Astrit Desku

SIGURIA E TË DHËNAVE TE APLIKACIONET SOFTUERIKE NË .NET

SIGURIA E QASJES ME KOD *(ang. Code Access Security CAS)*

Sistemet kompjuterike të lidhura në rrjetë, në ditët e sotme, shpesh janë të pambrojtura nga kode të krijuara nga burime pothuajse të panjohura. Kodi mund të jetë i bashkangjitur me email (ang.*e-mail*), mund të jetë i përmbajtur në dokumente apo mund të jetë i shkarkuar (ang.*downloaded*) nga Interneti. Fatkeqësisht, shumica e shfrytëzuesve të kompjuterëve janë të sprovuar drejtpërsëdrejti nga efektet e kodeve mobile keqdashëse sikurse janë viruset (ang.*viruses*) dhe krimbat (ang.*worms*) e të cilët i dëmtojnë apo edhe mund ti shkatërrojnë shënimet. E në të njëjtën kohë na sjellin dëme duke na kushtuar në kohë dhe para.
Shumica e mekanizmave të zakonshëm të sigurisë i japin të drejta shfrytëzuesit duke u bazuar në kreditencialet e tij (zakonisht fjalëkalimi) gjatë kyçjes në sistem dhe i kufizojnë resurset (më së shpeshti kufizimin ndaj qasjes në foldera dhe fajlla) në të cilat shfrytëzuesi është i lejuar t'i qaset. Në këto raste kodi është ekzekutuar duke shfrytëzuar identitetin, rolet dhe të drejtat e shfrytëzuesit i cili e ekzekuton atë kod, siç ishte rasti te siguria e bazuar në role.
Mirëpo kjo qasje ndaj sigurisë dështon në disa aspekte kështu: shfrytëzuesit përdorin kod nga shumë burime, disa prej tyre mund të jenë të dyshimta si; kodi mund të përmbajë gabime (ang.*bug*) apo dobësi (ang.*vulnerabilities*) e të cilat mundësojnë që kodi të jetë i shfrytëzueshëm nga kode të ndryshme keqdashëse dhe kështu kodi pastaj mund të bëhet i pakontrollueshëm nga ne pasi që mund të bëjë disa veprime të cilat ne nuk i dimë se kodi ynë mund ti bëjë. Si rezultat i kësaj sistemet kompjuterike mund të dëmtohen dhe shënimet private mund të rrjedhin në momentin kur shfrytëzuesi i kujdesshëm me maturi ekzekuton aplikacionin e mbushur me gabime apo me kode keqdashëse. Kodi i dëmtuar, pastaj mund t'i dëmtoj fajllat apo shënimet e tjera në të cilat shfrytëzuesi ka qasje për përdorimin e tyre, apo mund t'i merr përparësitë e autoritetit të shfrytëzuesit për t'i kryer të gjitha llojet e dëmeve të mundshme që ai mund t'i realizoj. Duke pasur parasysh këto lloje te rrezikut, edhe shfrytëzuesi me besueshmërinë më të lartë nuk mundet lirshëm që të shkarkoj dhe të ekzekutoj aplikacione nga burime të panjohura, nga frika se çfarë mund të bëjë kodi i aplikacionit të ekzekutuar në sistemin e tij kompjuterik apo në sistemet e tjera kompjuterike të lidhura në rrjetë, pjesë e të cilit është edhe ai.

Disa nga organizatat, problemin e sigurisë së kodit përpiqen që ta zgjedhin duke e "bllokuar" shfrytëzuesin e tyre, duke i vënë restrikcione në instalimin dhe ekzekutimin e aplikacioneve të ndryshme. Mirëpo kjo mënyrë "e vrazhdë" e sigurisë së kodit, përveç që kërkon administrim të gjerë të resurseve, zvogëlon edhe fleksibilitetin dhe përdorshmërinë e kompjuterit nga shfrytëzuesit gjatë realizimit të detyrave të tyre.
Shumica e mekanizmave të sigurisë së sistemeve operative kërkojnë që secila pjesë e kodit duhet plotësisht të jetë e besuar (ang.*trusted*) gjatë ekzekutimit. Prandaj akoma është e nevojshme për një zbatueshmëri të gjerë të mekanizmit të sigurisë i cili mundëson që kodi nga një sistem kompjuterik të ekzekutohet me sistemin e vet të mbrojtjes në

sistemin tjetër kompjuterik, me përjashtim atëherë kur nuk ka relacione të ashtuquajtura të mirëbesimit ndërmjet sistemeve kompjuterike.

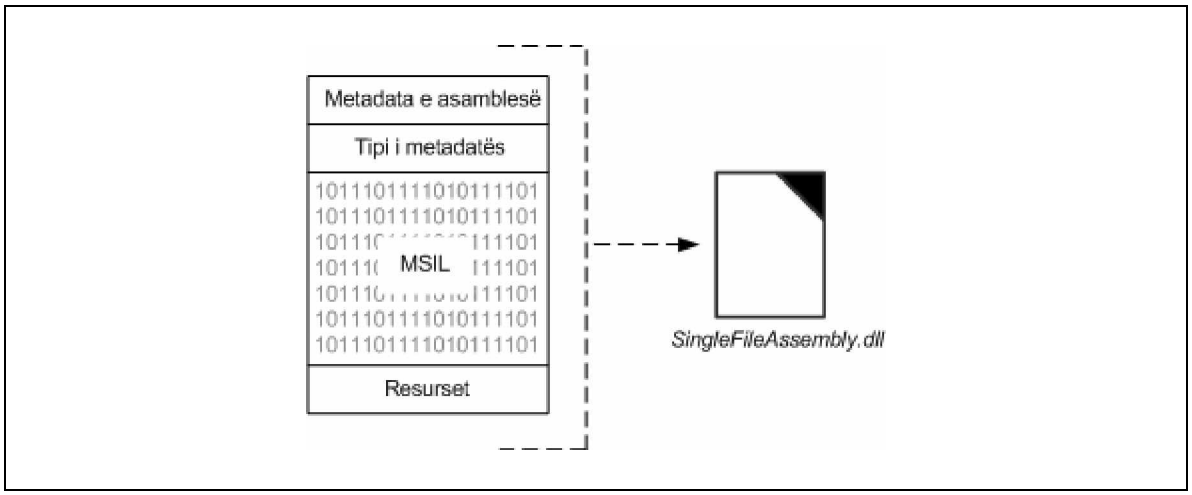
Për të ndihmuar në mbrojtjen e sistemeve kompjuterike nga kodet e lëvizshme (mobile) keqdashëse, për të lejuar kodin që nga ndonjë burimi i panjohur të ekzekutohet me sistemin e vet të mbrojtjes dhe për të ndihmuar në parandalimin e rrezikimit (të qëllimtë apo të rastësishëm) të sigurisë së kodit, .Net jep një mekanizëm të sigurisë i cili quhet **C**ode **A**ccess **S**ecurity (CAS) e që do të mund të përkthehej si Siguria e Qasjes me Kod.

CAS mundëson që kodi të jetë i besueshëm në nivele të caktuara, varësisht nga origjina e kodit dhe në varësi nga aspektet e identitetit të kodit. Gjithashtu CAS, siguron nivele të caktuara të besueshmërisë në kod, në këtë mënyrë minimizohet sasia e kodit që duhet të jetë plotësisht e besueshme në mënyrë që të ekzekutohet.
Duke përdorur CAS mund të zvogëlohen gjasat që kodi të keqpërdoret nga kodet keqdashëse apo nga kodi i mbushur me gabime. CAS, gjithashtu mund ta zvogëloj përgjegjësinë e shfrytëzuesit, sepse shfrytëzuesi mund të specifikoj vendosjen e operacioneve që i lejohen kodit t'i kryej si dhe operacionet të cilat kodi nuk do të mund t'i kryente asnjëherë. Gjithashtu CAS, mund të ndihmoj në minimizimin e dëmeve të cilat mund të jenë si rezultat i dobësimit të sigurisë në kod.
Në CAS, qasja është e bazuar në identitetin e kodit e jo në identitetin e shfrytëzuesit i cili e ekzekuton atë kod. Disa shembuj të operacioneve të cilat mbrohen nga CAS mund të ishin:

- Krijimi, fshirja dhe modifikimi i fajllave dhe folderave
- Leximi dhe shkruarja në regjistrat e e sistemit operativ Windows
- Inicializimi i lidhjes në rrjetë të kompjuterëve
- Krijimi një aplikacioni të ri
- Ekzekutimi i kodit burimor (ang.*native*) apo të pamenaxhueshëm

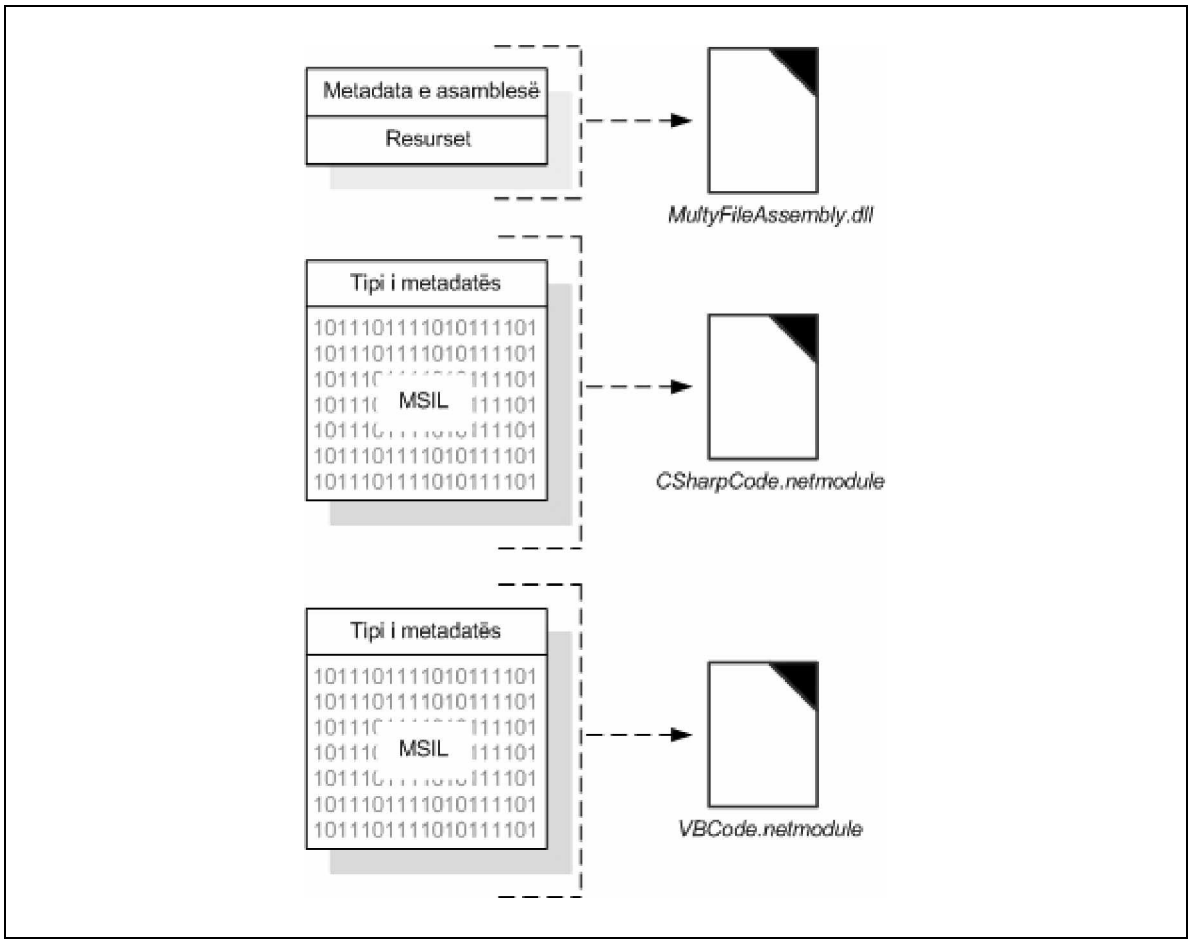
ASAMBLETË (ang.*ASSEMBLIES*)

Një asamble përmban një apo më shumë tipe të dhënash të kompajluara në ***M**irosoft **I**ntermeditae **L**anguage* (*MSIL*) që do të mund të përkthehej si gjuha e ndërmjetme e Mikrosoftit, me fjalë të tjera, një asamble përmban kodin të cilin e ekzekuton ***C**ommon **L**anguage **R**untime* (CLR) në shqip do të mund të ishte gjuha e përgjithshme e ekzekutimit në kohë.
.NET-i e përdorë asamblenë si njësinë themelore ekzekutuese dhe interpretuese. Ekzistojnë dy tipe të asamblesë, secila prej tyre ndryshon varësisht nga ajo se ku ruhen përmbajtjet e asamblesë. Tipi më i thjeshtë dhe më i zakonshëm është asambleja me fajll të vetëm (ang. *single-file assembly*), ku tërë përmbajtja e asamblesë ruhet në një fajll të vetëm. Siç shihet nga figura e mëposhtme, të katër komponentët e asamblesë ruhen në një fajll të quajtur *SingleFileAssembly.dll*.



Asambleja me fajll të vetëm

Tipi i dytë i asamblesë është asambleja më shumë fajlla (ang. *multifile assembly*), ku përmbajtjet e asamblesë ruhen në më shumë se një fajll, si në figurën në vijim.

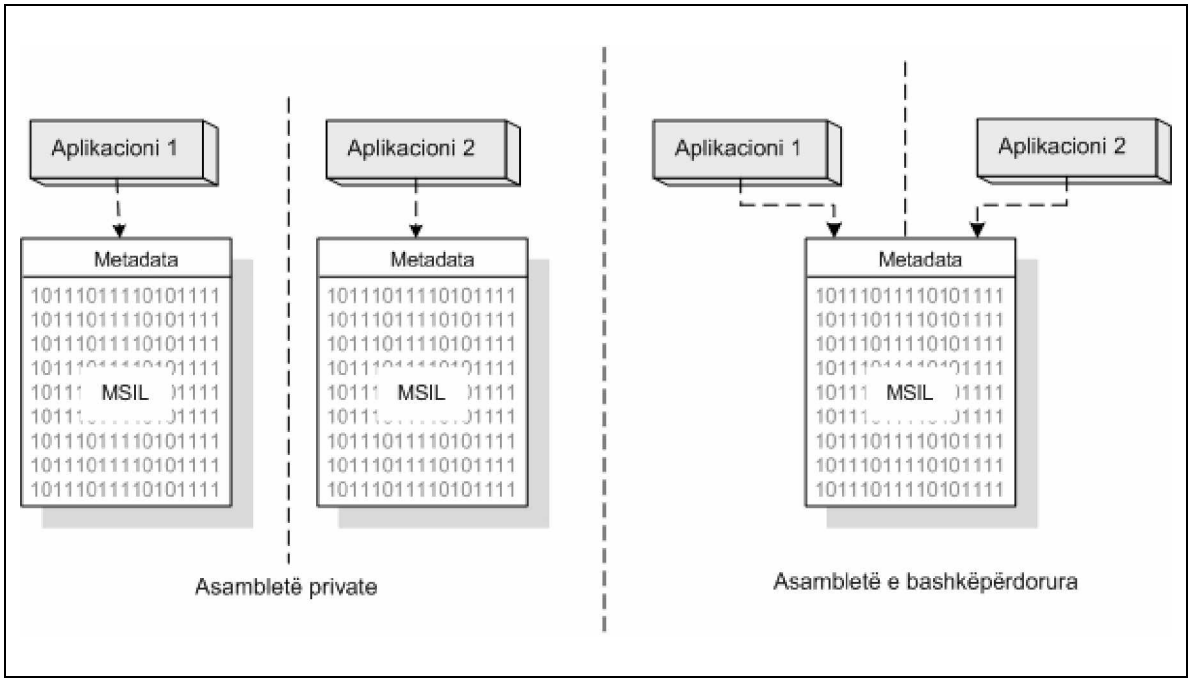


Asambleja me më shumë fajlla

Në asamblenë me më shumë fajlla, metadata e asamblesë dhe ndonjë resurs tjetër ruhen në një fajll (*MultiFileAssembly.dll*), derisa kodi i MSIL përmbahet në një apo më shumë module, ku secili prej tyre përmban një apo më shumë tipe të dhënash dhe një tip metadata.
Përparësia e asamblesë me më shumë fajlla është se, secili modul mund të përmbajë tipet e shkruara në gjuhë të ndryshme të .NET –it (p.sh. C#, VB.NET etj.). Siç shihet nga figura më lartë, në asamble janë përdorur fajllat *CsharpCode.netmodule* dhe *VBCode.netmodule* duke paraqitur tipet e të dhënave të shkruara në C# dheVB.NET (Visual Basic.NET).

