中央大学博士論文

潜在的要因を考慮した SQL インジェクション攻撃検知システムの開発

Michio Sonoda
園田　道夫

博士（工学）

中央大学大学院
理工学研究科
情報セキュリティ科学専攻

平成 28 年度
2016 年 12 月

あらまし

ICT の発展やインターネット利用人口の増加にともない，Web アプリケーションにもどつく Web サービスの開発者や提供者は増加している．一方で，サイバー攻撃の被害に遭うユーザや提供者も年々増加している．このような Web サービスを対象とするサイバー攻撃のことを，Web アプリケーション攻撃という．Web アプリケーション攻撃の中でも，データベース (DB) の実装された Web サーバ上の個人情報を搾取したり改ざんしたりすることを目的とする，SQL インジェクション攻撃の被害は特に深刻である．

SQL インジェクション攻撃への対策手法については数多くの研究があり，効果的な防御手法が確立されているにもかかわらず，この種の攻撃による被害は後を絶たない．その理由として，攻撃者がどのようにして SQL インジェクション攻撃を実現しているか正確に把握することなく，開発者が Web アプリケーションを開発したり，脆弱性を含む Web アプリケーションのソースコードを安易に利用したりしているためと考えられる．SQL インジェクション攻撃は，原理的には適切な知識ある者が Web アプリケーションに入力される文字列を常に正確に監視することができれば，防御することは難しくない．しかしながら，このような監視作業を人手で行うことはほとんど不可能である．

そこで本研究では，Web アプリケーションに入力される文字列のうち，特に記号の部分に着目することで，入力文字列が SQL インジェクション攻撃であるかどうか自動的に判別する攻撃検知システムを開発した．このような Web アプリケーション攻撃への自動検知システムで問題となるのが，誤検知の問題である．この誤検知には 2 つの種類が存在し，すなわち，Web アプリケーションへの正常入力を攻撃としてしまう誤検知 (False Positive) と，攻撃入力を正常としてしまう誤検知 (False Negative) である．情報セキュリティにおける自動検知システムにおいては，これら 2 種類の誤検知がなるべく少ないほうが望ましい．一方，統計学における仮説検定の問題においては，両者はそれぞれ，第一種の過誤，第二種の過誤と呼ばれており，さらにそれらの確率の大小はトレードオフの関係にあって，両者を同時に小さく保つことは難しいとされている．

本研究では，潜在曲線モデルを応用することで，SQL インジェクション攻撃の場合と正常な場合の両者の入力文字列の特徴を学習する手法を提案し，オープンソースソフト

ウェア (OSS) の Web サーバである Apache のモジュールとして使用できる攻撃検知システムを開発した．OSS で同じく Apache のモジュールである ModSecurity は，世界中で使用されている攻撃検知システムであるが，ModSecurity と本研究の提案システムを対象に性能比較実験を行った．その結果，False Negative については，提案システムは ModSecirity よりわずかに大きくなったものの，False Positive については，提案システムは ModSecurity よりもはるかに小さくなった．したがって SQL インジェクション攻撃の検知問題について，提案システムは ModSecurity よりも誤検知の少ない優れたパフォーマンスを発揮することが実験的に明らかになった．

目次

表目次

図目次

第 1 章

序論

クラウドサービスの充実，ビッグデータを扱うための環境整備により，ネットワークや情報システムに関するセキュリティは今後ますます重要性を増していくことが指摘されている．現に，文部科学省は，平成 28 年度に AIP プロジェクト（人工知能／ビッグデータ／ IoT ／サイバーセキュリティ統合プロジェクト）[1] を発表し，サイバーセキュリティが関連する分野において，多種膨大な情報の利活用を可能とする統合化技術の創出を目指している．上記の AIP プロジェクトについて，関連するキーワードを挙げた場合，機械学習という単語は必ずなんらかの形で関係してくるものと考えられる．

機械学習 [2] は，サンプルとなるデータからその特徴を抽出したり，ルールを見出したりすることで，人間の思考と同等，またはそれに類似した機能をアルゴリズムによって再現することを目的とする計算技術である．このような技術の中には，例からの学習とも呼ばれる，観測されたデータから未来に観測されるデータを予測するということにも利用されている．特に，サイバーセキュリティの分野では，過去の攻撃パターンから未知の攻撃パターンを予測して，既に知られている攻撃を防御するだけでなく，まだ知られていない攻撃に対する防御対策を講じることは，サイバー攻撃の被害を軽減させるために必要かつ重要な技術である．機械学習では，このような未知のデータに対する予測性能のことを汎化という用語を用いて表現する．

ところで，一言でサイバー攻撃といっても，その種類や目的は非常に多岐に渡る．実際にサイバー攻撃の種類やキーワードを挙げて行くと，コンピュータウィルス，標的型攻撃，ランサムウェア，ゼロデイ攻撃，パスワードリスト攻撃，ブルートフォースアタック，DoS/DDoS 攻撃，ウェブアプリケーション攻撃など多岐にわたり，全てを網羅して列挙することは大変難しい．これらの攻撃は共通する点も共通しない点も同時に持ち合わせている．昨今のようなビッグデータ時代においては，データを攻撃者から守るための技術は非常に重要であると言える．そこで本研究では，データベースへの侵入を目的とする SQL インジェクション攻撃にターゲットを定め，SQL インジェクション攻撃を検知する

数理モデルに基づくシステムを開発することを目的とする．SQL インジェクション攻撃は，ウェブアプリケーションに関する脆弱性のうち，攻撃者に狙われ易い脆弱性がランク付けされている OWASP Top 10[3] において，もっとも注意すべき脆弱性として紹介されている．SQL インジェクション攻撃は悪意のある SQL 文法をウェブアプリケーションに埋め込む手法の攻撃であり，攻撃が成功するとウェブアプリケーションのデータベースを不正に操作することが可能となる．

SQL インジェクション攻撃の歴史を遡ると，1998 年の電子雑誌 Phrack[4] に 発表された NT Web Technology Vulnerabilities[5] にたどり着く．しかし，攻撃が大々的に観測され始めたのは 2005 年以降 [6] であり，その後，ワームやボットネットなどを利用した大規模攻撃へと変貌を遂げ，現在でも継続的に多数の被害が出続けている．SQL インジェクション攻撃の基本的な原理は決して複雑なものでなく，どちらかといえば単純である．一言でいえば，SQL 文法として正しく解釈される文字列を，ウェブアプリケーションのデータベースに挿入することができれば，攻撃者はデータベースに不正にアクセスすることができる．しかしながら，ウェブアプリケーションの開発者にとって，ユーザーが入力する文字列を完璧に把握することは容易なことではない．そこで，SQL インジェクション攻撃に対して，これまでに多くの専門家によって様々な対策が研究されている．例えば，自動静的分析ツールによってウェブアプリケーションを構成するプログラムのソースコードを解析することで脆弱性を検知する方法 [7]，入力データから SQL 文を組み立てる際に，文字列連結演算でなくプリペアドステートメントとバインド機構を使用することで根本的に SQL インジェクションの脆弱性を解消する方法などが開発されている [8]．ウェブアプリケーションの開発においては，セキュリティ面での安全性を考慮することは重要であることは当然であるが，そのために運用上で不都合が生じることも問題である．プリペアドステートメントによって同じパターンの SQL 文を繰り返し使用する場合は，アプリケーションの処理に関するパフォーマンスの向上が期待される反面，同じパターンでない SQL 文を複数同時に使用する場合などにはパフォーマンスの低下を招く場合もある．また，複雑なアプリケーションでは，開発時の不備によって適切な対策とならない場合も考えられる．SQL インジェクション攻撃の場合は，ウェブアプリケーションにとって悪意のある SQL 文が挿入されるため，ウェブアプリケーションに入力される文字列を確認して攻撃を検知する，ウェブアプリケーションファイアウォール (WAF) と呼ばれるソフトウェアが開発されている．WAF による攻撃検知方法は，ブラックリストやホワイトリストによるリスティング形式，正常 な入力パターンを学習するパターン認識を行うことが一般的であるが，最近では Support Vector Machine (SVM)[9] や Bayesian Network[10] などの機械学習の手法を取り入れて開発された WAF も運用されるようになっている [11][12][13]．機械学習の手法を WAF に応用することで，攻撃検知のために参照するデータの量が減少するため，プログラムの処理効率を高めることができる．この

ような手法は WAF 以外にも，マルウェアの検知 [14] にも応用されている．機械学習の手法のメリットとして，未知の攻撃や予期せぬ脆弱性に対応できる可能性も挙げられている．しかしながら，攻撃検知では，攻撃文字列を正常文字列と判断する第一種過誤，正常文字列を攻撃文字列と判断する第二種過誤の問題がある．これらの誤検知が攻撃検知で起こる原因は，SQL インジェクション攻撃の特徴を何らかの形で決定するためであるが，これはリスティング方式やパターン認識，機械学習の方法に問わず共通する問題である．したがって，攻撃検知の問題においては，検知のために参照するデータ量を少なく，かつ誤検知をなるべく少なくする技術を確立させることが重要である．

本研究では，上述の問題に対する一つのソリューションとして SQL インジェクション攻撃の特徴を，SQL 文法において意味を持つ特殊記号から抽出するための特徴抽出モデルと潜在曲線モデル [15] を応用した攻撃検知モデルを提案し，それに基づく WAF を開発することを目的とする．研究の成果は大きく分けると以下の 2 点である．

- 提案法における特徴抽出方法は線形分類器の 1 種であり，特徴抽出モデルはその分類性能に関する確率を表すものと考えられること
- SVM が正常を攻撃と誤検知するデータに対しても，潜在曲線モデルは正確に正常と検知できること

