6W-8

細粒度 DB フィルタリングルールの生成システム提案

小野 雅章　　田中 英彦
{mgs054512, tanaka}@iisec.ac.jp
情報セキュリティ大学院大学 情報セキュリティ研究科 *

1． はじめに

近年、DB と連携する Web アプリケーションは増大しており、主に SQL インジェクションを使った不正アクセスによる企業の被害件数も同様に増えている。

本稿では、DB 内情報の漏洩を阻止するため、DB フィルタリング技術を用いたフィルタリングルールの生成システムを提案する。提案するフィルタリングシステムは特定の RDBMS に依存しないため、広範囲な DB アプリケーションに適用することができ、任意のカラムを組み合わせて検索する際のアクセス権など、粒度の細かい設定を可能とする。また、事前に設定したポリシーと現在の DB 内情報を照らし合わせ、利用者にポリシーに乗っ取ったフィルタリングルールの提案も行う。

2． DB フィルタリング技術

2．1． 概要

SQL インジェクションに対する手法として次のものが挙げられる[1]。

a) **Input Validation**
ユーザが入力した文字列を検証し、危険な文字群を無害化(サニタイジング)する。ソースコードを開発する段階での防止策。

b) **Web Application Gateway (WAG)**
Web アプリケーションのクライアントサーバ間に配置する。クライアントから POST されるデータ、クッキーを検証し、異常値であればエラーとする。

c) **SQL Driver Proxy**
SQL 文の問い合わせが実行される DB の前に配置し、DB に渡す SQL 文を検証する。

上記の中で Web アプリケーションのソースコードを変更せずに情報をフィルタリングできるのは、WAG または SQL Driver Proxy である。

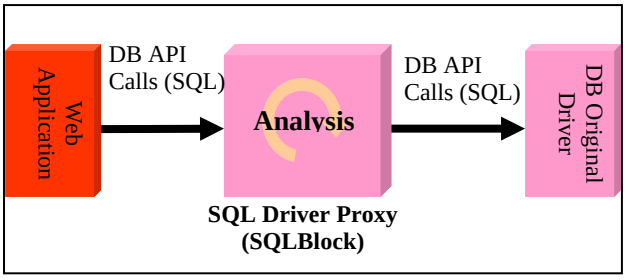

図 1: SQL Driver Proxy 概要図

本稿では Web アプリケーションの入力値を検証する WAG ではなく、DB に渡る SQL 文を直接検証できる SQL Driver Proxy を使い、DB 情報の漏洩を防止する。

2．2． SQLBlock について

SQLBlock は ODBC (Open Database Connectivity) の SQL Driver Proxy である[2]。ODBC は DB にアクセスするためのソフトウェア標準仕様 API であり、RDBMS の違いを吸収できる。

3． 提案システムについて

3．1． 動作概要とシステム構成

SQLBlock にはブラックリストの SQL 文を設定することが出来る。本ルール生成システムはそのブラックリストとなる SQL 文を生成する。次に生成されるルールの例を示す。

```
SELECT username FROM userinfo
```

上記のルールファイルは userinfo テーブルの username 参照を阻止する。また、次のようなルール

```
SELECT username, address FROM userinfo
```

は、userinfo テーブルの username と address を同時に参照するものを阻止する。このように DB の機能に依らず、細かい粒度で DB のアクセス権を設定することができる。

---

* “Proposal of a System Generating Fine Grain Database Filtering Rules”
Masaaki Ono, Hidehiko Tanaka
Graduate School of Information Security, INSTITUTE of INFORMATION SECURITY
2-14-1,Tsuruya-cho,Kanagawa-ku, Yokohama-city, Japan

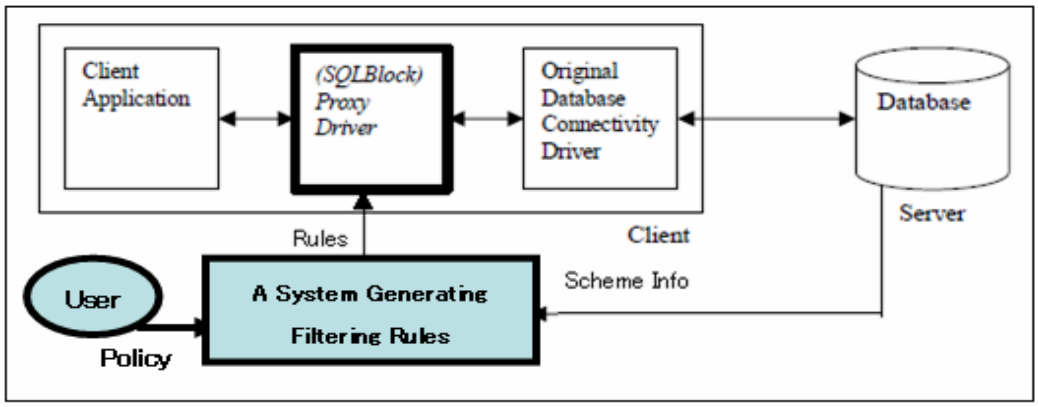

図 2: 提案システム構成図

本ルール生成システムの機能をまとめると「データベースの情報とユーザが設定したポリシーを読み取り、粒度の細かいフィルタリングルールを生成する」といえる。

なお、システムの構成図を図 2 に示した。

３．２．　想定する利用者と利用手順

想定する利用者は次のとおりである。

- システム設計者
- DB 設計者

また、開発現場における本ルール生成システムの利用手順は次のような流れとなる。

I. DB と Web アプリケーションを開発
II. SQLBlock と本ルール生成システムを導入
III. テストデータ含め、一定のデータが収められている DB のスキーマ情報を本ルール生成システムに入力
IV. ブラックリストとなるフィルタリングルールを生成し、DB 情報の漏洩を阻止する

４．　問題点

現状、解釈された内部スキーマと設定したポリシーとの結び付け精度に問題がある。例えばカラム名は等しいがテーブルが異なり、データの意味が異なる例を次の表に示す。(この場合のポリシーは個人情報の漏洩を防ぐものである)

テーブル	カラム名	個人情報
User	Name	氏名
Item	Name	商品名であるが、氏名と誤認識

ここで Item テーブルの Name を氏名と認識するのは間違いである。他、開発の段階でカラム名と実際のデータ内容が剥離してしまう状況もありえる。これらを防ぐには、意味辞書を事前に用意し、実際に収められているデータと内容を照合する方法が考えられる。

５．　まとめと今後の課題

本稿では、DB フィルタリング技術を利用するにあたり SQLBlock を採用し、特定の RDBMS に依存せず、粒度の細かいアクセス設定を可能とするフィルタリングルール生成システムを提案した。このシステム構成を用いることで、簡易に DB 情報の漏洩リスクを軽減できるものと考える。また、SQLBlock 同様、ブラックリスト方式で SQL 文のフィルタリングが行えるシステムならば、提案したルール生成システムはすべからく適用できる。

今後は、現在実装中のルール生成システムについてユーザビリティを向上させ、前述の問題点、ポリシー（個人情報保護法に準拠するポリシーなど）の拡大整備を検討したい。

参考文献

[1] OWASP, “Advanced Topics on SQL Injection Protection,” http://www.owasp.org/index.php/Image:Advanced_Topics_on_SQL_Injection_Protection.ppt

[2] Sam M.S. NG, “SQLBlock: SQL Injection Protection by Variable Normalization of SQL Statement,” http://www.sqlblock.com/sqlblock.pdf