Asambletë e përbashkëta (Shared Assemblies)

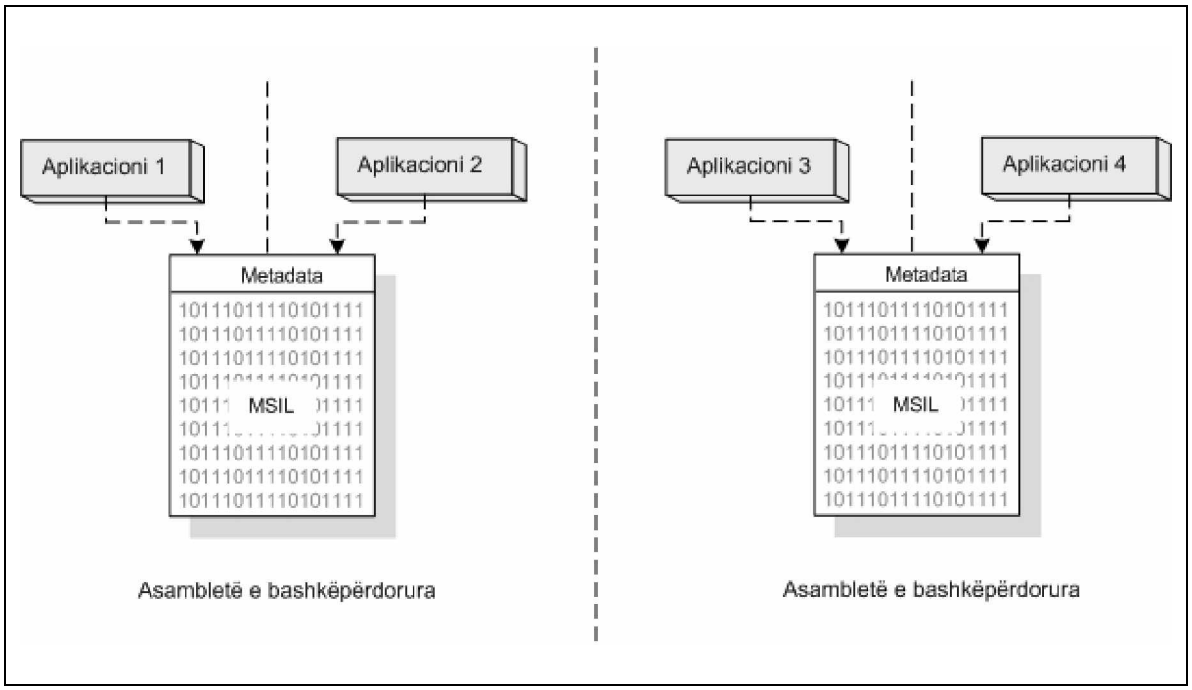
Asambletë mund të jenë private ose të përbashkëta (ang.shared), si në figurën në vijim. Secili aplikacion i cili e përdorë asamblenë private e përmban një kopje të saj, e cila ruhet pranë komponentëve të aplikacionit. Nëse janë dy aplikacione të instaluara në të njëjtin kompjuter dhe secili aplikacion i referohet të njëjtës asamble private, atëherë do të jenë të instaluara dy kopje të asamblesë. Sa herë që instalohet një aplikacion i ri që i referohet një asambleje private, do të krijohet një instancë e re e asamblesë në disk. Secila asamble është e pavarur nga tjetra dhe secili aplikacion është tërësisht vetë funksional apo i pavarur.



Asambletë private dhe të përbashkëta

Përndryshe, disa aplikacione mund ta shfrytëzojnë një instancë të vetme të asamblesë së përbashkët. Asambleja vendoset në një vend të përgjithshëm (ndonjë follder *share* ose në server në rrjetë) ku mund të kenë qasje shfrytëzuesit dhe çdo aplikacion të cilit i nevojitet asambleja e përdorë asamblenë e njëjtë. Mund të instalohen më shumë se një asamble të

përbashkëta dhe grupet e aplikacioneve mund të referohen në instanca të ndryshme, si në figurën në vijim.



Grupe aplikacionesh përdorin asamblenë e njëjtë

GAC

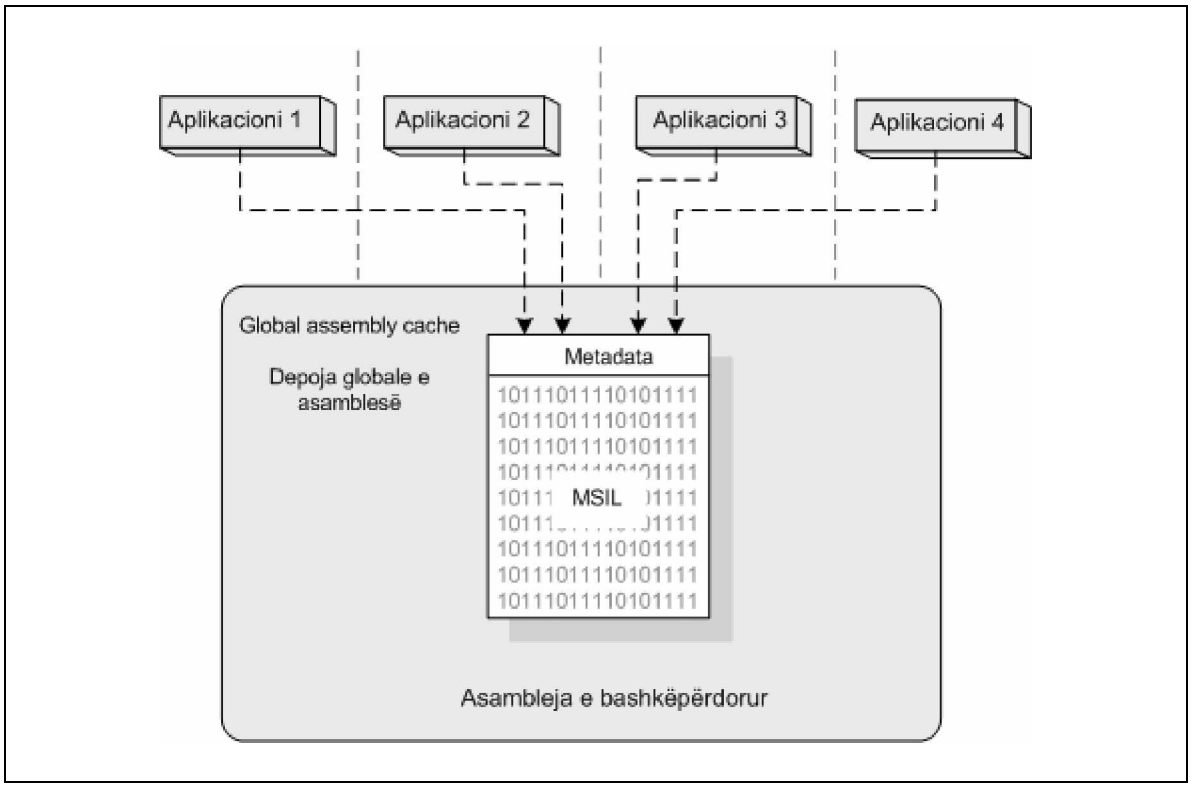
.NET përmban të ashtuquajturin ***G****lobal* ***A****ssembly* ***C****ache* (*GAC*) që është një depo qendrore për asambletë e përbashkëta. Të gjitha aplikacionet të cilat i referohen asamblesë së përbashkët e të ruajtur në GAC përdorin të njëjtin fajll, si në figurën në vijim. Kështu që nuk është e nevojshme që të konfigurohen aplikacionet për të gjetur asamblenë e përbashkët në lokacionin e caktuar, GAC gjithmonë është i përdorshëm për të gjitha aplikacionet.

Pra GAC mund të përkufizohet si një makinë ruajtëse e asambleve të përbashkëta, në të cilën mund të instalohet një asamble e përbashkët dhe e cila shfrytëzohet nga disa aplikacione.

Ekzistojnë disa arsye pse duhet instaluar një asamble në GAC:

- *Konfigurimi i aplikacionit*
  .NET automatikisht kërkon për ndodhjen e asambleve në GAC dhe për këtë arsye nuk është e nevojshme që aplikacioni të konfigurohet për të kërkuar asamblenë në vendin e caktuar, siq është rasti kur një folder përdoret për të ruajtur asambletë.
- *Sigurimi i fajllit*
  Lokacioni fizik i cili përdoret nga GAC për të ruajtur fajllat e asamblesë ndodhet në follderët e *WINNT* ose të *Windows*-it. Këta folldera zakonisht (nga administratorët e sistemit) mbrohen me nivelin më të lartë të sigurisë së Windows-it se sa folderët tjerë të zakonshëm. Andaj për të instaluar një asamble në GAC kërkohet që shfrytëzuesi të ketë autorizimet e administratorit, kështu që zvogëlohen shanset e keqpërdorimit.

- Ø *Administrimi*
  Instalimi i një asambleje të përbashkët në GAC është një proces i lehtë dhe relativisht i thjeshtë administrativ, mirëpo rëndësia qëndron në atë se i afekton të gjitha aplikacionet të cilat varen në atë asamble.
- Ø *Interpretim i njëkohshëm*
  Shumë kopje të asambleve me emër të njëjtë, mirëpo me përmbajtje të ndryshme të versionit, mund të mbahen në GAC.



GAC i strehon asambletë e përbashkëta

- o *Shënim*: Kur thuhet se aplikacionet e kanë një asamble "të përbashkët", me këtë nënkuptohet që ato përdorin të njëjtin fajll të diskut. Secili aplikacion e merr një kopje të tipit të të dhënave që përmbahen në asamble dhe jo që të dhënat të bashkëpërdoren ndërmjet aplikacioneve.

Përparësitë e përdorimit të asamblesë së përbashkët janë administrative. Konfigurimi i sigurisë aplikohet një herë në asamble dhe pastaj i afekton të gjitha aplikacionet të cilat referohen në atë asamble, përderisa në asambletë individuale secila instancë e saj duhet të konfigurohet individualisht. Apo në rast ndryshimi të asamblesë së përbashkët, ndryshimet gjithashtu afektohen në të gjitha aplikacionet që i referohen asaj asambleje, e kundërta ndodhë te asambletë private, ku ndryshimet duhen të bëhen në secilën asamble individualisht.

Emrat e fortë (ang.Strong Names)

Çdo asamble e përbashkët duhet të ketë të ashtuquajturin *emër i fortë* (*strong name*). Një emër i fortë përbëhet nga emri i asamblesë, versioni dhe metadata e kulturës si dhe një çelës publik kriptografik dhe nënshkrim digjital. Pasi që është e vështirë që dy asamble të ndryshme të kenë të njëjtë emrin e fortë, Mikrosofti konsideron që emri i fortë të jetë unik për të identifikuar një asamble. Këtë pozicion e mundësojnë dy karakteristika të nënshkrimit digjital:

- Nënshkrimet digjitale janë të ndërlidhura me gjenerimin e një të ashtuquajturi *hash code* (që në shqip do të ishte si *kod i përzier*) nga përmbajtja e asamblesë. Ky kod i përzier vepron si "gjurmë gishtrinjësh" për përmbajtjen e asamblesë së caktuar dhe në këtë mënyrë është e vështirë që të gjenden dy asamble me "gjurmë gishtrinjësh" të njëjtë.
- Çiftat e çelësave kriptografik të gjeneruar në mënyrë të rastit, përdorën për të krijuar nënshkrimin digjital.

Rastësia e gjenerimit të çiftit të çelësave dhe unifikimi i gjurmëve të gishtrinjëve të mundësuara nga kodi i përzier dhe nënshkrimi digjital, nënkupton që një emër i fortë, për të gjitha qëllimet praktike siguron identifikim unik për një asamble.

Unifikimi i një emri të fortë i mundëson .NET-it që të verifikoj përmbajtjen e një asambleje në mënyrë që ta mbrojë kundër keqpërdorimeve. Kodi i përzier i gjeneruar nga përmbajtja e asamblesë nënkupton që duke ndryshuar madje një formulim të vetëm të MSIL –it do të zhvlerësohet kodi digjital, ku si pasojë .NET nuk e merr më parasysh atë asamble. Kjo mbrojtje kundër keqpërdorimeve është karakteristikë fundamentale e sigurisë së emrave të fortë të asambleve, si dhe është çelësi i funksionimit të ciklit jetësor të aplikacioneve të .NET. Emrat e fortë, gjithashtu janë karakteristikë themelore për sigurinë e qasjes me kod, andaj edhe u trajtuan në këtë paragraf.

Çiftët e çelësave

Hapi i parë në drejtim të krijimit të një Emri të fortë është gjenerimi i çelësave privat dhe publik. Çelësi privat përdoret për të krijuar nënshkrimin digjital dhe duhet të mbahet sekret, deri sa çelësi publik përmbahet në asamble si pjesë e emrit të fortë dhe përdoret nga .NET-i për të validuar nënshkrimin digjital dhe për të përcaktuar nëse përmbajtja e një asambleje ka qenë e keqpërdorur më parë apo jo.

