未知のSQLインジェクション攻撃検知システムの構築

八木達哉†　　秋岡明香††　　吉永努†††
電気通信大学 情報システム学研究科

1 はじめに

近年WEBアプリケーションが多様化しており，その増加とともに脆弱性を突いた攻撃が増加している．中でもWEBアプリケーションへの攻撃の中でデータベース(以降DB)の情報を不正に閲覧，削除，改ざんが行えるSQLインジェクションは脅威となっている．

通常,SQLインジェクション攻撃をWEBサーバが受けた場合,WEBサーバのアクセスログに記録が残る．そのため，アクセスログの解析をすることで攻撃を受けているかどうかの判断を行うことができる．しかし，近年CookieにSQLインジェクションのリクエストを混入させる方法や,IDS/IPS/WAFのような攻撃検知システムを回避する方法なども出現してきている．今後もしアクセスログに記録を残さず,DBにアクセスできる未知の攻撃手法が出現した場合，管理者は攻撃の検知ができない可能性がでてくる．

本稿ではDBサーバから出力される一般クエリログからDB内部の情報を狙う未知の攻撃が発生した場合の攻撃検知方法を提案する．その後，提案した検知システムのプロトタイプを実装し，実験を行う．また，今後どのようなシステムにしていくのかについて議論する．

2 未知攻撃検知システム

2.1 SQLログ解析

DBサーバから出力される一般クエリログを用いてログ解析を行う．一般クエリログはクライアントが実行した全てのSQL文が記録される．ログ解析は以下の順番で行うものとする．

1. SQL文を構文解析し，要素に分解する
2. 構文解析した*Where*節の部分木を抽出する
3. 部分木に属する要素の出現状態を表にする

図1に解析手順を示した．まず，クエリログからSQL文を抜き出し構文解析を行う．今回，構文解析にはオープンソースであるSQL Parse Convert to Tree Array[1]を用いた．このツールを用いてSQL構文を解析木の状態に変換し出力する．その際に各ノードに要素が振り分けられる．次にSQLのレコード検索で用いられる*Where*節以降の部分木を抽出する．ユーザーが任意で変更できる部分は*Where*節にあるため,*Where*節に注目する．図1の例の場合,*Where*節の部分木に*AND*,*name* ="*abc*",*pass* ="*xyz*"があることがわかる．

最後に条件文で現れた要素を表にまとめ，パターンDBに保存する．ノード終端の*name* ="*abc*",*pass* ="*xyz*"はpredicate1,2とする．各predicateは左辺，演算子(演算子でない場合もあり)，右辺の3つに分類できる．左辺はWEBアプリケーションで使われるDBのカラムであるので，ユーザーの任意にならない．しかし右辺はユーザーの任意になる可能性がある．今回の場合,"*abc*"と"*xyz*"は任意の文字列であるため，ここではVALと表現する．また,*Where*節の部分木のすべての要素をまとめたものを文節パターン(以後pattern)とする．図1の場合はpredicate1,2と*AND*が使われているので"*predicate*1 AND *predicate2*"と表現する．

2.2 提案検知手法

本稿の検知手法は未知攻撃の検知であるので，ブラックリストでは対処できない．よってホワイトリストを用いて未知の攻撃検知を行う．WEBアプリケーションがSQLインジェクション攻撃を受けるとき，通常とは違うリクエストを受ける．よって，パターンDBに無いリクエスト情報が解析中のログから発見された場合，怪しいリクエストを含んだSQL文として警戒を行う．また，この手法と似た検出手法がElisaらの研究[2]で提案さ

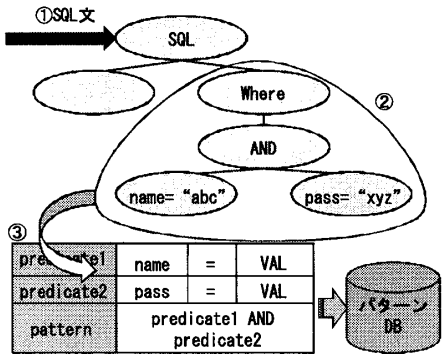

図1: 解析からパターンDBに保存するまで

Unknown SQL Injection Attack Detection System
†Tatsuya Yagi　††Sayaka Akioka　†††Tsutomu Yoshinaga
Graduate School of Infomation Systems, The University of Electro-Communications

れている. しかし,Elisa らの提案はシンプルであったが, シンプルにしたために誤検知や検知ミスが多い結果となってしまっている. また未知攻撃の検知という観点から Elisa らの手法は適さない.

