Emiel Duys en Loek Duys

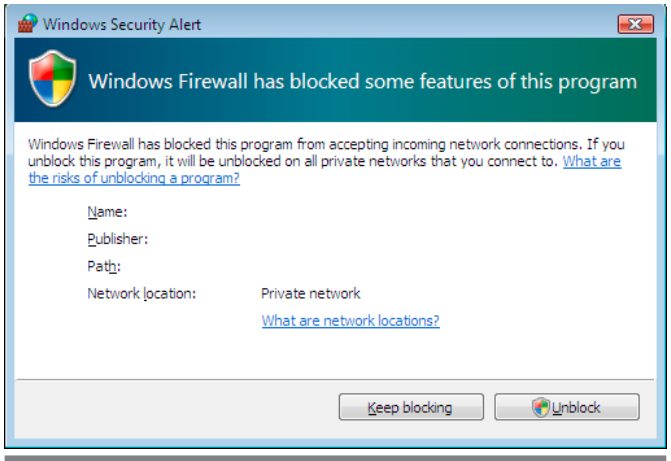
Het Windows Filtering Platform

OP WEG NAAR DE EERSTE FIREWALL-APPLICATIE IN MANAGED CODE

Een nieuw onderdeel in Windows Server 2008 en Windows Vista is het Windows Filtering Platform (WFP). Dit is de opvolger van Network Filtering Interfaces en wordt gebruikt voor het filteren van netwerkverkeer. De mogelijkheden die de WFP API biedt zijn enorm en het is een behoorlijke uitdaging om een deel van deze functionaliteit vanuit een .NET-programma te gebruiken. In dit artikel laten we enkele mogelijkheden zien.

Wanneer je voor een opdrachtgever een Windows Forms-applicatie bouwt, is de kans groot dat deze gebruikmaakt van een netwerk om met SQL Server of webservices te communiceren. Wanneer de beheerder jouw applicatie installeert, wil je daarbij de Windows Vista Firewall configureren om jouw applicatie de juiste verbindingen te laten maken. Hierdoor krijgt de eindgebruiker geen blokkeringsmelding - zoals in afbeelding 1 - bij het opstarten van de applicatie. Doorgaans heeft de eindgebruiker niet voldoende rechten om deze blokkering zelf op te heffen. Misschien wil je in C# een applicatie maken die internettoegang voor je kinderen thuis alleen tussen zes en zeven uur 's avonds toestaat. Of je wilt een eigen firewall ontwikkelen in managed code. Dit alles was in Windows XP nogal een uitdaging, maar is door de komst van Windows Vista goed mogelijk geworden, dankzij het Windows Filtering Platform.

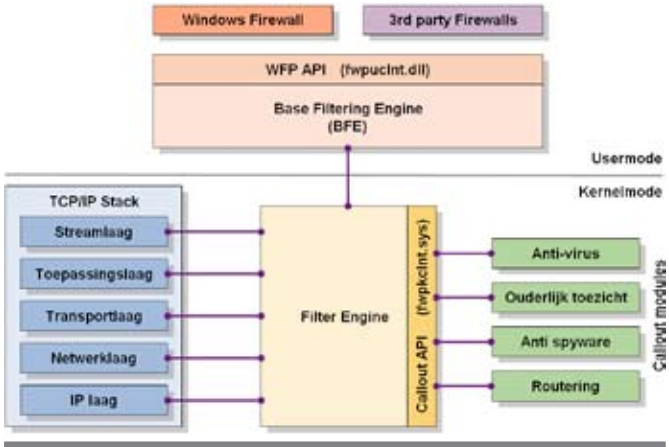
Een van de vele nieuwe ontwikkelingen in Windows Vista en Windows Server 2008 is het Windows Filtering Platform, de sterk uitgebreide opvolger van Network Filtering Interfaces in Windows XP. Deze platforms dienen voor het filteren van netwerkverkeer. Ze worden onder andere gebruikt door firewalls en antivirusprogramma's. Het Windows Filtering Platform heeft interfaces op verschillende niveaus. Dat maakt het mogelijk volledige protocollen te blokkeren, data te inspecteren en op het laagste niveau netwerkpakketjes te analyseren met zelf ontwikkelde modules. In dit artikel beschrijven we twee eenvoudige filterimplementaties die netwerkverkeer blokkeren of toestaan in jouw applicaties. Eerst een voorbeeld over indirect gebruik, via de Windows Firewall API. Daarna geven we een voorbeeld van een applicatie waarin WFP direct vanuit C# wordt aangeroepen. Hierbij maken we gebruik van een zelf ontwikkelde managed wrapper om de WFP API, die je samen met voorbeeldcode kunt downloaden vanaf de website van .NET Magazine.