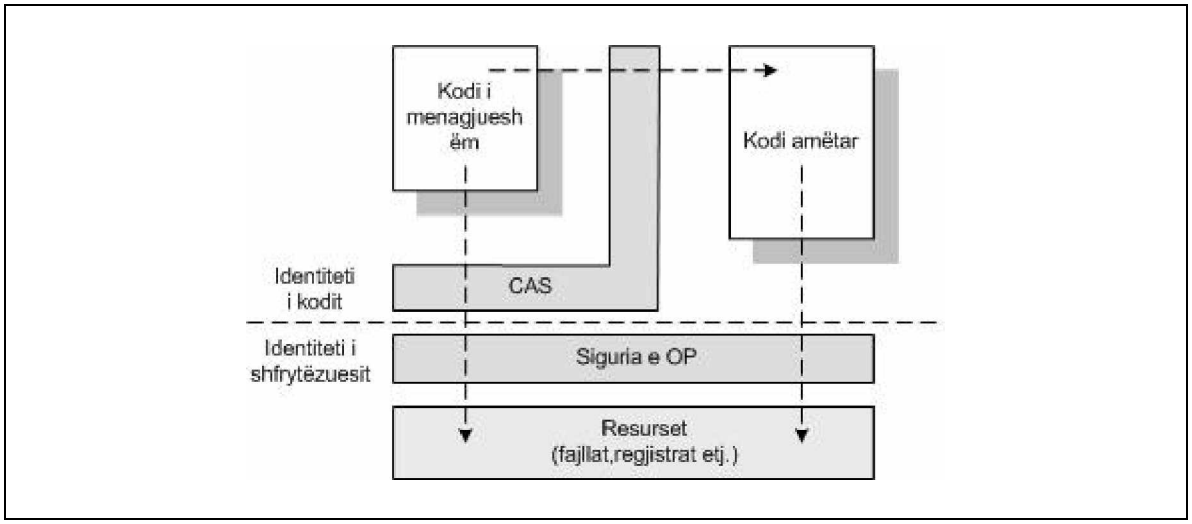
Botuesi i certifikatave

Emrat e fortë sigurojnë identitetet unike për asambletë dhe i mbrojnë ato kundër keqpërdorimeve, mirëpo nuk përmbajnë ndonjë informacion për botuesin e identitetit të asamblesë. .NET përkrah skemën e të ashtuquajturit *Signcode,* në shqip do të mund të ishte si *Kodi i nënshkruar*, i cili kërkon një publikues për të vërtetuar se a është identiteti në autoritetin e mirëbesimit dhe e përmban softuerin për certifikimin e publikuesit (ang. *Software publisher's certificate* SPC). Emrat e fortë dhe kodi i nënshkruar janë dy teknologji komplementare ndërmjet vete dhe të dyja mund të aplikohen në të njëjtën asamble. Kur të përdoret Kodi i nënshkruar, atëherë nënshkrimi digjital gjenerohet duke shfrytëzuar komponentët e çelësit privat të certifikatës dhe futet në asamble së bashku me komponentët publike të SPC. Nënshkrimi digjital kujdeset për lidhjen ndërmjet SPC dhe asamblesë së nënshkruar. Pjesët tjera të komponentëve nuk mund ta nënshkruajnë

asamblenë me SPC e njëjtë, vetëm nëse e njohin vlerën e çelësit privat i cili nuk përfshihet në asamble.

Siguria e Windowsit dhe Siguria e Qasjes me Kod

Siguria e qasjes me kod (CAS) nuk e zëvendëson apo nuk e hedhë poshtë sigurinë e siguruar nga sistemi operativ i Windows-it. CAS efektivisht është një shtresë tjetër e sigurisë në të cilën kodi i menaxhueshëm duhet të kalojë, përpara se ai të bashkëveproj me resurset e mbrojtura të sistemit siç janë hard disku dhe regjistrat e Windows-it. Ndryshimi më i rëndësishëm në mes të CAS dhe sigurisë së Windows-it është se CAS, sigurinë e bazon në identitetin e kodit i cili e ekzekuton ndonjë veprim, përderisa Windows-i sigurinë e mbështetë në identitetin e shfrytëzuesit në interes të të cilit ekzekutohet kodi. Në figurën e mëposhtme është paraqitur ky relacion.



Mardhënjet ndërmjet CAS dhe sigurisë së OS (Sistemit Operativ)

Mardhënjen në mes këtyre dy sistemeve të sigurisë e ilustron shembulli në vijim. Kështu, nëse një aplikacion i menaxhueshëm tenton që të shkruaj një fajll në diskun e ngurtë, atëherë CAS së pari duke vlerësuar autorizimet e lejuara në asamble në rastin kur ajo është ngritur, përcakton nëse kodi ka autoritetin e duhur që i mundëson të shkruarit e fajllit. Nëse kodi nuk ka autoritetin e nevojshëm, atëherë operacioni i të shkruarit do të dështoj dhe runtime plason një të ashtuquajtur qortim sigurie. Në anën tjetër, nëse kodi ka autorizim, atëherë runtime do të bashkëveproj me sistemin operativ ku me anë të shfrytëzuesit aktual do t'i qaset fajllit. Nëse autorizimet e Windows-it nuk e lejojnë shfrytëzuesin që të shkruaj në fajllin e caktuar, atëherë qasja refuzohet dhe runtime sërish jep një vërejtje apo qortim.
Siq shihet edhe nga figura e mëparshme, CAS kontrollon gjithashtu edhe mundësinë që kodi i menaxhueshëm të thërret kodin e pamenaxhueshëm apo kodin amtar (ang.*native code*) siq është WIN32 API. Kjo është e rëndësishme sepse në momentin kur një aplikacion i menaxhueshëm e thërret një kod të pamenaxhueshëm, atëherë nuk ekziston ndonjë shtresë e sigurisë së CAS-it që do t'i kufizoj veprimet e kodit të pamenaxhueshëm që do t'i kryente në emër të kodit të menaxhueshëm. Në të tilla raste, në vend që t'i

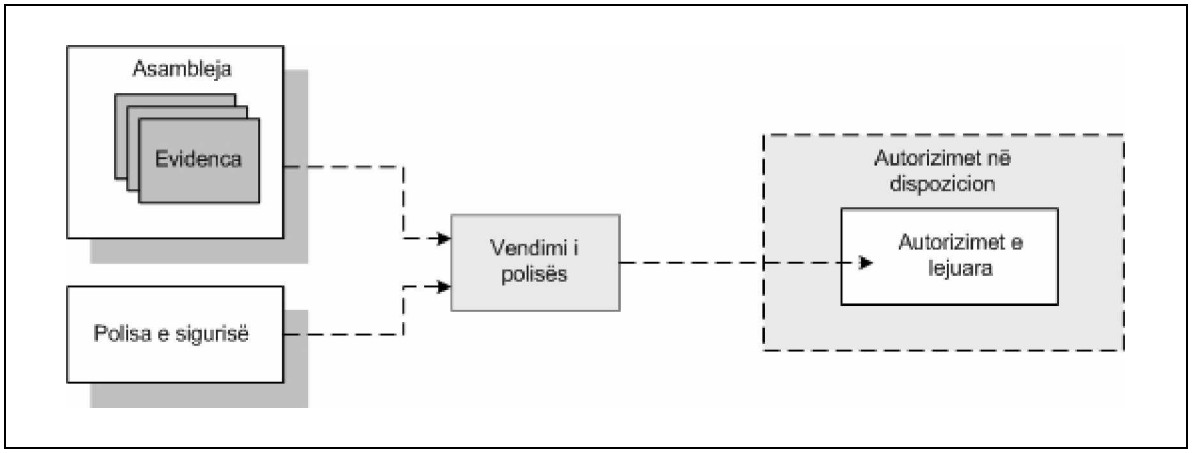
menaxhojmë autorizimet duke u bazuar në identitetin e kodit, vijmë në situatë kur vetëm autorizimet e shfrytëzuesit aktual mund t'i kufizojnë veprimet e kodit.

ELEMENTET E SIGURISË SË QASJES ME KOD

Janë tri elemente kryesore nga të cilat përbëhet siguria e qasjes me kod e ato janë:

- *Evidenca* (ang.*evidence*),
- *Polisa e sigurisë* (ang.*security policy*) dhe
- *Autorizimet* (ang.*permissions*).

Në momentin kur ngritët apo ngarkohet një asamble, runtime i shqyrton karakteristikat e ndryshme të asamblesë përkatësisht evidencës, për të përcaktuar identitetin e një kodi. Duke u bazuar në vënjën e konfigurueshme të disa rregullave (polisa e sigurisë), runtime e shfrytëzon evidencën e asamblesë si parametër hyrës për një proces tjetër të quajtur *policy resolution* e që do të mund të përkthehej si *vendim i polisës*. Rezultati i fituar nga vendimi i polisës është një grup i operacioneve dhe resurseve të mbrojtura, në të cilat kodi i një asambleje ka qasje apo autorizim.
Gjatë ekzekutimit, kodi tenton që të kryej veprime të ndryshme. Nëse kodi tenton që të ekzekutoj një operacion të mbrojtur apo operacion i cili është i pajisur me sistemin e mbrojtjes, atëherë runtime do të kërkoj autorizimet e lejuara në asamble dhe përcakton, nëse ato autorizimet janë apo jo të nevojshme për të lejuar veprimin që të ekzekutohet më tutje.
Në vijim do të diskutohen me detajisht këto tri elemente, ndërsa mardhënjet në mes tyre janë ilustruar në figurën vijuese.



Mardhënjet ndërmjet evidencës, polisës së sigurisë dhe autorizimeve

EVIDENCA (ang.EVIDENCE)

Sistemet e kompjuterizuara për të vërtetuar identitetin e një personi, shpesh përdorin karakteristikat biometrike siç janë gjurmët e gishtrinjëve, struktura e retinës së syrit etj. Në analogji me këtë .NET vërteton identitetin e një asambleje duke u bazuar në karakteristikat e nxjerrura nga përbërja e asamblesë, struktura dhe lokacioni burimor i saj. .NET-i i referohet këtyre karakteristikave identifikuese si *evidencë* dhe i përdorë ato për të përcaktuar veprimet dhe resurset në të cilat kodi i asamblesë ka autorizim që t'u qaset. Pra evidenca i përmban karakteristikat e një asambleje të përdorura nga runtime për të vërtetuar identitetin e asamblesë.

Disa tipe të evidencës trashëgohen në strukturë dhe përbërje të asamblesë, siç janë emri i fortë i asamblesë, vlera e përzier (*hash*) e përbërjes së asamblesë etj. Tipet tjera të evidencës përcaktohen nga runtime, duke u bazuar në karakteristika si veb faqet ose *URL* nga ku asambleja është ngarkuar.

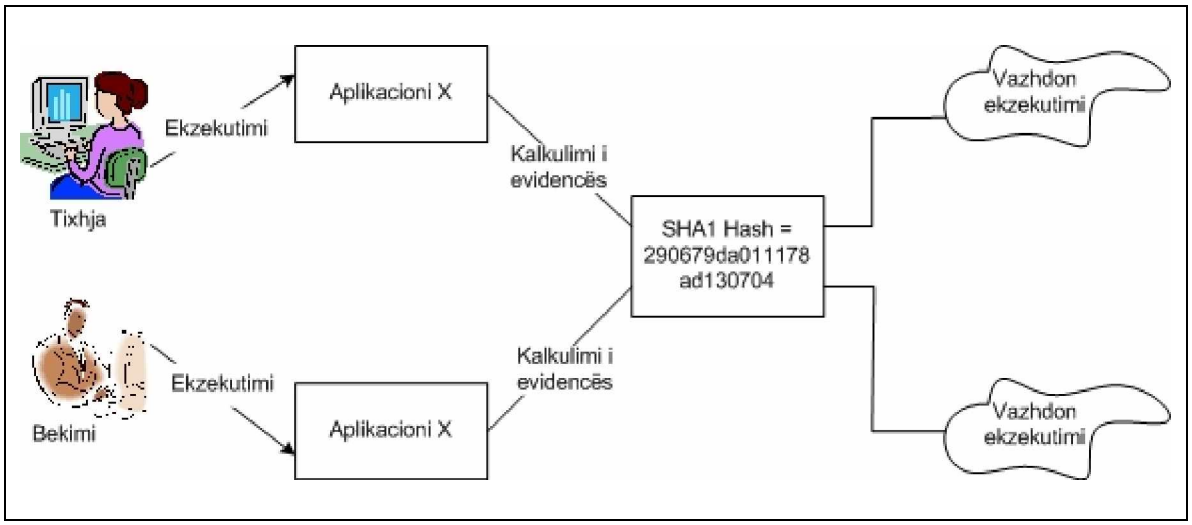
Kombinimi i evidencave vërteton identitetin e asamblesë. Kur runtime ngarkon një asamble, ai krahason identitetin e asamblesë me rregullat e definuara në polisën e sigurisë të cilat përcaktojnë autorizimet që i jepen apo dhurohen asamblesë e që quhen *grant set*. *Grant set* definon operacionet të cilat mund t'i kryej kodi i cili përmbahet në asamble.

Është e rëndësishme të kuptohet se, asambleja e njëjtë mund të ketë identitete shumë të ndryshme bazuar në evidencën e runtime. Kështu nëse një asamble ekzekutohet nga interneti, evidenca e saj do të jetë e ndryshme nga evidenca e asamblesë e cila ekzekutohet në diskun e ngurtë, nëse ajo (asambleja) më parë është shkarkuar nga interneti.

Aplikimi i evidencës në ekzekutimin e kodit

Siç u cek edhe më lartë, evidenca është një koleksion informacionesh i mbledhur rreth ekzekutimit të kodit, pa marrë parasysh se kush e ekzekuton atë. Një shembull i evidencës rreth kodit është vlera e përzier e tij *SHA1*. Kjo vlerë është një pjesë e kriptografuar e informacionit i cili është unik për pjesë të ndryshme të kodit.

Në figurën e më poshtme është paraqitur një shembull për këtë. Nëse marrim dy shfrytëzues *Tixhe* dhe *Bekim* e të cilët e ekzekutojnë të njëjtin aplikacion, por nga vende të ndryshme p.sh. aplikacionin X, mekanizmi i sigurisë do të siguroj saktësisht evidencën e njëjtë lidhur me aplikacionin X. Kur *Tixhja* e ekzekuton aplikacionin X, vlera e përzier *SHA1*(*hash value*) e saj nuk ndryshon nga ajo kur *Bekimi* e ekzekuton atë.



Kalkulimi i evidencës për aplikacionin e njëjtë, kur ai ekzekutohet nga dy shfrytëzues të ndryshëm

Në këtë mënyrë aplikacionet e .NET mund të shihen si shfrytëzues të ndryshëm, kur evidenca përdoret si kreditencial për një aplikacion. Pasi që është thënë edhe më herët se nuk mund të ekzistojnë dy aplikacione me evidencë të njëjtë. Pra me anë të evidencës siguria e .NET-it e dallon një pjesë të kodit nga tjetra.

Gjithashtu është e rëndësishme të theksohet se evidenca aplikohet vetëm te kodet të cilat janë në ekzekutim apo aktualisht janë duke u ekzekutuar. Pra .NET-i nuk e parallogaritë evidencën për kodin i cili nuk është në ekzekutim, ajo kalkulohet dhe aplikohet në mënyrë dinamike, vetëm atëherë kur kodi është duke u ekzekutuar. Faktikisht disa evidenca është e pamundur që të njihen përpara se kodi të ekzekutohet. P.sh. origjina e URL-së për një aplikacion është e pamundur të dihet përpara se se ai të ekzekutohet pasi që aplikacioni mund të ekzistoj në lokacione të ndryshme ose mund të jetë i adresuar nga URL të ndryshme që të dërgojnë në lokacion të njëjtë. P.sh. ftp://www komtel-pe.com dhe http://www.komtel-pe.com të dërgojnë te veb faqja e njëjtë.

