



.NET Framework Security

P. Libro - F. Pellegrini

Indice

- Introduzione .Net Platform e Framework
 - Design goals
 - Obiettivi
- Versioni del .Net Framework
- Componenti del Framework
 - CLR
 - BCL
- Architettura del .Net Framework
 - Assemblies
 - Metadata
 - Class Library
 - Security
- .NET Security: CAS
 - Struttura
 - Come lavora
 - Permessi
- Policy: Gestione
 - Dichiarative
 - Imperative
- Autenticazione e Autorizzazione
- Access Control List
- Reflection & Obfuscation
- Conclusioni

Le versioni di .Net Framework e di J2SE a cui fa riferimento questa presentazione sono rispettivamente la 2.0 e la 1.4.

Microsoft .NET Platform 1/2

Introduzione

Quando

Annunciata nel luglio 2000 da Microsoft insieme al C#

Cosa

Framework di sviluppo che fornisce una nuova Interfaccia di Programmazione ai servizi e alle APIs dei classici S.O. Windows, con l'intento di raccordare insieme un numero disparato di nuove tecnologie emerse negli ultimi anni tra cui:

- servizi di componenti COM+
- ambiente di sviluppo Web ASP.NET
- XML e programmazione OO
- protocolli web services SOAP,WSDL,UDDI
- focalizzazione su Internet e sulla Windows DNA Architecture (distributed business applications using Internet Technologies)

Microsoft .NET Platform 2/2

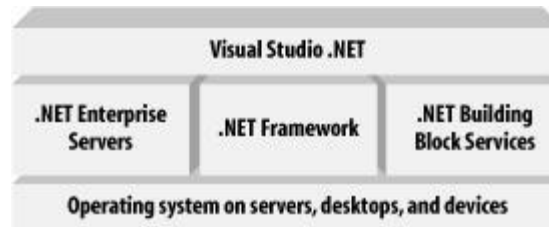
Introduzione

Strategia

Rendere il software utilizzabile come un servizio

Inoltre punta a

- Distributed computing
- Componentization
- Enterprise services
- Web paradigm shifts
- (Application of) Maturity of IT industry (lessons learned)



Microsoft .NET Framework

Introduzione: **Design Goals**

	COM	.NET
Components infrastructure	√(rewrite)	√√ (Assembly)
Language integration	X	√ (CTS)
Application interoperation across the Web	X	√ (SOAP)
Simplified development and deployment	X	√ (DII-Manifest)
Improved reliability	-	√ (Type Safety)
Greater security	-	√ (CAS)

Microsoft .NET Framework

Introduzione

Cosa

Fornisce un'astrazione tale da permettere la specializzazione del codice e delle funzioni d'uso generiche.

Dove

Componente integrante nei sistemi operativi Microsoft Windows (e ad oggi installabile anche su sistemi linux-based, MAC attraverso il Progetto MONO(ver. 2.4) : <http://www.mono-project.com>).

Permette

La generazione e l'esecuzione di applicazioni di nuova generazione, Web Services XML, RIA (Rich Internet Applications) etc...

Microsoft .NET Framework

Introduzione: **Obiettivi 1/2**

OO

Fornire un ambiente di programmazione orientato agli oggetti che supporti :

- esecuzione locale
- esecuzione locale e distribuzione su Internet
- esecuzione in modalità remota

Prestazioni - 1

Fornire un ambiente di esecuzione del codice che minimizzi:

- distribuzione del codice
- conflitti di versione

Prestazioni - 2

Fornire un ambiente di esecuzione del codice che elimini i problemi di prestazioni

- degli ambienti basati su script
- degli ambienti interpretati

Microsoft .NET Framework

Introduzione : **Obiettivi 2/2**

Sicurezza

Fornire un ambiente che permetta un'esecuzione sicura di:

- codice creato da produttori sconosciuti
- codice creato da produttori semi-trusted

Varietà

Rendere diversificata l'esperienza dello sviluppatore supportando tipi molto differenti di applicazioni basate su Windows, applicazioni basate sul Web, dispositivi mobili o embedded (.NET MicroFramework)

Integrazione

Generare tutte le comunicazioni in base agli standard industriali per assicurare che il codice basato su .NET Framework possa integrarsi con qualsiasi altro codice.

Microsoft .NET Framework

Componenti essenziali

CLR (Common Language Runtime) Motore di Esecuzione

Implementazione dello standard Microsoft *Common Language Infrastructure* (CLI)
Esegue il codice e fornisce i servizi che semplificano il processo di sviluppo

CLI

COS'E'

- specifica aperta (pubblicata sotto ECMA-335 e ISO/IEC 23271)

COSA FA

- descrive il codice eseguibile e l'ambiente in cui dovrà essere eseguito il codice (runtime)
- si trova anche alla base del progetto intrapreso da Mono e Portable.Net

✓ → è possibile utilizzare codice su diverse architetture senza doverlo riscrivere.

Class Library

Insieme di classi, interfacce, tipi-valore. Fornisce l'accesso alle funzionalità del sistema ed è stata progettata per essere la base per lo sviluppo di applicazioni, componenti e controlli .NET Framework.

Microsoft .NET Framework

CLR 1/2

Base

Esegue codice particolare *bytecode* denominato (CIL) (o MSIL) *Common Intermediate Language*

Compile time

Il compilatore .Net provvede a convertire il codice del programma in codice CIL.

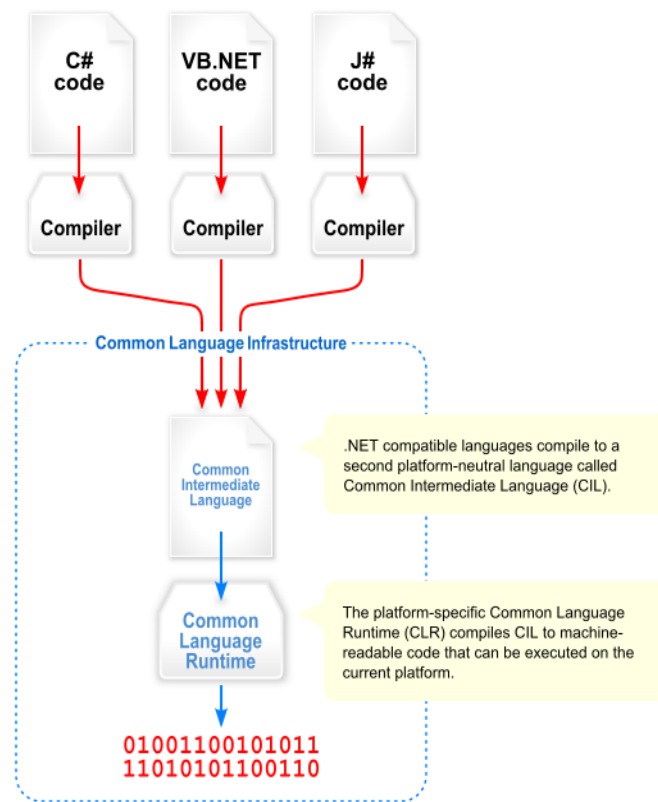
Runtime

Il compilatore j-i-t esegue la conversione del CIL nel codice nativo eseguibile dal SO.

Peculiarità

Sviluppo software utilizzando il proprio linguaggio preferito (.NET compatibili) :C#, VB.NET, J#, COBOL.NET, etc....

E' possibile eseguire la produzione del codice nativo in una fase precedente a quella di esecuzione (Runtime).



