

CAS by examples

Di Matteo Italia AKA MIItaly – mitaly@techtown.it

Prima puntata: cos'è CAS

L'iniziativa .NET ha portato con sé moltissime novità, principalmente per gli sviluppatori: la semplicità con cui si riescono a sviluppare applicazioni complesse e potenti nei linguaggi .NET, come VB.NET e C#, è impressionante, e lo stesso deployment è stato reso estremamente semplice grazie alla tecnologia ClickOnce e la semplicità con cui si può effettuare il cosiddetto xcopy deployment. Tuttavia Microsoft ha pensato anche ai sistemisti e alle problematiche di sicurezza che devono affrontare quotidianamente nella gestione della rete, per cui è stato sviluppato il modello di protezione CAS (*Code Access Security*), grazie al quale è possibile impostare con estrema precisione quali applicazioni possono fare cosa. In questo primo articolo di taglio molto teorico esamineremo a grandi linee il funzionamento di CAS e gli strumenti con cui gestirlo, ponendo le basi la comprensione degli articoli successivi, in cui ci occuperemo di problematiche concrete risolubili tramite un utilizzo corretto di CAS e della realizzazione di componenti in grado di funzionare correttamente anche se richiamati da assembly parzialmente trusted.

Il funzionamento di CAS

CAS funziona grazie al fatto che tutte le applicazioni .NET sono compilate in codice intermedio (non nativo) e che dipendono in tutto e per tutto dalle dll del .NET Framework: il runtime può quindi individuare esattamente quali metodi di quali classi il programma va a chiamare, e può perciò decidere di consentire o di negare la loro esecuzione sollevando una *SecurityException*.

Supponendo che il programma tenti di chiamare un certo metodo del Framework il runtime per decidere se permettere o negare questa azione verificherà tramite uno *stack walk* che il codice in esecuzione e tutti i suoi chiamanti possiedano le autorizzazioni necessarie per richiamare tale metodo; queste ultime sono solitamente determinate da una serie di attributi applicati al metodo (*protezione dichiarativa*) o da una richiesta specifica di controllo effettuata da parte del metodo stesso (*protezione imperativa*). I permessi assegnati al codice dell'applicazione e ai suoi chiamanti derivano invece dalle impostazioni dei *criteri di protezione runtime*, modificabili tramite gli strumenti descritti più in basso in questo articolo. I criteri di protezione runtime si basano sulla divisione del codice eseguito in *gruppi di codice*, per i quali sono definiti i *criteri di appartenenza* e il *set di autorizzazioni* associato. Se del codice appartiene a più gruppi di codice (risponde ai criteri di appartenenza di tutti) otterrà solo il subset di permessi comune a tutti i gruppi di codice (a questa regola c'è però un'eccezione che vedremo in seguito).

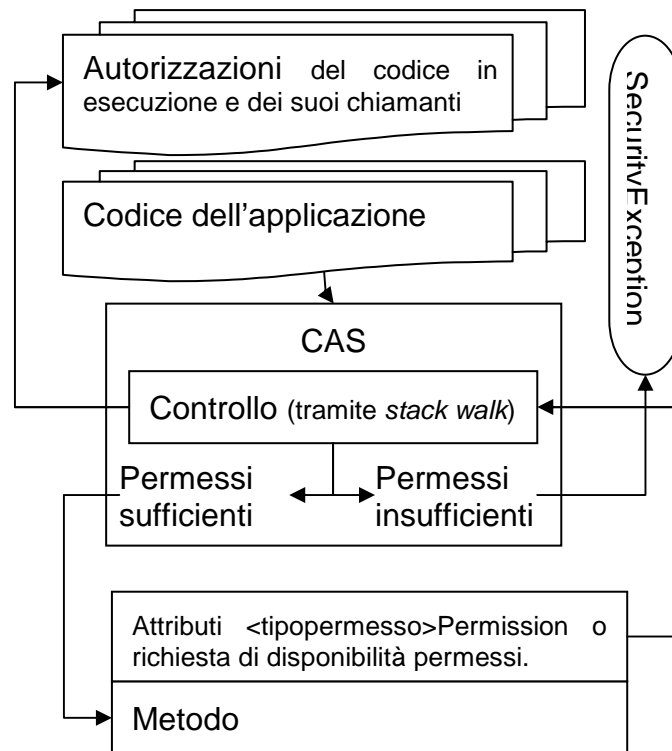


Figura 1 - Controlli effettuati da CAS alla chiamata di una funzione.