- *Shënim*: URL është shkurtesë nga ***U****niform* ***R****esource* ***L****ocator* e cila është adresë globale e dokumenteve dhe e resurseve të tjera në Worl Wide Web.

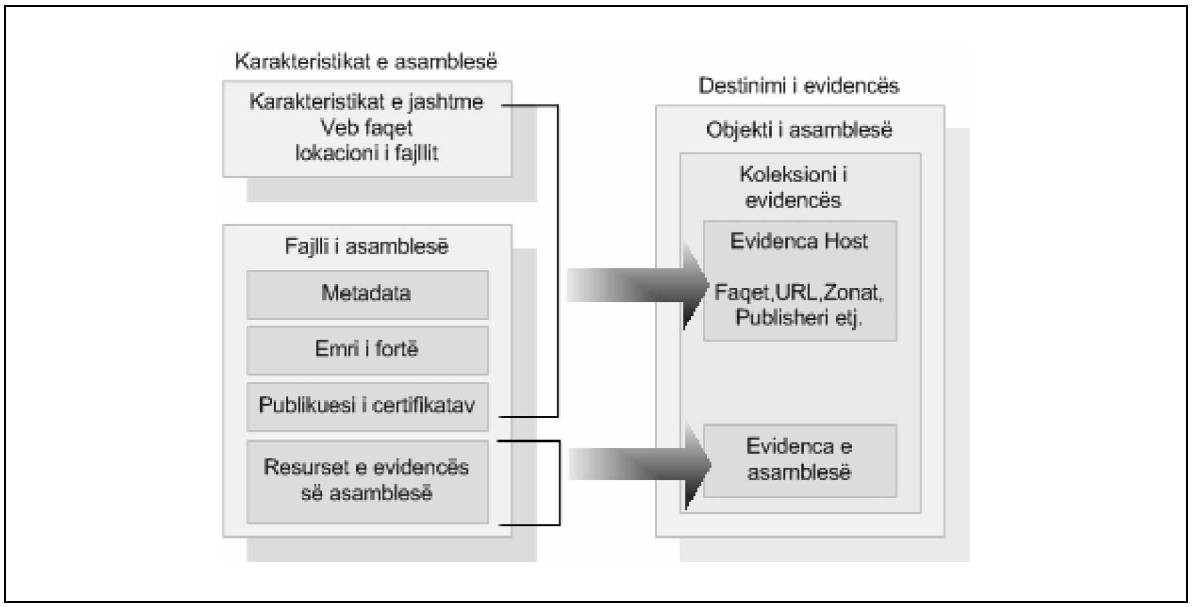
Llojet e evidencës

Ekzistojnë dy lloje të evidencës:

- *evidenca host* (ang. *host evidence*) dhe
- *evidenca e asamblesë* (ang. *asembly evidence*)

Qëllimi i të dyja llojeve të evidencës është që të japin identitetin e asamblesë, të cilën runtime e përdorë për t'i përcaktuar autorizimet që i lejohen asaj. Sipas vlerës së përcaktuar (ang.*by default*) runtime i .NET-it e anashkalon evidencën e asamblesë, vetëm në rastet kur ajo mundësohet (*enabled*) përmes implementimit të CAS me anë të procesit të polises së sigurisë.

Në figurën e mëposhtme janë paraqitur karakteristikat e gjetura në asamble dhe mënyra se si runtime i përdorë ato për të krijuar host evidencën dhe evidencën e asamblesë.



Burimet dhe destinacionet e evidencës

Evidenca Host

Host-i në bashkëpunim me bartësin e komponentëve të asamblesë së CLR, e mundësojnë host evidencën për një asamble. Host-i duhet të jetë në njërën nga standardet e host-eve të .NET (si Komand Shell,Internet Explorer,ose ASP.NET), ose të jetë pjesë e besueshme e kodit të menaxhueshëm i cili e ngarkon asamblenë në runtime.
Pra "host" mund të jetë diçka e cila inicion CLR-në, mund të jetë kod i pa menaxhueshëm, apo host mund të jetë pjesë e kodit të menaxhueshëm i cili e hedhë (*launch*) një pjesë tjetër të kodit të menaxhueshëm.

Evidenca e asamblesë

Qëllimi i evidencës së asamblesë është serializimi (ndarja pjesë – pjesë) i objekteve të evidencës dhe futja e tyre në asamble. Kur runtime e ngritë (*loads*) asamblenë ai i ekstrakton dhe i deserializozon objektet e futura dhe i fikson ato në asamble si evidencë të asamblesë.
Në shembullin e mëposhtëm, praktikisht mund të shihet përdorimi i të dy llojeve të asamblesë.

```
using System;
using System.Collections;
using System.Reflection;
using System.Security;
using System.Security.Policy;

public class ShowEvidence
```

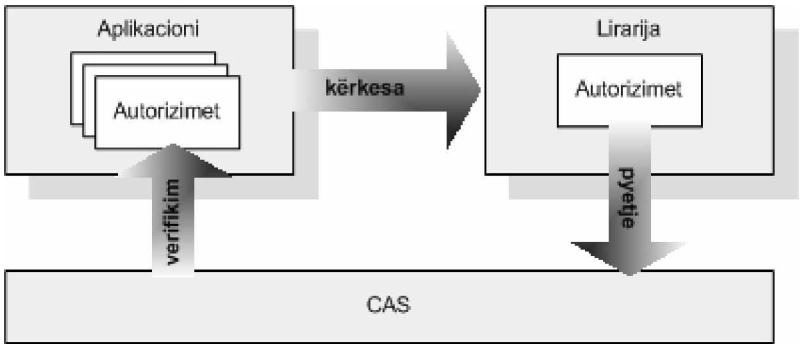
```
{
  public static void Main()
  {
    Assembly thisAssembly = Assembly.GetExecutingAssembly();
    Evidence ev = thisAssembly.Evidence;

    Console.WriteLine("Host Evidence:");
    IEnumerator enumerator = ev.GetHostEnumerator();
    while (enumerator.MoveNext())
    {
      Console.WriteLine(enumerator.Current + Environment.NewLine);
    }
    Console.WriteLine(Environment.NewLine);



    Console.WriteLine("Assembly Evidence:");

    enumerator = ev.GetAssemblyEnumerator();
    while (enumerator.MoveNext())
    {
      Console.WriteLine(enumerator.Current + Environment.NewLine);
    }
  }
}
```

AUTORIZIMET (ang. PERMISSIONS)

Mekanizmi i CAS-it në brendinë e tij bazohet në konceptin e autorizimeve. Autorizimet mund të kërkohen, mohohen, lejohen dhe anulohen nga asambletë dhe polisa e sigurisë. Mirëpo qëllimi kryesorë i CAS është që t'i siguroj shfrytëzuesit kontrollimin e veprimeve dhe resurseve në të cilat kodi ka qasje. P.sh. Administratori i sistemit dëshiron që ta ndalë ekzekutimin e aplikacioneve nga Interneti, apo dëshiron që vetëm kodet e shkruara nga Mikrosofti të kenë mundësi që të shkruajnë në regjistrat e Windows-it.
Shumica e autorizimeve paraqesin qasjen në veprimet dhe resurset të cilat janë subjekt në kontrollimin e sigurisë, sikurse janë mundësia për të krijuar domene të aplikacionit, ose mundësia për të shkruar në logun e Windows-it. Autorizimet e tjera paraqesin identitetin e kodit të nxjerrur nga evidenca e asamblesë, sikurse janë botuesi i certifikatave dhe emrat e fortë që i kemi diskutuar në paragrafet e mëhershme.

Kodi i cili kujdeset për funksionalitetin ose akcesibilitetin e resurseve duhet të jetë i mbrojtur dhe e shfrytëzon sigurinë për të pyetur nëse ndonjë kod i thirrur i ka autorizimet specifike. Runtime dhe klasat e librarisë së .NET e përdorin këtë mundësi gjerësisht për të mbrojtur qasjen në veprime dhe resurse, duke përfshirë këtu edhe mundësinë për të ekzekutuar kod të pamenaxhueshëm apo qasjen në diskun e fortë dhe regjistrat e Windows-it.
Në figurën e mëposhtme është ilustruar përdorimi i autorizimeve në zbatimin e sigurisë në nivel të kodit dhe demonstrimi i veprimit më të zakonshëm të CAS të quajtur *security demand* e në shqip do të mund të ishte *kërkesë e sigurisë.*

Kur një kod i asamblesë e thërret një anëtar të librarisë së mbrojtur, libraria duke përdorur objektin e autorizimeve kërkon që runtime ta sigurojë që asambleja thirrëse i ka autorizimet e nevojshme. Runtime vlerëson objektet e autorizimit të lidhura me asamblenë përkatëse dhe i konfirmon librarisë, nëse asambleja ka autorizimet e caktuara.



Kërkesa e sigurisë

Siç shihet nga figura më lartë, është përgjegjësi e librarisë që të kërkoj nga CAS që të garantoj që aplikacioni thirrës (ai i cili e thirrë librarinë) ka autorizimin e nevojshëm për përdorimin e funksioneve të saj. Libraria gjithashtu përcakton veprimin e duhur që duhet të ndërmerret nëse kërkesa e sigurisë dështon. Ky proces do të diskutohet në paragrafet e ardhshme.

Tipet e autorizimeve

Klasat e librarisë së .NET i përdorin autorizimet për të kontrolluar qasjen në funksionet e siguruara apo lejuara. Runtime gjithashtu i përdorë autorizimet për të ndaluar qasjen në disa veprime të cilat janë kritike për sigurimin e të gjithë ambientit të runtime-it.
Këto operacione përfshijnë mundësinë për ekzekutimin e kodit, ekzekutimin e kodit të pamenaxhueshëm si dhe anashkalimin e verifikimit.
Klasat e autorizimeve të implementuara në .NET ndahen në tri kategori: autorizimet e CAS, autorizimet e identitetit dhe autorizimet e bazuara në role.

Autorizimet e qasjes me kod

Klasat e autorizimeve të qasjes me kod paraqesin veprimet dhe resurset të cilat janë subjekt në kontrollimin e sigurisë. Klasat e autorizimeve të përfshira në librarinë e klasave të .NET janë listuar në vijim.

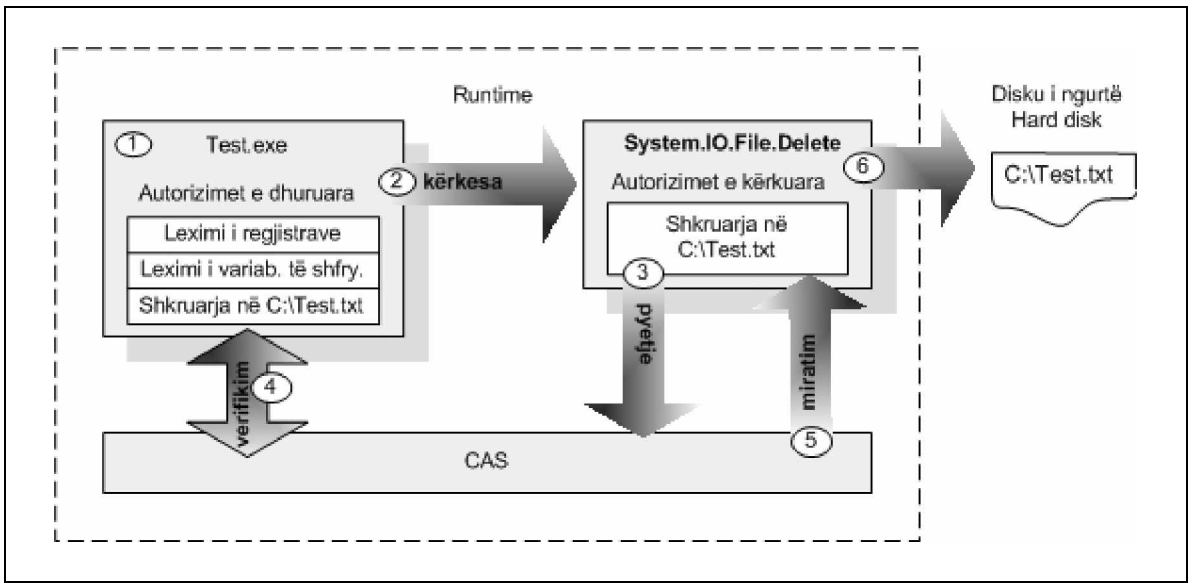
Hapësira e emrave dhe klasat	**Përshkrimi**
System.Data.Common DBDataPermission	Është klasë abstrakte e cila siguron funksion të përgjithshëm për autorizimet e klasave të provajderëve të shënimeve, duke përfshirë klasat *OdbcPermission*, *OleDbPermission* dhe *SqlPermmision*

System.Data.Odbc	
OdbcPermission(1.1)	Kontrollon qasjen në burimet e shënimeve përmes provajderit ODBC.
System.Data.Oledb	
OledbPermission	Kontrollon qasjen në burimet e shënimeve përmes provajderit OLE DB.
System.Data.OracleClient	
OraclePermission(1.1)	Kontrollon qasjen në bazën e shënimevet të Oracle-it e përmes provajderit të shënimeve Oracle.
System.Data.SqlClient	
SqlClientPermission	Kontrollon qasjen në bazën e shënimevet të SQL Server përmes provajderit të shënimeve SQL Client.
System.Diagnostic	
EventLogPermission	Kontrollon qasjen në ditarin e ngjarjeve të Windows-it
PerformanceCounterPermission	Kontrollon qasjen performansat e njehsorit të Windows -it
System.DirectoryServices	
DirectoryServisesPermission	Kontrollon qasjen në sevist udhëzuese, sikurse janë LDAP dhe Udhëzuesi aktiv i Mikrosoftit (ang.Microsoft Active Directory)
System.Drawing.Printing	
PrintingPermission	Kontrollon qasjen në printera
System.Net	
DnsPermission	Kontrollon qasjen në domenin e emrave të serverave në rrjetë (DNSs).
WebPermission	Kontrollon qasjen në resurset e Internet-it.
SocketPermission	Kontrollon qasjen në komunikimet e rrjetës.
System.Messaging	
MessageQueuePermission	Kontrollon qasjen në radhën e mesazheve të Mikrosoft-it
System.Security.Permission	
EnvironmentPermission	Kontrollon qasjen në leximin, krijimin, dhe ndryshimin e variablave të ambientit.
FileDialogPermission	Kontrollon qasjen në fajllat dhe folderët me qasje dhe funksionalitet të kufizuar të kutisë standarte dialoguese të Win.
FileIOPermission	Kontrollon qasjen në diskun e ngurtë lokal duke kufizuar qasjen e krijimit, ndryshimit dhe fshirjes së fajllave dhe folderëve .
ReflectionPermission	Kontrollon qasjen në fonksionet reflektuese të siguruara nga Runtime.
RegistryPermission	Kontrollon qasjen në regjistrat e Windows-it.
ResurcePermissionBase	Është klasë themelore abstrakte e cila siguron funkionalitetin e përgjithëshëm të Windows-it.
SecurityPermission	Kontrollon qasjen në karakteristikat e sigurisë të ndërlidhura me aplikacionet e domenit si polisa, evidenca, thredi, principalet, ekzekutimi, infrastruktura, verifikimi, kodi i pamenagjueshëm
UIPermission	Kontrollon qasjen në manipulimet në mes shfrytëzuesit të interfejsit dhe klipbordit (ang.clipboard).
System.ServiceProcess	
ServiceControllerPermission	Kontrollon qasjen në shërbimet e brëndeshme të Windows.
System.Web	
AspNetHostingPermission(1.1)	Kontrollon qasjen në ambientet e hostuara të ASP.NET .