Microsoft .NET Framework

CLR 2/2

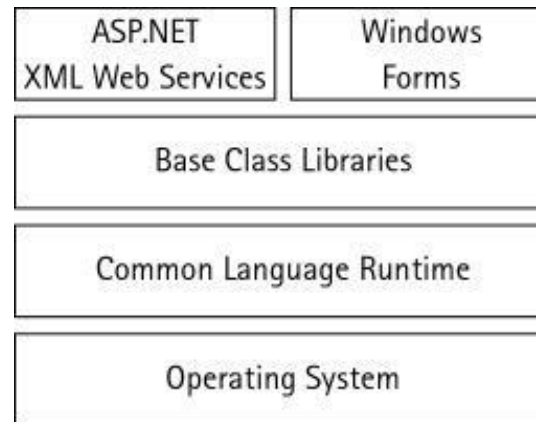
Osservazioni

- ✓ Utilizzo del CLR permette agli sviluppatori di ignorare tutti i dettagli specifici della CPU su cui dovrà essere eseguita l'applicazione.
- X Non tutti i tipi di applicazioni scritte in Microsoft .NET possono essere eseguite al di fuori dei suoi SO (ad esempio applicazioni Windows Form).
- ✓ Mono ha implementato CLI (Common Language Infrastructure) utilizzato per eseguire applicazioni .NET compilate.

Ulteriori compiti svolti

- Gestione della memoria (Memory management)
- Gestione dei thread (Thread management)
- Gestione delle eccezioni (Exception Management)
- Garbage Collection
- Sicurezza (approfondito più avanti nel corso di questa presentazione)

Microsoft .NET Framework Overview



CLR e JVM

E' l'equivalente della JVM per Java.

Alcune differenze

CLR Supporta qualsiasi linguaggio (a differenza di JVM ufficialmente) che può essere rappresentato in CIL è non è mai interpretato a differenza di Java.

Il codice Java è eseguito su qualsiasi piattaforma con una JVM mentre il codice .NET solo su piattaforme che supportano il CLR.

Microsoft .NET Framework

Versioni 1/2

Start Development

Fine anni '90 con il progetto
Next Generation Windows
Services (NGWS).

First Version

Durante l'inizio dell'anno
2002 è rilasciata la prima
versione ufficiale.

Versione	Ver. Number	Rilascio	Ver. Visual Studio	Default in Windows
1.0	1.0.3705.0	13/02/2002	Visual Studio .Net	
1.1	1.1.4322.573	24/04/2003	Visual Studio .Net 2003	Windows Server 2003
2.0	2.0.50727.42	07/11/2005	Visual Studio .Net 2005	
3.0	3.0.4506.30	06/11/2006	Visual Studio .Net 2008	Windows Vista, Windows Server 2008
3.5	3.5.21022.8	19/11/2007	Visual Studio 2008	Windows 7
4.0 (Beta 1)	4.0.1534	18/05/2009	Visual Studio 2010 Beta1	

Nel seguito di questo lavoro faremo riferimento alla versione 2.0 del .NET Framework. La Versione 3.5 è costituita una serie di estensioni che si appoggiano sulla versione 2.0 del Framework.

Microsoft .NET Framework

Versioni 2/ 2

Versione	Ver. Number	Rilascio	Ver. Visual Studio	Default in Windows
1.0	1.0.3705.0	13/02/2002	Visual Studio .Net	
1.1	1.1.4322.573	24/04/2003	Visual Studio .Net 2003	Windows Server 2003
2.0	2.0.50727.42	07/11/2005	Visual Studio .Net 2005	
3.0	3.0.4506.30	06/11/2006	Visual Studio .Net 2008	Windows Vista, Windows Server 2008
3.5	3.5.21022.8	19/11/2007	Visual Studio 2008	Windows 7
4.0 (Beta 1)	4.0.1534	18/05/2009	Visual Studio 2010 Beta1	

Altre note sulla tabella precedente

La versione del .Net Framework non combacia 'strettamente' con il number version. Dopo la prima versione sono stati rilasciati service pack per le diversi edizioni, alcuni anche anche corposi.

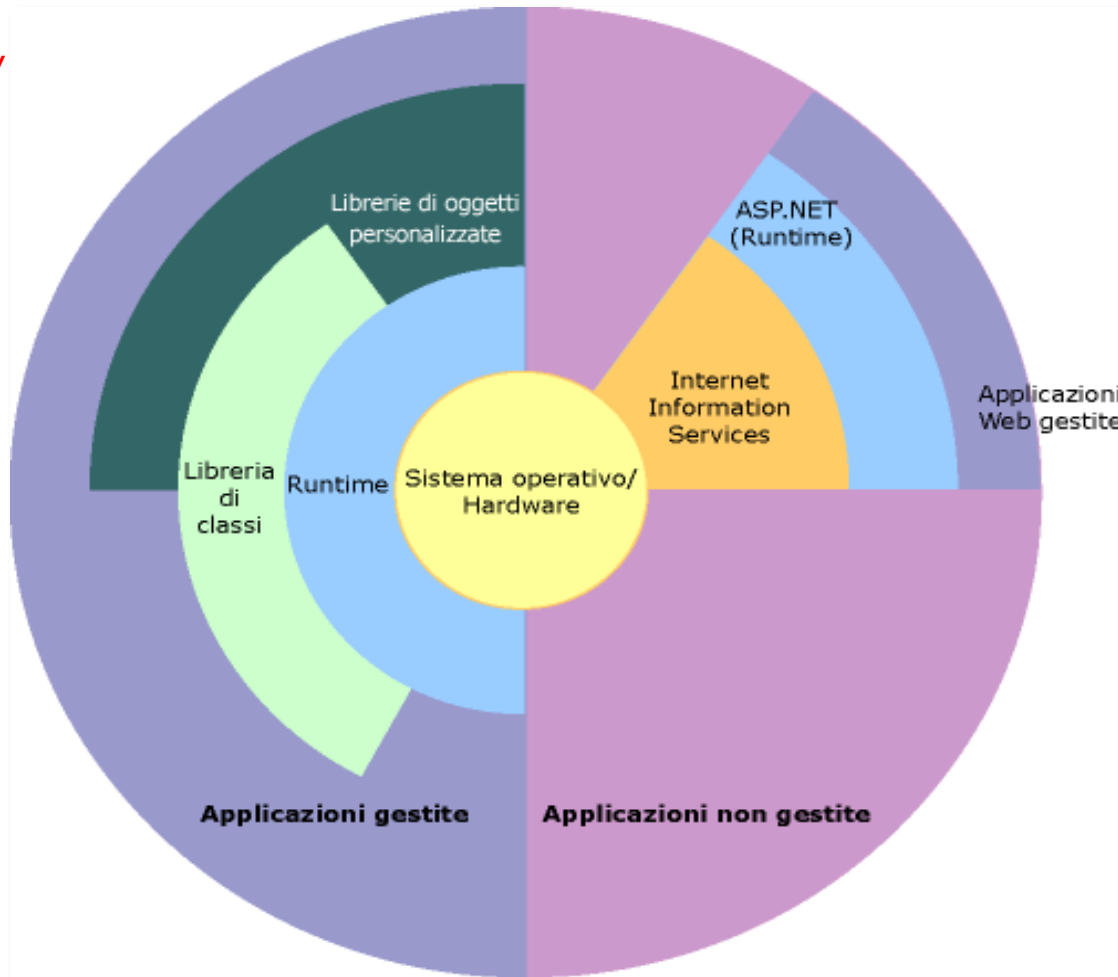
Ad oggi è stata rilasciata il 18 maggio 2009 la versione 4.0 Beta 1 del Framework e la versione 2010 Beta 1 di Visual Studio.