Stack walk parziali

Da questo discorso può emergere però un'obiezione: i metodi "sensibili" del .NET Framework sono certamente dotati di attributi o di controlli che consentono al CAS di evitare che applicazioni senza permessi sufficienti possano effettuare certe operazioni, ma l'invocazione di codice unmanaged tramite `PInvoke` consentirebbe di aggirare agilmente qualunque protezione di questo tipo; il problema è stato risolto consentendo solo agli assembly dotati dei massimi privilegi ("*FullTrust*") di invocare codice unmanaged, disattivando questa caratteristica per tutti gli altri gruppi di codice. Tuttavia anche questo fatto può dar luogo a perplessità: infatti tutto il codice gestito alla lunga si risolve in chiamate a codice unmanaged, per cui teoricamente le applicazioni che non sono abilitate a richiamare procedure non gestite non potrebbero interagire in alcun modo con il sistema: infatti il CAS ad ogni chiamata delle dll del Framework a codice nativo genererebbe uno stack walk che arriverebbe inevitabilmente al codice a privilegi ridotti, e genererebbe immediatamente una eccezione `SecurityException`; questo problema viene evitato grazie all'attributo `SuppressUnmanagedCodeSecurity`, che, applicato ad una dichiarazione di metodo unmanaged o alla classe che lo contiene, indica al CAS di non effettuare lo stack walk alla ricerca delle autorizzazioni necessarie, ma di controllare soltanto le autorizzazioni del codice che richiama direttamente la procedura non gestita; ovviamente l'utilizzo di questo attributo apre una vulnerabilità nel codice, e va quindi utilizzato solamente se si nasconde il metodo unmanaged dichiarandolo come privato della classe e consentendo l'accesso ad esso solo tramite metodi gestiti che effettuano tutti i controlli di protezione del caso, come fanno i metodi del .NET Framework.

Esistono inoltre altri metodi per indicare a CAS di non effettuare uno stack walk completo ma di riferirsi solamente al gruppo di codice a cui appartiene l'assembly corrente (e non i suoi chiamanti), come i vari Assert e LinkDemand; ne parleremo approfonditamente nella puntata dedicata alla creazione di librerie che ammettano chiamanti parzialmente trusted.

I gruppi di codice

I gruppi di codice possono essere definiti a tre livelli:

1. **Livello User:** gruppi definiti per l'utente attualmente loggato; memorizzato nel file `%appdata%\Microsoft\CLR Security Config\v<versione>\security.xml`.
2. **Livello Computer:** gruppi definiti a livello di computer, solitamente modificabili solo dall'amministratore di sistema; memorizzato nel file `%windir%\Microsoft.NET\Framework\v<versione>\CONFIG\security.xml`.
3. **Livello Enterprise:** gruppi definiti a livello di azienda, solitamente modificabili solo dall'amministratore di sistema; memorizzato nel file `%windir%\Microsoft.NET\Framework\v<versione>\CONFIG\enterprisesec.xml`.

Nota: `<versione>` è la versione del .NET Framework da configurare, le due più diffuse sono la 2.0.50727.42 e la 1.1.4322.

L'ordine in cui vengono esaminati questi gruppi è Enterprise, Machine, User; questo è importante da sapere nel caso in cui uno dei gruppi di codice disponga del flag "Final", ossia di un flag che dice al CAS di non andare oltre nella ricerca di gruppi di codice cui il codice in esame potrebbe appartenere; negli esempi che vedremo nelle prossime puntate questo flag si rivelerà molto utile, poiché consente al CAS di ignorare i criteri definiti a livello Machine e User se ne esistono già a livello Enterprise.

Abbiamo detto che per appartenere ad un gruppo di codice del codice deve rispondere a dei criteri: ecco quelli che sono usati più di frequente:

- **Tutto il codice** include qualunque assembly;
- **Directory dell'applicazione** include il codice degli assembly situati nella directory specificata;
- **GAC** include il codice degli assembly registrati nella *Global Assembly Cache*;
- **Hash** include il codice dell'assembly con l'hash specificato;
- **Sito** include il codice degli assembly provenienti dal sito specificato (si applica per gli assembly senza installazione aperti grazie alla tecnologia ClickOnce);
- **Nome sicuro** include tutti gli assembly con lo strong name (tutto o solo parte di esso) specificato;

- **Zona** include tutti gli assembly provenienti da una zona specifica (come *Computer Locale*, *Intranet Locale*, *Internet* e così via; sono le stesse zone definite per Internet Explorer).

Ecco invece i set di autorizzazioni assegnati più spesso:

- **FullTrust** dà al gruppo di codice tutti i permessi disponibili;
- **LocalIntranet** dà al gruppo di codice un set di permessi limitato ma abbastanza ampio; tra le limitazioni più importanti segnaliamo l'impossibilità di invocare dll .NET condivise (a meno che queste non abbiano l'attributo *AllowPartiallyTrustedCallers*), codice non gestito, la limitazione di alcune funzionalità della reflection e la disabilitazione di alcune funzionalità avanzate tra cui la gestione di thread, oggetti principal, appdomains e serializzatori.
- **Internet** dà al gruppo di codice un set di permessi molto limitato: rispetto al set *LocalIntranet* è possibile leggere files solo dietro conferma dell'utente (tramite *OpenFileDialog*), non è possibile accedere agli appunti globali, non è possibile influenzare le finestre di primo livello, la dimensione dello spazio di archiviazione isolata è limitata a 512 KB e le funzionalità di stampa sono limitate.

