence to help grant the right permissions to the right assembly. Location-related evidence includes:

* **Strong name**: This applies to assemblies with a strong name. Strong names are one way to digitally sign an assembly using a private key as I outline earlier.
* **Publisher**. The Authenticode signature; based on the digital certificate used to sign code, representing the development organization.

**Important**:  Publisher evidence (the Authenticode signature) is ignored by the ASP.NET host and therefore cannot be used to configure code access security policy for server-side Web applications. This evidence is primarily used by the Internet Explorer host.

* **Hash**: The assembly hash is based on the overall content of the assembly and allows you to detect a particular compilation of a piece of code, independent of version number. This is useful for detecting when third party assemblies change (without an updated version number) and you have not tested and authorized their use for your build.

PERMISSIONS CLASSES - Permissions represent the rights for code to access a secured resource or perform a privileged operation*.* The .NET Framework providescodeaccesspermissions *and* codeidentitypermissions. Code access permissions encapsulate the ability to access a particular resource or perform a privileged operation. Code identity permissions are used to restrict access to code, based on an aspect of the calling code's identity such as its strong name. Your code is granted permissions by code access security policy that is configured by the administrator. An assembly can also affect the set of permissions that it is ultimately granted by using permission requests. Together, code access security policy and permission requests determine what your code can do.

### Assert, deny, and permitonly methods - Code access permission classes support the ****Assert****, ****Deny****, and ****PermitOnly**** methods. You can use these methods to alter the behavior of a permission demand stack walk. They are referred to as stack walk modifiers. Like the Access modifiers within a class, these restrict how the class is accessed itself.

### Policy - Code access security policy is configured by administrators and it determines the permissions granted to assemblies. Policy can be established at four levels:

* **Enterprise**. Used to apply Enterprise-wide policy.
* **Machine**. Used to apply machine-level policy.
* **User**. Used to apply per user policy.
* **Application** **Domain**. Used to configure the application domain into which an assembly is loaded.

Policy settings are maintained in XML configuration files that set on the server in a secure folder structure. Example:

<?xml version="1.0"?>

<configuration>

<runtime>

<**LegacySecurityPolicy enabled="true"/>**

</runtime>

</configuration>

Implementation **-** For any enterprise level security architecture to be successful in application and system development, the following items must be understood and followed: strategic alignment, process enhancement, business enablement, and security effectiveness. Strategic alignment means the business drivers and the regulatory and legal requirements are being met by the security and application architecture. Security efforts must provide and support an environment that allows the company survives if unforeseen events happen. Business enablement requirement of the application and system security architecture exist for the solo purpose to help the company not only be secured, but to make money. This is done my building robust application and systems that protects themselves when attacked. So the controls and policies need to be in place in order to have security effectiveness.

**OPERATION/MAINTENANCE –** So you built the application, what next? Performance measurement and inspection must be a living methodology. In the near future, companies, governments, and individuals will start demanding certification and accreditation of your software. Governments are already asking questions of corporations who want to do business with the government, and consumers given some of the high profile identity thefts over the last couple of years. Consumers are now asking companies - just what are you doing with my personal information? Who are you sharing that data with? What other purposes are you using it for?

Certification is the comprehensive technical evaluation of the security components and their compliance for the purpose of accreditation. A certification process may use safeguard evaluation, risk analysis, verification, testing, and auditing techniques to assess the appropriateness of a specific system.

Accreditation is the formal acceptance of the adequacy of a system’s overall security and functionality by management. The certification information is presented to management, or the responsible body, and it is up to management to ask questions, review the reports and findings, and decide whether to accept the product and whether any corrective action needs to take place.

**CONCLUSION -** The need for security requirements analysis, identification, and prioritization of the vulnerabilities within applications is a growing function of any software company. Checklists and policies need to be created along with graphical models and flow charts making it easy for business people and developers to follow at each step of the product life-cycle. These will help reduce the number of coding errors and help focus the work on the task of building a secure application. We know from studies that key errors in most applications happen in the development and implementation phases. It's a fact that diagrams and flow charts will not only help spot coding errors, but will streamline the testing phase and troubleshooting process. These types of documents will also help in building best practices for other tasks like, configuration, patch management, disposal, and ongoing maintenance.