Microsoft .NET Framework Architettura

Componenti principali

- *Common Language Infrastructure* ✓
- *Assemblies*
- *Metadata*
- *Security*
- *Base Class Library*

Metadata as sometimes seen
 standard and available at all
 times. The framework
 assemblies are the blocks
 that constitute the
 runtime. Each application
 contains one or more
 assemblies. An assembly
 can contain one or more
 files (e.g., .dll, .exe, .resources,
 .xml, etc.).



Microsoft .NET Framework Architettura : **Assembly 1/3**

Definizioni

- Rappresenta il costituente fondamentale di un'applicazione .NET.
 - È una collezione di funzionalità sviluppate e distribuite come una singola unità applicativa (uno o più file).
 - Tutti i tipi e le risorse gestiti sono contrassegnati o come accessibili solo all'interno della propria unità implementativa o come esportati per essere utilizzati da codice al di fuori di tale unità.
 - Unità funzionale per la condivisione ed il riuso di codice nel Common Language Runtime. E' l'equivalente dei file JAR (Java Archive) in Java.
 - Un assembly può essere formato da uno o più file. I file di codice sono chiamati moduli, un assembly può contenere uno o più moduli ed è inoltre possibile utilizzare diversi linguaggi per creare ogni modulo.
- ✓ E' teoricamente possibile utilizzare diversi linguaggi per creare un assembly.
- X Microsoft Visual Studio non supporta l'utilizzo di differenti linguaggi per lo sviluppo dello stesso assembly.

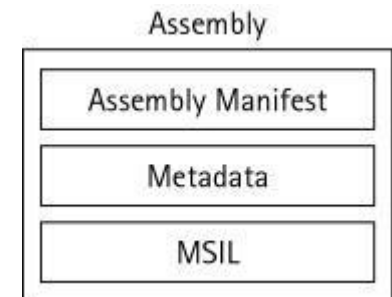
Microsoft .NET Framework

Architettura : Assembly 2/3

Uso

Una libreria di codice parzialmente compilata utilizzabile per:

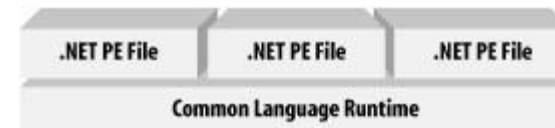
- Deployment
- Versioning
- Sicurezza applicazioni



Composto in .Net Win

da un PE (*portable executable*) → formato file per eseguibili, object code e DLL per SO Win a 32- e 64-bit (EXE o DLL)

Il termine *portable* è riferito alla versatilità del formato in numerosi ambienti dell'architettura software del SO.



Il formato PE

Incapsula tutte le informazioni necessarie al loader di un sistema operativo Win per gestire codice eseguibile *wrappato*.

Include riferimenti per il linking dinamico delle librerie, API, gestione delle risorse e i dati thread-local storage.

Microsoft .NET Framework Architettura : **Manifest 3/3**

Manifest

- Gli assembly si **autodescrivono** tramite il proprio **manifest**, che costituisce una parte integrante di ogni assembly.

Compiti del Manifest

- Stabilisce l'identità dell'assembly in termini di nome, versione, livello di condivisione tra applicazioni diverse, firma digitale.
- Definisce quali file (nome e file hash) costituiscono l'implementazione dell'assembly.
- Specifica le dipendenze in fase di compilazione da altri assembly.
- Specifica i tipi e le risorse che costituiscono l'assembly, inclusi quelli che vengono esportati dall'assembly.
- Specifica l'insieme dei permessi necessari al corretto funzionamento dell'assembly.

Il Manifest in breve descrive l'assembly stesso, la sua versione, le sue relazioni con gli altri etc..

Microsoft .NET Framework Architettura : **Metadata**

COS'E'

- Strutture dati incorporate all'interno del codice CIL
- Dati che descrivono tutte le classi e i membri definiti in un assembly e i membri delle classi che vengono richiamate all'interno dello stesso

Method metadata contengono

- la descrizione completa del metodo
- la classe contenente il metodo di tipo di ritorno e tutti i parametri (con relativi tipi) passati al metodo

Generati

dal compilatore .Net e immagazzinati (stored) nell'assembly

Creati

Dagli sviluppatori i quali possono aggiungere metadati al proprio codice attraverso attributi (custom attributes e pseudo-attributes).

Esempio :

```
[Description ("Questa funzione gestisce il codice per il Load della Windows Form")]  
private void Form1_Load(object sender, EventArgs e) {...}
```

Microsoft .NET Framework

Architettura : **Base Class Library**

COS'E'

Libreria standard disponibile per tutti i linguaggi utilizzati dal .Net Framework.

INCLUDE funzioni comuni per

- leggere e scrivere file
- eseguire il rendering grafico
- eseguire interazione con DB
- manipolazione documenti XML
- altre facilities che agevolano il lavoro giornaliero dello sviluppatore

Note

Molto più grande rispetto ad altri linguaggi standard come ad esempio C++. Spesso si fa riferimento erroneamente alla BCL con FCL (Framework Class Library) la quale è invece un superinsieme incluso nei namespaces Microsoft che la contiene.

Microsoft .NET Framework Architettura : **CTS**

Come già detto

Oltre alla indipendenza nell'utilizzo di linguaggi è supportata anche l'integrazione di linguaggi.

E' permessa

l'ereditarietà da classi, catch exceptions e sfruttamento del polimorfismo tra linguaggi diversi.

Grazie a

delle specifiche chiamate *Common Type System* (CTS) alle quali tutte le componenti .NET devono obbedire. In .NET ogni entità rappresenta un oggetto di una specifica classe che deriva dalla classe root chiamata *System.Object*.

CTS supporta e implementa i concetti generali di classes, interfaces, delegates (which support callbacks), reference types, and value types. Inoltre .NET include una *Common Language Specification* (CLS) la quale detta una serie di regole di base richieste per l'integrazione fra diversi linguaggi. La CLS determina I requisiti minimi per essere un linguaggio .NET.

I compilatori che sono conformi alla CLS creano oggetti che interoperano l'uno con l'altro, inoltre l'intera FCL (e quindi non solo la BCL) può essere utilizzata da qualsiasi linguaggio conforme.

Microsoft .NET Framework Security-AppDomain 1/4

Problema

Gli sviluppatori hanno bisogno di utilizzare codice in assembly esterni all'applicazione ma ciò può :

- Causare uso inefficiente delle risorse
- Introdurre vulnerabilità nel sistema

Soluzione .NET

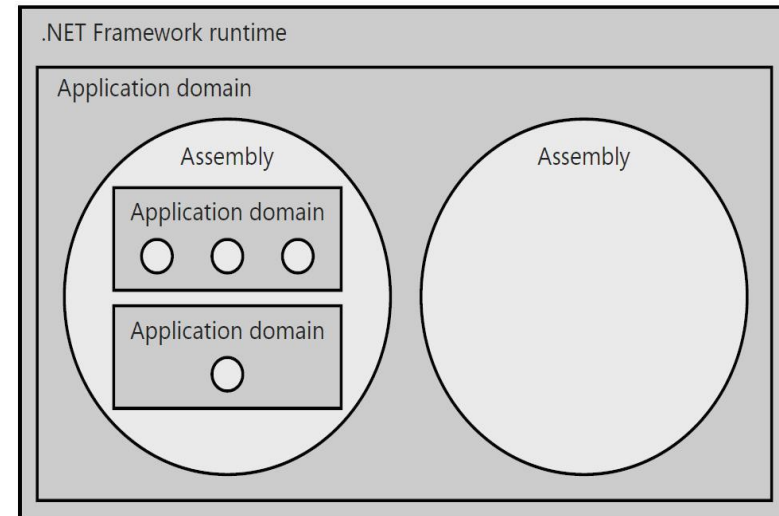
Creare un *AppDomain* e chiamare l'assembly all'interno di un ambiente protetto.