Dopo l'installazione del .NET Framework di default sono creati i seguenti gruppi di codice:

- **Livello enterprise**
 - *All_Code* include tutto il codice e gli dà il set di autorizzazioni *FullTrust*.
- **Livello Computer**
 - *All_Code* include tutto il codice e gli dà il set di autorizzazioni *FullTrust*.
 - *My_Computer_Zone* include il codice situato sulla macchina locale (criterio di appartenenza "Zona Risorse del Computer") e gli dà il set di autorizzazioni *FullTrust*.
 - *LocalIntranet_Zone* include il codice situato nella intranet (criterio di appartenenza "Zona Intranet Locale") e gli dà il set di autorizzazioni *LocalIntranet*.
 - *Internet_Zone* include il codice situato in internet (criterio di appartenenza "Zona Internet") e gli dà il set di autorizzazioni *Internet*.
 - *Restricted_Zone* include il codice situato su siti non fidati (criterio di appartenenza "Zona Siti non attendibili") e gli dà il set di autorizzazioni *Nothing* (nessun permesso).

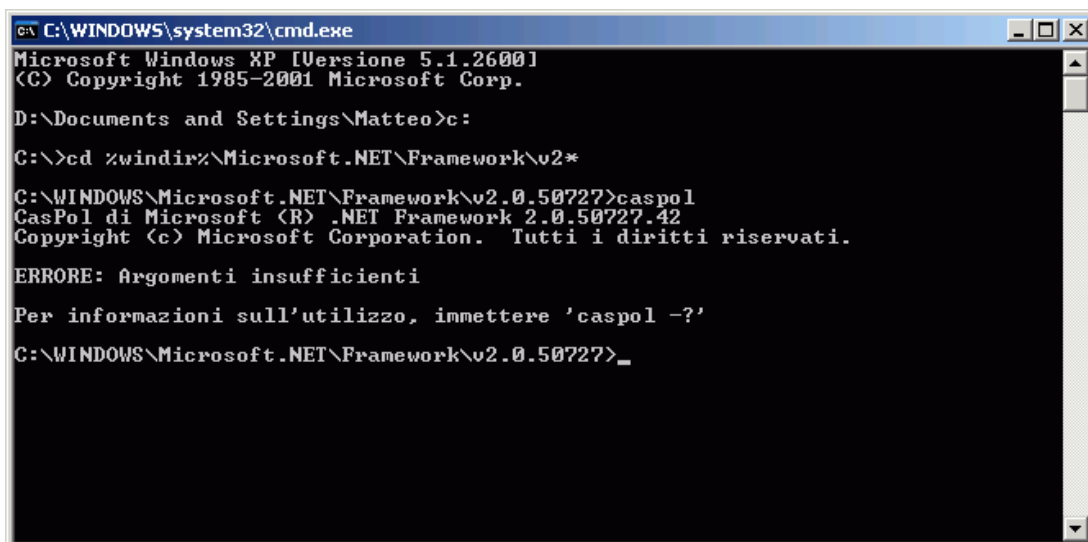
- *Trusted_Zone* include il codice situato su siti attendibili (criterio di appartenenza "Zona Siti attendibili") e gli dà il set di autorizzazioni *Internet*.
- **Livello Utente**
 - *All_Code* include tutto il codice e gli dà il set di autorizzazioni *FullTrust*.

Si potrà notare come in realtà gli unici gruppi di codice che effettivamente limitano l'esecuzione degli assembly siano quelli impostati a livello di Computer, mentre i vari "All_Code" si limitano a fare da contenitori degli altri eventuali sottogruppi di codice.

Strumenti di modifica

Ci sono più strumenti per modificare i gruppi di codice; riportiamo i più diffusi:

- **CasPol**, strumento di modifica a linea di comando di gruppi di codice e più in generale della impostazioni di sicurezza .NET. A mio avviso estremamente scomodo e rischioso da usare, poiché dispone di un numero di opzioni e di modalità piuttosto elevato ed il rischio di confondersi è sempre in agguato; ha tuttavia il grande vantaggio di essere installato assieme al .NET Framework e quindi di essere sempre installato su qualunque macchina su cui si renda necessario intervenire. Da usare se non avete niente di meglio. *Per accedervi: Start->Esegui->Digitare cmd->Ok->Digitare cd %windir%\Microsoft.NET\Framework\v<vers> e premere Invio->digitare caspol seguito dai parametri e premere Invio.*



```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Versione 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

D:\Documents and Settings\Matteo>c:
C:\>cd %windir%\Microsoft.NET\Framework\v2*
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727>caspol
CasPol di Microsoft (R) .NET Framework 2.0.50727.42
Copyright (c) Microsoft Corporation. Tutti i diritti riservati.

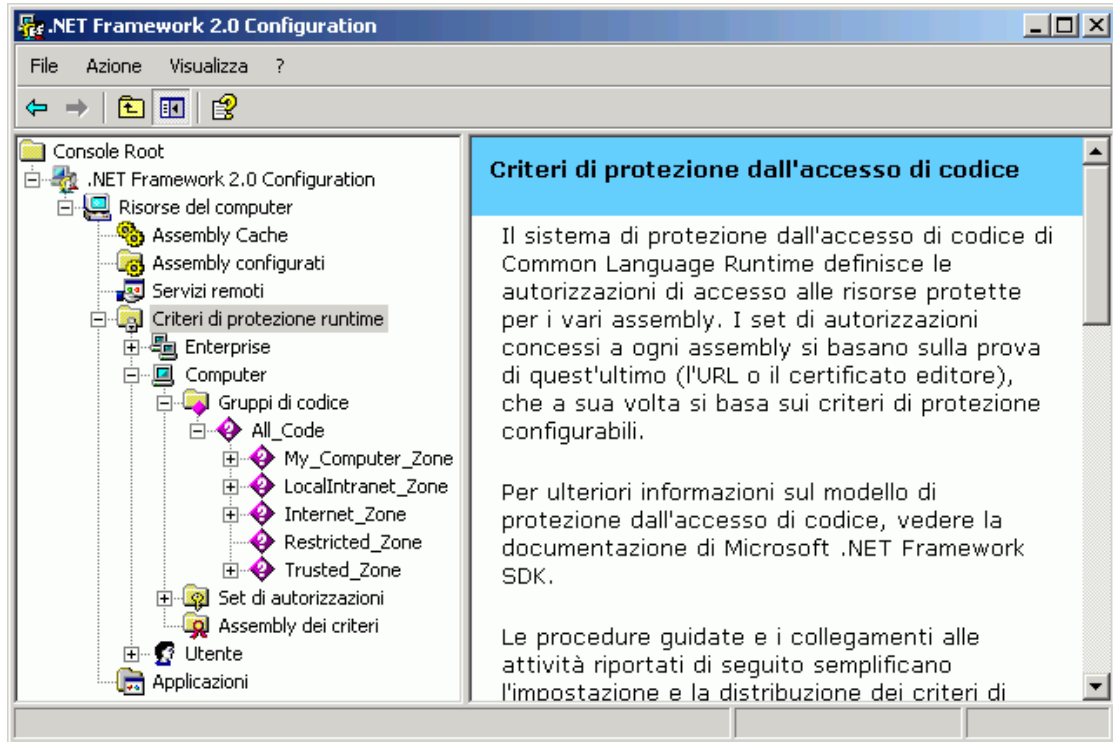
ERRORE: Argomenti insufficienti

Per informazioni sull'utilizzo, immettere 'caspol -?'
C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727>_

```

- **.NET Framework Configuration**, snap-in della console mmc che consente di configurare in modalità grafica e in maniera abbastanza semplice e intuitiva la protezione e altre impostazioni del .NET Framework. Molto comodo da usare, purtroppo dalla versione 2.0 del Framework non è più incluso nell'installazione standard del Framework ma solo con l'SDK. Resta lo strumento di modifica più semplice e sicuro,

forse un po' tortuoso per effettuare alcune operazioni; da usare sempre se se ne ha la possibilità. Per accedervi: Start->Impostazioni->Pannello di controllo->Strumenti di amministrazione->Microsoft .Net Framework <versione> Configuration (o "Configurazione di Microsoft .NET Framework <versione>").



- **Notepad** o altro editor di testo, o anche meglio di XML, consentono di modificare manualmente i vari files delle impostazioni dei gruppi di codice, operazione che in molti casi è più rapida della consultazione del lungo help screen di CasPol. Naturalmente è consigliabile effettuare una copia di backup dei files che si intende modificare prima di lavorarci sopra.

Strong Name

Non è direttamente collegato al CAS, ma spesso i due argomenti si intrecciano, per cui è bene parlarne subito; uno strong name (tradotto da Microsoft come "nome sicuro") rappresenta l'identità di un assembly, ed è costituito da nome, numero di versione e localizzazione dell'assembly, dal manifesto dell'assembly (contenente l'hash dei files dell'assembly) cifrato con una chiave privata e dalla relativa chiave pubblica (si veda a questo proposito lo schema che segue questo paragrafo). Un assembly con strong name quindi dà una serie di importanti garanzie, tra cui la garanzia che l'assembly non è stato modificato dal momento della firma, che tutti gli assembly con la stessa chiave pubblica provengono da un unico produttore (quello che ha anche la chiave privata) ma soprattutto garantisce l'unicità dell'assembly: infatti solo chi dispone della chiave privata può crearne uno uguale dagli stessi sorgenti; questo risolve il problema del cosiddetto "inferno delle dll", perché gli assembly che dipendono da un assembly con nome sicuro lo sanno individuare con esattezza e non

possono essere ingannati da altri assembly con lo stesso nome, come invece accade con le dll tradizionali e COM.