提案手法は，SQL インジェクション攻撃の実データに基づいて特徴を抽出し，攻撃検知を実行するため，機械学習分野における，いわゆる例からの学習を攻撃検知に応用したものである．一般的に機械学習を用いる場合は，特徴抽出と判定の 2 つのフェーズについて考える必要があることが多い．SQL インジェクション攻撃のデータを扱う場合，攻撃データは文字列であるため，文字列を数値データに置き換えるためにも特徴抽出のフェーズが必要である．さらにいえば，SQL インジェクション攻撃においては，文字列を数値データに変換する，この特徴抽出フェーズは重要であると考えられる．例外はあるものの，多くの SQL インジェクション攻撃には，特殊記号が一定の割合含まれている．したがって，はじめに，SQL インジェクション攻撃のデータのこの性質を特徴抽出モデルとして定義する．目標は，ある特徴を持つときに，それが攻撃であるかどうかを判定することであるから，特徴抽出モデルを確率モデルにより記述すれば，ベルヌーイ分布に従うものと考えられる．本論文では，まずこの特徴抽出モデルが，Soft Confidence-Weighted (SCW) Learning [16] と呼ばれる線形分類器と類似するものであることを示す．

次に，SQL インジェクション攻撃の文字列から得た特徴を，有効に攻撃検知に活かすための攻撃検知モデルを提案する．このフェーズは機械学習における判定のフェーズであり，教師あり学習のアルゴリズムを適用することになる．本研究では，特徴抽出によって得られた特徴をうまく組み合わせる方法を，潜在曲線モデルを応用することでモデル化し，これを攻撃検知モデルとして定義する．特徴抽出のフェーズを経ているため，攻撃検

知には様々な教師あり学習のアルゴリズムを適用することができる．そこで，提案モデルの有用性を比較するために，汎化能力が高いとされる SVM を用いた場合の検知実験を行う．

このような攻撃検知に関する話題において機械学習の手法を応用する最大のメリットとして，未知の攻撃への対策となり得ることが挙げられる．攻撃者は攻撃を成功させるためにあらゆる工夫を行うため，例えば，WAF による検知をくぐり抜けるバイパス攻撃を攻撃中に思いつくということも想定しなければならない．このようなバイパス攻撃は，単純なパターンマッチングやリスティング方式による WAF が苦手とする攻撃として有名であるため，汎化能力を持つと期待される機械学習の手法を攻撃検知に活用できる技術を確立させることは重要な課題であるといえる．

本研究で用意した学習用データと検証用データを用いて提案手法と SVM による手法を比較したところ，学習データに含まれていなかったパターンの検証用データに対して，SVM による手法ではカーネル関数の利用の有無に関わらず正しく検知することができず，提案手法では比較的精度良く検知できることを確認することができた．なお，本研究では，オープンソースソフトウェア (OSS) の Web サーバである Apache[17] に組み込むことのできる WAF に提案手法のアルゴリズムを実装し，その一部のコアコードをインターネット上に公開している．

本論文の構成は以下の通りである．

2 章では，SQL インジェクション攻撃のメカニズムを具体例を用いながら解説する．3 章では，本研究と関連のある機械学習のアルゴリズムとその性質について紹介する．4 章では，SQL インジェクション攻撃の特徴抽出方法と関連する特徴抽出モデルを提案する．5 章では，潜在曲線モデルを応用した SQL インジェクション攻撃の検知モデルを提案する．6 章では，実データを用いることで，提案手法を従来手法として知られている機械学習のアルゴリズムを用いた手法と比較するための実験を行う．7 章では，提案手法と実験結果に対する考察を行い，最後の 8 章で本研究のまとめを行う．

第 2 章

SQL インジェクション攻撃

本章では，ウェブアプリケーションのデータベースに不正アクセスする攻撃手法として知られている SQL インジェクション攻撃について解説する．

2.1 SQL インジェクション

インターネットは様々な場面で欠かすことができないツールとなっており，日々，様々なデータがやりとりされている．インターネットを介して様々なサービスを提供するウェブアプリケーションは，この様なデータをデータベースに蓄積しているため，データベースには，ビジネス上有用であると考えられるデータから個人情報を含む多種多様なデータまで，様々な形態のものが存在する．

ウェブアプリケーションは，データベースとのやりとりを SQL という言語を用いて行い，ブラウザを通じてユーザーに様々なサービスを提供する．ウェブアプリケーションはユーザーからの入力を待ち，それに基づいて必要なときにデータベースとやりとりをするが，一般的に，ウェブアプリケーションの開発者が，すべてのユーザーがどのような入力をするか完全に想定することは困難である．SQL インジェクション攻撃 [18] は，このような開発者が想定し得ない入力によって引き起こされるサイバー攻撃である．

2.2 SQL インジェクション攻撃の具体例

ウェブアプリケーションは，ユーザーの入力したデータに基づいて，データベースとやりとりをするため，ユーザーが SQL 文を入力すると，SQL インジェクションの脆弱性をもつウェブアプリケーションは，ユーザーにデータベースを不正にアクセスされる恐れがある．

以下，具体例を用いて SQL インジェクション攻撃のメカニズムを説明する．ここでは，

ユーザの入力からデータベースへアクセスするための SQL 文を直接生成する Web アプリケーションを想定し，アプリケーションを利用する際に，URL にユーザを特定する情報が含まれている場合を考える．

具体的には以下のような例を考える．

```
http://www.sqlinjection.ac.jp?id=14990000
```

この場合，ウェブサーバはアプリケーションサーバに `id=14990000` という情報を送り，ウェブアプリケーションはデータベースに送るための SQL 文を生成する．その結果，`id=14990000` に紐づいたユーザの情報をウェブページに表示する．

ここで URL を，

```
http://www.sqlinjection.ac.jp?id=14990000' or 'A'='A
```

と変更すると，`A=A` は SQL 言語において常に真の命題であるため，`id=14990000` に紐づいた情報も含めて，データベース上のすべてのユーザの情報がウェブページに表示される危険性をもつことになる．このようにして，攻撃者はウェブアプリケーションのデータベースに不正に侵入し，悪意のある SQL 文を使用してその目的を達成する．

SQL インジェクション攻撃に対して脆弱なウェブアプリケーションは，任意の SQL 文をブラウザの入力欄から入力されることによってデータベースへの不正なアクセスを許すことになってしまう．したがって，ユーザーの入力を制限すればするほど，SQL インジェクション攻撃の被害に会う可能性は低くなると考えられる．しかし，入力の制限はアプリケーションのサービスの範囲を狭めることにも繋がるため，攻撃となり得る文字列だけを検知する手法を開発することは実用上も重要である．

もっともシンプルな対策は，危険な記号の入力をエスケープする方法であるが，これは完全な対策とならない場合がある．根本的な対策は，ユーザーの入力から SQL 文を組み立てないようにアプリケーションを開発することであり，プリペアドステートメントを適切に活用することである．しかしながら，このような根本的な対策が施されていない Web アプリケーションは実際に存在し，被害を受けている．

SQL インジェクション攻撃を実現するには，シングルクォートやセミコロンなどの特殊な記号が必要となるため，Web アプリケーションに対する通常の入力（正常データ）とは異なる特徴をもっている．

しかしながら，ブラウザ上の入力欄に入力される文字列は，例えば，英文，顔文字，Wiki の文法である場合も考えられるため，SQL インジェクション攻撃に含まれる記号を利用した攻撃検知法では，正常データを攻撃データと誤検知することをどのように防ぐかということが問題となる．

筆者らは，SQL インジェクション攻撃に含まれる記号に着目した攻撃検知法 [19] を提

案し，攻撃データを複数個収集して SQL インジェクション攻撃に含まれる記号の分布を調べると，記号の分布がベキ乗則に従う傾向にある [20] ことを確認している．本研究では，これらの手法 [19][20] に基づいて，攻撃データの特徴と正常データの特徴をそれぞれのサンプルを収集・比較することで抽出し，攻撃検知を行うアプリケーションを Apache のモジュール開発機能を用いて開発した．

第 3 章

既存手法のまとめ

SQL インジェクションの脅威に対しては，Web アプリケーションの開発時点での対策が非常に重要であり，基本的な対策方法として，バインドメカニズムを適切に使用して，根本的に SQL インジェクション攻撃を攻撃者にさせないことが重要である [8] という主旨の指摘がある．バインドメカニズムが SQL インジェクション攻撃の対策となる理由は，ユーザーが入力した文字列を SQL 文として扱わないためである．しかしながら，いつでもバインドメカニズムが利用できるわけではなく，検索条件の検索項目を動的に変更する場合などでは，SQL インジェクション攻撃を引き起こさないような対策を必要とする場合がある．このようなことから，ユーザーが入力した文字列をチェックして攻撃を成功させないような工夫が必要となる．本章では，Web アプリケーションに入力された文字列から攻撃を防御する既存手法を紹介する．

3.1 エスケープ処理

SQL インジェクション攻撃の例を用いてエスケープ処理の有効性を確認する．ここでは，次のようなユーザー認証のための SQL 文を組み立てる部分のプログラムを考える．

```
SELECT UserID FROM accountTable WHERE id='uid' AND pass='pasw'
```

Web ページの入力フォームの id 入力欄に，

```
abc' OR 1=1- -
```

が入力されると，

```
SELECT UserID FROM accountTable WHERE id='abc' OR 1=1- -' AND
pass='pasw'
```

となる．この SQL 文では，id=’abc’ はシングルクォートを入力されることにより，文字列定数を終わらせる効果があり，id が abc ではない場合でも，OR 1=1 は常に真であると判定され，- -でこれ以降のものをコメント文にして無効化する効果を持つ．これにより，本来認証されるべきでないユーザもログインできてしまう．

エスケープ処理とは，上述の例のシングルクォートのような，SQL 文法として意味をもつ記号を，SQL 文法として意味のない他の記号に置き換える処理のことをいう．具体的には，シングルクォートが含まれている場合に，シングルクォートをもう一つ追加するという手法がよく利用されている．

’ OR 1=1- - → " OR 1=1 - -

こうすることで，SQL 文

SELECT UserID FROM accountTable WHERE id=’abc" OR 1=1- -’ AND pass=’pasw’