AppDomain

Contenitore logico che permette a più assemblies di girare all'interno di un singolo processo, senza che un assembly abbia accesso diretto alla memoria gestita da altri assembly e senza che il processo venga influenzato dal suo comportamento.

AppDomain vs Processi

Offrono molte delle funzioni di un processo come spazi di memoria separati ed accesso alle risorse ma sono più efficienti dei processi dato che più assemblies possono essere eseguiti in più AppDomain senza l'overhead di lanciare diversi processi.



Microsoft .NET Framework

Security-AppDomain 2/4

Affidabilità

Nati per essere utilizzati per isolare tasks cause possibili di instabilità o terminazione di processi:

- se lo stato di AppDomain che sta eseguendo un task diventa instabile → esso può essere scaricato senza effetti 'collaterali' per il processo.
- Utilizzabile anche quando un processo deve essere eseguito per un lungo periodo di tempo senza essere riavviato (viene fermato solo l'AppDomain).
- Un ulteriore utilizzo è quello di isolare tasks che non dovrebbero condividere dati.

Efficienza

- Se un assembly è caricato all'interno dell'AppDomain di default, tale assembly non può essere scaricato dalla memoria mentre il processo è in esecuzione (**protezione del processo**).
- Se durante l'esecuzione dell'applicazione viene caricato ed aperto un secondo assembly, questo viene scaricato solo quando il corrispondente AppDomain è scaricato (**protezione del linkaggio**).

Uso adatto per tutti quei processi che fanno un largo uso di librerie a collegamento dinamico (DLL).

Microsoft .NET Framework Security-AppDomain 3/4

Uno o più AppDomain caricati da un'applicazione possono essere configurati in modo tale da creare dei veri e propri ambienti personalizzati per l'esecuzione degli assemblies. La più importante applicazione della personalizzazione è quella di restringere i privilegi per ridurre il rischio associato alle vulnerabilità di sicurezza, infatti AppDomain configurato in modo corretto → fornisce un'unità di isolamento e limita i danni dovuti ad attaccanti che sfruttino qualche exploit dell'assembly.

Scenario

*Supponiamo di acquistare un componente (assembly) di terze parti per interagire con un DB. Se questo ha delle vulnerabilità e noi ci fidiamo dell'assembly senza caricarlo in un opportuno AppDomain (comunque senza particolari restrizioni), un attaccante potrebbe approfittarne per compromettere l'applicazione, e a secondo dei casi, anche il sistema. In questo caso, il problema è di chi ha sviluppato l'applicazione che non si è preoccupato di caricare gli assembly con le opportune restrizioni. Con privilegi limitati (es. per scrittura file o db) la richiesta malevola verrebbe rifiutata. La tecnica vista di protezione tramite esecuzione dell'assembly con privilegi limitati è chiamata **Defense-in-depth**. Lo scopo principale di DID è quello di fornire più livelli di protezione cercando di proteggersi da un'eventuale vulnerabilità. Codice esterno infatti può presentare vulnerabilità che non possiamo controllare ma prevenire.*

Microsoft .NET Framework Security-AppDomain 4/4

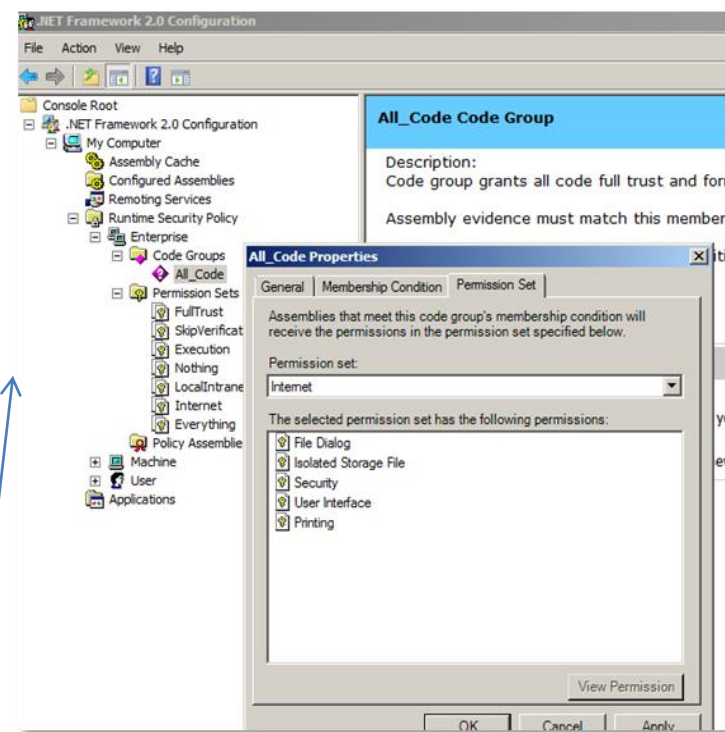
Quando eseguiamo un assembly abbiamo a disposizione le informazioni che il runtime raccoglie su di esso per determinare a quale *code group* appartiene.

Code Group a sua volta determina i privilegi che possono essere concessi all'assembly.

Un code group è un raggruppamento logico di codice che ha specifiche condizioni di *membership*.

Ad ogni *code group* sono associati un insieme di permessi che sono valutati durante il *grant* dei permessi.

Gli amministratori di sistema possono configurare le policy di sicurezza attraverso un apposito *tool* di gestione che permette di associare ai *code group* l'insieme di permessi desiderati.



Microsoft .NET Framework Security

CAS, Code Access Security

- CAS è un sistema di sicurezza che permette agli amministratori e sviluppatori di controllare le autorizzazioni delle applicazioni. Ad esempio si può permettere o negare l'accesso in lettura e/o scrittura ad un file o ad una chiave/valore di registro.
- CAS è diverso da RBS (Role Based Security, sicurezza basata sui ruoli) del sistema operativo. Con CAS è possibile restringere l'accesso a molte risorse, come :
 - FileSystem
 - Registro di configurazione
 - Stampanti
 - Log degli eventiDi solito gestibili anche con RBS
- CAS può essere applicato solo ad applicazioni .NET che utilizzano *managed code*. Per le applicazioni che utilizzano *unmanaged code* possono essere applicate solo restrizioni basate su RBS

Microsoft .NET Framework Security

CAS, Code Access Security

Se ad un assembly vengono applicate restrizioni CAS, l'assembly viene denominato *partially trusted* (parzialmente trusted). Questo tipo di assembly, ogni qual volta ha bisogno di accedere ad una risorsa viene verificato se ha le autorizzazioni per accedervi.