Lo strong name è importante per il CAS perché è uno dei criteri di appartenenza per i gruppi di codice più usato e sicuro: infatti garantendo in modo sicuro l'identità dell'assembly o della sua provenienza consente di creare gruppi di codice a cui appartengano solo determinati assembly o solo gli assembly con la medesima chiave pubblica, ossia dello stesso produttore.

Nota importante: gli assembly con strong name non possono avere riferimenti ad altri assembly senza strong name.

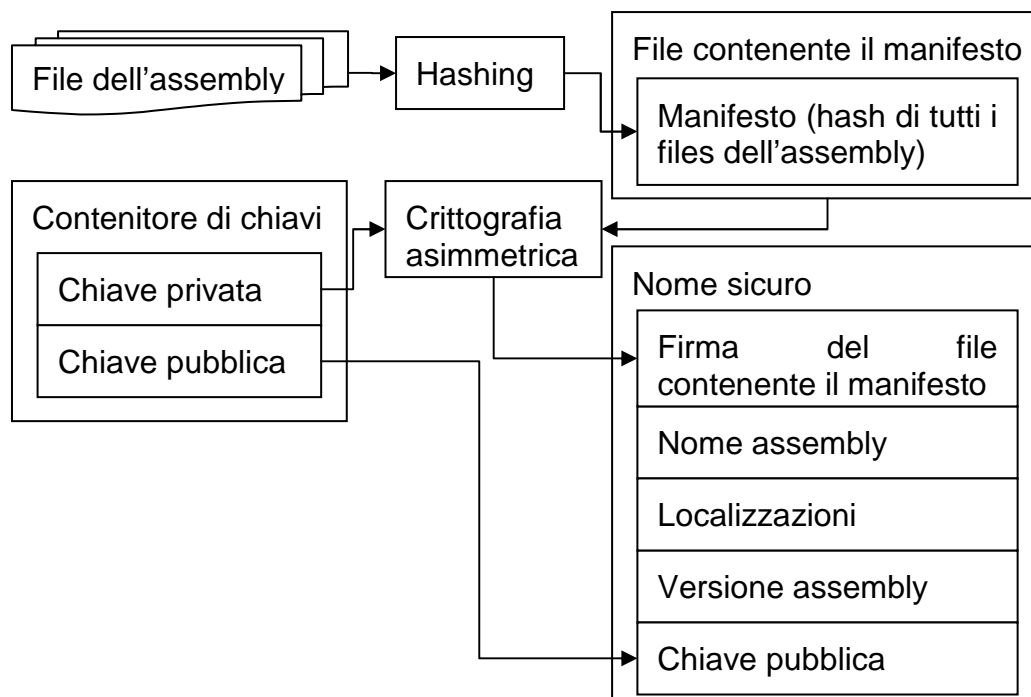


Figura 2 - Creazione di uno strong name

Seconda puntata: eseguire assembly "esigenti" da disco di rete

Quanto detto nella puntata precedente può spiegare uno dei problemi più diffusi che si verifica a causa della protezione CAS, ossia l'impossibilità di eseguire molte applicazioni .NET se esse sono situate su dischi di rete; esse infatti, trovandosi nel gruppo di codice "Machine\All_code\Localintranet_zone" sono soggette alle limitazioni citate in precedenza, delle quali la più importante è l'impossibilità di invocare codice unmanaged. In contesto aziendale questo problema può rivelarsi estremamente fastidioso: infatti è prassi diffusa condividere su una share del PDC piccole applicazioni utilizzate su tutti i client del dominio in modo da non dover agire su tutti i PC quando è necessario aggiornarle. In questa puntata vedremo come risolvere questo problema in due modi differenti, e al termine della loro descrizione vedremo come distribuire le impostazioni su tutto il dominio.

Nota: per lavorare comodamente in questa puntata è consigliabile aver installato lo snap-in di mmc ".NET Framework Configuration", che dalla versione 2 del framework viene installato soltanto con l'SDK (non con i normali redistribuitori). Per chi non l'avesse installato riporto comunque le modifiche da apportare direttamente ai file XML tranne nei casi in cui è necessario creare gruppi di codice la cui rappresentazione XML può variare molto a seconda dei casi, come nel caso di quelli che hanno come criterio di appartenenza lo strong name dell'assembly.

Prima possibilità: aumento dei permessi per tutta l'area intranet

Si tratta del metodo più "rude" e con cui si perde di più in sicurezza, ma ha il grande pregio della semplicità; potendo scegliere se mettere mano o alla sicurezza *Computer* o a quella *Enterprise*, opteremo ovviamente per la prima nel caso sia necessario intervenire su una singola macchina, mentre modificheremo le impostazioni *Enterprise* se si tratta di una modifica a livello di dominio.