は SQL の文法としてエラーとなる．エスケープ処理を適切に行うことで，SQL インジェクション攻撃を防ぐことができる一方で，対象となるすべての部分が適切にエスケープ処理されているかどうかをチェックする必要があり，ヒューマンエラーが起きやすいことが問題点として挙げられる．開発者がエスケープ処理をしたつもりでも，想定外の入力でそれを無効化されるケースも考えられるため，攻撃を検知する仕組みも合わせて活用することが望ましいといえる．

3.2　Web アプリケーションファイアウォール

Web アプリケーションファイアウォールは WAF(ワフ) とも呼ばれ，Web サーバー側に設置される防御システムである．適切に WAF を活用することで，脆弱性を抱えている Web アプリケーションも SQL インジェクションを防御することができる．攻撃を検知する仕組みとして最も単純なものが，実際に攻撃に利用される文字列をブラックリスト化する手法である．そのような文字列を正規表現で表したり，パターンマッチングをしたりすることで同種の攻撃を防ぐことができる．しかしながら，攻撃者はこのような検知ルールをバイパスする新たな攻撃手法を開発することが問題である．現在では，クラウドサービスを利用した格安な商用の WAF が販売されている [21]．WAF の最大の問題は攻撃検知の設定やメンテナンスであり，Web アプリケーションは比較的誰でも簡単に作ることができることに比べると，WAF を適切に導入する技術的知見は広く浸透しているとはいえない．さらに，保護すべき Web アプリケーションの対象はケースバイケースであることが通常である．Web アプリケーションを防御する手法としてホワイトリストと呼ばれる，

特定の文字列を入力することを禁止する手法も広く活用されているが，多種多様の Web アプリケーションが必要とされている実情を考えると，正常なサービスを提供することが困難になる恐れもあるため，ブラックリストやホワイトリストなどのリスティング手法だけに頼った防御機構を活用するにはこれらのトレードオフの関係をどのようにするかなどの様々な知見が必要となる．加えて，IT 技術の進化と共に新たな攻撃が開発されることで，未知の攻撃に対する脅威にも対抗できる WAF の必要性は高まっているといえる．このような未知の攻撃に対する技術として，機械学習が注目を浴びている．

3.3 機械学習

機械学習 [2] は与えられたデータの特徴を抽出したり，そのような特徴を用いて何らかの判断を用いるときに利用される技術である．ラベルがないデータをクラスタリングするために利用される教師なし学習と，ラベルがあるデータを学習して未知のデータに予測ラベル付けするために利用される教師あり学習がある．攻撃検知を機械学習で行う場合は，教師あり学習を考えることになる．しかしながら，教師あり学習を行うためには，データにラベルをつける必要がある．攻撃検知では，攻撃文字列であるか正常文字列であるかのいずれかである．一般的に，SQL インジェクション攻撃を成功させるには，攻撃データとしては不完全な文字列を Web アプリケーションに与えてその挙動を把握する必要がある．本研究では，攻撃が成功するしないに関わらず，攻撃を成功させるために利用される可能性がある文字列はすべて攻撃文字列であるとラベル付けすることにする．それ以外の文字列はすべて正常文字列である．

攻撃検知において機械学習を活用したいというモチベーションは，Web アプリケーションに入力される文字列にはラベルがついていないためである．SQL インジェクション攻撃は一定の条件は必要であるが，記号を含む任意の文字列で構成されるため，その構成要素を考えると正常文字列とオーバーラップする部分は少なくない．したがって，SQL インジェクション攻撃の検知を機械学習で行う場合は，最初に行う特徴抽出が重要となる．本研究は，このような特徴抽出を行う手法を目的としているが，本研究で提案するアルゴリズムやモデルとの比較を行うため，機械学習のアルゴリズムとして知られている Soft Confidence-Weighted (SCW) アルゴリズム [16] を紹介する．SCW はデータが与えられる毎に逐次にパラメータを更新していくオンラインアルゴリズムであり，原点を通る線形な方程式を用いて空間を 2 つに分割する線形分類器である．次の章で提案するモデルにおける攻撃検知手法は線形分類器の一種であり，データは 2 次元ユークリッド空間上に与えられることから，ここでは 2 次元データに関する SCW のアルゴリズムを紹介する．

データが 2 次元 $\boldsymbol{p} = (p_1, p_2)$ の場合，SCW は以下の式 (3.1) を用いて学習を行う．

$$w_1 p_1 + w_2 p_2 = 0, \tag{3.1}$$

また，ラベルデータを $y \in \{1, -1\}$ とし，$y = 1$ を攻撃文字列，$y = -1$ を正常文字列と割り当てることにする．SCW のパラメータ w_i $(i = 1, 2)$ は，平均ベクトル $\boldsymbol{\mu}$，共分散行列 Σ からなる 2 次元ガウス分布に従うものとして定義されている．SCW の学習では，学習が進むと正規分布の分散と共分散の値が小さくなるようになり，具体的には $t = 1, 2, \ldots$ として μ_t と Σ_t は，以下の式 (3.2),(3.3) を用いて更新される．

$$\mu_{t+1} = \mu_t + \alpha_t y_t \Sigma_t p_t \,, \tag{3.2}$$

$$\Sigma_{t+1} = \Sigma_t - \beta_t \Sigma_t p_t^{\mathrm{T}} p_t \Sigma_t \,, \tag{3.3}$$

ただし，

$$\alpha_t = \min\left\{C, \max\left[0, \frac{1}{v_t \zeta}\left(-m_t \psi + \sqrt{m_t^2 \frac{\phi^4}{4} + v_t \phi^2 \zeta}\right)\right]\right\}, \tag{3.4}$$

$$\beta_t = \frac{\alpha_t \phi}{\sqrt{u_t} + v_t \alpha_t \phi}, \tag{3.5}$$

$$u_t = \frac{1}{4}\left(-\alpha_t v_t \phi + \sqrt{\alpha_t^2 v_t^2 \phi^2 + 4 v_t}\right)^2, \tag{3.6}$$

$$v_t = p_t^{\mathrm{T}} \Sigma p_t, \tag{3.7}$$

$$m_t = y_t \cdot < \mu_t, p_t >, \tag{3.8}$$

$$\phi = \Phi^{-1}(\eta)\,, \tag{3.9}$$

$$\psi = 1 + \frac{\phi^2}{2}, \tag{3.10}$$

$$\zeta = 1 + \phi^2, \tag{3.11}$$

であり，式 (3.8) の右辺の記号 $<, >$ はベクトルの内積を表しており，式 (3.9) の右辺における Φ は正規分布の累積分布関数である．

SCW のアルゴリズムでは，式 (3.4) の右辺の C，式 (3.9) の右辺の η の 2 つの定数の値を適当に決める必要があり，C はパラメータの更新の値を調整する効果があり，η は確率 $1 - \eta$ の誤分類を許容するという意味がある．

第 4 章

SQL インジェクション攻撃と記号分布

この章では，SQL インジェクション攻撃の文字列に含まれる記号に着目した SQL インジェクション攻撃の特徴抽出と，その特徴を用いた攻撃検知手法 [23][24][25] についてまとめる．本論文では，SQL インジェクション攻撃の検知に有効的に作用しうる記号のことを攻撃特徴記号と呼ぶ．

4.1 攻撃特徴記号とその分布

2, 3 章で SQL インジェクション攻撃のいくつかの具体例を紹介したが，この攻撃は Web アプリケーションに入力された文字列の中に SQL の文法として意味をなす文字列が入力されることによって引き起こされるサイバー攻撃である．Web アプリケーションはユーザーに様々なサービスを提供するため，ユーザーに文字列の入力を要求する場面がある．ユーザーに求める入力は，文字列や数値であることが圧倒的に多いと考えられる．SQL インジェクション攻撃の基本的な対策は，ユーザーの入力した文字列を使って SQL 文を組み立てないようにすることである．SQL では文字リテラルはシングルクォートで囲むきまりになっているので，攻撃者はこれを悪用し，Web アプリケーションの開発者がユーザーの入力を受け付けるために作成した SQL 文を巧みに利用して，Web アプリケーション側が生成する SQL クエリとして矛盾がないように攻撃文字列を注入してデータベースに不正にアクセスを実現させる．その際によく観測されるのがシングルクォートやコメントなどの記号である．以下に，SQL インジェクション攻撃の具体例をいくつか列挙する．

- `';select if( user() like 'root@%',`

```
        benchmark(100000,sha1('test')), 'false');
```

- `;SELECT CHR(65)||CHR(66);`
- `1,UNION ALL INSERT INTO mytable(mycol) VALUES`
  `('<? pasthru($_GET[cmd])\ni?>');`
- `10; DROP TABLE members --`
- `BEGIN IF 1=1 THEN dbms_lock.sleep(3);`
  `ELSE dbms_lock.sleep(0); END IF; END;`
- `1,(CREATE TABLE mydata(t text);`
- `'SELECT inet_server_port();`
- `' or 'aaa'='aaa--`

SQL インジェクション攻撃の多くは記号を必要とすることから，このような記号が含まれている入力をチェックすることで攻撃を防御できる可能性はかなり高くなるといえる．しかしながら，シングルクォートやコメントで利用される記号は攻撃文字列以外にも含まれる可能性も存在する．攻撃を止めることは重要であるが，それによって Web アプリケーションの利用が制限されてしまうとサービスの提供ができなくなってしまうことも大きな問題である．本研究では，SQL インジェクション攻撃の文字列 (以下，攻撃文字列) とそれ以外の文字列 (以下，正常文字列) を収集・生成してそれぞれの文字列にどのような記号がどの程度含まれているか確認するための調査を行った．なお，正常文字列としては，様々な Web アプリケーションに対するユーザーの多様の入力を想定し，収集・生成した．

SQL インジェクション攻撃の重要な特徴ともいえる記号が実際の攻撃文字列にどの程度含まれ，また，そのような記号が正常文字列にどのように含まれるかを調べるために，624 個の攻撃文字列と 234 個の正常文字列を調査した結果を表 4.1 にまとめる．なお，表の数値データは以下のように計算したものである．