Gli assembly a cui non sono applicate restrizioni CAS, sono detti *trusted (fully trusted)* e come il codice non gestito possono accedere a qualsiasi risorsa di sistema a cui l'utente che esegue l'assembly ha il permesso di accedere (basandosi su RBS)

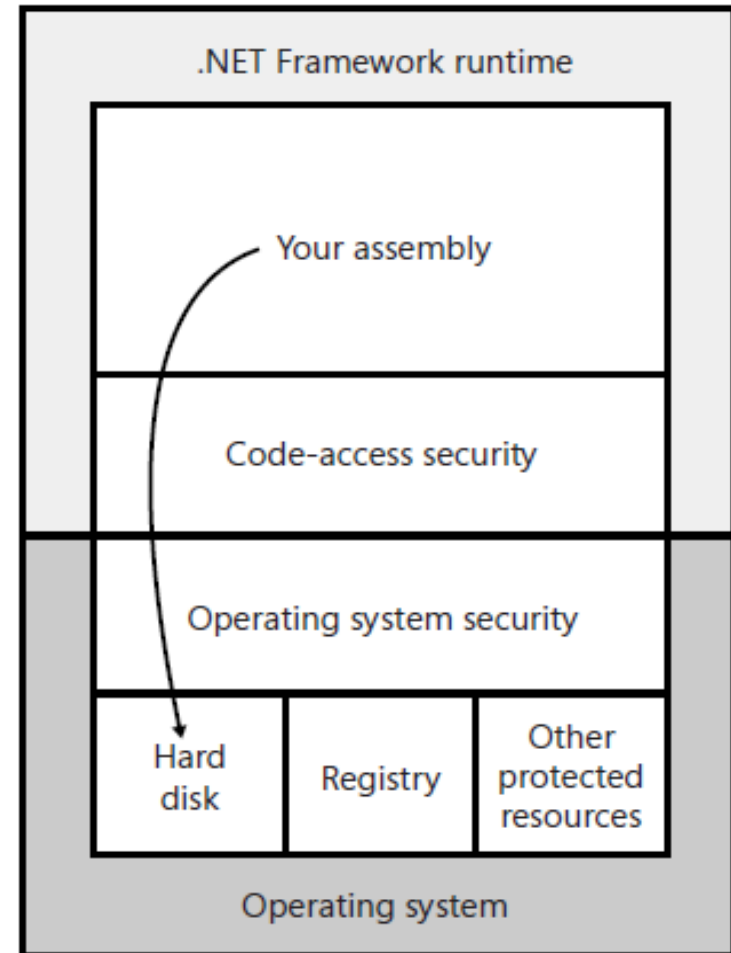
Microsoft .NET Framework Security **CAS, Struttura**

- Ogni sistema di sicurezza ha la necessità di trovare un modo per identificare l'utente e CAS non rappresenta un'eccezione
- CAS identifica e assegna i permessi alle applicazioni e non agli utenti, quindi non utilizza username, password o access control lists (ACLs). CAS identifica gli assemblies utilizzando gli evidence
- Un evidence permette ad un assembly di essere identificato, di individuare la locazione in cui viene memorizzato, di individuare la firma dell'assembly ed il suo hash, nonché identificare il code group a cui esso appartiene. (Application Directory, Hash, Publisher, Site, Strong Name, Zone)
- Esistono due tipi di evidence:
 - Host Evidence, che identifica o meglio descrive l'origine dell'assembly (come la directory dell'applicazione o l'URL)
 - Assembly Evidence che invece descrive l'identità dell'assembly, come l'hash, lo strong name e il publisher

Microsoft .NET Framework Security **CAS, Come lavora**

CAS è completamente indipendente dal sistema di sicurezza del sistema operativo sottostante, infatti CAS lavora al di sopra di questo livello. Quando bisogna determinare se un assembly ha il permesso di eseguire una determinata azione, vengono valutati separatamente, CAS e la sicurezza del sistema operativo basata su RBS. Se ad esempio un assembly ha il permesso di leggere una directory di sistema, ma a livello del sistema operativo questo permesso è negato, l'assembly non avrà il permesso d'interagire con la directory.

CAS non sostituisce la sicurezza basata sui ruoli, la integra.



Microsoft .NET Framework Security **CAS, Permessi**

- Le piattaforme Java e .NET forniscono permessi molto simili per il controllo dell'accesso a risorse come File System, file, directory, stampanti etc. Differiscono per:
 - Permessi per le risorse specifiche della piattaforma
 - Granularità di controllo su specifiche operazioni e sulle risorse a cui è concesso l'accesso.
- In .NET un permesso è una specifica entry di CAS per controllare l'accesso ad una specifica risorsa. Per default, in .NET sono disponibili 19 tipi di permessi, configurabili utilizzando un apposito tool.

Microsoft .NET Framework Security CAS, Permessi

Permesso	Descrizione
Directory Services	Permette ad un assembly di accedere ad Active Directory. Possono essere specificati percorsi utilizzabili, permessi di scrittura e di esplorazione (Browse)
DNS	Abilita o restringe l'accesso ad un assembly a sottomettere richieste DNS
Environment Variables	Permette ad un assembly di accedere alle variabili d'ambiente come Path, Username. Possono essere specificate le variabili d'ambiente a cui l'assembly può avere accesso.
Event Log	Permette ad un assembly di accedere all'event log di sistema. Possono essere permessi accessi limitati o restringere gli accessi a determinate eventi/applicazioni
File Dialog	Permette (o nega) ad un assembly di visualizzare finestre di dialogo predefinite come la finestra per l'apertura o il salvataggio di un file
File IO	Restringe l'accesso a file o cartelle. Può essere specificato di restringere l'accesso a tutti i file o cartelle o possono essere indicati liste di percorsi a cui l'assembly può accedere (in lettura e/o scrittura, Append o discovery delle directory)
Isolated Storage File	Permette ad un assembly di accedere ad zone di memorizzazione isolate. Ogni zona di memorizzazione può essere configurata con un livello di isolamento e la dimensione della quota assegnata
Message Queue	Permette ad un assembly di accedere alle code di messaggi del sistema operativo
Performance Conuter	Permette ad un assembly di leggere e/o scrivere i performance counter
Printing	Limita le capacità di stampa di un assembly
Reflection	Permette ad un assembly di eseguire il discovering dei membri e dei tipi contenuti in alti assembly

Microsoft .NET Framework Security **CAS, Permessi**

Permesso	Descrizione
Registry	Restringe l'accesso alle chiavi di registro
Security	Fornisce un accesso granulare alle capacità di un assembly di accedere alle varie funzioni della CAS. Questo tipo di permesso può controllare se gli assembly possono eseguire codice non gestito, concedere permessi, controllare threads etc..
Service Controller	Specifica quali servizi, se presenti, un assembly può visualizzare e/o controllare
Socket Acces	Permesso utilizzato per concedere o negare ad un assembly il permesso di inizializzare connessioni TCP/IP. Può essere controllato destinatario, numero di porta e protocollo
SQL Client	Controlla se un assembly può accedere a SQL Server e se blank passwords sono permesse
User Interface	Determina se un assembly può creare nuove finestre o se ha accesso alla clipboard
Web Access	Determina se un assembly può accedere a siti internet e a quali siti ha accesso
X509 Store	Permette ad un assembly di accedere alla zona in cui sono memorizzati i certificati X509. Secondo dei permessi, un assembly può avere accesso per l'aggiunta, rimozione e apertura.

Microsoft .NET Framework Security **CAS, Permessi**

Generalmente, .NET fornisce una granularità più fine nella concessione dei permessi. Ad esempio entrambe le piattaforme possono restringere l'accesso al file system, ma mentre in Java esistono i permessi di lettura e scrittura, in .NET i permessi sono diversificati in lettura, scrittura e *append*.