Afbeelding 1. Applicatie geblokkeerd

Wat is het Windows Filtering Platform?

Het Windows Filtering Platform biedt de mogelijkheid netwerkverkeer te filteren. Het platform is uitbreidbaar en configureerbaar, zodat het als basis kan dienen voor bijvoorbeeld een firewall. Maar omdat netwerkverkeer op alle niveaus geanalyseerd kan worden, is het WFP ook bruikbaar om bijvoorbeeld een antivirus- of antispyware-applicatie op te bouwen. Het platform bestaat uit een centrale verwerkingsmodule (de filter-engine) en een verzameling aanknooppunten (hooks). De hooks worden gebruikt om de filterfunctionaliteit aan applicaties aan te bieden. Applicaties kunnen zich bij de filter-engine registreren, zodat zij worden aangeroepen op het moment dat netwerkverkeer gefilterd moet worden. De filter-engine is verantwoordelijk voor het aanroepen van alle geregistreerde hooks, het verzamelen van alle resultaten en het bepalen van de uiteindelijke actie. In afbeelding 2 wordt een vereenvoudigde versie van de architectuur van WFP getoond. Boven de horizontale lijn staan de componenten die in usermode draaien. Doorgaans worden applicaties in usermode uitgevoerd. Dat wil zeggen dat zij beperkte toegang tot systeembronnen hebben. Onder de lijn staan de componenten die in kernelmode draaien, meestal in de vorm van hardware-drivers. In kernelmode is het mogelijk hardware, zoals de netwerkinterface, direct aan te roepen. Maar door de onbeperkte toegang kunnen programmafouten in drivers tot onherstelbare systeemfouten leiden. Zulke componenten moeten dus zeer nauwkeurig gebouwd en uitvoerig getest worden. In dit artikel beperken we ons dan ook tot voorbeelden die in usermode worden uitgevoerd. In het geval van WFP kan het simpelweg blokkeren van een protocol in usermode



Afbeelding 2. WFP-architectuur

gedaan worden, maar als je losse pakketjes wilt onderscheppen of aanpassen, moet je dat in kernelmode doen.

WFP bestaat uit 4 hoofdonderdelen:

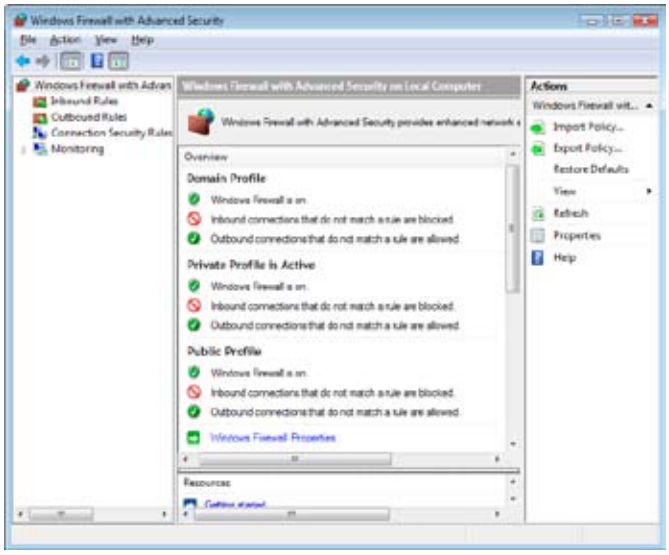
1. **De WFP API** - Het WFP is door eigen code aan te sturen. De functies van WFP worden ontsloten via een Application Programming Interface (API). De Windows Firewall in Vista maakt gebruik van deze API. Ook de code van voorbeeld 2 in dit artikel maakt direct gebruik van deze API.
2. **De Base Filtering Engine (BFE) -** De Base Filtering Engine is een Windows-service die wordt gebruikt om het netwerkverkeer in Vista te filteren op basis van geconfigureerde regels. Deze usermode-component handelt netwerkfilterregistratieopdrachten af die via de API binnenkomen. Hierbij maakt hij gebruik van de Generic Filter Engine.
3. **Call-outmodules** - Soms is de beschrijving van een pakket niet voldoende en moet ook de inhoud van een pakket worden bekeken om te bepalen of deze moet worden doorgelaten of geblokkeerd. Hiervoor zijn de call-outmodules. Een voorbeeld van een call-outmodule is een antivirusprogramma. Ongeacht welk protocol wordt gebruikt, moet binnenkomende informatie worden geanalyseerd om te kijken of er geen virus in is verborgen. Call-outmodules zijn geïmplementeerd als Windows-drivers en draaien in kernelmode.
4. **De Filter Engine** - Deze component is als een onderdeel van het besturingssysteem in de Vista TCP/IP-stack gebouwd. Hierin zijn de filters aanwezig die via de Base Filtering Engine worden aangemaakt. Via de call-out-API worden hier ook de call-outmodules aangestuurd. Wanneer een netwerkpakket binnenkomt en de stack doorloopt, wordt door ieder niveau aan de filter-engine gevraagd of het pakket moet worden doorgelaten of geblokkeerd. Hij bepaalt dit door de geregistreerde en ingebouwde filters en eventuele call-outmodules aan te roepen.