Klasat e autorizimeve në qasje të kodit të .NET

Zbatimi i CAS (Enforcing CAS)

Kur kodi e përdorë kërkesën e sigurisë (ang.*security demand*), atëherë ai e udhëzon runtime që të siguroj atë, nëse kodi i thirrur i ka autorizimet e caktuara. Në figurën në vijim mund të shihet se çfarë ndodhë kur një aplikacion i quajtur *Test.exe* provon që të fshij një fajll me emrin *Test.txt* i cili ndodhet në diskun C (*C:\Test.txt*), duke përdorur metodën *System.IO.File.Delete* nga libraria e .NET.



Procesi i sigurisë së kërkuar

Procesi i kërkesës së sigurisë nga figura vepron si ne vijim.

1. Kur të ekzekutohet aplikacioni *Test.exe*, runtime e ngarkon asamblenë e *Test.exe*, e vlerëson evidencën e tij dhe përcakton se çfarë autorizimesh i dhurohen. Në këtë shembull runtime i dhuron aplikacionit *Test.exe* tri autorizime, me të cilat krijohen objektet e autorizimit që i atribohen asamblesë:
    - Objekti *RegistryPermission* i cili paraqet autorizimin për të lexuar në regjistrat e Windows-it
    - Objekti *EnvironmentPermission* i cili paraqet autorizimin për të lexuar variablat e shfrytëzuesit.
    - Objekti *FileIOPermission* i cili paraqet autorizimin për të shkruar në fajllin *C:\Test.txt*

2. Në filim, gjatë ekzekutimit aplikacioni *Test.exe* e thërret *File.Delete* që të fshij fajllin *C:\Test.txt*

3. Metoda *File.Delete* e krijon objektin *FileIOPermission* i cili paraqet autorizimin për të shkruar në fajllin *C:\Test.txt* dhe e thërret metodën e saj *Demand* që të kërkoj sigurimin nga runtime që *Test.exe* e ka autorizimin e nevojshëm.

4. Runtime shqyrton objektet e caktuara të autorizimeve të *Test.exe* dhe konfirmon që ato janë përfshirë në *FileIOPermission* që paraqet autorizimin për të shkruar në fajllin *C:\Test.txt*

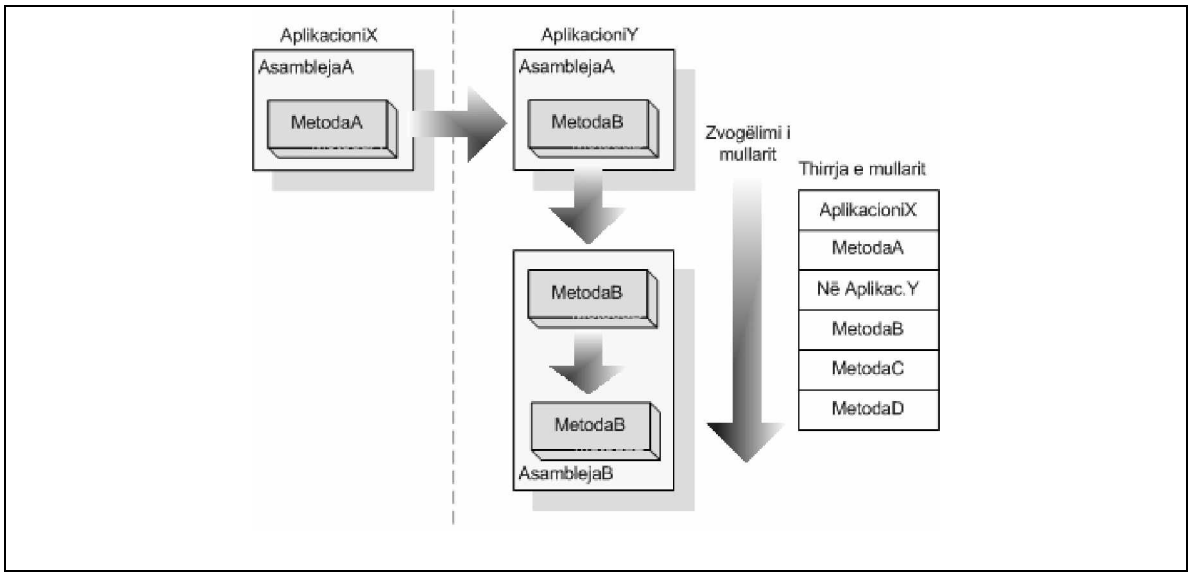
5. Runtime i përgjigjet me rezultat pozitiv metodës *File.Delete*, që do të thotë se *Test.exe* i ka kërkuar autorizimet.

6. Pasi që *Test.exe* ka autorizimin e nevojshëm, metoda *File.Delete* tenton që të fshij fajllin *C:\Test.txt*. Ky veprim do të jetë i suksesshëm, vetëm nëse siguria e sistemit operativ i lejon shfrytëzuesit aktual që ta fshij fajllin *C:\Test.txt*.

Në këtë shembull, *Test.exe* ishte vetëm një aplikacion ekzekutiv dhe vetëm asambleja e tij ishte në shqyrtim. Mirëpo, kur një kod e bënë kërkesën e sigurisë, atëherë runtime duhet të siguroj që jo vetëm shfrytëzuesi aktual të ketë autorizimet e caktuara, por i tërë vargu i shfrytëzueseve të mëparshëm t'i kenë autorizimet e nevojshme. Kjo siguron që kodi me besueshmëri më te pakët nuk mundet që t'a shfrytëzoj kodin me besueshmëri më të lartë, për t'i realizuar veprimet në të cilat ai nuk ka autorizim. Runtime shqyrton autorizimet e të gjithë shfrytëzueseve duke përdorur të ashtuquajturin *stack walk* që në shqip mund të ishte si *shëtitje nëpër stek*.

Shëtitjet nëpër stek (ang. Stack walks)

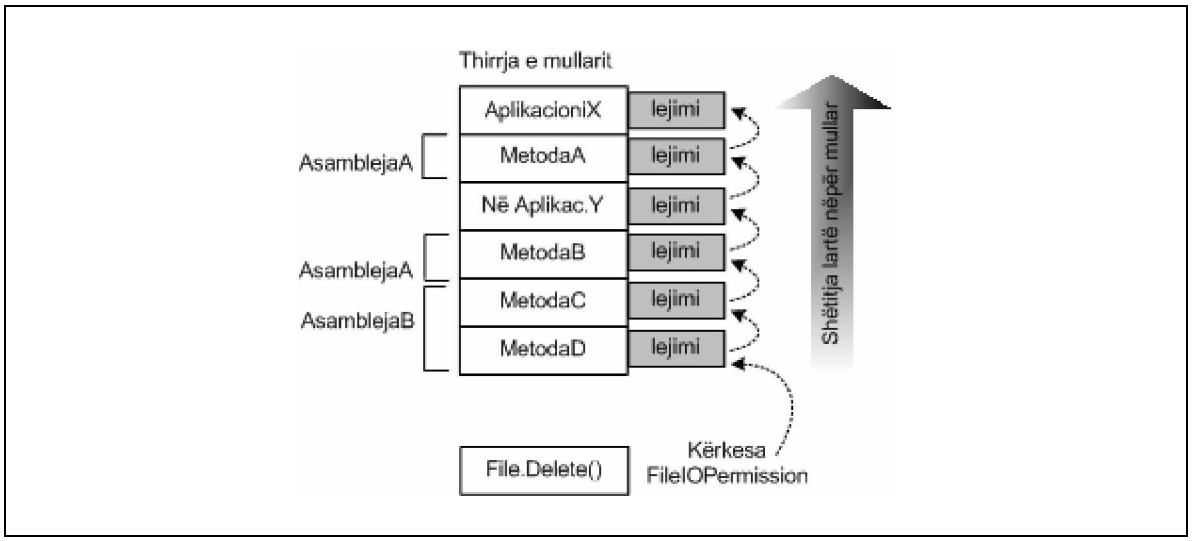
Runtime e mban një stek (ang. *stack*) për secilën shtresë (ang.*thread*) të ekzekutuar. Kur një shtresë aktive e thërret një metodë ose funksion, atëherë runtime e shton një rekord të ri aktiv në fillim (apo maje) të stekut të shtresës.
Rekordi aktiv mbanë detalet e metodës së thirrur, argumentet e vjetra të metodës si dhe adresën se ku duhet të kthehet kur metoda të kompletohet. Me komplementimin e metodës së thirrur, runtime e largon rekordin aktiv nga maja e shtresës dhe e kthen kontrollin në instruksionet e caktuara në rekord.
Steku rritet dhe tkurret (ang.*shrink*) vazhdimisht gjatë jetës së aplikacionit.
Në figurën e mëposhtme është paraqitur gjendja e stekut pasi që janë thirrur sekuencat e metodës të cilat i takojnë asamblesë.



Struktura e stekut (ang. stack)

Le të shqyrtohet përdorimi i stekut në shqyrtimin e autorizimeve.
Kur kodi bënë një kërkesë të sigurisë, atëherë runtime shëtitë teposhtë stekut që nga rekordi më i ri e deri te ai më i vjetri. Është e rëndësishme të theksohet se steku nuk e përfshinë rekordin aktiv të metodës që e ka bërë kërkesën e sigurisë. Rekordi i parë aktiv e përcakton thirrësin aktual.
Runtime duke shëtitur nëpër stek, i krahason autorizimet e lejuara për secilën metodë dhe aplikacion për të parë nëse ata përmbajnë autorizimet e kërkuara. Duhet ditur se autorizimet e metodës janë të njëjta me autorizimet e asamblesë në të cilën ato përmbahen.
Nëse metoda dhe aplikacioni i domenit në stek i kanë autorizimet e kërkuara, atëherë runtime e kompleton shëtitjen nëpër stek pa ndonjë koment apo veprim dhe kthehet te metoda e cila e ka bërë kërkesën e sigurisë me të cilin vazhdohet kompletimi i operacionit të sigurisë. Mirëpo nëse runtime e takon ndonjë metodë ose aplikacion pa kërkesën e autorizimeve, atëherë runtime përmes klasës *System.Security.SecurityException* kthehet te metoda që e ka bërë kërkesën e sigurisë, me të cilën tani kalohet të kodi i cili e ka thirrur atë.

Në figurën në vijim është paraqitur shëtitja nëpër stek që është si rezultat kur *MetodaD* nga figura e mësipërme e thërret metodën *File.Delete.*
Metoda *File.Delete* kërkon që thirrësit të kenë autorizimin *FileIOPermission* për të shkruar në fajllin e caktuar. Në përgjigjije të kërkesës së sigurisë, runtime shëtit nëpër stek dhe i shqyrton autorizimet e dhuruara së pari në *MetodaD*, pastaj *MetodaC*, *MetodaB, AplikacioniY, MetodaA* dhe përfundimisht *AplikacioniX*.
Thirrja e metodës *File.Delete* do të lejohet, me kusht që të gjitha nga këto të kenë autorizimin e nevojshëm *FileIOPermission.*



Shëtitja nëpër stek

PROGRAMIMI I CAS

Intenca kryesore në këtë paragraf do të jetë të kuptuarit e funksionalitetit të klasave standarde të cilat ofrojnë objektet e sigurisë, si dhe implementimi i tyre në krijimin e kërkesave të sigurisë për të siguruar që aplikacioni ynë të ketë autorizimet e nevojshme që të ekzekutohet.
Për implementimin apo programimin e implementimit ekzistojnë dy mënyra për të shprehur paraqitjen e sigurisë. E para siguria *Imperative*, e cila përdorë metodat standarde të implementuara nga klasat e autorizimeve dhe e dyta siguria *Deklerative*, i përdorë atributet e klasave homologe për secilën klasë të autorizimeve. Dhe tani më detalisht për këto dy mënyra, në paragrafet në vijim.

Siguria Ipmerative

Siguria imperative paraqitet në trupin e metodave dhe funksioneve, që do të thotë se e mbarojnë formimin e pjesëve të CIL-së (**C**ompiled **I**ntermeditae **L**anguage), të kodit i cili përmbahet në asamble.
Hapat e parë drejt përdorimit të sigurisë imperative janë ilustrimet e objekteve të autorizimeve me shembuj konkret për tipet e kërkuara për të pasqyruar autorizimet me të cilat dëshirohet të punohet, pastaj thirren metodat përkatëse të cilat janë të implementuara në të gjitha klasat e autorizimeve të CAS , për të ekzekutuar sigurinë e dëshiruar. P.sh. për sigurinë e kërkuar (ang. *security demands*) përdorim metodën *Demand*, për manipulimet nëpër stek (ang.*stack*) zakonisht përdoren metodat *Assert*, *Deny* dhe *PermitOnly*.
Në shembullin në vijim është paraqitur sintaksa e përgjithshme e përdorimit të sigurisë imperative. Fillimisht përmes objektit *FileIOPermission* konfigurohet qasja për të shkruajtur në fajllin *C:\Diqka.txt* dhe pastaj thirret metoda *Demand* për të iniciuar kërkesën e sigurisë imperative. Runtime do të parqesë ngjarjen e mos lejimit të shkruarjes në *C:\Diqka.txt*, vetëm nëse thirrësit e metodës *CreateFile* nuk kanë autorizimet e nevojshme për të shkruar në atë fajll.

```
public void FileCreate()
 {
      // Krijohet një objekt i ri i autorizimeve FileIO
      FileIOPermission perm= new FileIOPermission (
      FileIOPermissionAccess.Write,@"C:\Diqka.txt");

      try
      {
            // Kërkohet FileIOPermission
            perm.Demand();
      }

      catch (SecurityException se)
      {
            // Thirrësit nuk kanë autorizimet e duhura
      }

      // Implementimi i metutjeshëm i metodës.....

}
```

Duhet të jet e qartë se të gjitha veprimet e sigurisë nuk mund të shprehen duke përdorur vetëm sintaksën e sigurisë imperative. Megjithatë siguria imperative ofron një nivel më të lartë të fleksibilitetit dhe të verifikimit se sa mund të arrihet me anë të sintaksës së sigurisë deklerative, që do të diskutohet në paragrafin e ardhshëm.
Përveç kësaj, siguria imperative mundëson që siguria të bazohet në informata të cilat janë të gatshme apo të përdorueshme vetëm në runtime (kohën e ekzekutimit të procesit), pra kur parametrat janë dinamik dhe jo në kohën e dizajnit (ang.*design time*).