The ISO/IEC 18028-2 and ITU-T X.805 documents are important to security because they help uncover vulnerabilities in existing products, networks, and solutions, and they offer a framework that can be used as a basis for developing a common methodology for defining robust security deployment programs for converged, multivendor next-generation networks. But they are not the only ones, because the National Institute of Standards and Technology (NIST) has been putting out major papers on everything from best practices for picking an encryption package to designing cyber-security frameworks.

As an example as to just how hard it is to build in security I have created a diagram of the system I work on at Infor/Lawson. It’s our ERP system called “Landmark,” within this diagram I have numbers just how and average user would access the system. This ERP system – “LandMark,” it’s made up of many different computer languages, and processes. We use 3rd party products like Bouncy Castle for encryption to Perl scripting to run utilities, and updates. We use IBM’s WebSphere nick named "WAS," This is the web server which handles all the traffic in and out to the World-Wide-Web as well as all the user interfaces to Landmark through Rich Client. Rich Client is a web portal designed to interact with the different applications within LandMark. For example: The Human Resource application, Procurement application, Financial Systems, Manufacturing systems, and customer service apps.

The next layer is the core technology environment which includes the various databases to custom servers to handle event management, batch services, COBOL code and server utilities. There are two major databases depending on the setup, one called Lawson, and the other called Logan, each track and record user actions and transactions within the system. There is also a database for security, which is kept in sync using what is called Infor Security Service (ISS).

The next layer is called the "Grid," it’s a custom management application that helps manage links to all the different servers running within LandMark. It also controls how memory is handled and configured. Within the GRID are programs called Async, and Queue that track errors within the different servers. This Grid, is a virtual server within a server, and can be scaled to handle workloads from hundreds to thousands of people hitting the servers at any given time. LandMark is a very complex system to build and maintain. It has many parts that make it hard to protect. If not setup and configured correctly, hacker can gain access to a company’s financial records or its procurement systems...etc. That is why it's critical that when building systems like these, developers and project managers have to be skilled in multiple disciplines from database design, network/application development and security architecture.

In this paper I was only able to cover a small part at what goes into creating applications like “LandMark.” So by including this diagram you can get some idea the tasks of programming, maintaining, and building a system like this, and trying to make it a secure platform. It’s a work in process.

Lance Westberg

**REFERENCES**

Harris, Shon. (2013). All-In-One CISSP Exam Guide Sixth Edition

Publisher: McGraw Hill

NIST Publication: Improving Critical Infrastructure Cybersecurity Executive Order 13636

Retrieved from:

<http://www.whitehouse.gov/the-press-office/2013/02/12/executive-order-improving-critical-infrastructure-cybersecurity>

NIST Special Publication 800-38G: Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving

Encryption: Retrieved from:

<http://csrc.nist.gov/publications/drafts/800-38g/sp800_38g_draft.pdf>

Information technology -- Security techniques -- IT network security -- Network security architecture ISO/IEC 18028-2; Retrieved from:

<http://www.iso.org/iso/catalogue_detail.htm?csnumber=40009>

Zachary Zeltsan, ITU-T Recommendation X.805 and its application to NGN

Retrieved from:

<https://www.itu.int/ITU-T/worksem/ngn/200505/presentations/s5-zelstan.pdf>

W. Timothy Polk, Donna F. Dodson, NIST Special Publication 800-78-4, Cryptographic Algorithms and Key Sizes for Personal Identity Verification

Retrieved from:

<http://csrc.nist.gov/publications/drafts/800-78-4/sp800_78-4_draft.pdf>

W. Timothy Polk, Hildegard Ferraiolo, NISTIR 7863; Cardholder Authentication for the

PIV Digital Signature Key

Retrieved from:

<http://csrc.nist.gov/publications/drafts/nistir-7863/draft_nistir_7863.pdf>

Martin, J. (1991). Rapid application development. Macmillan Publishing Co., Inc.

Retrieved from: <http://dl.acm.org/citation.cfm?id=103275>

Gellersen, H. W., & Gaedke, M. (1999). Object-oriented web application development. IEEE Internet Computing, 3(1), 60-68. Retrieved from:

<http://dl.acm.org/citation.cfm?id=613368>

Zhu, H., & Zhou, M. (2003, October). Methodology first and language second: a way to teach object-oriented programming. In Companion of the 18th annual ACM SIGPLANconference on Object-oriented programming, systems, languages, and applications (pp. 140-147). ACM. Retrieved from: <http://dl.acm.org/citation.cfm?id=949389>