- l_i : i 番目の文字列 (ただし $i = 1, 2, \ldots$)
- $|l_i|$: l_i の文字列長
- L_A : 攻撃文字列全体のサンプル集合
- L_N : 正常文字列全体のサンプル集合
- I: 攻撃文字列の総数
- J: 正常文字列の総数
- $|L_A| = \sum_{i=1}^{I} |l_i|$: 攻撃文字列の総文字列長 (ただし $l_i \in L_A$)
- $|L_N| = \sum_{j=1}^{J} |l_j|$: 正常文字列の総文字列長 (ただし $l_j \in L_N$)
- s_k : 攻撃特徴記号 (ただし k は自然数)
- $x_{k,i}$: 記号 s_k の文字列 l_i における出現頻度

以上の準備のもとで，攻撃文字列および正常文字列を含むデータのサンプル集合について，

$$P_A(k) = \frac{\sum_{i=1}^{I} x_{k,i}}{|L_A|}, \tag{4.1}$$

$$P_N(k) = \frac{\sum_{j=1}^{J} x_{k,j}}{|L_N|}, \tag{4.2}$$

を求める．

式 (4.1) は，k 番目の特徴記号がそのデータのサンプル集合中のすべての攻撃文字列の総文字列長を全体としたときにどれくらいの割合で含まれるか (攻撃特徴含有率) を定量化している．同様に式 (4.2) は，k 番目の特徴記号がそのデータのサンプル集合中のすべての正常文字列の総文字列を全体としたときにどれくらいの割合で含まれるか (正常特徴含有率) を表している．$k = 1, 2, \ldots, 20$ までの 20 種類の特徴記号についての攻撃特徴含有率および正常特徴含有率をまとめると，表 4.1 のようになる [23][24]．

表 4.1 において，$P_A(k)$ の値が高く，かつ $P_N(k)$ の値が低い記号 s_k が攻撃検知に向いている記号であるといえる．そのような記号を選ぶ一つの基準として，

$$|P_A(k) - P_N(k)| > 0.30, \tag{4.3}$$

となる k を選ぶと，$k = 1, 2, 3, 4, 5$ つまり，

- s_1：半角スペース
- s_2：セミコロン
- s_3：シングルクォート
- s_4：右側丸括弧
- s_5：左側丸括弧

が選び出される．これらの記号は文字リテラルや SQL 文の連結，関数の利用など攻撃文字列を生成する上でも重要な意味をもっている．しかしながら，これらの記号は正常文字列にも含まれるため，これらの記号を用いて攻撃検知を成功させるには何か工夫が必要である．そこで，上述の 5 つの記号 $\{s_1, s_2, s_3, s_4, s_5\}$ の組み合わせを考えて，どの組み合わせが攻撃検知に向いているかということを，収集したデータから選別するためのアルゴリズムを提案した [23][24]．

表 4.1　主な特徴記号の攻撃含有率と正常特徴含有率

記号	$P_A(k)$	$P_N(k)$
s_1：半角スペース	0.971	0.098
s_2：セミコロン	0.664	0.000
s_3：シングルクォート	0.483	0.000
s_4：右側丸括弧	0.459	0.060
s_5：左側丸括弧	0.414	0.090
s_6：右側中括弧	0.000	0.060
s_7：左側中括弧	0.000	0.026
s_8：右側大括弧	0.000	0.026
s_9：左側大括弧	0.000	0.026
s_{10}：シャープ	0.032	0.021
s_{11}：パーセント	0.024	0.026
s_{12}：ダブルクォート	0.010	0.000
s_{13}：アンパサンド	0.019	0.021
s_{14}：バックスラッシュ	0.032	0.000
s_{15}：パイプ	0.040	0.103
s_{16}：等号	0.294	0.060
s_{17}：大なり不等号	0.000	0.090
s_{18}：小なり不等号	0.000	0.038
s_{19}：アスタリスク	0.128	0.094
s_{20}：スラッシュ	0.112	0.073

5 つの記号の組み合わせは，$(1+1)^5 - 1 = 31$ 通りになるため，

$$
\begin{aligned}
&S_1 = \{s_1\}, S_2 = \{s_2\}, \ldots, S_5 = \{s_5\},\\
&S_6 = \{s_1, s_2\}, S_7 = \{s_1, s_3\}, \ldots, S_{15} = \{s_4, s_5\},\\
&S_{16} = \{s_1, s_2, s_3\}, S_{17} = \{s_1, s_2, s_4\}, \ldots, S_{25} = \{s_3, s_4, s_5\},\\
&S_{26} = \{s_1, s_2, s_3, s_4\}, S_{27} = \{s_1, s_2, s_3, s_5\}, \ldots, S_{30} = \{s_2, s_3, s_4, s_5\},\\
&S_{31} = \{s_1, s_2, s_3, s_4, s_5\},
\end{aligned}
\tag{4.4}
$$

と記号の集合をおいて，$|S_k|$, $k = 1, 2, \ldots, 31$ で入力文字列 l_i における集合 S_k に属する記号の出現頻度を表すものとするとき，入力文字列 l_i における記号集合 S_k の含有率，

$$
p_{ik} = \frac{|S_k|}{|l_i|}, \tag{4.5}
$$

を計算し，式 (4.5) の含有率がある閾値 α_j より大きくなる個数を数えるための関数，

$$h_j(p_{ik}) = \begin{cases} 1 & (p_{ik} > \alpha_j); \\ 0 & (p_{ik} \leq \alpha_j), \end{cases} \tag{4.6}$$

を考える．$h_j(p_{ik}) = 1$ であるときに l_i を攻撃と検知し，$h_j(p_{ik}) = 0$ であるときに l_i を正常と検知するように定義する．

検知成功率の評価については，I 個の攻撃文字列と J 個の正常文字列に対し，

$$x_{kj} = \frac{\sum_{i=1}^{I} p_{ik}}{I}, \tag{4.7}$$

$$y_{kj} = \frac{\sum_{j=1}^{J} p_{jk}}{J}, \tag{4.8}$$

を計算する．式 (4.7) は攻撃文字列の検知成功率，式 (4.8) は正常文字列の検知成功率をそれぞれ表している．x_{kj}, y_{kj} それぞれの値は 1 に近ければ近いほど検知精度は高いといえ，これらの値は記号集合の選び方 k と閾値 j の選び方に依存して変化する．

ヒューリスティックな探索手法により，任意の $0 \leq \beta \leq 1$ について μ が安定的に高い値をとる記号の組として，

$$S_{12} = \{s_1, s_2, s_4\},$$

が，攻撃検知の閾値として $\alpha = 0.08$ が経験的に選び出されている [23][24]．

さらに筆者らは，与えられたサンプル集合に対し，記号集合 S_k と攻撃検知の閾値 α_j を，経験的な手法によらずに最小二乗法を応用して計算するためのアルゴリズム [26] を提案した．

アルゴリズム 4.1.1（記号集合 S_k と攻撃検知の閾値 α_j の計算アルゴリズム [26]）

1. J 個の閾値候補 $0 \leq \alpha_j \leq 1\,(j = 1, 2, \ldots, J)$ を固定
2. すべての i, k に対して，x_{kj}, y_{kj} を計算することを R 回繰り返す
3. すべての S_k に対して，

$$\sum_{r=1}^{R} \frac{x_{kjr}}{R} \geq 0.90, \tag{4.9}$$

   を満足する α_j を選んでできる集合を A_k とする．
4. A_k のうち，

$$\mu = \max_j \left\{ \left(\sum_{r=1}^{R} \frac{x_{kjr}}{R} \right) + \beta \left(\sum_{r=1}^{R} \frac{y_{kjr}}{R} \right) \right\}, \tag{4.10}$$

   となる閾値 α_j を選び，特徴抽出を行う記号集合 S_k を決定する．ただし，$0 \leq \beta \leq 1$ とする．

□

攻撃検知に最適な記号の組は用意するサンプルに依存するため，SQL インジェクション攻撃を検知するために必要である記号の組み合せを固定することは容易ではない．しかしながら，SQL インジェクション攻撃のデータを収集して，全データに含まれる記号の頻度を数えて降順に並べると，ゼータ分布のように急減少する単調減少な関数で記号が分布することが確認されている [20][27]．SQL インジェクション攻撃を成功させるためには，シングルクォートやセミコロンなどの特殊文字が必要になるケースが多いため，記号の分布を調べるとこのような性質が観測されるものと考えられる．そこで本研究では，SQL インジェクション攻撃に含まれる記号の分布の性質 [23][24] と，そのような記号を含む文字列が SQL インジェクション攻撃である確率を結びつける確率モデルを提案する．

4.2　攻撃文字列の確率モデル化

4.1 節では，SQL インジェクション攻撃に含まれる記号に着目して攻撃を検知することの有用性について議論した．ユーザーによって Web アプリケーションに入力された文字列における攻撃特徴記号の含有率は，0.10 程度の小さな値の場合でも攻撃になりうるデータが多いことを確認している [19]．本節では，上記のことを表現する確率モデルを提案する．

4.2.1　提案モデル

$l_i, i = 1, 2, \ldots$ を i 番目の入力文字列とし，x_i を l_i に含まれる攻撃特徴記号の総数，$|l_i|$ を l_i の文字列長とすると，入力文字列 l_i に含まれる攻撃特徴含有率は，

$$z_i = \frac{x_i}{|l_i|}, \tag{4.11}$$

と表すことができる．実数 z, b を $0 < z < 1, 0 < b < 1$ とすると，z に関する関数 z^b は 0 から 1 に値をとる単調増加関数で，z が 0 に近いときでも z^b は 1 に近い値をとる．この関数は，含有率が小さい値でも攻撃である可能性が高いという，攻撃特徴記号のもつ性質を表すものと考えることができる．これとベルヌーイ分布の性質を合わせると，攻撃特徴含有率が z_i である入力文字列 l_i が，攻撃入力である確率は，

$$\Pr\left(y \mid b\right) = (z^b)^y (1 - z^b)^{1-y}, \tag{4.12}$$

であると表現することができる．ただし，入力文字列 l_i が攻撃であるときは $y = 1$，正常であるときは $y = 0$ とラベル付けする．b は式 (4.12) で定義される確率モデルの未知のパラメータであり，一般的には与えられたデータからパラメータの値を推定することに