Siguria Deklerative

Siç është thënë edhe më lartë, paraqitja e sigurisë deklerative shprehet duke përdorur atributet, të cilat janë formulime të kompajluara në formë të një metadate të asamblesë. Për secilën klasë të CAS, libraria e klasave të .NET përmban atributet korrespondente të klasës, të cilat mundësojnë përdorimin e sintaksës së sigurisë deklerative me autorizimet e caktuara. Secila klasë e atributeve është pjesëtare e hapësirës së emrave të njëjtë sikurse klasa e autorizimeve korresponduese të saj dhe ka emër të njëjtë sikurse klasa e autorizimit por më fjalën "Attribute" të shtuar apo si prapashtesë. P.sh. atributi korrespondues për klasën *FileIOPermission* është klasa *FileIOPermissionAttribute* dhe të dyja janë pjesëtare të hapësirës së emrave (ang.*namespace*) *System.Security.Permission*

Të gjitha atributet për autorizmet e qasjes në kod rrjedhin nga *CodeAccessSecurity*. Ngjashëm sikur te autorizimet, këto atribute mund të përdoren për të lejuar apo për të kufizuar qasjen në autorizimet individuale, siç mund të shihet nga shembulli në vijim.

```
[FileIOPermission (SecurityAction.Deny, Unrestricted=true)]
[EnvironmentPermission (SecurityAction.Deny, Unrestricted=true)]
public void ProcessRequestDenyAll (string iFile, string iEnv)
{
      ReadFile (iFile);
      ReadEnv (iEnv);
      return;
}


[FileIOPermission (SecurityAction.Assert, Unrestricted=true)]
[EnvironmentPermission (SecurityAction.Assert, Unrestricted=true)]
public void ProcessRequestAllowAll (string iFile, string iEnv)
{
      ReadFile (iFile);
      ReadEnv (iEnv);
      return;
}

private void ReadFile (string iFile)
{
      FileStream lStream;
      try
      {
      lStream = new FileStream (iFile, FileMode.Open, FileAccess.Read);
      }
```

```
	catch (Exception lException)
	{
		MessageBox.Show (lException.Message, "Gabim");
		return;
	}
```

```
	int lFileLen = (int)lStream.Length;
	byte[] lBytes = new byte [lFileLen+1];

	lStream.Read (lBytes, 0, lFileLen);

	string lDataRead;
	lDataRead = System.Text.Encoding.ASCII.GetString (lBytes);
	MessageBox.Show (lDataRead, iFile);
	lStream.Close ();

	return;
}

private void ReadEnv (string iEnv)
{
	try
	{
		string lEnvVal = Environment.GetEnvironmentVariable (iEnv);
		if (lEnvVal == null)
			lEnvVal = "E papërcaktuar";
		MessageBox.Show (lEnvVal, iEnv);
	}
	catch (Exception lException)
	{
		MessageBox.Show (lException.Message, "Gabim");
	}

	return;
}
```

Siç mund të shihet gjithashtu nga ky shembull, për të pohuar apo mohuar autorizimet janë përdorur klasat *FileIOPermissionAttribute* dhe *EnvironmentPermissionAttribute*. Megjithatë duhet cekur se këto vlera duhet caktuar në kohën e dizajnit (ang.*design time*) dhe jo në kohën e ekzekutimit (ang.*run time*) si te siguria imperative.

Nëse dëshirojmë që të kufizojmë qasjen në fajll të veçantë ose variabla rrethuese, mund të shihet në kodin në vazhdim.
Në të cilën veçanërisht mohohet qasja në variabilen "`USERNAME`", qasja në variablat tjera duhet të jetë përcaktuar nga polisa e sigurisë.

```
[EnvironmentPermission (SecurityAction.Deny, Read="USERNAME")]
public void DenyLogonUser (string iFile, string iEnv)
```

```
{
      ReadEnv (iEnv);
      return;
}
```

Përparësia e sigurisë deklerative nga ajo imperative është se të gjitha të dhënat e fituara përfshihen në metadaten e asamblesë dhe mund të lexohen përmes veglës duke përdorur refleksionin. Kjo mund të jetë mënyrë e shkëlqyeshme për të kompajluar të dhëna statistikore ose informacione të tjera rreth asambleve dhe përdorimit të autorizimeve të tyre. Kjo nuk është e mundur më sigurinë imperative. Megjithatë si u tha edhe më herët siguria imperative ka përparësi në atë se mund t'i caktoj autorizimet në kohën e ekzekutimit (varësisht nga kushtet), përderisa te siguria deklerative të gjitha aranzhimet duhet të jenë të caktuara në kohën e dizajnimit.

POLISA E SIGURISË (ang. SECURITY POLICY)

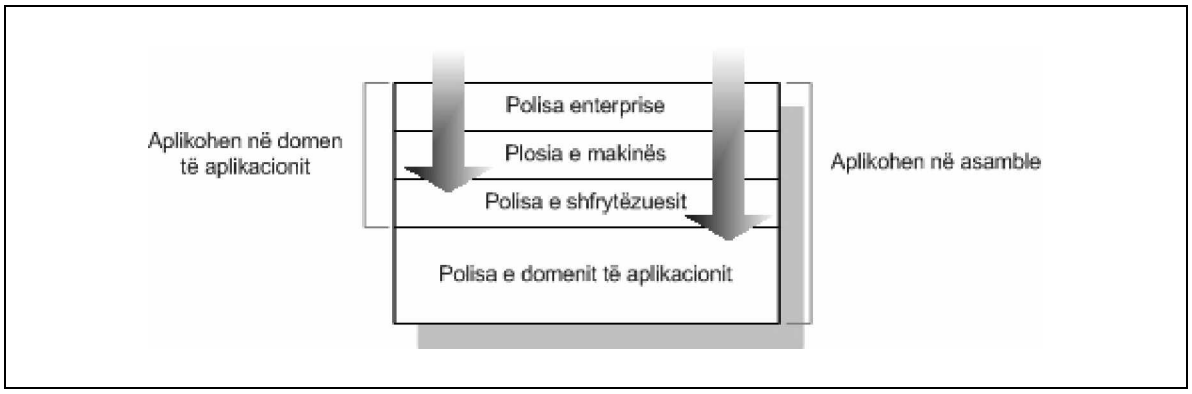
Në këtë paragraf do të diskutohet në lidhje me atë se si runtime e shfrytëzon polisën e sigurisë për të përcaktuar se cilat autorizime duhet dhuruar një asambleje ose një aplikacioni duke u bazuar në identitetin e tyre.
Polisa e sigurisë është një trajtë e rregullave e cila siguron hartimin apo planifikimin ndërmjet evidencës së asamblesë dhe autorizimeve. Përkatësisht runtime e shfrytëzon polisën e sigurisë për të përcaktuar se cilat autorizime të CAS i jepen apo i dhurohen një asambleje apo aplikacioni duke u bazuar në evidencën të cilën asambleja apo aplikacioni e paraqet.
Mekanizmi i polisës së sigurisë është mjaft fleksibil dhe gjithëpërfshirës. Ai i dhuron administratorëve dhe shfrytëzuesve të sistemeve kontroll të plotë rreth operacioneve dhe resurseve në të cilat kodi ka qasje. Me një konfigurim të plotë të polisës së sigurisë, shfrytëzuesit në mënyrë konfidenciale (të besuar) mund të ekzekutojnë kod nga ndonjë burim, duke ditur se runtime do ta ndalojë kodin që të ndërmarrë veprime të padëshiruara. Fleksibiliteti i siguruar nga mekanizmi i polisës së sigurisë është esencial, sepse jo vetëm një kufizim i sigurisë mund t'i takojë kërkesave të çdonjërit shfrytëzues. P.sh. një shfrytëzues dëshiron që kodi nga Interneti kurrë mos të lejohet që të shkruaj në diskun e ngurtë, një tjetër dëshiron që vetëm aplikacionet e shkruara nga Mikrosofti të lejohen që të shkruajnë në regjistrat e sistemit operativ Windows, apo ndonjë shfrytëzues do të dëshironte që të gjitha aplikacionet mund të bëjnë çdo gjë.
Me fjalë të tjera mekanizmi i polisës së sigurisë është mjaft fleksibil për të plotësuar nevojat e shfrytëzuesve rreth sigurisë së kodit në të gjitha situatat.

Nivelet e Polisës së sigurisë

Siç shihet edhe nga figura në vijim polisa e sigurisë në .NET ndahet në katër nivele:

1. Ndërmarrje (ang.*Enterprise*)
2. Makinë
3. Shfrytëzues dhe
4. Domen të aplikacionit

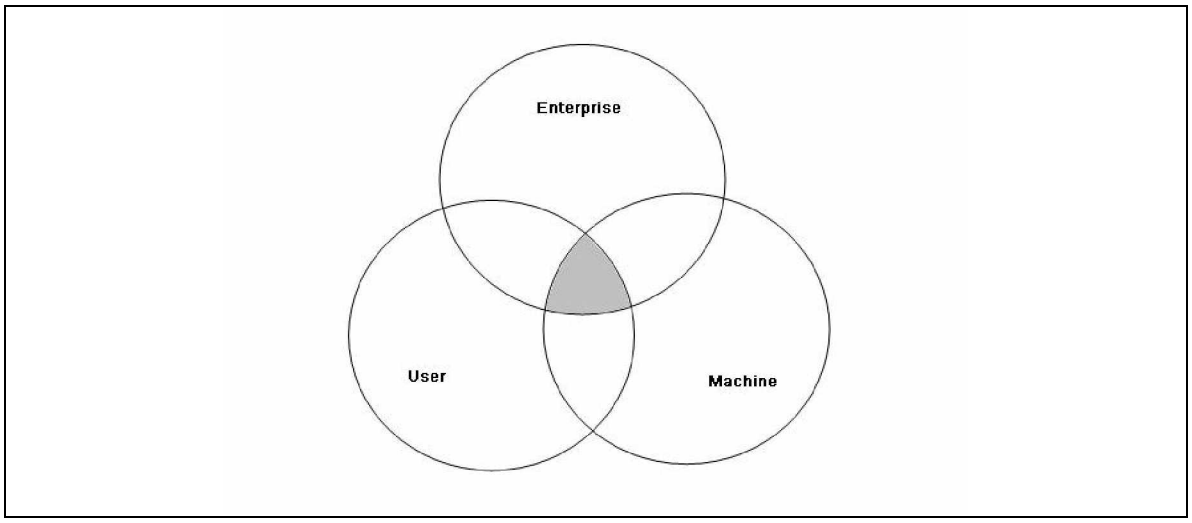
Nivelet ndërmarrje, makinë dhe shfrytëzues konfigurohen pavarësisht nga njëra tjetra duke përdorur veglat administrative të siguruara nga .NET. Pasi që domeni (rrethina) i aplikacionit ekziston vetëm në kohën e ekzekutimit, nuk është e mundur që ai të konfigurohet statikisht duke përdorur veglat administrative. Mirëpo domeni i aplikacionit mund të konfigurohet me programim.



Nivelet e polisës së sigurisë.

Kur runtime e ngarkon një asamble ose e krijon një domen të aplikacionit, ai përcakton sëpari autorizimet e dhuruara nga polisa e ndërmarrjes, pastaj autorizimet nga polisa e makinës dhe në fund autorizimet e dhuruara nga polisa e shfrytëzuesit. Në rastin e asamblesë, runtime gjithashtu do të përcaktoj autorizimet e dhuruara nga polisa e domenit të aplikacionit në të cilin ngarkohet asambleja. Runtime i kryqëzon autorizimet e dhuruara nga secili nivel i polisës për të përcaktuar autorizimin final të CAS për asamblenë apo domenin e aplikacionit. Me këtë nënkuptohet që secili nivel i polisës mund t'i kufizoj edhe më shumë autorozimet nga niveli paraprak i polisës, por asnjëherë nuk mund t'i dhuroj më shumë autorizime se sa niveli paraprak.

Qëllimi i polisës së ndërmarrjes është që të siguroj një siguri të gjerë në të cilën mbështetën makinat, mirëpo të dyja polisat si ajo e ndërmarrjes si polisa e makinës janë makina të caktuara. Pra nuk ka ndonjë mekanizëm në .NET i cili do ta menaxhonte polisën e ndërmarrjes nga një lokacion i centralizuar.
Polisa e shfrytëzuesit është polisë specifike për secilin shfrytëzues. Në një makinë e cila përdoret nga shfrytëzues të ndryshëm, secili shfrytëzues do të ketë konfiguracionin e tij të sigurisë.



Kryqëzimi niveleve të polisës për dhurimin përfundimtar të autorizimeve për një asamble

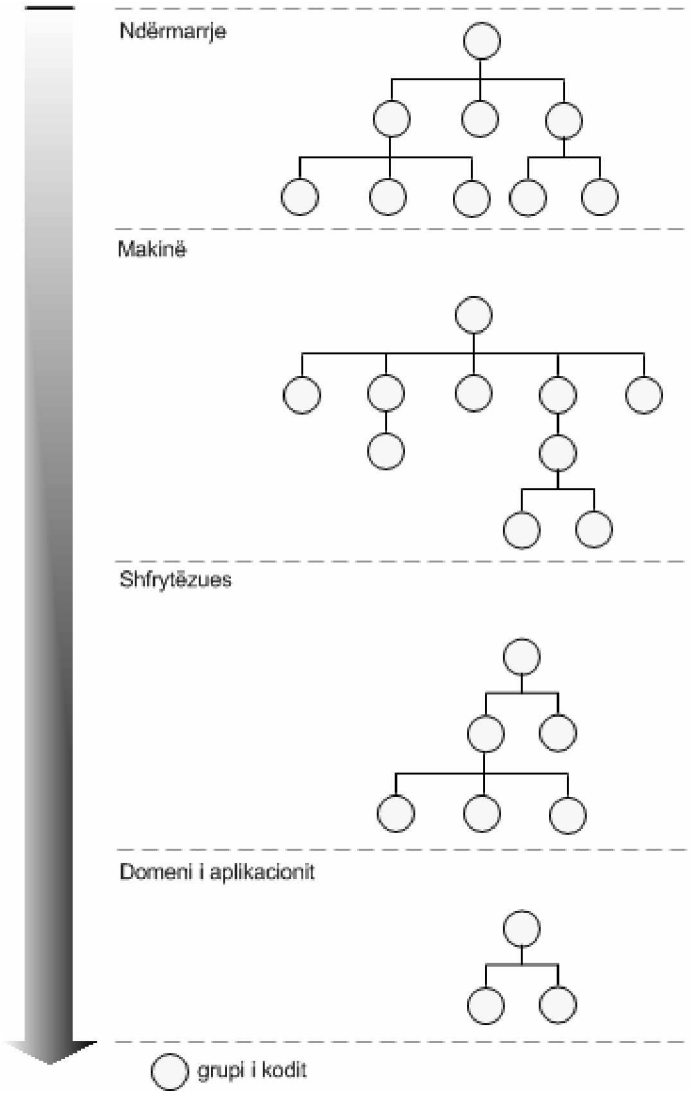
Ndarja e polisës së sigurisë në nivele të ndryshme në shikim të parë duket se e komplikon administrimin e polisës së sigurisë, mirëpo fleksibiliteti i dhuruar nga këto ndarje është më i vlefshëm apo shprehur ndryshe fleksibiliteti peshon më shumë se sa kompleksiteti në këto raste, e kjo vërehet veçanërisht në ambiente të korporatës apo organizatës. Me më shumë nivele të polisës së sigurisë është e mundur delegimi i detyrave drejtuesit të sigurisë. P.sh. Qendra e IT ose departamenti i sigurisë mund ta përdorë polisën e ndërmarrjes për të caktuar minimumin e standardeve të sigurisë për një organizatë, përderisa departamentet e tjera, ekipet apo edhe individët brenda organizatës pastaj mund t'i vendosin polisat e sigurisë në nivel të makinave lokale dhe në nivel të shfrytëzuesve, varësisht nga nevojat e caktuara të tyre.
Secili nivel i polisës përmban tri elemente kryesore :

1. Grupet e kodit
2. Autorizimet e emëruara dhe
3. Asambletë më besueshmëri të plotë

Seclia nga këto tri elemente do të diskutohet shkurtimisht në vazhdim.

