7.10.2 Code Injection

Come riconoscerla

L'applicazione esegue del codice ricevuto attraverso l'input che non è stato sufficientemente verificato. Un utente in grado di inserire codice arbitrario può prendere il controllo dell'applicazione e del server, se non sono state adottate tecniche di difesa in profondità.

Come difendersi.

- È vietata qualsiasi esecuzione dinamica di codice ricevuto da canali non attendibili. Se è proprio necessario compilare ed eseguire del codice dinamico, occorre allora predisporre una sandbox isolata, ad esempio AppDomain di .NET o un thread isolato.
- Devono essere effettuati tutti i controlli possibili per validare il codice in ingresso.
- Configurare l'applicazione da eseguire utilizzando un account utente limitato che non disponga di privilegi non necessari.
- Se è possibile optare per isolare tutta l'esecuzione dinamica utilizzando un account utente separato e dedicato che abbia privilegi solo per le operazioni e i file specifici utilizzati dal codice da eseguire, in base al principio denominato "Principle of Least Privilege". Il principio stabilisce che agli utenti venga attribuito il più basso livello di "diritti" che possano, detenere rimanendo comunque in grado di compiere il proprio lavoro.

Esempio:

Il codice seguente mostra come del codice VB.NET può essere passibile di code injection. Il codice utente viene accettato senza verifiche e compilato "al volo", quindi eseguito.

```
Function EsecuziondeDinamicaCodiceUtente_NonSicuro(request As HttpRequest) As
Integer
  Dim exitCode As Integer
  Dim codiceUtente As String = request.Form("Codice")

  Dim compiler As New VBCodeProvider
  Dim parameters As New CompilerParameters
  parameters.GenerateInMemory = True
  parameters.GenerateExecutable = True

  Try
          Dim results As CompilerResults =
compiler.CompileAssemblyFromSource(parameters, codiceUtente)

          Dim compiledAssembly As Assembly = results.CompiledAssembly
          exitCode = CInt(compiledAssembly.EntryPoint.Invoke(Nothing, New
Object()))
  Catch ex As Exception
          HandleExceptions(ex)
  End Try

  Return exitCode
End Function
```

La seguente implementazione invece risolve il problema della code injection. Non viene eseguito del codice, ma l'input dell'utente è usato per selezionare del codice precompilato che viene lanciato in un nuovo processo:

```
Function EsecuziondeStaticaCodiceUtente(request As HttpRequest) As Integer
  Dim exitCode As Integer
  Dim parametriUtente As String = request.Form("ExeParams")

  Using proc As Process = New Process()
          proc.StartInfo.FileName = PATH_TO_PRECOMPILED_EXTERNAL_PROGRAM
          proc.StartInfo.Arguments = SanitizeForProcess(parametriUtente)
          proc.StartInfo.UseShellExecute = False

          proc.Start()
```