なる．パラメータを推定する方法としては，最尤推定法やベイズ推定法などが知られているが，式 (4.12) の確率モデルのパラメータの場合，最尤推定量を解析的に求めることは困難である．そこで本研究では，式 (4.12) のモデル最尤推定量の近似値を求める定理 4.2.1[25] を与える．

4.2.2 最尤推定量の近似

式 (4.12) で定義される確率モデルの尤度関数は，r 個の攻撃文字列から得られる攻撃特徴含有率 $z_1, z_2, \ldots, z_r$ と $n-r$ 個の正常文字列から得られる正常特徴含有率 $z_{r+1}, z_{r+2}, \ldots, z_n$ が与えられると，攻撃文字列のラベルは $y_i = 1$ で正常文字列のラベルは $y_i = 0$ であるから，

$$L(b) = \prod_{i=1}^{n} (z_i^b)^{y_i} (1 - z_i^b)^{1-y_i} \tag{4.13}$$

$$= \left(\prod_{i=1}^{r} z_i^b \right) \left(\prod_{i=r+1}^{n} (1 - z_i^b) \right), \tag{4.14}$$

となる．

したがって，対数尤度関数は，

$$LL(b) = \log L(b) = \left(\sum_{i=1}^{r} z_i \right) b + \sum_{i=r+1}^{n} \log(1 - z_i^b), \tag{4.15}$$

となる．

最尤推定量は方程式，

$$\frac{d}{db} LL(b) = \left(\sum_{i=1}^{r} z_i \right) + \sum_{i=1}^{n} \frac{z_i^b \log z_i}{1 - z_i^b} = 0, \tag{4.16}$$

を解くことで求められる．

しかしながら，式 (4.16) の解を計算することは困難であるため，対数尤度関数を近似する多項式を求めることで最尤推定量の近似値を計算する方法を与える．なお，式 (4.15) の対数尤度関数において，最右辺第 1 項 $\left(\sum_{i=1}^{r} z_i\right) b$ の係数部分は攻撃文字列のみから得られるため，この係数を，

$$Z_{\text{attack}} = \sum_{i=1}^{r} z_i, \tag{4.17}$$

と表わすことにする．

定理 4.2.1（確率モデルのパラメータの近似最尤推定 [25]）

$$\Pr\left(y \mid b\right) = (z^b)^y (1 - z^b)^{1-y}, \tag{4.18}$$

で表される確率モデルについて，$y = 1$ となるデータが $z_1, z_2, \ldots, z_r$ で，$y = 0$ となるデータが $z_{r+1}, z_{r+2}, \ldots, z_n$ であるとき，その最尤推定量 b_{mle} は式 (4.19) で近似することができる．

$$b_{\mathrm{amle}} = A + B - \frac{\beta}{3\alpha}, \tag{4.19}$$

ただし，

$$A = \sqrt[3]{\frac{-q + \sqrt{q^2 + 4p^3}}{2}}, \tag{4.20}$$

$$B = \sqrt[3]{\frac{-q - \sqrt{q^2 + 4p^3}}{2}}, \tag{4.21}$$

$$p = \frac{1}{3}\left(\frac{\gamma}{\alpha} - \frac{\beta^2}{3\alpha^2}\right), \tag{4.22}$$

$$q = \frac{2\beta^3}{27\alpha^3} - \frac{\beta\gamma}{3\alpha^2} + \frac{\delta}{\alpha}, \tag{4.23}$$

$$\alpha = \frac{1}{6}g^{(4)}(b_1), \tag{4.24}$$

$$\beta = \frac{1}{2}\left[g^{(3)}(b_1) - b_1 g^{(4)}(b_1)\right], \tag{4.25}$$

$$\gamma = g^{(2)}(b_1) - b_1 g^{(3)}(b_1) + \frac{1}{2}b_1^2 g^{(4)}(b_1), \tag{4.26}$$

$$\delta = Z_{\mathrm{attack}} + g^{(1)}(b_1) - b_1 g^{(2)}(b_1) + \frac{1}{2}b_1^2 g^{(3)}(b_1) - \frac{1}{6}b_1^3 g^{(4)}(b_1), \tag{4.27}$$

である．

□

式 (4.12) の提案モデルにおける最尤推定量は，定理 4.2.1 の公式を用いて計算することは可能であるが，そのためには b_1 の値をうまく設定する必要がある．4.2.4 節で考察するが，展開の中心が最尤推定量に近い値であればあるほど，近似精度は向上する．そのような値を見つけるために，テイラー展開に基づいて近似の項を増やし，その結果，近似最尤推定の精度が向上することを示す．

4.2.3　定理 4.2.1 の証明

ここでは，定理 4.2.1 の証明を与える．提案モデル中の式 (4.12) の最尤推定量は，式 (4.16) から求められるが，この方程式の解を求めることは難しい．そこで，対数尤度関数 (4.15) の $\sum_{i=r+1}^{n} \log(1 - z_i^b)$ の部分を多項式で近似することによって方程式 (4.16) の右辺を近似多項式で置き換えることを考える．

具体的には，

$$g(b) = \sum_{i=r+1}^{n} \log(1 - z_i^b), \tag{4.28}$$

とおくと，式 (4.28) は n 階微分可能であるため，対数尤度関数 $LL(b)$ は $0 < b < 1$ の領域において，テイラーの定理によって多項式，

$$\begin{aligned} F(b) = Z_{\text{attack}} + g(b_1) + g^{(1)}(b - b_1) + \frac{1}{2}g^{(2)}(b_1)(b - b_1)^2 \\ + \frac{1}{6}g^{(3)}(b_1)(b - b_1)^3 + \frac{1}{24}g^{(4)}(b_1)(b - b_1)^4, \end{aligned} \tag{4.29}$$

で近似することができる．ただし，$g^{(n)}(b)$ は $g(b)$ の n 階導関数であり，$0 < b_1 < 1$ であるとする．

この近似式により，式 (4.12) で表される提案モデルの最尤推定量の近似値は，式 (4.30)，すなわち，

$$\frac{d}{db}F(b) = 0, \tag{4.30}$$

から求めることができる．

いま，式 (4.29) の多項式 $F(b)$ の係数部分を，

$$\alpha = \frac{1}{6}g^{(4)}(b_1), \tag{4.31}$$

$$\beta = \frac{1}{2}\left[g^{(3)}(b_1) - b_1 g^{(4)}(b_1)\right], \tag{4.32}$$

$$\gamma = g^{(2)}(b_1) - b_1 g^{(3)}(b_1) + \frac{1}{2}b_1^2 g^{(4)}(b_1), \tag{4.33}$$

$$\delta = Z_{attack} + g^{(1)}(b_1) - b_1 g^{(2)}(b_1) + \frac{1}{2}b_1^2 g^{(3)}(b_1) - \frac{1}{6}b_1^3 g^{(4)}(b_1), \tag{4.34}$$

とおくと，最尤推定量の近似値は式 (4.35) の 3 次方程式，

$$\frac{dF(b)}{db} = \alpha b^3 + \beta b^2 + \gamma b + \delta = 0, \tag{4.35}$$

の解となる．

そこで 3 次多項式，

$$G(b) = \alpha b^3 + \beta b^2 + \gamma b + \delta, \tag{4.36}$$

を考える．

変換，

$$t = b + \frac{\beta}{3\alpha}, \tag{4.37}$$

によって，$G(b) = 0$ は，

$$t^3 + 3pt + q = 0, \tag{4.38}$$

と書き換えることができる．ただし，

$$p = \frac{1}{3}\left(\frac{\gamma}{\alpha} - \frac{\beta^2}{3\alpha^2}\right), \tag{4.39}$$

$$q = \frac{2\beta^3}{27\alpha^3} - \frac{\beta\gamma}{3\alpha^2} + \frac{\delta}{\alpha}, \tag{4.40}$$

である．

ここで，$\alpha < 0, \delta > 0$ であり，$G(b)$ は連続関数であることから，方程式 $G(b) = 0$ は 0 より大きな実数解を少なくとも 1 つもつことがわかる．したがって，カルダノの定理により，提案モデルの最尤推定量の近似値は，

$$b_{\mathrm{amle}} = A + B - \frac{\beta}{3\alpha}, \tag{4.41}$$

と求めることができる．

ただし，

$$A = \sqrt[3]{\frac{-q + \sqrt{q^2 + 4p^3}}{2}}, \tag{4.42}$$

$$B = \sqrt[3]{\frac{-q - \sqrt{q^2 + 4p^3}}{2}}, \tag{4.43}$$

$$p = \frac{1}{3}\left(\frac{\gamma}{\alpha} - \frac{\beta^2}{3\alpha^2}\right), \tag{4.44}$$

$$q = \frac{2\beta^3}{27\alpha^3} - \frac{\beta\gamma}{3\alpha^2} + \frac{\delta}{\alpha}, \tag{4.45}$$

$$\alpha = \frac{1}{6}g^{(4)}(b_1), \tag{4.46}$$

$$\beta = \frac{1}{2}\left[g^{(3)}(b_1) - b_1 g^{(4)}(b_1)\right], \tag{4.47}$$

$$\gamma = g^{(2)}(b_1) - b_1 g^{(3)}(b_1) + \frac{1}{2}b_1^2 g^{(4)}(b_1), \tag{4.48}$$

$$\delta = Z_{\mathrm{attack}} + g^{(1)}(b_1) - b_1 g^{(2)}(b_1) + \frac{1}{2}b_1^2 g^{(3)}(b_1) - \frac{1}{6}b_1^3 g^{(4)}(b_1), \tag{4.49}$$

である．

4.2.4　数値実験

本研究で扱っている確率モデルのように，最尤推定量を解析的に計算することが困難である場合は，Newton Raphson 法による数値計算によって最尤推定量を求める手法が用いられる．最尤推定量を数値計算の手法によって求める際に問題となるのは，最大値や最小値ではなく，極大値や極小値の解に収束してしまう場合が存在してしまうことと，いつ

