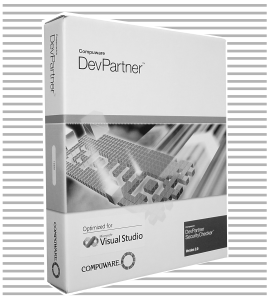

セキュアなASP.NETアプリケーションの構築を支援するセキュリティ検査ツール

DevPartner Security Checker 2.0日本語版

株式会社CSKシステムズ　IT生産技術部
藤田 聡 FUJITA, Satoshi

Software Information

OS	Windows 2000/XP Windows Server 2003
開発環境	Visual Studio .NET 2003 Visual Studio 2005
対応言語	Visual Basic Visual C#
価格	644,000円（1指名ユーザー） 1,931,000円（1コンカレントユーザー）

問合先
日本コンピュウェア株式会社
TEL ：03-5473-4530
FAX ：03-5473-4528
URL ：http://www.compuware.co.jp/
MAIL：marketingjapan@compuware.com

Webシステムの脆弱性

インターネットが広く普及し、いつでもどこからでもアプリケーションを利用できるという手軽さから、多くのビジネスアプリケーションがWeb化されています。これによりビジネスのスピードアップや生産性向上、運用コストの軽減が実現されています。しかしながら、このようなWebアプリケーションの多くには、セキュリティ面の脆弱性があるといわれています。近年、さまざまな企業システムにおいて情報漏洩などの問題が発生し、システムに対する個人情報保護やコンプライアンス対応などのセキュリティ面での要求が、よりいっそう高まっています。そのため、Webアプリケーションの脆弱性を克服し、情報漏洩などの問題を防ぐことは、企業にとって非常に重要な命題となっています。

言語 ››› Language
- Visual Basic

ツール ››› Tool
- Visual Studio 2005 Professional
- SQL Server 2005

これに対して、Webサーバーのセキュリティ対策は、一般的にファイアウォール、IDS（不正侵入検知システム）、ウィルス対策といった手段が主流です。しかし残念ながら、バッファオーバーフローやSQLインジェクションといったWebアプリケーションへの攻撃を、ファイアウォールやIDSだけでは防ぐことができません。そのため、開発段階でWebアプリケーションそのものに脆弱性を作らないように対策を練り、システムを作り込むことが不可欠となります。

また、このような脆弱性を検査するツールもありますが、その多くは“運用環境”での利用に焦点を当てており、本番稼働中のアプリケーションのセキュリティ問題を検出するためのものです。しかし、本番稼働中のアプリケーションを修正するには、多くの時間とコストがかかります。そのため、より少ないコストで対応できる“開発段階”で、Webアプリケーションの脆弱性に対するセキュリティ対策を行なうのがよりよい手段と考えられるでしょう。

このように、システム開発の段階でWebアプリケーションのセキュリティ対策を行なうことが非常に大切ですが、セキュリティに詳しい開発者が少ないという現実があります。

ではこのような状況下で、Webアプリケーションのセキュリティ対策を考慮し、効率的にシステム開発を進めるにはどうしたらいいのでしょうか。これらの問題を解決する手段を提供する製品のひとつに「DevPartner Security Checker 2.0日本語版」（以下Security Checker）があります。

以降で、このSecurityCheckerがどのようなツールなのかを具体的に紹介します。

SecurityCheckerの機能

SecurityCheckerは、ASP.NETアプリケーションの開発時に脆弱性を検査し、その修正方法を提示するツールです。このツールを利用することで、システム開発の実装段階から、Webアプ

リケーションの脆弱性を検出し、修正することができます。また、Visual Studio上からSecurityCheckerを実行できるため、とても効率的に脆弱性の検出／修正作業を行なうことができます。