Di seguito è riportata una tabella che riassume in che modalità le due piattaforme gestiscono le restrizioni per una stessa risorsa

Microsoft .NET Framework Security CAS, Permessi

Risorsa	Restrizione	Java Permissions	.Net Permissions
File System	Leggere/Scrivere/eseguire/Cancellare files	FilePermission	FileIOPermission, SecurityPermission
	Append, accesso alle informazioni dello stesso file	Permessi non separati (append=scrittura)	FileIOPermissionAccess (Append, PathDiscovery)
	Accedere ai dati di una directory da un programma in esecuzione	Può leggere qualsiasi file della directory corrente o sub-directory della directory corrente	IsolatedStoragePermission
Network	Accettare/connettersi/ascoltare/risolvere un host all'interno di un range di porte	SocketPermission	Socket Permission
Display	Mostrare un applet, creare una finestra senza warning, restringere l'accesso alla coda degli eventi	AWTPermission	Eventi gestiti separatamente senza permessi speciali
	Controllare differenti proprietà di una finestra	Non fornito	UIPermission.Window
Reflection	Uso di reflection	ReflectionPermission	ReflectionPermission
	Reflection di membri visibili ed invisibili di un tipo	Livelli di controllo non permessi (tutto o niente)	ReflectionPermission Differenti valori di flag per controllarne l'uso
System Clipboard	Leggere/scrivere la clipboard (tutto o niente)	AWTPermission	UIPermission.Clipboard
	Leggere la clipboard (senza restrizioni)	Livello di controllo non permesso (tutto o niente)	UIPermission.Clipboard (OwnClipboard)

Microsoft .NET Framework Security CAS, Permessi

Risorsa	Restrizione	Java Permissions	.Net Permissions
Threading	Controllo dei threads	RuntimePermission	SecurityPermission
	Controllo di qualsiasi thread, codice di un thread, controllo di un gruppo di thread	RuntimePermission Differenti valori per controllare diversi livelli di privilegi	ThreadPool fornisce la sicurezza attraverso l'implementazione
Database	Log delle operazioni SQL	SQLPermission	Configurazione attraverso registro o attraverso strumenti di configurazione esterni
	Password blank per l'accesso a database utenti	Specifico delle API di database non incluse per default	OdbcPermission OleDbPermission OraclePermission SqlClientPermission
Printer	Stampa	RuntimePermission	PrintingPermission
	Stampare su qualsiasi stampante (solo quella di default) e attraverso finestre di dialogo con parametri ristretti	Tutto o niente, attraverso RuntimePermission	PrintingPermission

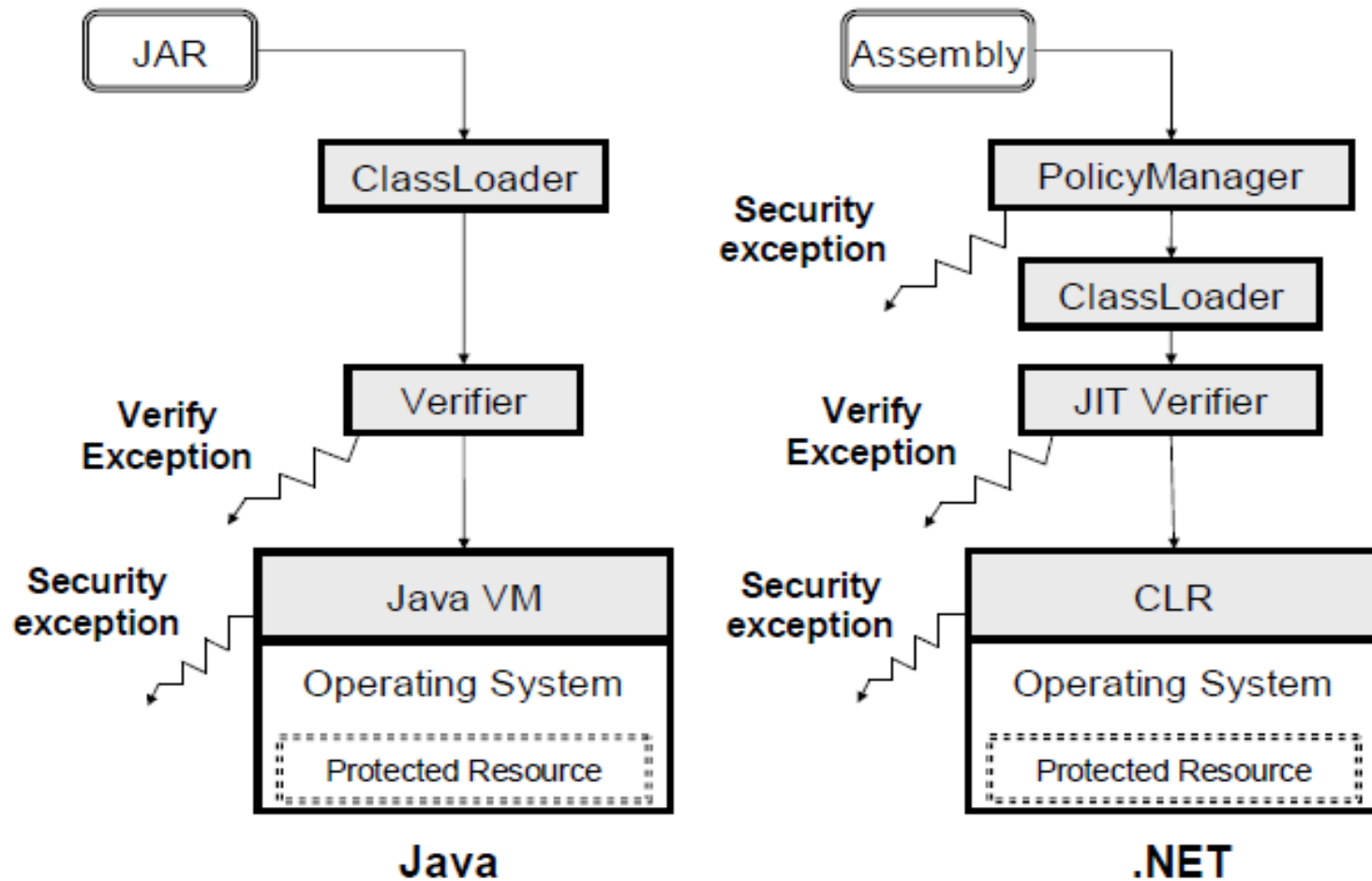
Microsoft .NET Framework Security **CAS, Permessi**

Nessuna delle due piattaforme offre un supporto completo per la gestione dei permessi, solo le azioni associate ad un permesso sono testate. Molte altre risorse, come ad esempio l'allocazione della memoria non hanno un permesso associato, inoltre non è previsto un supporto per la restrizione alla quantità di risorse consumate. In questo modo, molti attacchi di tipo *Denial-of-Service* sono possibili, evitando le policy di sicurezza. Queste limitazioni sono molto importanti e costituiscono un'attiva area della ricerca

Microsoft .NET Framework Security Policy

- Applicazioni con differenti trust levels possono essere eseguiti sulla stessa VM. Il meccanismo di sicurezza della VM deve determinare quale policy deve essere utilizzata per mediare l'accesso alle risorse protette.
- Esistono differenze sostanziali sulla gestione delle policies nelle due piattaforme:
 - Inizialmente il codice poteva essere solo *trusted* o *untrusted*. Le versioni successive di Java hanno esteso questo modello, ma si è reso necessario mantenere la retro compatibilità con il codice precedente.
 - .NET essendo un progetto nuovo è nato con un modello di sicurezza più ricco ed estensibile
- In entrambe le piattaforme esistono due tipi di permessi: statici e dinamici

Microsoft .NET Framework Security Policy, Gestione



Microsoft .NET Framework Security Policy, Gestione

- Java e .NET includono meccanismi complessi per la risoluzione delle policy
- Sebbene .NET non fornisca lo stesso livello di flessibilità come Java nella personalizzazione delle policy di sicurezza, uno sviluppatore può creare nuovi permessi, ma deve stare attento ad evitare errori
- Un file delle policy di Java è costituito da una lista di entry, una per ogni *grant*. Ogni entry specifica il contesto nel quale il *grant* può essere applicato e l'insieme di permessi concessi in quel specifico contesto.
- Una policy di .NET è specificata da un gruppo di livelli di policy
 - *Enterprise*: inteso per amministratori di sistema
 - *Machine*: amministratore della macchina
 - *User e Application Domain*.