Nota: in realtà questa distinzione viene attuata non tanto perché le modifiche al livello *Computer* non siano distribuibili sulla rete (operazione tecnicamente fattibilissima), quanto perché ritengo sia sempre meglio evitare di confondere le impostazioni *machine-specific* da quelle condivise a livello di azienda. Inoltre il mantenimento di tutte le impostazioni condivise nell'azienda a livello *Enterprise* garantisce sia che esse abbiano una priorità maggiore rispetto a quelle a livello *Computer* sia che, qualora si rendesse necessario aggiornare le impostazioni condivise, non si tocchi il livello *Computer*, che potrebbe essere stato modificato per esigenze specifiche della macchina.

Modifica a livello Computer

Vediamo in primo luogo il caso più semplice, ossia la modifica a livello *Computer*:

1. aprire .NET Framework Configuration e portarsi al livello Console Root\ .NET Framework <vers> Configuration\Risorse del computer\Criteri di protezione runtime\Computer\Gruppi di codice\All_code;
2. fare click di destro su "LocalIntranet_zone" e selezionare "Proprietà";
3. selezionare la scheda "Set di autorizzazioni" e dalla lista sottostante la scritta "Set di autorizzazioni" selezionare "FullTrust";
4. premere OK e chiudere mmc.

Dal momento che lo snap-in della console mmc dalla versione 2 del Framework è disponibile solo con l'SDK fornisco anche una procedura per chi non avesse voglia di scaricarsi 150 MB di software e di documentazione di cui non conta di farsi nulla:

1. aprire con Blocco Note o con qualunque altro editor di codice il file %windir%\Microsoft.NET\Framework\v<versione>\CONFIG\security.config;
2. individuare il nodo configuration\mscorlib\security\policy\policylevel;

3. modificare il sottonodo codegroup la cui proprietà "Name" sia "LocalIntranet_Zone" da così:

```
<CodeGroup class="UnionCodeGroup" version="1"  
PermissionSetName="LocalIntranet" Name="LocalIntranet_Zone"  
Description="Il gruppo di codice (eccetera).">(eccetera)</CodeGroup>
```

a così:

```
<CodeGroup class="UnionCodeGroup" version="1"  
PermissionSetName="FullTrust" Name="LocalIntranet_Zone" Description="Il  
gruppo di codice (eccetera).">(eccetera)</CodeGroup>
```

4. Salvare e chiudere.

Modifica a livello Enterprise

La modifica a livello Enterprise è leggermente più laboriosa, poiché richiede la creazione di un nuovo gruppo di codice:

1. aprire .NET Framework Configuration e portarsi al livello Console Root\ .NET Framework <vers> Configuration\Risorse del computer\Criteri di protezione runtime\Enterprise\Gruppi di codice\All_code;
2. fare click di destro su "All_code" e selezionare "Nuovo...";
3. inserire come nome "Localintranet_zone" (o quello che si preferisce, lo stesso dicasi per la descrizione) e premere Avanti;
4. come condizione selezionare "Zona", e come Zona scegliere "Intranet locale"; premere Avanti;
5. verificare che come set di autorizzazioni sia selezionato "FullTrust" e premere Avanti e quindi Fine.

Si potrebbe pensare di aver finito, ma bisogna tenere conto del fatto che il codice dell'area Intranet finirebbe con avere solo i permessi in comune tra il gruppo di codice appena creato e quello già esistente tra le impostazioni a livello di Computer, per cui è necessario impostare il flag *Final* per evitare che CAS cerchi altri gruppi di appartenenza per il codice oltre che questo:

6. fare click di destro sul gruppo di codice appena creato e scegliere "Proprietà";
7. sbarrare la casella "I livelli dei criteri al di sotto di questo non verranno valutati";
8. premere OK e chiudere mmc.

Anche qui fornisco l'equivalente modifica al file XML:

1. aprire con Blocco Note o con qualunque altro editor di codice il file %windir%\Microsoft.NET\Framework\v<versione>\CONFIG\enterprisesec.config;
2. individuare il nodo `configuration\mscorlib\security\policy\policylevel`;

3. individuare il seguente sottonodo

```
<CodeGroup class="UnionCodeGroup" version="1"
PermissionSetName="FullTrust" Name="All_Code" Description="Il gruppo di
(eccetera) ">
  <IMembershipCondition class="AllMembershipCondition" version="1" />
  <!-- eventuale altro codice -->
</CodeGroup>
```

e modificarlo inserendo il sottonodo che rappresenta il nostro nuovo gruppo di codice (il codice aggiunto è evidenziato):

```
<CodeGroup class="UnionCodeGroup" version="1"
PermissionSetName="FullTrust" Name="All_Code" Description="Il gruppo di
(eccetera) ">
  <IMembershipCondition class="AllMembershipCondition" version="1" />
  <CodeGroup class="UnionCodeGroup" version="1"
PermissionSetName="FullTrust" Attributes="LevelFinal "
Name="Localintranet_zone" Description=" ">
  <IMembershipCondition class="ZoneMembershipCondition" version="1"
Zone="Intranet" />
  </CodeGroup>
  <!-- eventuale altro codice -->
</CodeGroup>
```