Grupet e kodit (ang. Code groups)

Grupet e kodit janë blloqe themelore për ndërtimin e polisës së sigurisë, ato e personifikojnë logjikën me të cilën kontrollohet procesi i zgjidhjes së polisës. Më saktësisht, grupet e kodit mundësojnë ndërlidhjen ndërmjet evidencës dhe autorizimeve të cilat procesi i vendosjes së polisës i përdorë për t'i caktuar se cilat autorizime të CAS-it i jepen apo i dhurohen një asambleje apo domeni të aplikacionit.
Struktura e grupeve të kodit është e organizuar në formë të pemës për secilin nivel të polisës, siq shihet edhe në figurën në vijim.



Hierarkia e grupit të kodit

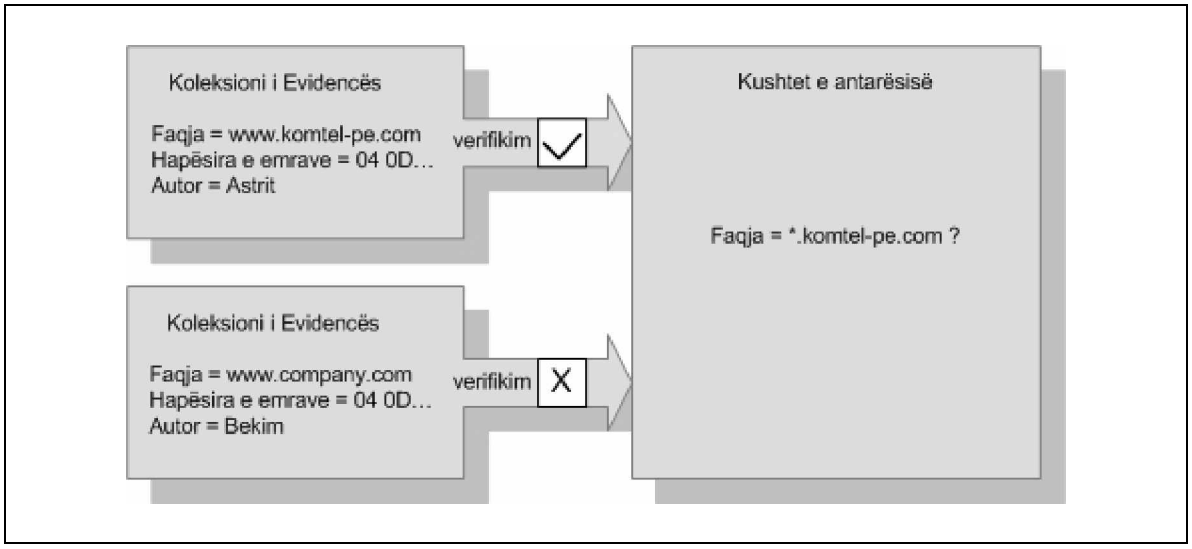
Gjatë zgjidhjes së polisës, runtime e përshkon pemën e grupeve të kodit për secilin nivel të polisës dhe e krahason evidencën e prezantuar nga asambleja ose domeni i aplikacionit me *konditën e anëtarit* (ang.*membership condition)* për secilin grup të kodit. Nëse evidenca i takon konditës së antarësisë së grupit të kodit, atëherë runtime i jep asamblesë apo domenit të aplikacionit autorizimet që përmbahen në *vendimin e autorizimit* (ang.*permission set*) të grupit të kodit.
Secili grup i kodit ka emrin dhe përshkrimin dhe përbëhet nga këto elemente: kondita e anëtarësisë, vendimi i autorizimit dhe grupet e kodit fëmijë.

- *Kondita e anëtarësisë* (ang. *membership condition*) :
  Sikurse shihet edhe nga figura në vijim, konditat apo kushtet e anëtarësisë definojnë evidencën të cilën duhet t'a kenë asambletë apo domenet e aplikacionit për t'u kualifikuar si anëtar i grupit të kodit. Kushtet e anëtarësisë i përkrahin të gjitha tipet

standarde të evidencës. Kushtet e anëtarësisë mund të përfshijnë për shembull:
Faqja = "* komtel-pe.com", Zona = Internet, Autor = Astrit
Si në figurën në vijim.

Ø *Vendimi i autorizimit* (ang. *permission set*) :
Vendimi i autorizimit është vendosje e autorizimeve për t'i lejuar një asambleje apo domeni të aplikacionit që të kualifikohen për anëtarësi të grupit të kodit.

Ø *Grupet e kodit fëmijë* (ang.*Child code groups*) :
Siç shihet nga hierarkia e grupit të kodit, secili nivel i polisës përmban një pemë të vetme të grupeve të kodit me secilin kod të grupit që ka një apo më shumë grupe të kodit fëmijë. Një asamble ose domen i aplikacionit i cili është anëtar i grupit të kodit, është testuar për anëtarësinë e fëmijëve të grupit të kodit dhe kështu teposhtë nëpër pemë.



Konditat apo kushtet e anëtarësisë

Autorizimet e emëruara (ang. Named permission sets)

Sikurse sugjeron edhe vetë emri, autorizimet e emëruara janë një grumbull i thjeshtë i autorizimeve të cilave iu caktohet një emër.
Kur të përdoren veglat administrative të .NET për të konfiguruar autorizimet e dhuruara nga grupi i kodit, atëherë patjetër duhet të përdoret autorizimi i emëruar nga niveli i polisës të cilit i takon grupi i kodit.
Polisa e sigurisë e paracaktuar i definon standardet e autorizimeve të emëruara për nivelet e polisës për ndërmarrje, makinë dhe shfrytëzues, të cilat janë listuar si më poshtë. Të gjitha autorizimet më përjashtim të atij *Everything* janë të përhershme, pra të pandryshueshme.

Autorizimet	**Përshkrimi**
FullTrust	Qasje e pakufizuar në të gjitha resurset dhe veprimet
SkipVerification	S'ka qasje në ndonjë resurs apo operacion, përveç në

	autorizimin për ta tejkaluar verifikimin.
Execution	S'ka qasje në ndonjë resurs apo operacion, përveç në autorizimin për ekzekutim
Nothing	S'ka qasje në ndonjë resurs apo operacion, madje as në autorizimin për ekzekutim.
LocalIntranet	Janë autorizime të definuara në polisën e paracaktuar të sigurisë, të cilat janë të përshtatshme për ngritjen e kodit nga Intranet-ti lokal.
Internet	Janë autorizime të definuara në polisën e paracaktuar të sigurisë, të cilat janë të përshtatshme për ngritjen e kodit nga Interneti.
Evrything	Është i vetmi standart i autorizimeve i cili mund të modifikohet. Sipas vlerës së paracaktuar, Evrything përmban të gjitha autorizimet me qasje të pakufizuar, përveç elementit SkipVerification nga SecurityPermission.

Lista standarde e autorizimeve të emëruara

Asambletë me besueshmëri të plotë (ang. Fully trusted assembly)

Gjatë vendosjes së polisës, runtime krijon objekte për të paraqitur informatën e sigurisë, nëse ajo është duke punuar, kjo mund të përfshij klasa nga shumë asamble të ndryshme. Nën rrethana normale, kur runtime i ngarkon (ngritë) këto asamble, atij i duhet që t'i vendosë polisat për secilën prej tyre, në mënyrë që t'i përcaktoj autorizimet e tyre. Mirëpo, nëse vendojsa e polisës të këtyre asambleve kërkon që runtime t'i ilustroj klasat e sigurisë të përmbajtura në asamble të njëjtë, atëherë rutime –it do t'i duhej që të vendosë sërish polisën për asamblenë e njëjtë, si rezultat i kësaj procesi i vendosjes së polisës kurrë nuk do të mund të del nga ciklimi (ang.*loop*).
Për të tejkaluar këtë problem, secili nivel i polisës përmban listën e asambleve me besueshmëri të plotë. Kur runtime e ngarkon ndonjërën nga këto asamble gjatë vendosjes së polisës, ai automatikisht u atribuon atyre besueshmërinë e plotë përbrenda atij niveli të polisës.
Krijimi i një asambleje me besueshmëri të plotë në njërin nivel të polisës nuk nënkupton që ajo asamble të ketë besueshmëri të plotë edhe gjatë ekzekutimit të programit. Sepse siç dihet dhurimi i autorizimeve përfundimtare të asamblesë llogaritet nga kryqëzimi i autorizimeve të dhuruara nga secili nivel i polisës.

Konfigurimi i Polisës së sigurisë

Polisa e sigurisë ruhet në *XML* fajll dhe është një fajll i vetëm për secilin nivel të polisave (ndërmarrje, makinë dhe shfrytëzues). Ekzistojnë katër mënyra për ta konfiguruar polisën e sigurisë:

1. Duke përdorur veglën *.NET Framework Configuration* (*Mscorcfg.msc*) ku *Microsoft Managment Console* (*MMC*) siguron interfejs grafik me të cilën administrohet polisa e sigurisë.

2. Duke përdorur veglën *Code Access Security Policy* (Caspol.exe) që është vegël komandore e siguruar gjithashtu nga .NET Framework
3. Në mënyrë programore, duke përdorur klasat e sigurisë të cilat përmbahen në librarinë e klasave të .NET
4. Manualisht, duke edituar fajllin e *XML* i cili përmbahet në fajllat individual të polisës së sigurisë.

Rekomandohet që kurrë mos të provohet të editohet manualisht fajlli i polisës së sigurisë. Pasi që fajlli i polisës së sigurisë është një dokument i vogël, ai lehtë mund të bëhet i madh dhe konfuz. Andaj shkruarja e një gabimi sado të thjeshtë mund t'a ndryshoj konfigurimin e sigurisë dhe mandej asnjë kod nuk mund të ekzekutohet.

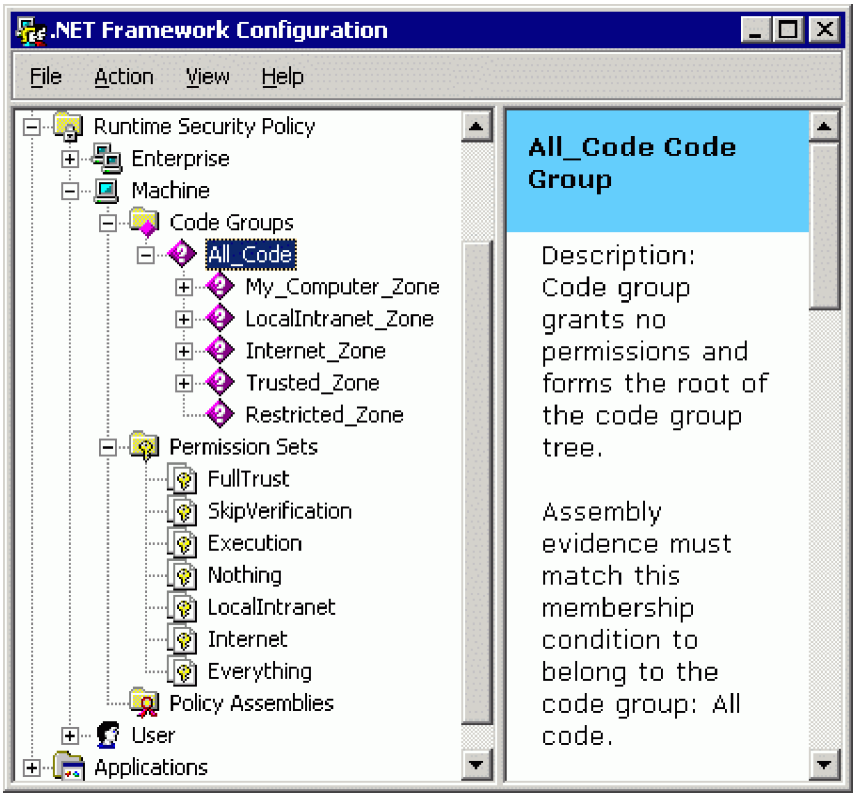
Në vazhdim më gjërësisht do të diskutohen dy mënyrat e para pasi që janë edhe më të lehta për administrim dhe gjithashtu janë më të përdorshmet, rrjedhimisht edhe rëndësia e tyre është më e madhe.

Përdorimi i veglës .NET Framework Configuration

Vegla NET Framework Configuration (Mscorcfg.msc) është konsolë menaxhuese e Mikrosoftit (ang.Microsoft Menagment Console MMC) e cila është e pajisur me .NET Framework e që siguron një interfejs grafik për të menaxhuar aspekte të ndryshme të konfigurimit në .NET. Në këtë diskutim do të shikohet vetëm aspekti i konfigurimit të polisës së sigurisë.

- *Shënim* : *Mscorcfg.msc* mund të ekzekutohet vetëm nëse në kompjuter është i instaluar verzioni 1.2 i MMC apo ndonjë version më i fundit.

Mënyrat apo hapat për ekzekutimin e *Mscorcfg.msc* varen nga sistemi operativ i instaluar në kompjuter, mirëpo veprimet e përgjithshme janë si në vijim.
Duhet shkuar në grupin e veglave administrative të Control Panel dhe nga atje zgjedhet vegla *Administrativ Tools* dhe pastaj *Microsoft .NET Framework Configuration* (*Control Panel --> Administrative Tools --> Microsoft .NET Framework Configuration*).
Apo nga opcioni *Start* zgjedhet opcioni *Programms*, mandej nga kjo zgjedhet programi *Microsoft Visual Studio .NET*, nga grupi i opcioneve zgjedhet *Visual Studio .NET Tools* dhe *Visual Studio .NET Command Prompt* (*Start --> Programms --> Microsoft Visual Studio .NET --> Visual Studio .NET Tools --> Studio .NET Command Prompt*) ku në editorin e hapur më këtë rast shkruhet komanda *Mscorcfg.msc*
Si rezultat i të dy mënyrave të treguara do të shfaqet figura si në vijim.



Dritarja kryesore e *Mscorcfg.msc*

Është thënë më lartë se secili nivel i polisës përbëhet nga tri elemente themelore, menaxhimi i këtyre elementeve bëhet përmes veglave të administrimit të cilat i ofron .NET. Në vazhdim do të tregohet mënyra e menaxhimit të elementit të quajtur Asambleja me besueshmëri të plotë.