In dit artikel worden de volgende niveaus gebruikt:

A. **De streamlaag -** In de streamlaag wordt een stroom van data aangeduid, bijvoorbeeld de stroom bytes die tijdens het downloaden van een bestand over het netwerk gaat. In deze laag kan bijvoorbeeld de binnenkomende informatie door een antivirusproduct worden doorgezocht op ongewenste scripts.

B. **De transportlaag** - De transportlaag is verantwoordelijk voor het transporteren van berichten tussen partijen, op basis van de protocoldefinitie, via een stabiele verbinding. Hier wordt ervoor gezorgd dat berichten in de juiste volgorde, eenmalig en zonder fouten worden uitgewisseld. In deze laag hebben WFP-filters de beschikking over losse pakketjes, voorzien van een protocolheader. Via de transportlaag kan een applicatie beschikken over een betrouwbare openstaande verbinding (een *sessie*). De onderliggende lagen werken samen om dit beeld te vormen. Hierdoor hoeft een applicatie die TCP als transmissieprotocol gebruikt, geen rekening te houden met de onderliggende netwerkarchitectuur en eventuele fouten in de communicatie. Dit protocol wordt gebruikt voor internetverkeer, in combinatie met het Internet Protocol. Door al het uitgaande TCP-verkeer te blokkeren, kan een internetverbinding worden geblokkeerd.

C. **De netwerklaag -** Deze laag ontvangt pakketten van de datalinklaag. Hij vertaalt de communicatie tussen twee partijen naar een stroom netwerkpakketten. Een bericht wordt daarvoor opgedeeld in blokken die afzonderlijk verstuurd worden. De netwerklaag is ook verantwoordelijk voor de adressering en route die de pakketten afleggen in een netwerk. Een protocol dat in deze laag wordt gebruikt, is het Internet Control Message Protocol (ICMP). Dat wordt in dit artikel gebruikt om andere computers te pingen. Door al het uitgaande ICMP-verkeer te blokkeren, kan het pingen van andere machines worden geblokkeerd.



Afbeelding 3. Schermafdruk WF-configuratiescherm

WFP-gebruik via de Windows Firewall

Eerst gaan we op hoog niveau met het WFP integreren door gebruik te maken van de Windows Firewall (WF). Deze applicatie, gebouwd op het WFP, bewaakt de toegang tot en vanaf netwerken. Het is mogelijk vanuit code met WF te communiceren. WF communiceert op zijn beurt weer met het WFP. Door gebruik te maken van de WF API kan bijvoorbeeld een applicatie tijdens installatie al worden geconfigureerd voor netwerktoegang.

Poorten en regels

De Windows Firewall reguleert netwerkverkeer met hulp van het WFP. Het doorlaten of blokkeren van netwerkverkeer wordt bepaald op basis van zogenaamde rules. Een voorbeeld van een rule is het openzetten van een poort. Een poort kan bijvoorbeeld worden opengezet voor al het netwerkverkeer binnen het TCP- of UDP-protocol, optioneel beperkt op onder andere subnet- of IP-adresbereik. Via een openstaande poort kan worden gecommuniceerd door iedere applicatie. Het is veiliger om de toegang te beperken op applicatieniveau. Een verzameling filterregels of *rules* vormt het machinebeleid, of *policy*. Standaard is in de policy al een aantal rules aanwezig voor bijvoorbeeld het delen van bestanden en printers, of hulp op afstand. Al het netwerkverkeer wordt tegen de rules geëvalueerd. Of een rule van toepassing is, wordt bepaald door zijn *conditon*. De condition kan worden opgebouwd uit de volgende kenmerken:

- Applicatienaam
- Domein, het domein waar de machine lid van is
- Private, privélocaties zoals een thuisnetwerk
- Public, publieke locaties, zoals een openbaar draadloos netwerk

Als een conditie geldig is, wordt zijn gekoppelde *action* uitgevoerd. De action kan zijn 'doorlaten' of 'blokkeren'. Naast de evaluatie van de rule geldt een procedure die gevolgd wordt als geen van de regels van toepassing is op netwerkverkeer. Deze is standaard 'doorlaten' voor uitgaand- en 'blokkeren' voor inkomend verkeer.
Om eenvoudig te kunnen zien welke poorten en regels geconfigureerd zijn in de Windows Firewall, kun je de managementconsole (afbeelding 3) 'wf.msc' openen. De rules die worden aangemaakt in de voorbeeldcode, zullen hier ook verschijnen. Het toevoegen van een rule aan de Windows Firewall wordt uitgewerkt in de voorbeeldapplicatie. In de code wordt een rule gemaakt die 'internettoegang' (uitgaand TCP-verkeer op alle poorten en IP-adressen) voor de voorbeeldapplicatie blokkeert. Op een vergelijkbare manier kan een applicatie ook juist expliciet worden toegestaan TCP- verkeer te mogen versturen of ontvangen.

WFP direct

Via de Windows Firewall kan lang niet alle functionaliteit van het WFP worden benut en bovendien gebruikt niet iedereen de ingebouwde firewall. Daarom is het soms nodig direct met het WFP

te integreren. We zullen dezelfde filters als in het eerste voorbeeld maken, maar dan iets ingewikkelder en direct in het WFP. De API die we in dit hoofdstuk gebruiken, is de 'FWPUCLNT.DLL'. Deze API stelt de Base Filter Engine (BFE) filterfunctionaliteit beschikbaar aan applicaties. In de voorbeeldcode is een managed wrapper meegeleverd, waarin de benodigde functies als managed code beschikbaar worden gesteld. In de wrapper worden door *platform invoke* de aanroepen naar de API en het geheugengebruik voor je geregeld.
Een filter in de BFE staat niet op zichzelf. Er is een bepaalde context nodig, zodat de BFE kan bepalen van wie het filter is (*provider*) en waar in de stack het filter is geplaatst (*layer* en *sublayer*). In usermode worden alle BFE-gerelateerde objecten voorzien van een GUID (uniek identificatie-token) om ze vanuit externe code te kunnen refereren.
De context van een filter in de BFE bestaat uit verschillende elementen. Allereerst is er een sessie of *session object*, waarin alle andere objecten worden aangemaakt in de BFE. Een session kan dynamisch zijn. Dat betekent dat alle objecten die zijn aangemaakt in zo'n session, automatisch worden verwijderd als hij is afgelopen. Objecten kunnen ook blijvend in de BFE worden opgenomen in *permanent sessions*. Sommige objecten, zoals een *provider-object*, kunnen alleen worden aangemaakt als een permanent object. Alle interactie met de BFE, zoals het aanmaken en verwijderen van filterobjecten, moet in de context van een transactie worden uitgevoerd. Voor de transactie kan, vergelijkbaar met een databasetransactie, een 'commit', 'roll-back' of 'time-out' plaatsvinden. Filterobjecten kunnen aan elkaar gerelateerd worden via een provider-object en een *sublayer*-object. Een provider-object geeft details over de entiteit die de filters heeft aangemaakt in de BFE. De Windows Firewall is bijvoorbeeld een provider.
Een *layer* is het niveau waarop in de TCP/IP-stack een filter wordt geplaatst. Sublayer-objecten groeperen functioneel gerelateerde filters. De layer heeft geen directe relatie met een sublayer. Men kan bijvoorbeeld twee auditfilters schrijven, één voor binnenkomend en één voor uitgaand verkeer. De auditfilters kunnen in de zelfde sublayer geplaatst zijn, maar in twee verschillende layers.