解を求めるための繰り返し計算が終わるか見積もりができない場合があることである．本研究の提案モデルについては，尤度関数は単峰であって Newton Raphson 法を用いても収束までに時間がかからないが，最尤推定量を代数的に求めることができるため，システムに実装する際の手間は定理 4.2.1 によってかなり単純化することができる．

以下，簡単に Newton Raphson 法の手続きをまとめておく．Newton Raphson 法ではモデルのパラメータ b_v の値 $(v = 1, 2, \ldots)$ を次の更新式により計算する．

$$b_{v+1} = b_v - \alpha \frac{LL^{(1)}(b)}{LL^{(2)}(b)}. \tag{4.50}$$

ただし，$LL^{(n)}(b), n = 1, 2, \ldots$ は $LL(b)$ の n 階導関数であり，b_v は v 回目に更新されたパラメータの値で，初期値 b_0 は適当に与える必要がある．α は更新ステップを調整するパラメータであり，本研究では対象となる尤度関数が単峰であることも考慮して $\alpha = 1$ とした．この更新式を繰り返し利用していくと，b_{v+1} と b_v の値はほとんど同じ値になり，本研究では，

$$|b_{v+1} - b_v| < 0.001, \tag{4.51}$$

となるところで更新を停止させ，そのときの b_{v+1} の値を提案モデルの最尤推定量とすることにした．あとは，データを固定することで最尤推定量を計算することができる．

ここでは，表 4.2, 4.3 に示す学習データを用いることで，定理 4.2.1 の近似精度を確認するために Newton Raphson 法で最尤推定量を求めた場合と比較するための実験を行う．ここで，表 4.2 は 65 個の攻撃文字列から攻撃特徴含有率を計算したものであり，表 4.3 は 35 個の正常文字列から攻撃特徴含有率を計算したものである．

Newton Raphson 法によって求められた最尤推定量を b_{nr} とすると，表 4.2, 4.3 のデータから，

$$b_{\mathrm{nr}} = 0.101, \tag{4.52}$$

という結果が得られた．一方，定理 4.2.1 において b_1 の値と近似に利用する多項式の項数を変化させることで，表 4.4 の結果が得られた．

本実験では，2 次まで，3 次まで，4 次までの近似を考えている．n 次までの近似によって，対数尤度関数を近似する多項式のグラフの形が変化する．$b_1 = 0.10$ の場合において，2 次まで，3 次まで，4 次までの近似で最尤推定量の近似値に差異が見られないということは，b_1 が対数尤度関数の最大値をとる座標 b の近くであることを意味している．特に，$0.05 \leq b_1 \leq 0.15$ の区間において，Newton Raphson 法と定理 4.2.1 の公式を用いて得られる近似値の差を調べても，

$$|b_{\mathrm{nr}} - b_{\mathrm{amle}}| \leq 0.03, \tag{4.53}$$

となることから，定理 4.2.1 の近似精度は高いことを数値実験から確認することができる．

表 4.2　学習データ (攻撃入力) $(z_i, i = 1, 2, \ldots, 65)$

i	1	2	3	4	5	6	7	8
z_i	0.50	0.50	0.50	0.46	0.44	0.40	0.40	0.38
i	9	10	11	12	13	14	15	16
z_i	0.36	0.36	0.35	0.34	0.33	0.22	0.22	0.28
i	17	18	19	20	21	22	23	24
z_i	0.28	0.27	0.55	0.53	0.25	0.25	0.25	0.25
i	25	26	27	28	29	30	31	32
z_i	0.25	0.24	0.24	0.24	0.23	0.23	0.23	0.23
i	33	34	35	36	37	38	39	40
z_i	0.23	0.22	0.20	0.20	0.20	0.19	0.19	0.19
i	41	42	43	44	45	46	47	48
z_i	0.19	0.19	0.19	0.19	0.19	0.19	0.17	0.16
i	49	50	51	52	53	54	55	56
z_i	0.16	0.16	0.12	0.12	0.12	0.12	0.12	0.12
i	57	58	59	60	61	62	63	64
z_i	0.12	0.06	0.06	0.06	0.06	0.06	0.05	0.04
i	65							
z_i	0.04							

4.2.5　線形分類器と提案モデルに関する考察

前節では，Web アプリケーションへの入力文字列にどの程度攻撃特徴記号が含まれるかということに着目して SQL インジェクション攻撃を検知する手法を提案した．入力文字列 l に対する攻撃特徴記号の含有率 z は $0 \leq z \leq 1$ という値をとるため，l に対して $y = 1 - z$ という量を考えると，(z, y) は 2 次元平面における直線 $y = 1 - z$ 上の点となり，含有率で表されるデータはこの直線上にプロットされることになる．攻撃特徴記号の含有率による攻撃検知では，攻撃か正常であるかの閾値を設定する必要があるが，これは直線 $y = 1 - z$ と原点を通る直線 $y = Az$ の交点に対応する．つまり，含有率による攻撃検知は，正常か攻撃であるかを線形分類器として分類することに対応しており，本研究で提案する攻撃特徴記号の含有率による攻撃検知は線形分類器の一種であるといえる．本章

表 4.3　学習データ (正常入力) $(z_j, j = 1, 2, \ldots, 35)$

j	1	2	3	4	5	6	7	8
z_j	0.25	0.15	0.10	0.05	0.02	0.14	0.13	0.14
j	9	10	11	12	13	14	15	16
z_j	0.07	0.07	0.12	0.08	0.00	0.00	0.00	0.00
j	17	18	19	20	21	22	23	24
z_j	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
j	25	26	27	28	29	30	31	32
z_i	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
j	33	34	35					
z_j	0.00	0.00	0.00					

表 4.4　近似最尤推定量 b_{amle} の計算結果

b_1 の値	0.05	0.10	0.15	0.20
2 次近似	0.075	0.101	0.077	0.002
3 次近似	複素数	0.101	0.096	0.078
4 次近似	0.083	0.101	0.100	0.096

で提案している含有率による攻撃検知では，攻撃か正常を判定するための閾値を設定する必要があるが，この閾値は上述のことに注意すると，別の線形分類器から攻撃検知の閾値を計算できることになる．本研究では，SCW[16] と呼ばれる機械学習のアルゴリズムを用いて攻撃検知のための閾値を計算し，経験的な手法 [23][24] で求められた攻撃検知閾値との比較を行い，提案モデルを用いた考察を行った．

SCW のパラメータ更新に必要な値を $C = 0.5$, $\eta = 0.8$ とし，パラメータの初期値は次のように設定した．

$$\mu_0 = (0, 0), \tag{4.54}$$

$$\Sigma_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{4.55}$$

表 4.2 と表 4.3 のデータを繰り返し利用して，3,000 回の繰り返しの後でパラメータの値

は以下のようになった．

$$\mu_{3000} = (0.643, -0.071), \tag{4.56}$$

$$\Sigma_{3000} = \begin{pmatrix} 0.012 & 0 \\ 0 & 0.001 \end{pmatrix}. \tag{4.57}$$

したがって，分離直線は，

$$0.643p_1 + (-0.071)p_2 = 0, \tag{4.58}$$

となる．本研究では，文字列に含まれる攻撃特徴記号の含有率による攻撃検知手法を提案しており，このことから，$\boldsymbol{p} = (p_1, p_2) = (x, y)$ において，p_1 を攻撃特徴含有率，$p_2 = 1 - p_1$ とすることで SCW を用いた攻撃検知も可能となる．また，$\boldsymbol{p} = (p_1, p_2) = (p_1, 1 - p_1)$ であるから，含有率で表現されるデータは xy 平面では直線 $y = -x + 1$ 上にプロットされることが分かる．含有率による攻撃検知では，閾値を設定する必要があるが，データは直線 $y = -x + 1$ にあり，SCW による分離直線は $0.643x + (-0.071)y = 0$ であることから，攻撃検知に必要な閾値はこの 2 直線の交点からも求められることが分かる．$y = \frac{0.643}{0.071}x$ であるから，$\frac{0.643}{0.071}x = -x + 1$ となり，$\frac{0.643+0.071}{0.071}x = 1$ より，

$$x = \frac{0.071}{0.714} \fallingdotseq 0.0994, \tag{4.59}$$

となるため，攻撃検知に使用する攻撃特徴含有率はおよそ $A = 0.1$ と求めることができる．経験的に閾値を定める手法 [23][24] では，その値が $A = 0.08$ であることから，SCW を用いた場合でも近い値を求めることができているといえる．SCW において，$0.643x + (-0.071)y \geq 0$ を満たす場合は攻撃文字列，$0.643x + (-0.071)y < 0$ を満たす場合は正常文字列として，表 4.2 と表 4.3 のデータを用いて実験を行うと，攻撃文字列と正常文字列を合わせて 87.7% のデータを正しく分類することができた．

また，本章で提案したモデルに最尤推定量の近似値と SCW で求めた閾値 $A = 0.1$ を代入すると，

$$p(t = 1 \mid z = 0.100, b_{\mathrm{amle}} = 0.101) \fallingdotseq 0.793\,, \tag{4.60}$$

という値が得られる．これは，攻撃特徴含有率が 0.1 の場合にその文字列が攻撃である確率は 0.793 程度であることを意味しており，SCW の検知率 87.7% にも近い値であると考えられる．

第 5 章

潜在曲線モデルによる攻撃検知アルゴリズム

本章では，潜在曲線モデル [15] を攻撃検知に応用する手法を提案する．潜在曲線モデルとは，時系列データ分析に利用される推定手法の一つであり，個体差のあるデータ群を解析する際によく利用されている．潜在曲線モデルの基本的な考え方は回帰分析と同じであり，パラメータ推定の方法も最尤推定法を利用することができるが，モデルのパラメータに個体差を表現するための潜在変数が導入されるところが回帰分析と異なる点であると言える．