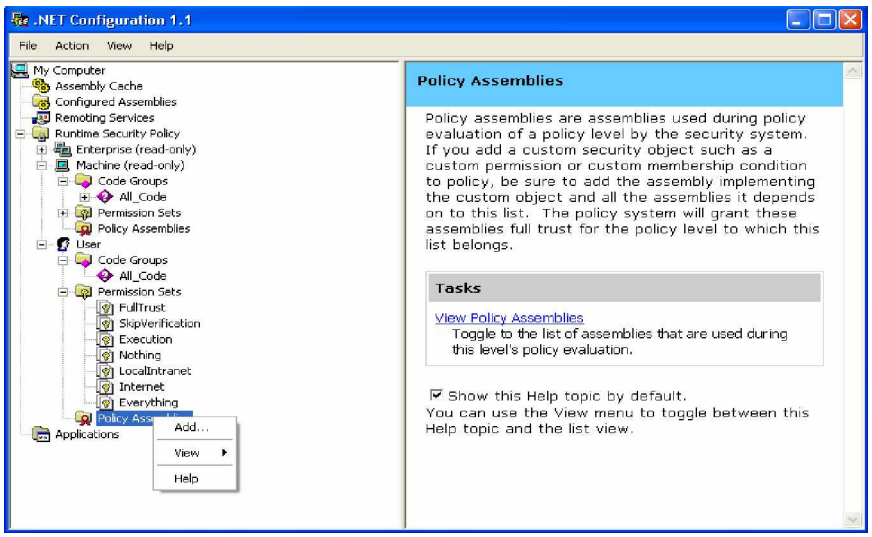
Menaxhimi i asamblesë me besueshmëri të plotë

Përpara se të shtohet një asamble në listën e asambleve me besueshmëri të plotë, ajo asamble fillimisht duhet të ekzistoj në depon e përgjithshme të asambleve (shih paragrafin *GAC*)
Pas zgjedhjes së opsionit *Add* si në figurën e mëposhtme, do të listohen të gjitha asambletë të cilat ekzistojnë në depon e përgjithshme të asambleve. Pastaj zgjidhet asambleja e dëshiruar dhe klikohet në preklën “Select”

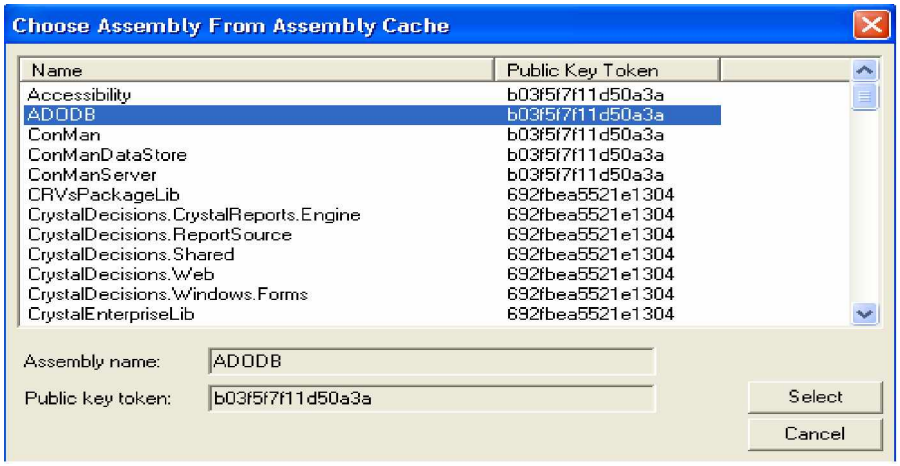
- *Shënim*: Një asamble e besueshmërisë së plotë duhet të bëhet për secilin nivel të polisës në të cilin dëshirojmë që t’i përdorim klasat e sigurisë që i përmban asambleja.

Procesi i fshirjes së një asambleje nga lista e asambleve me besueshmëri të plotë, pothuajse është i ngjajshëm më atë kur shtohet një asamble në atë listë.
Nga opsionet nga figura e mëposhtme zgjedhim atë *View*, ku më pastaj do të shohim listën e asambleve me besueshmëri të plotë, pastaj klikojmë me tastin e djathtë të miut

dhe zgjedhim opsionin *Delete*, nëse e konfirmojmë këtë opsion duke klikuar *Yes* në mesazhin në vijim atëherë procesi i fshirjes së asamblesë do të kompletohet.



Menaxhimi i asambleve me besueshmëri të plotë



Lista e asambleve në depon globale të asambleve

Përdorimi i veglës Code Access Security Policy

Vegla *Code Access Security Policy* (*Caspol.exe*) është gjithashtu një vegël e cila përkrahet nga .NET Framework e që mundëson administrimin e sistemit të CAS – it me anë të rreshtave të komandave (ang.*Command line*) dhe nga fajllat paketë (ang.*batch files*).
Funksionet për menaxhimin e elementeve të nivele të polisës janë të njëjta sikur me *Mscorcfg.msc*. Andaj nuk do të hyhet në diskutimin e detajishëm të menaxhimit të këtyre elementeve me *Caspol.exe*.

Përcaktimi i objektivit të nivelit të polisës

Një pjesë mjaft e rëndësishme drejt administrimit me *Caspol.exe* është përcaktimi i nivelit të polisës. Pra fillimisht duhet caktuar nivelin e polisës të cilin dëshirojmë ta administrojmë, komandat në vijim e mundësojnë një gjë të tillë.

-all
: Afekton polisat e ndërmarrjes dhe të makinës si dhe polisën e shfrytëzuesit për shfrytëzuesin aktual.

-enterprice
: afekton vetëm polisën në nivel të ndërmarrjes

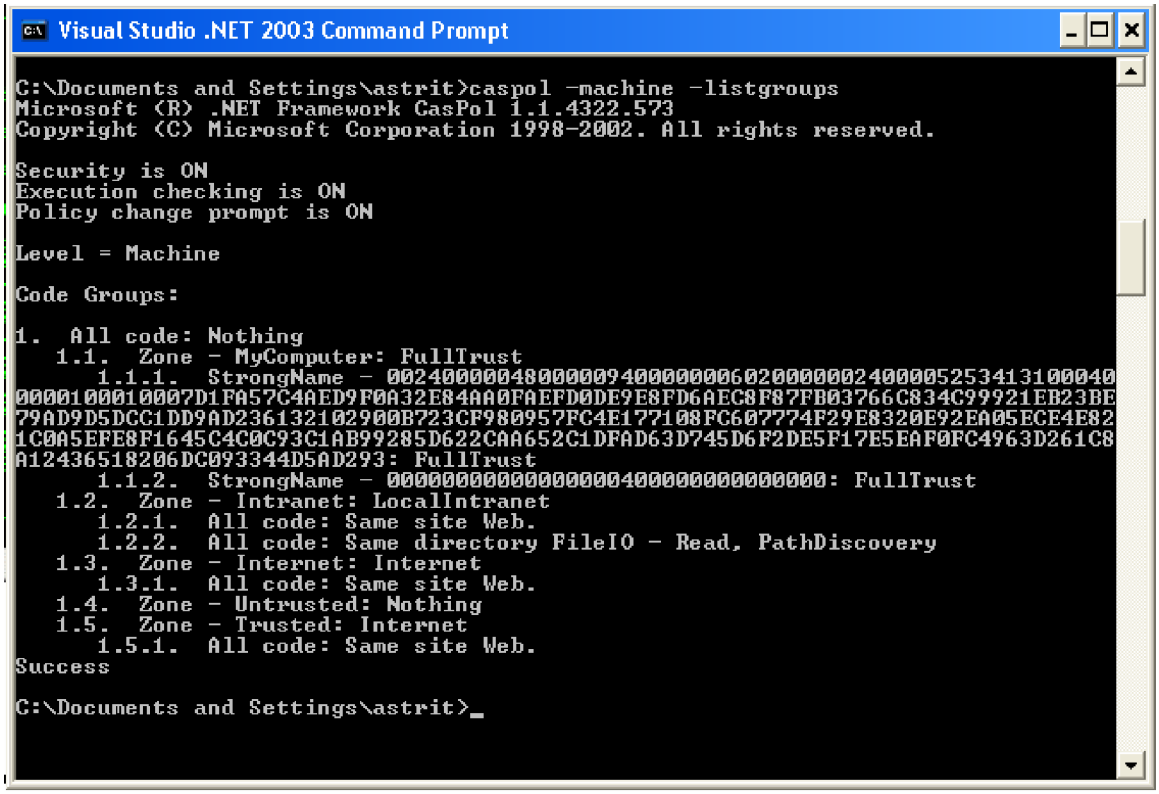
-machine
: Afekton vetëm polisën në nivel të makinës

-user
: Afekton vetëm polisën në nivel të shfrytëzuesit për shfrytëzuesin aktual

-costumer path
: Afekton vetëm polisën shfrytëzuesit e cila përmbahet në fajllin e polisës së sigurisë të lokalizuar në shtegun e specifikuar

-customall path
: Afekton vetëm polisën në nivel të ndërmarrjes dhe në nivel të makinës si dhe polisën e shfrytëzuesit e cila përmbahet në fajllin e polisës së sigurisë të lokalizuar në shtegun e specifikuar.

Në figurën në vijim është paraqitur lista e grupeve në nivel të polisës së makinës duke përdorur *Caspol.exe.*



Lisat e grupeve të kodeve duke përdorur *Caspol.exe*

Për më shumë detaje rreth përdorimit të *Caspol.exe* mund të shikohet në dokumentacionin e *MSDN* apo ndonjë broshurë tjetër në *Internet*, ndërsa nga librat do të specifikoja librin *Programming .NET Security* kapitulli i nëntë (9).

PËRDORIMI I *RBS* DHE *CAS*

Në paragrafët e mëparshme është diskutuar rreth dy sistemeve të sigurisë të cilat i ofron .NET, pra sigurisë së bazuar në role (RBS) dhe sigurisë së qasjes me kod (CAS).
Ku për secilën në veçanti janë treguar mundësitë e realizimit të tyre për sigurimin e aplikacioneve si dhe me anë të shembujve janë ilustruar praktikisht këto mundësi. Gjatë tërë diskutimit në këto paragrafe jam përpjekur t'i përgjigjem pothuajse një pyetje të vetme e ajo ishte " *Si*". Pra si të realizoheshin këto dy mundësi të sigurisë së aplikacioneve, mirëpo thuajse nuk u diskutua fare edhe për tri pyetje të cilat janë më të zakonshmet nga zhvilluesit e aplikacioneve pra : *Pse*, *Kur* dhe *Ku*. Këtyre tri pyetjeve do të përpiqem t'iu përgjigjem në vijim.

Pse të përdoret RBS ose CAS

Fillimisht është e rëndësishme të kuptohet se në programet e ndërtuara e të cilat kanë zbatueshmëri të gjerë në praktikë zakonisht do të përdorën të dyja CAS dhe RBS. Përdorimi i RBS në të shumtën e rasteve ka të bëjë me zbatimin e sigurisë si pjesë interne e vetë aplikacionit. Dhe se menaxhimi i roleve është pjesë e politikës së brendshme të aplikacionit, varësisht nga komplekisteti i tij bëhet edhe ndarja dhe autorizimi i tyre në pjesë të veçanta të kodit. Me përdorimin e RBS gjithashtu bëhet edhe ndarja e përgjegjësive të shfrytëzuesve gjatë realizmit të detyrave të tyre. Gjithashtu sigurohet kodi apo resurset nga përdorimi i pa autorizuar nga shfrytëzues të ndryshëm.

Në të kaluarën çdo pjesë e kodit ekzekutohej në kompjuter nga një burim i vetëm. Dhe nuk ishte e nevojshme brengosja rreth lidhjeve online, emailave, apo shfrytëzuesve të pandërgjegjshëm (*cracker*) të cilët i dinin portat e sigurisë. Në ditët e sotme shfrytëzuesit e kompjuterëve ballafaqohen me rrebeshe të kodit nga burime të panumërta. Nganjëherë kodi mund të hyjë në sistem pa dijeninë e shfrytëzuesit. Me fjalë tjera sistemet kompjuterike në ditët e sotme janë shumë komplekse.
Një prej arsyeve se pse duhet përdorur CAS është të siguruarit që asnjë shfrytëzues mos t'a përdor kodin në mënyrë të parregullt. CAS iu lejon përcaktimin se si dikush mund ta përdorë kodin tuaj dhe çka është më e rëndësishme, se cilat lloje të detyrave nuk mund t'i realizojnë duke përdorur këtë kod. P.sh. Mund të realizohet që kodi juaj nuk mund t'i qaset disqeve të ngurta.
Shkurtimisht, mund të sigurohet që edhe pse krateret disi i qasen kodit, ata nuk mund ta detyrojnë këtë kod që të ndërmarr veprime të cilat nuk janë parashikuar që ky kod mund t'i bëj.
Një karakteristikë gjithashtu shumë e rëndësishme të cilën e ofron CAS është që kodi juaj të insistoj që shfrytëzuesi i tij të ketë nënshkrim digjital. Përveç kësaj, ekzistojnë autoritete (tregtar) të ndryshme të nënshkrimeve digjitale, të cilët e japin nënshkrimin digjital si evidencë për qasje në resurse të caktuara.
Tjetër arsye e fortë e kërkesës së sigurisë është edhe trajtimi i gabimeve. Që do të thotë të menaxhohen të gjitha problemet potenciale të sigurisë të cilat mund të shfaqen.

Kur të përdoret RBS ose CAS

Është e rëndësishme të gjykohet se kur duhet të përdoret mënyra e veçantë e sigurisë. Një tendencë e zhvillueseve të aplikacioneve në ditët e sotme është të krijojnë ambient të sigurt për shfrytëzuesin.
Le të shihet kjo kërkesë nga një perspektivë tjetër. Kështu nëse dihet që i tërë kodi në sistem është i nënshkruar dhe ndonjë kod i panënshkruar vjen nga ndonjë burim i pasigurtë dhe dëshirojmë që ai mos të ekzekutohet.
Pasi që kodi i cili ekzekutohet mund të kontrollohet nga ndonjë kod tjetër i .NET, atëherë mund të mbahet siguria më e mirë vetëm kur shfrytëzuesi shkarkon ndonjë virus nga Interneti. .NET siguron një numër të mënyrave për të verifikuar identitetin e thirrësit dhe burimin e kodit. P.sh. mund të shfrytëzohen klasat *SiteIdentityPermission* ose *URLIdentityPermission* për të verifikuar burimin e ndonjë thirrje nga Interneti.
Prania e kodit të pamenaxhueshëm nëpër lokacione të ndryshme të kodeve bënë, që CAS mos të jep rezultate komplete në sigurinë e aplikacioneve. Atëherë pse nuk kufizohet kodi i pamenaxhueshëm? Nuk do të ishte praktike sepse edhe vetë sistemi operativ i Windows-it është kod i pamenaxhueshëm. Këtë mos funksionim të CAS mund ta përmbushë RBS i cili jo vetëm që përcakton nivelin e besueshmërisë që kemi në shfrytëzues, por gjithashtu definon edhe kualifikimin e shfrytëzuesve. Andaj RBS mund ta përmbushë këtë nevojë duke u bazuar në përcaktimet e Windows-it.

Ku të përdoret RBS ose CAS

Një nga karakteristikat më të rëndësishme të sigurisë në .NET është që administratorët e rrjetës kanë kontroll mjaft të mirë mbi autorizimet. Administratori i rrjetës gjithashtu mund ta përdorë veglën *.NET Framework Configuration* , për të konfiguruar ndonjë kod të krijuar, megjithatë ndryshimet të cilat i bëjnë administratorët janë manuale.
Prandaj një përgjigjeje e shpejtë e pyetjes se ku të përdoren RBS dhe CAS është kudo. Megjithatë aplikimi i sigurisë kudo është problematik. Realisht duhet caktuar lokacionet se ku duhet vendosur CAS ose RBS. Përveç tjerash duhet të vendoset se cila kërkesë e nivelit të sigurisë është zgjidhje më e mirë dhe të vendoset gjithashtu se cila nga siguritë ajo imperative apo deklerative është zgjidhje më e mirë.