Filterprioriteiten

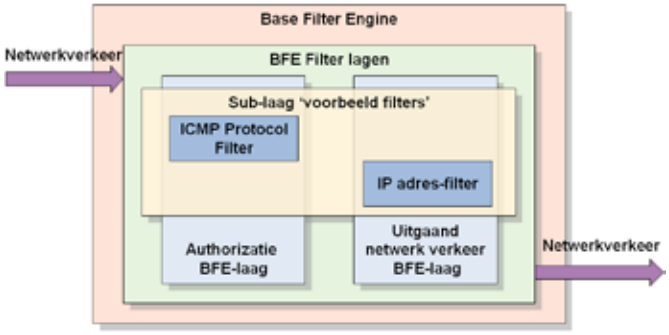
Om te voorkomen dat tegenstrijdige filteracties kunnen ontstaan, krijgen filters een prioriteit. Bij conflicten bepaalt de BFE welke actie wordt uitgevoerd op basis van zijn prioriteit. Zowel sublayers als filters hebben een bepaalde prioriteit. Netwerkverkeer stroomt van de sublayer met de hoogste prioriteit door naar de sublayer met de laagste prioriteit. De filters die gedefinieerd zijn in sublayers werken volgens hetzelfde principe. Het verschil tussen de filter- en de sublayer-prioriteitsbepaling is dat het systeem altijd alle sublayers evalueert voordat bepaald wordt welke sublayer wordt gekozen. Voor filterevaluatie geldt dat het filter met de hoogste prioriteit wordt gekozen waarin een 'block' of 'permit' action wordt gedefinieerd, en dat eventuele andere filters met een lagere prioriteit niet verder worden geëvalueerd.

```
BaseFilterEngine myBFE;

using (myBFE = BaseFilterEngine.GetBaseFilterEngine())
{
   string myProviderName = "Watertight Firewalls inc.";
   string myProviderDesc = "We provide firewalls";
   string myProviderSvc = "n/a";

   myBFE.CreateProvider(providerKey, myProviderName, myProviderDesc,
myProviderSvc);
```

Codevoorbeeld 1.

```
string mySublayerName = "DotNetMag Example Sublayer";
   string mySublayerDesc = "Contains the two WFP example filters";
   ushort mySublayerPriority = 10;

   myBFE.CreateSublayer(providerKey, sublayerKey, mySublayerName,
      mySublayerDesc, mySublayerPriority);
```

Codevoorbeeld 2.



Afbeelding 4. De aangemaakte filters in de Base Filtering Engine

Filterimplementatie

In codevoorbeeld 1 kun je zien hoe via de meegeleverde managed wrapper filters in de BFE kunnen worden geplaatst.
Een instantie van het BaseFilterEngine-object stelt de methoden beschikbaar die nodig zijn om het filter te kunnen aanmaken. Er wordt eerst een provider gecreëerd. Een provider heeft een token, een naam, een beschrijving en een (optionele) Windows Service-naam. Als een provider wordt gehost in een Windows Service, komt het vaak voor dat policies voor die provider alleen gelden wanneer de Windows Service actief is. De BFE kan dit controleren door middel van een eventueel gedefinieerde servicenaam. Het Base Filter Engine-object stelt slechts een beperkt aantal instellingen aan de gebruiker beschikbaar om het gebruik te vereenvoudigen. In dit geval wordt bijvoorbeeld automatisch de providerstatus op permanent gezet, zodat de provider na een herstart niet uit de BFE verdwijnt. De volgende stap, te zien in codevoorbeeld 2, is het aanmaken van de sublayer. De methode die de sublayer aanmaakt, heeft het token nodig van de provider, de sleutel, de naam en de beschrijving van de sublayer, en ten slotte nog de prioriteit.
Nu kunnen de filters worden aangemaakt, dit wordt gedaan in codevoorbeeld 3 en 4. De condition en action worden samen met de verwijzing naar de provider en de sublayer aan de BFE doorgegeven. Ook wordt een bepaalde prioriteit toegewezen. Het eerste filter is weer het TCP- filter dat netwerkverkeer blokkeert dat afkomstig is van IP-adressen in het bereik 192.168.0.[*1 – 255*] in het subnet 255.255.255.0. Dit filter wordt geplaatst in de 'FWPM_LAYER_OUTBOUND_IPPACKET_V4' BFE Layer - in afbeelding 2 de netwerklaag. Informatie die van de server naar de client gaat, zal door deze layer stromen. Het is dus de juiste plaats om uitgaand netwerk-

```
#region Create TCP Filter

string myFilterName = "TCP Protocol Filter";
string myFilterDesc = "Filters all TCP network traffic";

FwpmLayerIdentifier bfeLayerID =
   FwpmLayerIdentifier.FWPM_LAYER_OUTBOUND_IPPACKET_V4;

FwpmFieldIdentifier bfeFieldID =
   FwpmFieldIdentifier.FWPM_CONDITION_IP_PROTOCOL;

FWP_MatchType bfeMatchType = FWP_MatchType.FWP_MATCH_EQUAL;

FwpmActionType actionType = FwpmActionType.FWP_ACTION_BLOCK;

BaseFilterEngine.ConditionValue conditionValue =
   BaseFilterEngine.ConditionValue.CreateUint8Condition(
      (byte)RFC1700Protocol.TCP);
BaseFilterEngine.Condition condition =
   new BaseFilterEngine.Condition(bfeFieldID, bfeMatchType, conditionValue);

myBFE.CreateFilter(filterKey, providerKey, sublayerKey, myFilterName,
   myFilterDesc, bfeLayerID, actionType, condition, 0);
```