3 システムの実装・実験

3.1 設定

まず, 攻撃の痕跡がないクリーンなログをパターン DB に学習させる. 十分学習させた後にログ検知の設定を行う. 設定は通常稼動用とアプリケーション追加用, 手動用の 3 種類を用意した. 今回は学習を継続せずに異常なログを発見した場合に警告を出す通常稼動時の場合に設定し,SQL インジェクション攻撃を受けた一般クエリログを解析した場合, 警告がでるかどうか試みた.

3.2 検知システムの実験と結果

一般クエリログ 100MB 分を二つ用意し, 一つを学習用, もう一つを攻撃を含んだログとした. 攻撃を含んだログには.where 節の部分を *name="abc" OR pass ="xyz"* とした物を含んだ. なお,*name="abc" AND pass ="xyz"* は学習用ログに含まれているものとする.

図 2 に検知システムでの実行結果を示す. predicate1 と predicate2 はパターン DB に学習されている. しかし pattern の"predicate1 *AND* predicate2"と"predicate1 *OR* predicate2" が異なっており学習されていない. よって, 検知できたことが示された.

4 まとめ

本稿では未知の SQL インジェクション攻撃を想定した攻撃検知システムの構築を提案し, プロトタイプを実装した. さらに, 攻撃を含んだ一般クエリログを解析させる実験を行った. この実験でログに攻撃の可能性があった場合に警告として出力されることを示した.

今後作成する検知システムの外部仕様を図 3 に示す. アクセスログや一般クエリログを Web サーバや DB サーバから独立したストレージに保管する. その後, ネットワークに接続された別の PC で解析スクリプトを実行する. 出力された各 pattern や predicate の増加値を XML に変換し, クラウドコンピューティング上に送信する. もしくは一般クエリログとアクセスログを変換せずにクラウド上に送信する.

一般クエリログは情報量が非常に多い. しかし既知の安全な SQL 文を振り分けし, 高速でログを処理できる環境を整えれば, サービスを継続したままの一般クエリログの解析は可能である. このためクラウド上にログを高速に処理する環境や複数の WEB サーバ群を管理するシステムの構築を行う必要がある. そのシステム上で複数の WEB サーバから得た XML を解析処理し, 各 WEB サーバから検出した攻撃に関する情報共有を行う. もし未知の攻撃が発見された場合, 即座に所属する全ての WEB サーバに警告, もしくはパッチ処理を行うことができる. また,WEB ページやメールなどを送信し, 警告を呼びかけるシステム構築を予定している.

```
#php sqlquery.php
predicate1:name, ope:=,VAL
predicate2:pass, ope:= ,VAL
pattern:predicate1 OR  predicate2
*********WARNING STRING**********
SELECT * FROM user WHERE name="abc"
OR pass ="xyz"
```

図 2: 検知システムでの実行結果

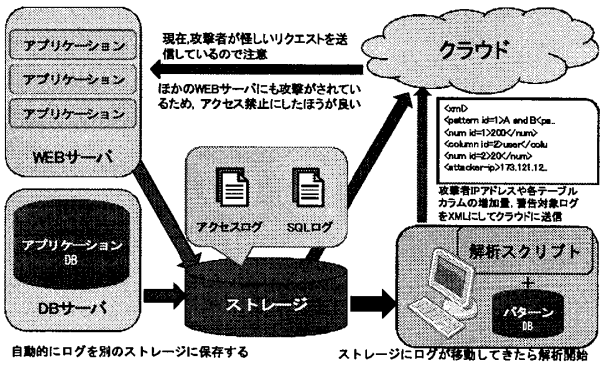

図 3: 今後予定している攻撃検知システムの仕様

謝辞

本研究の一部は, 文部科学省研究費補助金特定領域研究「情報爆発時代に向けた新しい IT 基盤技術の研究」(計画研究 A02-00-04, 情報爆発に対応する高度にスケーラバブルなモニタリングアーキテクチャ) による助成を受けた.

参考文献

[1] SQL Parse Convert to Tree Array. http://phpclasses.controloye.com/browse/package/4916.html.

[2] Elisa Bertino, Ashish Kamra, James P. Early. Profiling Database Application to Detect SQL Injection Attacks. In *IPCCC ,pp449-458*, 2007.