5.1 潜在曲線モデル

ここでは，線形な場合の潜在曲線モデルについて簡単に紹介する．線形回帰モデルでは，与えられるデータが $(\boldsymbol{x}, y) = (x_1, x_2, \ldots, x_n, y) \in \boldsymbol{R}^n \times \boldsymbol{R}$ である場合は，以下のような線形な方程式を考える．

$$y = f(\boldsymbol{x}) = b + \sum_{j=1}^{n} a_j x_j, \tag{5.1}$$

ここで，$a_1, a_2, \ldots, a_n, b$ は実数に値をとるモデルのパラメータである．線形回帰モデルでは，I 個のデータ $\boldsymbol{D}_I = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{I}$ が与えられたとき，

$$H(\boldsymbol{a}, b) = \sum_{i=1}^{I} \left(y_i - b - \sum_{j=1}^{n} a_j x_j \right)^2, \tag{5.2}$$

を最小にする $\boldsymbol{a} = (a_1, a_2, \ldots, a_n), b$ を計算する．このパラメータ推定の方法は，最小二乗法と呼ばれている．

潜在曲線モデルでは，データの個体差を示すラベルデータが付与されたデータ，

$$(\boldsymbol{x}, y, t) \in \boldsymbol{R}^n \times \boldsymbol{R} \times \boldsymbol{N}, \tag{5.3}$$

を考える．また，考えるモデルは，

$$y = b + \sum_{j=1}^{n} a_j x_j, \tag{5.4}$$

$$b = q_0 + q_1 t, \tag{5.5}$$

$$a_j = p_{0j} + p_{1j} t, \tag{5.6}$$

となる．パラメータ推定の方法は基本的には回帰分析と同じであるが，次のように 2 段階に分けて行う．始めに，I 個のデータ $\boldsymbol{D}_i = \{(\boldsymbol{x}_i, y_i, t_i)\}_{i=1}^{I}$ から，

$$\{(\boldsymbol{a}_i, b_i, t_i)\}_{i=1}^{I}, \tag{5.7}$$

を計算し，次にこのデータの組を用いてパラメータ $q_0, q_1, p_{01}, p_{11}, \ldots, p_{0n}, p_{1n}$ を最小二乗法で求める．本研究では，潜在曲線モデルを用いて SQL インジェクション攻撃の記号分布の特徴を抽出する手法を提案する．

5.2　特徴抽出モデル

SQL インジェクション攻撃のサンプルを収集して記号の出現頻度を考えると，その分布はべき乗則の一種であるゼータ分布に似た形となることが多い [20]．このようなデータの持つ傾向は，表 4.1 にもみられ，その理由として多くの攻撃文字列は文字リテラルや SQL 文を繋げるために記号が必要であり，その結果，SQL インジェクション攻撃には区切り文字が多く含まれるということが考えられる．一方，正常文字列はデータの集め方に依存すると考えられるが，記号を多く含む文字列を収集してもゼータ分布のような記号分布がみられることはほとんどない．本研究では，上述の攻撃文字列と正常文字列の性質を利用し，潜在曲線モデルを用いて攻撃と正常の特徴を抽出する手法を提案する．

5.2.1　特徴抽出アルゴリズム

SQL インジェクションの攻撃文字列に含まれる記号は，当然のことながら正常文字列にも含まれる．そこで，攻撃文字列に含まれる記号の関係性をモデル化することで，攻撃文字列と正常文字列の記号の特徴のオーバーラップを少しでも解消することが期待される．また，攻撃文字列，正常文字列におけるそれぞれの記号間の関係を調べることで，これらのオーバーラップはさらに解消されると考えられる．そのため，ここでは，攻撃データと正常データの特徴を抽出するためのアルゴリズムを提案する．

アルゴリズム 5.2.1（攻撃および正常データの特徴抽出アルゴリズム）

Step1 攻撃データを I_A 個，正常データを I_N 個準備する．

Step2 攻撃，正常それぞれのデータの集合に対し，出現するすべての記号 (攻撃データ集合については J_A 個, 正常データ集合については J_N 個とする) とその頻度を計算し，記号を出現頻度の大きい順に並べ替え，その記号を順に $s_1, s_2, \ldots, s_J$ とする．データ l_i に含まれる記号 s_j の個数を $z_i(s_j)$ とおくと，攻撃データ集合における記号の頻度分布は, $j = 1, 2, \ldots, J_A$ に対して，

$$T_A(s_j) = \sum_{i=1}^{I_A} \frac{z_i(s_j)}{|l_i| \cdot I_A}, \tag{5.8}$$

正常データ集合における記号の頻度分布は, $j = 1, 2, \ldots, J_N$ に対して，

$$T_N(s_j) = \sum_{i=1}^{I_N} \frac{z_i(s_j)}{|l_i| \cdot I_N}, \tag{5.9}$$

と表すことができる．ここで，l_i は文字列のデータ，$|l_i|$ は文字列 l_i に含まれる記号の総数を表すものとする．4 章までは $|l_i|$ を l_i の文字列長としていたが，より記号間の関係性をモデルに抽出させやすくするために，ここでは記号を含むデータのみを対象とすることにした．本研究では，攻撃データ集合における記号の出現頻度分布のうち，出現頻度が大きい記号から順に x 個の記号を利用して攻撃の特徴を抽出する．

Step 3 攻撃データと正常データを 3 次元ユークリッド空間 $\boldsymbol{R}^3$ 上の点 (x, y, t) として表現する．$\boldsymbol{R}^3$ の第 1 座標 X は，以下のように攻撃データ集合に出現する記号に対応させるように定義する．具体的には，攻撃データ集合の記号出現頻度分布から上位 x 個のデータに着目し，それらを $s_1, s_2, \ldots, s_x$ とする．この x 個の記号の列を第 1 座標 X として，

$$1 = s_1, 2 = s_2, \ldots, x = s_x, \tag{5.10}$$

と表すことにする．これにより，記号 s_j $(j = 1, 2, \ldots, x)$ を, $\boldsymbol{R}^3$ の X 軸の $(j, 0, 0)$ という座標に対応させることができる．次に，第 2 座標の値を, 第 1 座標 j に対応する記号 s_j から $y_j(l_i) = \frac{z_i(s_j)}{|l_i|}$ を計算した値として定義する．ここまでの定義により，もし文字列 l に記号 s_{j_1}, s_{j_2} が含まれる場合，座標の組 $(j_1, y_{j_1}(l))$ と $(j_2, y_{j_2}(l))$ が得られる．最後に，第 3 座標 t の値は，データが攻撃である場合は $t = 1$, 正常である場合は $t = 0$ とする．

Step 4 Step 3 で作成したデータを用いて，以下の潜在曲線モデルのパラメータを最尤法

で推定する．

$$y = a_0 + a_1 x + a_2 x^2 + \epsilon_1, \tag{5.11}$$
$$a_0 = b_{00} + b_{10} t + \epsilon_2, \tag{5.12}$$
$$a_1 = b_{01} + b_{11} t + \epsilon_3, \tag{5.13}$$
$$a_2 = b_{02} + b_{12} t + \epsilon_4, \tag{5.14}$$

与えられたデータから，パラメータ $b_{00}, b_{10}, b_{01}, b_{11}, b_{02}, b_{12}$ を推定することが目標である．ただし，ϵ_i（$i = 1, 2, 3, 4$）は平均 0, 分散 $\sigma_i^2 > 0$ の正規分布に従う誤差項である．

□

図 5.1 より，SQL インジェクション攻撃には出現頻度が極端に高い記号がいくつか存在することが分かり，これらの記号は攻撃検知に有用であると考えられる．出現頻度の高い部分を表現するモデルとして最も単純なものは上記の $y = a_0 + a_1 x + a_2 x^2$ である．しかしながらこのモデルは，$a_2 > 0$ の場合，ある x を境目に微分係数 y' がある一定の正の数より大きくなるため，攻撃検知には微分係数が 0.152 より小さくなるする区間に含まれる記号のみを利用する．なお，0.152 という値は，図 5.3 において $x = 5$ のときの微分係数 y' の値がおよそ 0.151 となるために設定した数値である．このようにすることでモデル $y = a_0 + a_1 x + a_2 x^2$ は x の値が大きくなると y の値は小さくなり，x の値が小さい（攻撃に頻出する記号に対応する）ときに y の値が大きくなるため，y の値を攻撃特徴の重みとして使用することができる．

5.2.2　SQL インジェクションの攻撃特徴抽出

ここでは，5.2.1 節で提案した攻撃特徴抽出アルゴリズム 5.2.1 を用いて，実際に SQL インジェクション攻撃の特徴抽出を行う．

Step 1　本研究で使用するサンプルは以下の通りである．攻撃データのサンプルを OWASP のウェブサイト [28] から収集し，そこから攻撃データ再構成する攻撃データ生成機を作成して 2779 個 ($J_A = 2779$) の攻撃データを準備した．正常データのサンプルはブラウザに入力される個人情報に関連する仮想的なデータを用意し，顔文字や Wiki の文法など，特殊な記号を多く含む文字列を 444 個 ($J_N = 444$) の正常データを準備した．

Step 2　2,779 個の攻撃データ集合に含まれている記号のうち，出現頻度の高い 5 つの記号 ($x = 5$ とした) を選ぶと以下のようになる．

表 5.1　横軸 (x 軸) に対応する 5 つの記号

記号	Space	’	;	)	(
x 座標	1	2	3	4	5

本研究では，表 5.1 にある 5 つの文字を用いて SQL インジェクション攻撃の特徴抽出を行う．参考までに，攻撃と正常それぞれのデータ集合に含まれる記号の分布（出現頻度の総和で正規化したもの）は以下の図 5.1, 5.2 のようになる．ここで，図 5.1, 5.2 の曲線は，[20] の手法に基づいて計算したゼータ分布を表しており，それぞれの図の棒グラフのデータからゼータ分布のパラメータを推定して計算したものである．