Codevoorbeeld 3.

```
#region Create ICMP Filter

string myFilterName = "ICMP Protocol Filter";
string myFilterDesc = "Filters all Internet Control Message network
traffic";

FwpmLayerIdentifier bfeLayerID =
   FwpmLayerIdentifier.FWPM_LAYER_ALE_AUTH_CONNECT_V4;

FwpmFieldIdentifier bfeFieldID =
   FwpmFieldIdentifier.FWPM_CONDITION_IP_PROTOCOL;

FWP_MatchType bfeMatchType = FWP_MatchType.FWP_MATCH_EQUAL;

FwpmActionType actionType = FwpmActionType.FWP_ACTION_BLOCK;

BaseFilterEngine.ConditionValue conditionValue =
   BaseFilterEngine.ConditionValue.CreateUint8Condition(
      (byte)RFC1700Protocol.ICMP);
BaseFilterEngine.Condition condition =
   new BaseFilterEngine.Condition(bfeFieldID, bfeMatchType, conditionValue);

myBFE.CreateFilter(filterKey, providerKey, sublayerKey, myFilterName,
   myFilterDesc, bfeLayerID, actionType, condition, 0);
```

Codevoorbeeld 4.

verkeer naar bepaalde IP-adressen te blokkeren.
Op vergelijkbare wijze kan het pingfilter worden aangemaakt, wat te zien is in codevoorbeeld 4. Dit filter wordt geplaatst in de 'FWPM_LAYER_ALE_AUTH_RECV_ACCEPT_V4' (autorisatie) BFE-layer. Deze layer wordt op de server gebruikt om binnenkomende gegevensstromen te autoriseren. Omdat hij alleen voor netwerkverkeer op de server van toepassing is, zal het filter een ping naar bijvoorbeeld www.microsoft.com gewoon doorlaten, maar een ping naar 127.0.01 blokkeren. Dit illustreert hoe belangrijk het is om de juiste BFE-layer te kiezen voor een filter.
In afbeelding 4 worden de beschreven voorbeeldfilters weergegeven zoals deze in de BFE zijn gedefinieerd.

Tipje van de sluier

In dit artikel hebben we geprobeerd een overzicht te geven van enkele mogelijkheden van de nieuwe Vista Firewall en de WFP API die hiervoor wordt gebruikt. We hebben slechts een tipje van de sluier gelicht. De mogelijkheden die de WFP API biedt, zijn enorm en het is een behoorlijke uitdaging om een deel van deze functionaliteit vanuit een .NET-programma te gebruiken. Dankzij de managed wrapper wordt het een stuk gemakkelijker. De voorbeeldcode geeft een gedetailleerd beeld over hoe de integratie kan plaatsvinden. Wij hopen in ieder geval dat het artikel de interesse in dit onderwerp heeft doen toenemen en we verwachten dat er in de toekomst een grote hoeveelheid applicaties zal inhaken op de WFP API.

Loek Duys is architect en ontwikkelaar bij VX Company. Hij is te bereiken via lduys@vxcompany.com

Emiel Duys is werkzaam voor een consultancy firma in de Verenigde Staten. Hij richt zich hier voornamelijk op .NET-toepassingen voor de financiële sector. Hij is te bereiken via demandiebakt@gmail.com.

Referenties:

Meer informatie over de TCP/Stack van Vista en het OSI-model: http://www.microsoft.com/technet/community/columns/cableguy/cg0905.mspx
Algemene informatie over WFP: http://msdn2.microsoft.com/en-us/library/aa366510.aspx
De diverse layers in de BFE zijn beschreven in het volgende MSDN-document: http://msdn2.microsoft.com/en-us/library/Aa366492.aspx