Bericht uit 

Java security

De wortels van Java liggen in het idee van samenwerkende devices. Het eerste met Java uitgeruste apparaat was een afstandsbediening die op intelligente manier kon communiceren met de apparaten en systemen om hem heen. Dat soort gedistribueerde systemen vergde toen een geheel nieuwe aanpak. Je kunt van Java zeggen dat het niets meer is dan een samenraapsel van al lang bestaande principes, maar dit samenraapsel komt voort uit een belangrijk nieuw denkkader.

Zo is het gegeven van samenwerkende devices en applicaties doordrongen van een aantal voor Java belangrijke principes: platformonafhankelijkheid, multithreading, remote method invocation en ook security. Je wilt namelijk niet dat alle apparaten zomaar alles tegen elkaar mogen zeggen.

De eerste grote groeistuip van Java ontstond toen men met het concept Applet naar buiten kwam. De browser van dat moment, Netscape versie 2.0, kende een virtual machine voor Java applets en de hele wereld kon dus applets downloaden en lokaal opstarten. Het idee van deze applets was vooral, dat dit ongemerkt zou gebeuren. Stukken programmatuur die ongemerkt gedownload worden en ongemerkt worden opgestart stonden tot dan toe eigenlijk alleen bekend als virus. Om te voorkomen dat applets de voordeur zouden gaan vormen voor de cybernetische virologen onder ons, moest het concept applet met een zeer strikte vorm van security uitgerust zijn. Het lijkt er op dat de ingebouwde security mechanismen voldoende waren. Java applets zijn nooit op grote schaal in verband gebracht met security problematiek (daar hebben we andere platformen voor :-).

Al met al is security een belangrijk aspect van Java. In zowel de taal als het platform is security overal terug te vinden: Classloader, SecurityManager, Policy en AccesController zijn enkele van de in alle Java applicaties aanwezige classes die zich bezig houden met security. De gemiddelde Java ontwikkelaar komt deze classes nooit tegen, laat staan dat hij deze met opzet toepast. Het zijn classes die op de achtergrond een basaal soort 'gegeven' security implementeren. Het wordt anders, wanneer we te maken krijgen met zaken als encriptie, certificaten, PKI en KeyStores. Of wanneer we Enterprise Java Beans moeten deployen en onze applicatie ineens, samen met een LDAP server, zich bezig gaat houden met "wie mag wat?" en vooral "wie mag wat niet?". Dan opent zich een hele nieuwe wereld van uitgebreide libraries vol met ondoorgrondelijke classes waar we nooit eerder het nut van hebben ingezien.

De gemiddelde ontwikkelaar ontwerpt systemen alsof er geen security bestaat. In de dagelijkse praktijk kom ik in de meest kritische omgeving systeemontwerpen tegen waar geen woord gerept wordt over security, waar security uitgesteld wordt tot de laatste implementatieronde of liever nog tot de deployment. Dan is het te laat. Dan blijken al die Java security classes warrige, lelijke code op te leveren, de keurig uitgedachte modellen moeten op hun kop gezet worden en er beginnen gaten te vallen. OO-ers hebben altijd geleerd dat er begonnen moet worden met de business modellen, met de classes die voor hun organisatie specifiek zijn. Business classes zijn waardevolle classes, die kan je alleen zelf maken. Dat security classes business aspecifiek zijn, betekent niet dat security zelf business aspecifiek is. Security is over het algemeen zeer specifiek voor een organisatie. Ik ben er dan ook een voorstander van om security zo vroeg mogelijk in het businessmodel in te modelleren. Rollen, rechten en overtredingen zijn zaken die ik graag tegenkom in een vroeg systeemmodel. Door dit soort concepten vroeg mee te modelleren blijken die generieke security classes ineens zeer bruikbaar te zijn en ook eenvoudig toepasbaar te zijn. Security en systeem zouden naadloos in elkaar over moeten gaan. De implementatie van de Java virtuele machine geeft ons daar een mooi voorbeeld van: Security is een integraal en belangrijk deel van onze systemen. Laten we dat voorbeeld volgen en security de aandacht geven die het nodig heeft.

J. Meermans
is Java-kenner en te bereiken via
meermans@cibit.nl