4. Salvare e chiudere.

Nota: come si può vedere all'interno dei file XML è rappresentata la stessa identica struttura visibile tramite lo snap-in di mmc; gli utenti più esperti troveranno che spesso, per aggiustamenti di entità non eccessiva, la modifica diretta dei file XML risulta molto più rapida di tutti i click dell'utility .NET Framework Configuration.

Seconda possibilità: uso di uno strong name per riconoscere l'assembly cui concedere la fiducia

Un'alternativa più sicura a quest'approccio è dare selettivamente tutti i privilegi a degli assembly sulla base del loro strong name, e più in particolare in base alla loro chiave pubblica; questo approccio è particolarmente indicato in ambito aziendale, dove si può dire ai vari client del dominio di fidarsi ciecamente degli assembly che hanno la chiave pubblica usata dall'azienda per i suoi programmi interni. Questo approccio ha il vantaggio di essere più sicuro (si evita di dare pieni poteri a qualunque assembly situato nella LAN) ma più macchinoso: bisogna infatti ricompilare tutti i programmi di cui si vuole che i client si fidino aggiungendo lo strong name; vediamo ora come fare.

Creazione di una coppia di chiavi

La creazione di una coppia di chiavi è estremamente semplice, ma richiede che sul computer sia installato il .NET Framework SDK. Procedere in questa maniera:

1. Aprire il "Prompt dei comandi di SDK" (Start→[Tutti i] Programmi→.NET Framework SDK <versione>→Prompt dei comandi di SDK);

2. posizionarsi nella cartella in cui si vuole salvare la coppia di chiavi tramite i consueti comandi MS-DOS;
3. digitare `sn -k nomefile.snk` (dove *nomefile.snk* è il nome che si vuole assegnare al file che conterrà la coppia di chiavi) e premere Invio.

Aggiunta di uno strong name ad assembly di cui si possiede il sorgente

Aggiungere uno strong name ad assembly di cui si possiede il sorgente è un'operazione relativamente semplice: basta infatti aggiungere un attributo al file `assembly.cs` (o `.vb`, a seconda del linguaggio in cui è scritto il progetto).

Nota: è possibile procedere in questa maniera solamente se tutti gli assembly a cui l'eseguibile fa riferimento sono dotati di strong name: infatti gli assembly dotati di strong name possono richiamare solamente altri assembly con strong name. Se si utilizzano componenti di terze parti prive di strong name aggiungerglielo come descritto nel paragrafo successivo ("Aggiunta di uno strong name ad assembly di cui non si possiede il sorgente"), quindi seguire le istruzioni che seguono per l'eseguibile principale.

1. Aprire il progetto a cui aggiungere lo strong name;
2. aprire il file `assembly.cs` (o `.vb`, a seconda del linguaggio);
3. aggiungere in fondo al file l'attributo

```
[assembly: AssemblyKeyFile("path\\filename.snk")]
```

se il progetto è scritto in VB.NET la sintassi sarà leggermente diversa:

```
<Assembly: AssemblyKeyFile("path\\filename.snk")>
```

Tenere conto del fatto che *path* rappresenta il percorso in cui è situata la chiave relativo alla cartella in cui vengono creati i moduli oggetto e *filename.snk* è il nome del file che contiene la coppia di chiavi;

4. ricompilare il progetto.

Aggiunta di uno strong name ad assembly di cui non si possiede il sorgente

Può capitare che il proprio progetto faccia riferimento ad assembly privi di strong name di cui non si possiede il sorgente; questo può rappresentare un problema, perché, come già detto in nota, gli assembly con strong name (come dovrà essere la nostra applicazione) possono richiamare solamente assembly dotati anch'essi di strong name. Tuttavia esiste un modo per aggirare il problema, ossia il disassemblaggio in codice MSIL degli assembly privi di strong name e il loro riassettaggio con l'aggiunta dello strong name. Anche per questa operazione è richiesto il .NET Framework SDK.

Nota: molti contratti di licenza di eseguibili e di dll specificano a chiare lettere che ne è vietata la decompilazione e la modifica, quindi potrebbe non essere consentito eseguire il procedimento che segue.