たとえば、Webアプリケーションの脆弱性テストの代表例として、クロスサイトスクリプティングの脆弱性検査があります。このテストでは、Webアプリケーションを実行して、URLパラメータとしてスクリプトを送信することで動作を確認し、脆弱性が存在しないことを検証します。通常はこの作業を、すべてのURLパラメータに対して、さまざまなパターンの攻撃を想定して繰り返す必要があります。そのため、このようなテストは非常に手間がかかります。

しかしSecurityCheckerを利用すれば、WebアプリケーションのWebページを指定するだけで、クロスサイトスクリプティングの脆弱性の有無を検出することができます。脆弱性がある場合はその場所を指摘してくれるため、テストを実施する前にプログラムを修正することができます。

■検出可能な脆弱性

SecurityCheckerで検出できるWebアプリケーション上の脆弱性は、表1に示す5つに分類されています。

では、これらの脆弱性を、どのようにして検出／修正するのでしょうか。SecurityCheckerでは、Webアプリケーションの脆弱性を効率的に検出／修正するために、

・静的分析
・実行時分析
・保全性分析

という3つの機能が用意されています。

■静的分析

静的分析では、Webアプリケーションのソースコード、HTMLファイル、Web.Configファイルをスキャンし、セキュリティ上の既知の問題を検出します。実際のソースコードをスキャンすることにより、アプリケーションに対して攻撃をシミュレートするだけでは検出するのが難しい問題も発見することができます。

静的分析は、コードをコンパイルすることで実行されるため、開発サイクルのさまざまなタイミングで実行することができます。分析時間も短いため、開発の初期段階に利用すると効率的です。

■実行時分析

実行時分析では、Webアプリケーションの実行時の内部的な処理内容を検査し、処理コードの実行と並行して脆弱性を検出します。この分析では、アプリケーションが必要以上の特権を使用していないか、特別な権限が必要なディレクトリのファイルにアクセスしていないか、レジストリを不正に使用していないかなど、セキュリティに悪影響を与える可能性のある操作を発見することができます。

実行時分析を行なうには、動作するWebアプリケーションが必要となります。そのため、Webアプリケーションが動作するようになったら、比較的早い段階で実行するのがよいでしょう。また、アプリケーションに機能を追加するたびに実行することで、機能追加によって脆弱性が生じていないかどうかを検証することもできます。

■保全性分析

保全性分析では、Webアプリケーションに対して既知のセキュリティ攻撃をシミュレートし、脆弱性を検出します。このシミュレートにより、すべてのフィールドやリンク、ページに対して、クロスサイトスクリプティング、SQLインジェクション、バッファオーバーフロー、パラメータ改竄などの既知の問題が存在していないかどうかのテストを行なうことができます。

表1：SecurityCheckerで検出可能な脆弱性の種類

脆弱性の種類	脆弱性の種類
セキュリティコンテキストの問題	アプリケーションの動作時に、アプリケーションの認証で使うIDのセキュリティ上の脆弱性を検出する 例：読み込み操作しか必要ないのに読み書き可能な権限でレジストリを操作しているなど
安全でないコーディング	マネージコードとアンマネージコードの両方で、セキュリティ上の脆弱性の原因となる既知のコードパターンを検出する 例：バッファオーバーフロー攻撃の可能性のあるコードなど
実行エラー	アプリケーションの実行時に発生し、アプリケーションへの攻撃を許すセキュリティ上の脆弱性とプログラムエラーを検出する 例：SQLインジェクション攻撃を受ける可能性がある脆弱性やクロスサイトスクリプティングの脆弱性など
アプリケーションの保全性の問題	アプリケーションの保全性（クレジットカードの情報や個人情報など、重要な情報を保護する能力がアプリケーションにあること）に関わる問題を検出する 例：Cookieの有効期限が長いなど
導入時の問題	さまざまな構成の詳細（アプリケーション全体のセキュリティに影響する可能性があるファイルシステムのセキュリティ、リモートアクセス機能、Webサーバーの構成、および.NETランタイムの構成など）に関わる問題を検出する 例：Web.Configでデバッグが有効になっているなど