Microsoft .NET Framework Security **Policies, Dichiarative e Imperative**

Le dichiarazioni statiche di policy sono meglio conosciute come Dichiarative e sono contenute nel manifest dell'assembly, le policy dinamiche sono meglio conosciute come Imperative e vengono compilate in CIL per essere valutate a runtime . I permessi imperativi possono essere specificati a livello di classe o di metodo e possono essere utilizzate per descrivere tutte quelle costrizioni che non possono essere specificate utilizzando permessi dichiarativi (staticamente). Si osserva che in .NET, non è concesso creare una propria implementazione di SecurityManager, riducendo la flessibilità, ma aumentando il livello di sicurezza.

Microsoft .NET Framework Security Policies, Modalità Dichiarativa 1/2

Ci sono tre motivi per utilizzare la forma dichiarativa per un assembly:

1. Assicurarsi che il *runtime* non eseguirà mai un assembly se questo non ha accesso a tutte le risorse di cui ha bisogno per essere eseguito;
2. Creare una piccola *sandbox* per l'applicazione che dovrà essere seguita assicurandoci che un'attaccante non possa utilizzare la nostra applicazione per avere accesso a risorse a cui normalmente non può (il principio di concedere meno privilegi possibili riduce le *chances* di un'attaccante di abusare di un assembly per intraprendere azioni maliziose, come rilevare il contenuto di file privati, distruggere dati o propagare virus e/o worms)
3. Verificare che l'applicazione può essere eseguita con permessi limitati e quindi essere eseguita in zone parzialmente *trusted*.

Microsoft .NET Framework Security

Policies, Modalità Dichiarativa 2/2

Tutte le classi utilizzate per la dichiarazione dei permessi derivano da un'unica classe: *CodeAccessSecurityAttribute* ed hanno in comune due proprietà:

- *Action*, specifica l'azione di sicurezza che deve essere intrapresa
- *Unrestricted*, un valore booleano che specifica che tutti i permessi della classe sono abilitati

Action può assumere uno dei seguenti valori:

- *SecurityAction.RequestMinimum*: richiede il permesso (minimo) che permette all'assembly di essere eseguito.
- *SecurityAction.RequestOptional*: rifiuta tutti i permessi che non sono presenti nella lista delle dichiarazioni di *Action.RequestOptional* o *SecurityAction.RequestMinimum*. Definire i permessi con questa proprietà, ci assicura che l'applicazione eseguita non avrà più permessi di quelli dichiarati.
- *SecurityAction.RequestRefuse*: riduce i permessi assegnati all'applicazione. Questa dichiarazione deve essere utilizzata quando ci si vuole assicurare che l'applicazione non abbia accesso alle risorse critiche di sistema.

Microsoft .NET Framework Security

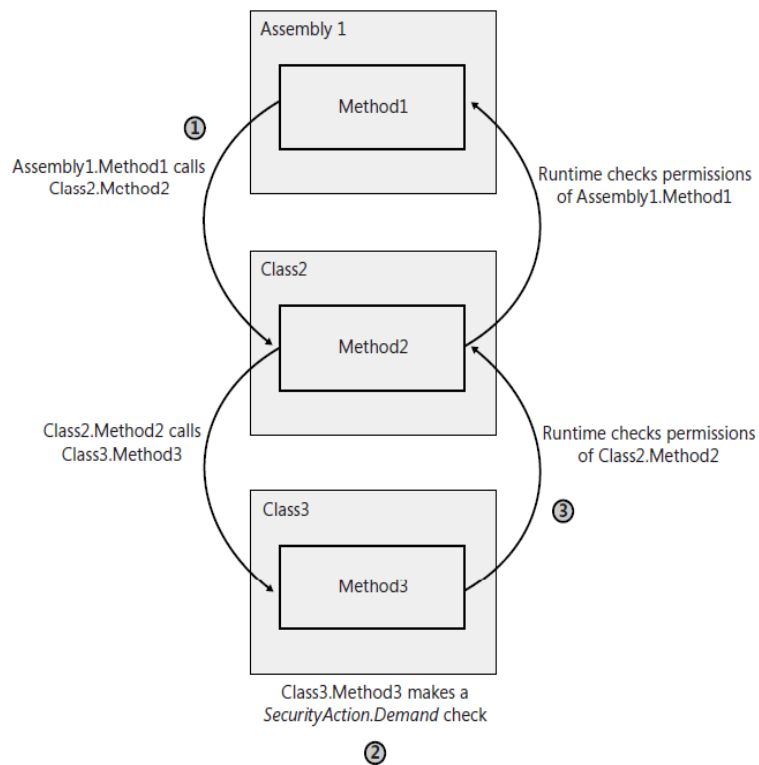
Policies, Modalità Imperativa 1/2

Sebbene siano presenti tre modalità per la dichiarazione di CAS per un assembly, esistono sei opzioni disponibili per la dichiarazione dei permessi dei metodi di un oggetto, sia in forma dichiarativa che imperativa.

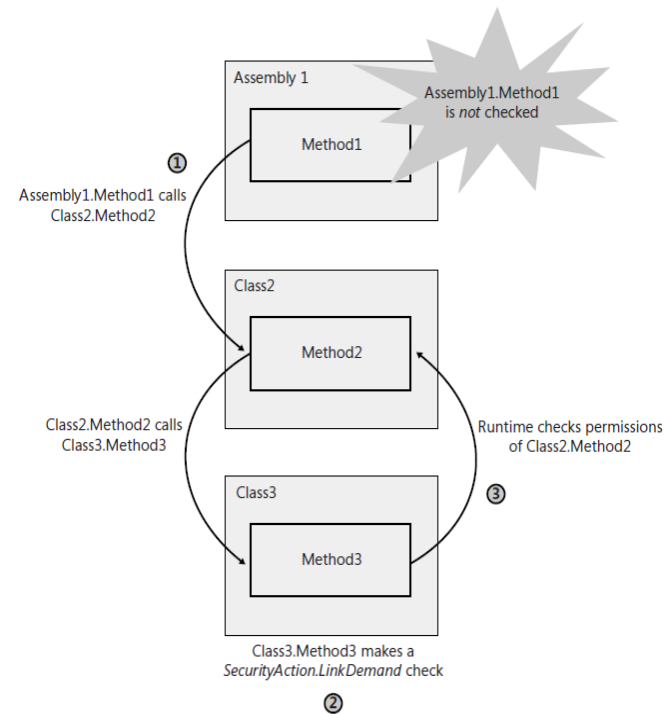
- *Assert*: Istruisce il runtime ad ignorare il fatto che i chiamanti potrebbero non avere i permessi specificati.
- *Demand*: Istruisce il runtime a sollevare un'eccezione se il chiamante e i chiamanti più alti nello stack, non hanno lo specifico permesso richiesto
- *Deny*: Causa il runtime a ridurre l'accesso ai metodi attraverso la rimozione dello specifico permesso
- *InheritanceDemand*: Istruisce il runtime a sollevare un'eccezione se l'assembly eredita da una classe che manca dello specifico permesso
- *LinkDemand*: Indica al runtime di sollevare un'eccezione se l'immediato chiamante, ma non i chiamanti più alti nello stack, mancano dello specifico permesso
- *PermitOnly*: Istruisce il runtime a ridurre l'accesso al metodo rimuovendo tutti i permessi ad eccezione di quello specificato