1. Aprire il "Prompt dei comandi di SDK" (Start→[Tutti i] Programmi→.NET Framework SDK <versione>→Prompt dei comandi di SDK);
2. posizionarsi nella cartella in cui è situato l'assembly a cui applicare lo strong name;
3. in primo luogo disassemblare l'assembly, digitando

```
ildasm /all nomeassembly.ext /out=nomefile.il
```

dove *nomeassembly.ext* è il nome dell'assembly seguito dall'estensione (ad esempio, componente.dll) e *nomefile.il* è il nome del file dove verrà salvato l'output (ossia l'assembly decompilato);

4. quindi riassemblarlo aggiungendo lo strong name, inserendo

```
ilasm /out=nuovoassembly.ext /tipo /key="percorso\filename.snk"  
nomefile.il
```

dove *nuovoassembly.ext* è il nome da dare al nuovo assembly (ad esempio, component_sn.dll), "*percorso\filename.snk*" è il percorso (relativo alla cartella corrente) e il nome del file contenente la coppia di chiavi, *nomefile.il* è il nome del file dove è stato salvato l'output di ildasm (vedi punto precedente) e *tipo* è il tipo di file da generare (exe o dll);

5. ricompilare il progetto che referencia questo assembly avendo cura di rimuovere il riferimento al vecchio componente e aggiungerlo a quello nuovo (dotato di strong name) e di aggiungere lo strong name come descritto nel paragrafo precedente.

Creazione del gruppo di codice

Il procedimento è molto simile a quello del paragrafo "Modifica a livello enterprise":

1. aprire .NET Framework Configuration e portarsi al livello Console Root\.NET Framework <vers> Configuration\Risorse del computer\Criteri di protezione runtime\Enterprise\Gruppi di codice\All_code;
2. fare click di destro su "All_code" e selezionare "Nuovo...";
3. inserire come nome "EnterprisePublicKey" (o quello che si preferisce, lo stesso dicasi per la descrizione) e premere Avanti;
4. come condizione selezionare "Nome sicuro" e premere il pulsante "Importa..."; aprire un assembly firmato con il nome sicuro che è stato creato ai paragrafi precedenti;
5. assicurarsi che la casella di testo "Chiave" sia stata riempita con una serie di numeri e che i checkbox situati a sinistra di "Nome" e "Versione" siano deselezionati, quindi premere Avanti;
6. verificare che come set di autorizzazioni sia selezionato "FullTrust" e premere Avanti e quindi Fine;

7. fare click di destro sul gruppo di codice appena creato e scegliere "Proprietà";
8. sbarrare la casella "I livelli dei criteri al di sotto di questo non verranno valutati";
9. premere OK e chiudere mmc.

Distribuzione delle impostazioni sulla rete

Una volta creato sulla macchina di test il gruppo di codice che ha le caratteristiche richieste si rende necessario distribuire le impostazioni su tutto il dominio; si può procedere o per semplice copia del file *enterprisec.xml* sulle singole macchine oppure si possono utilizzare strumenti più avanzati che Windows Server ci mette a disposizione. Per poterlo fare in primo luogo abbiamo bisogno di creare un pacchetto msi che installi le impostazioni enterprise per poi distribuirlo tramite i Criteri di gruppo.

Creazione del pacchetto di installazione

1. aprire .NET Framework Configuration e portarsi al livello Console Root\ .NET Framework <vers> Configuration\Risorse del computer;
2. fare click di destro su "Criteri di protezione runtime" e selezionare "Crea package di distribuzione...";
3. assicurarsi che l'opzione "Enterprise" sia selezionata, quindi specificare il percorso in cui si desidera salvare il file di installazione (manualmente o tramite il pulsante "Sfoggia...");
4. fare click su Avanti e quindi su Fine.

Distribuzione del pacchetto tramite i criteri di gruppo

Nota: le istruzioni che seguono si riferiscono alla versione inglese di Windows 2003 Server; tali informazioni si applicano praticamente allo stesso modo anche alla versione italiana (ovviamente traducendo i nomi citati) e a Windows 2000 Server.

1. Copiare il pacchetto di installazione creato nel precedente punto su una share del server accessibile a tutti i client su cui si intende distribuire le nuove impostazioni;
2. aprire mmc (Start→Run...→Digitare "mmc"→Premere OK);
3. dal menu File selezionare "Add/Remove snap-in...", quindi premere "Add...", selezionare dalla lista "Group Policy Object Editor" e selezionare "Add";
4. dalla nuova finestra scegliere "Browse..." e selezionare la policy da modificare (solitamente "Default Domain Policy");
5. selezionare "Finish", "Close" ed "OK";

6. nella gerarchia di cartelle visualizzata nella parte sinistra dello schermo portarsi in "Console Root\- 7. aprire con Windows Explorer la share in cui è stato copiato il pacchetto di installazione e trascinare il pacchetto nella finestra del Policy Editor;
- 8. verificare che nella finestra sia selezionata l'opzione "Assigned" e selezionare OK.
- 9. chiudere mmc. L'installazione dovrebbe avvenire su tutti i client in tempi brevi.