実際に作りこんでわかるWebアプリケーション

.NETによるビジネスアプリケーション開発講座

第4回 認証や許可、ロールなどセキュリティ関連について

日本ユニシス株式会社
.NETビジネスディベロプメント
尾島 良司 *OJIMA, Ryoji*

Technology Tools

- ☑ Visual Basic .NET
- ☑ Visual C# .NET
- ❑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ☑ ASP.NET
- ☑ Internet Information Services
- ❑ Other:

Level

★ ★ ★ ★ ★

Samples

はじめに

本連載では、サンプルアプリケーションの開発を通じて、ビジネスアプリケーションをどのように開発してゆけばよいのかを具体的に考察しています。

……と書いた後にアレなのですけど、今月は趣向を変えてセキュリティの“一般論”をやります。サンプルアプリケーションはお休み。ごめんなさい。

さて、今月のお題のセキュリティ。セキュリティは範囲の広い用語です。プロセス、インフラストラクチャ、アプリケーション。これらを全部やるのはどう考えても不可能なので、対象を絞らなければなりません。

今回は対象を“アプリケーションセキュリティ”に絞ることにしましょう。なんと言っても本連載のタイトルは「ビジネスアプリケーション開発講座」なのですから。

以下に、安全なアプリケーションを作るために最も重要なこと、「正しい利用者に正しい操作のみを許す」方法について解説します。

基礎知識

ここでは、以下の項目について解説します。

- **認証、許可、ロール**
- **Windowsのセキュリティ**
- **.NET Frameworkのセキュリティ**

セキュリティの実装方針を定めるには、さまざまな知識が必要です。最低限必要な用語である認証と認可とロールを定義して、Windowsと.NET Frameworkでのセキュリティ機構の基礎を解説します。

認証、許可、ロール

認証（Authentication）は、利用者が確かに本人であることを確認することです。許可（Authorization）は、認証された利用者に処理を許可するかどうかを確認することです。ロール（Role）は、アプリケーションにおけるユーザーの役割で、許可を制御する単位です。

認証と許可とロールは直交しているのが理想です。指紋認証やID／パスワードで認証した利用者を、ロールでたばねてASP.NETやファイルアクセスの許可を制御する。このようなことを可能にするために認証と許可とロールを分割して考えるわけです。

しかしまぁ、理想どおりにはゆかないのが世の常。認証の方式によって選択可能な認可やロールの方式が制限されたり、その逆が発生したりします。

なぜそのような制限が発生するのでしょうか？　答えは簡単。.NET Frameworkのセキュリティ機構とWindowsのセキュリティ機構がバラバラだから。……悲しいなぁ。

Windowsのセキュリティ

認証の結果がないと許可の制御ができないわけで、つまりは許可のためには認証が必要なわけで。でも、実行効率を考えれば許可のたびに認証なんかできないわけで。そもそも、許可のたびに認証がかかってパスワードを聞かれるシステムなんて誰も使ってくれないわけで。

認証の結果は、許可に引き継ぐために何らかの形で残しておかなければなりません。

Windowsのセキュリティ機構では、この認証結果をトークンと呼びます。そして、トークンには、プロセスに関係付けられた“プロセストークン”とスレッドに関係付けられた“スレッドトークン”の2つがあります（図1）。スレッドトークンは省略可能で、スレッドトークンが設定されていない場合はプロセストークンが利用されます。図1の場合であれば、スレッドA1はBさんの権限で、スレッドB1の場合にはDさんの権限で実行されるわけです。

で、このトークンの正当性は、LSA（Local Security Authority）というサービスで保証されます。ローカル（当該コンピュータ）の範囲では、トークンがあれば認証済みとみなされ、そのまま許可に進むことができるわけです。

.NET Frameworkのセキュリティ

上に述べたWindowsのセキュリティ機構は、Windowsの認証機構、具体的にはWindowsのアカウントの存在が前提になっています。

しかし、アプリケーションによってはWindowsのアカウントを使えない場合があります。たとえば、インターネット向けのWebアプリケーション。このような場合には、管理コストを考えて、Active DirectoryではなくRDBMS（Relational Database Management System）でアカウントを管理することになるでしょう。

そう、このようなケースに対応するために、.NET FrameworkはWindowsのセキュリティ機構の上に“独自”のセキュリティ機構を構築しているのです。

.NET Frameworkのセキュリティ機構では、認証結果をプリンシパルと呼び、CLR（Common Language Runtime）の

図1：プロセストークンとスレッドトークン

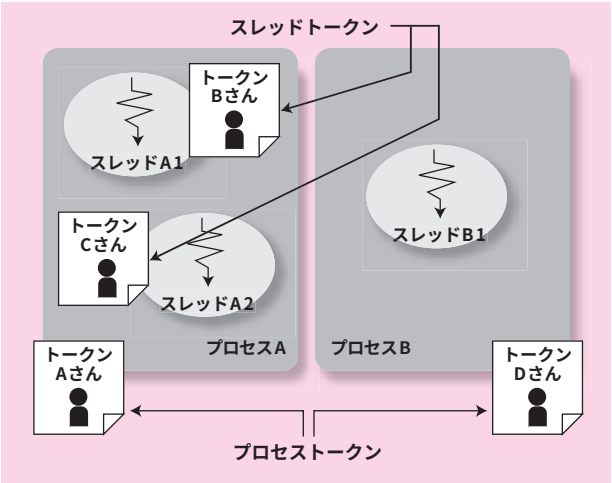

図2：.NET Frameworkのプリンシパル