Microsoft .NET Framework Security

Policies, Modalità Imperativa 2/2



Demand controlla i permessi di tutti i chiamanti

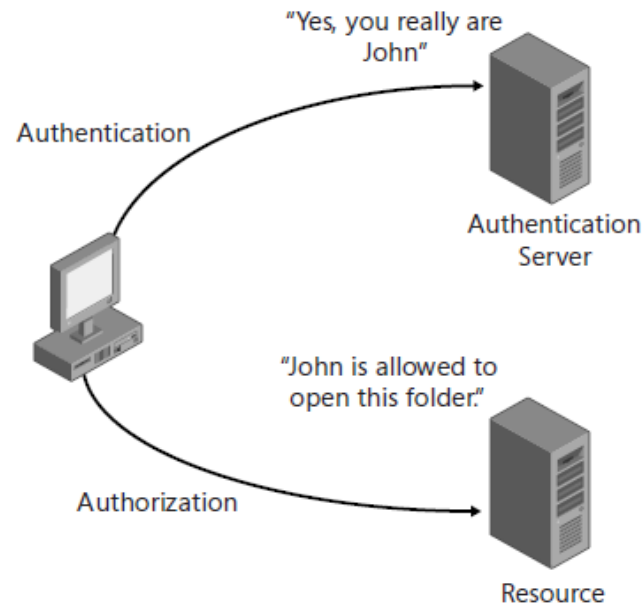


LinkDemand verifica i permessi solo del chiamante immediato

Microsoft .NET Framework Security

Autenticazione e Autorizzazione

- L'autenticazione è il processo d'identificazione di un utente ed è il concetto più visibile di sicurezza
- Con autorizzazione indichiamo il processo di restrizione dell'accesso a specifiche risorse basandosi sull'identità degli utenti.



Microsoft .NET Framework Security Autent. e Autoriz., Gestione

- *WindowsIdentity*, oggetto che rappresenta l'account di un utente Windows. Può essere creato mediante l'utilizzo di uno dei tre metodi seguenti:
 - *GetAnonymous*
 - *GetCurrent*
 - *Impersonate*
- *WindowsPrincipal*, fornisce l'accesso alle informazioni del gruppo a cui appartiene l'utente
- *PrincipalPermission*, permette di verificare il *principal*. Utilizzata per chiedere se l'utente che esegue il codice è autenticato o se appartiene ad uno specifico ruolo

Microsoft .NET Framework Security **Access Control Lists (ACLs)**

I sistemi operativi usano le *Access Control Lists* per restringere l'accesso a file, cartelle, stampanti, servizi e su tutte le risorse utilizzate e appartenenti al sistema operativo. Le ACLs possono essere utilizzate per:

- Restringere l'accesso a files, cartelle, e altre risorse che possono essere utilizzate dalle applicazioni
- Configurare l'accesso a risorse che normalmente non sono accessibili da parte dell'utente, ma che sono necessarie per l'esecuzione di determinate applicazioni.

In .NET Framework, può essere utilizzata l'enumerazione *FileSystemRights* per specificare i permessi ed i diritti su file e cartelle, con gli stessi permessi speciali e standard che possono essere visualizzati utilizzando la finestra delle proprietà di Windows Explorer

Microsoft .NET Framework Security Reflection & Obfuscation

Reflection

indica la possibilità di ottenere informazioni relative ai tipi contenuti in un assembly a *runtime*. In questo modo è possibile analizzare assemblies ed oggetti, consentendo addirittura di invocare direttamente i metodi di una classe, o di accedere alle sue proprietà.

Obfuscation

Come Java, anche .NET soffre dello stesso problema: attraverso semplici tool scaricabili da internet, tra i più famosi .Net Reflector (<http://www.red-gate.com/products/reflector/>) (o ad ILDASM), è possibile eseguire il reverse engineering di un'assembly.

Gli obfuscator attuali, eseguono:

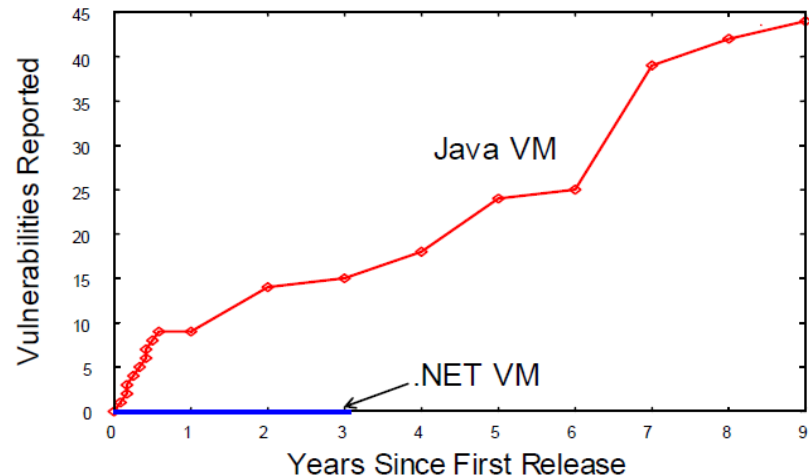
- La cifratura del codice sorgente
- La rimozione di tutte informazioni non necessarie dai metadati dell'assembly

Microsoft .NET Framework Security Conclusioni 1/2

Java e .NET hanno gli stessi obiettivi di sicurezza e, più o meno, gli stessi meccanismi di sicurezza.

Alcune delle vulnerabilità relative al .NET, non sono legate alla piattaforma stessa o alla Virtual Machine utilizzata, quanto a bug del sistema operativo su cui il Framework si appoggia.

Ad esempio, nel 2007 fu creato il primo virus che aveva come obiettivo i file .exe creati con il .NET Framework, W32.Donut. Il problema si presentava quando il virus riusciva a prendere il controllo dell'eseguibile prima del runtime di .NET. Per questo modo di agire non ha mai avuto una grande diffusione, ed è stato catalogato come di basso rischio



Microsoft .NET Framework

Security

Conclusioni 2/2

Perché sembrano esserci meno lacune in .NET rispetto a Java?

- è una piattaforma meno desiderabile rispetto a Java e quindi per il momento non ha suscitato un interesse tale da permettere di scoprire eventuali vulnerabilità. Questa teoria sembrerebbe improbabile dato che il .NET viene offerto come aggiornamento dei sistemi operativi Windows e comunque si trova integrato nei recenti sistemi operativi.
- sono state trovate più vulnerabilità nelle implementazioni in Java dato che sono presenti differenti Virtual Machine, mentre per .NET esiste solo l'implementazione Microsoft. Questo non è del tutto esatto per la presenza del progetto Mono.
- .NET ha evitato le vulnerabilità più critiche dato che queste erano già ben conosciute dalla precedente esperienza di Java, anche se in generale, non è così, dato che esistono differenze sostanziali tra le due piattaforme e le più gravi vulnerabilità di sicurezza non sono direttamente collegate.

Microsoft .NET Framework Security Bibliografia

- [1] Nathanael Paul, David Evans: *Comparing Java and .NET Security: Lessons Learned and Missed*, University of Virginia, Department of Computer Science, 2006
- [2] Tony Northrup, Shawn Wildermuth: *Microsoft .NET Framework 2.0 Application Development Foundation*, Microsoft Press 2008
- [3] MSDN, <http://msdn.microsoft.com>
- [4] Wikipedia, <http://it.wikipedia.org>

Microsoft .NET Framework Security

Grazie per l'attenzione. 😊