KPU Lab Security

Kommentarer til løsningsforslag

Der er 2 alternative løsningsforslag.

Version 1

Denne løsning består af projekterne `CautiousApp og SuspectAssembly.`

Dette er den simpleste, men den er også lidt urealistisk. For at kunne kalde en instans af en klasse i et andet AppDomain, så kræver .Net at klassen skal nedarve fra framework klassen `MarshalByRefObject`. Dette har jeg ladet Utility klassen gøre. Og det er det som er det urealistiske, da men sjældent vil opleve at 3. parts assemblies er forberedt for dette senarie. Men det kan bruges, hvis man er i stand til at definere grænsefladen mellem applikationen og eventuelle plug-inns / 3. parts komponenter.

Min fremgangsmåde er lidt anderledes end det eksempel, som der var henvist til:

1. Laver et nyt appDomain med begrænsede rettigheder –> sandbox'en.
2. Loader den mistænkelige assembly ind i sandbox'en.
3. Laver en instans af Utility klassen i sandbox'en og unwrapper den med samme funktionskald. Resultatet af denne operation er, at jeg nu har en instans af Utility klassen i Sandbox AppDomain'et, men jeg kan kalde den via en proxy (sandboxedUtility) i hoved AppDomain'et.
4. Kalder PrintDomain for at vise an den eksekvere i Sandbox'en, og når jeg kalder Foo får jeg en SecurityException.

Version 2

Denne løsning består af projekterne `CautiousAppEx, SSAWrapper og SuspectAssembly.`

Denne løsning kræver noget mere tastearbejde, men kan bruges når klasserne i den suspekte assembly ikke nedarver fra klassen `MarshalByRefObject.`

Det som adskiller denne løsning fra den første, er at der indskydes en wrapper-klasse mellem applikationen og Utility-klassen. Wrapper-klassen nedarver fra `MarshalByRefObject` så den kan kaldes fra hoved AppDomain'et. Da Wrapperen eksekverer i sandbox appDomain'et, så har den ingen problemer med at kalde Utility-klasssen direkte, selvom Utility-klassen ikke nedarver fra `MarshalByRefObject`.