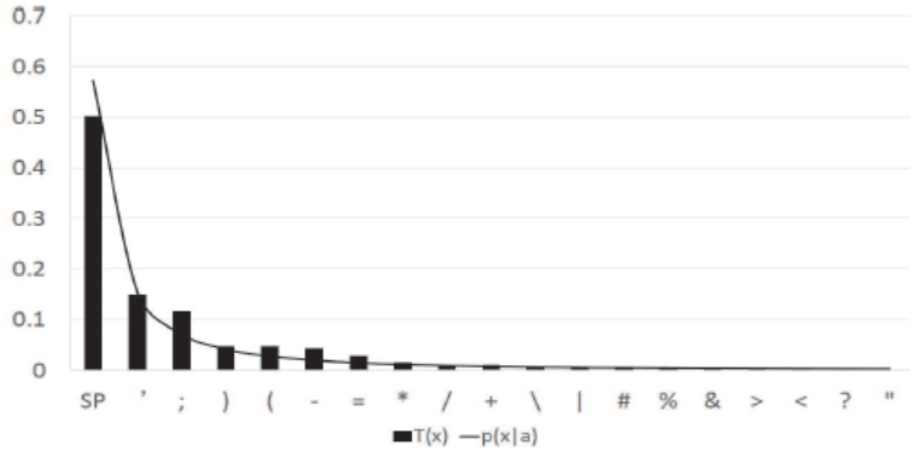

図 5.1　攻撃データの記号分布

Step 3, Step 4　本研究では 2,799 個の攻撃データと 444 個の正常データのサンプルからそれぞれのデータ集合を作成し特徴抽出を行った．図 5.3, 5.4 はこれらのデータ集合から計算したモデル $y = (b_{00} + b_{10}t) + (b_{01} + b_{11t})x + (b_{02} + b_{12}t)x^2$ の推定結果である．図 5.3, 5.4 の点は [Step 3] で構成したデータを表し，曲線は [Step 4] での推定結果のモデル（2 次関数）を表している．なお，図 5.3, 5.4 の横軸（x 軸）は，表 5.1 に示した記号に対応している．

表 5.2 は潜在曲線モデルから得られる記号の重みの算出結果である．

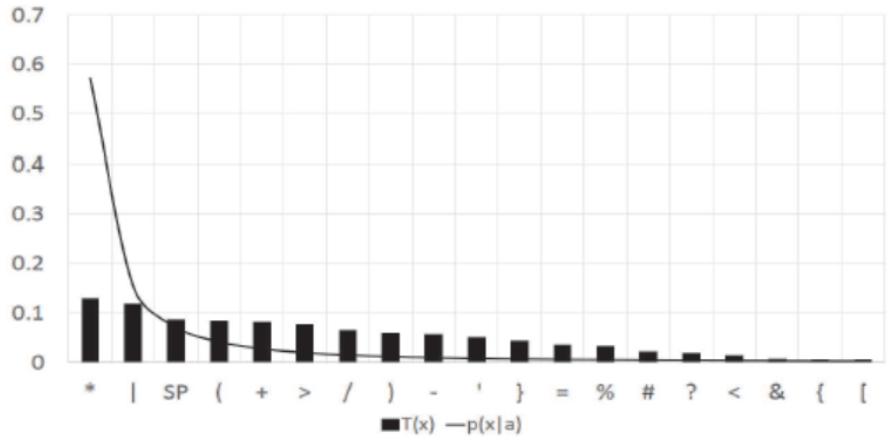

図 5.2　正常データの記号分布

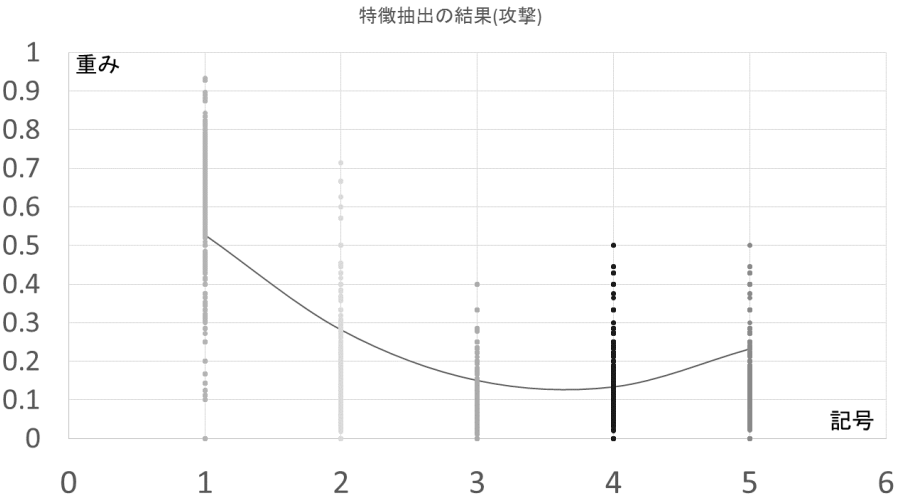

図 5.3　学習用データ（攻撃 $t = 1$）の分布と推定結果

5.2.3　SQL インジェクション攻撃の検知アルゴリズム

5.2.2 節の攻撃特徴抽出結果をふまえた SQL インジェクション攻撃の検知アルゴリズムは，次のアルゴリズム 5.2.2 のようになる．

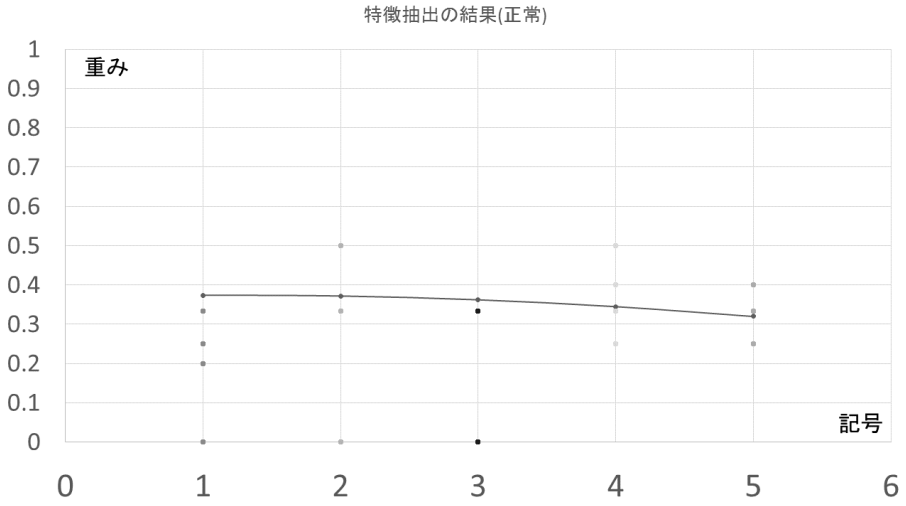

図 5.4 学習用データ（正常 $t = 0$）の分布と推定結果

表 5.2 出現記号と重み

記号	Space	’	;	)	(
重み (攻撃)	0.5274	0.2815	0.1503	0.1339	0.2323
重み (正常)	0.3732	0.3712	0.3616	0.3444	0.3196

アルゴリズム 5.2.2

Step A 入力文字列に含まれる文字 $s_{j_k}(k = 1, 2, \ldots, K)$ を求め，座標の組，

$$T_N = \{(j_k, y_{j_k}, 0)\}_{k=1}^{K}, \tag{5.15}$$
$$T_A = \{(j_k, y_{j_k}, 1)\}_{k=1}^{K}, \tag{5.16}$$

を作る．なお，記号 s_j が入力文字列に含まれない場合，そのときの y_j の値は欠損値として扱う．ただし本提案手法では，5 つの特徴記号のうち 3 つ以上の記号が出現しないデータについてはパラメータが一意に定まらないため，この条件を満足するデータを学習データとして使用する．なお，検証用のテストデータには上記のようなデータの制限を与えないため，仮想的に学習段階では出現しなかった未知のデータが含まれていることになっている．

Step B 前の Step で作成した座標の組のうち，どちらの方が特徴抽出アルゴリズムで求めた潜在曲線モデルの超平面に当てはまりが良いか調べる．当てはまりの良さを調

べる指標として残差を利用する．具体的には，座標の組 $T_A = \{(j_k, y_{j_k}, 1)\}_{k=1}^{K}$ の残差 $e(T_A)$ が $T_N = \{(j_k, y_{j_k}, 0)\}_{k=1}^{K}$ の残差 $e(T_N)$ より小さいときは入力された文字列を攻撃と，それ以外のときは正常と検知する．

□

5.2.4　攻撃データ

提案手法の有用性を検証するための攻撃データを OWASP のウェブサイト [28] から 200 個収集した．なお，これらのデータは特徴抽出の際には使用していないことに注意されたい．以下は検証のために使用した攻撃データの一部である．

- l_1
  - `id=3 union all select 1,2,3,4,5 from admin/*`
- l_2
  - `SELECT 'A' || 'B' FROM dual;`
  - `1) or ('ab' = 'a''b`
  - `SELECT UTL_INADDR.get_host_address FROM dual;`
- l_3
  - `%'SELECT CHR(65)||CHR(66);`

これらのデータから得られる座標の組はそれぞれ以下のように表現される．ただし，$y = 0$ であるときは欠損データとして扱う．

- l_1
  - $T_{A_1} = \{(1, 1, 1)\}$
  - $T_{N_1} = \{(1, 1, 0)\}$
- l_2
  - $T_{A_2} = \{(1, 0.48, 1), (2, 0.36, 1), (3, 0.08, 1), (4, 0.04, 1), (5, 0.04, 1)\}$
  - $T_{N_2} = \{(1, 0.48, 0), (2, 0.36, 0), (3, 0.08, 0), (4, 0.04, 0), (5, 0.04, 0)\}$
- l_3 :
  - $T_{A_3} = \{(1, 0.143, 1), (2, 0.143, 1), (3, 0.143, 1), (4, 0.286, 1), (5, 0.286, 1)\}$
  - $T_{N_3} = \{(1, 0.143, 0), (2, 0.143, 0), (3, 0.143, 0), (4, 0.286, 0), (5, 0.286, 0)\}$

これらの座標の組と，特徴抽出アルゴリズムによって得られた超平面との残差はそれぞれ以下の通りである．

- l_1
  - $e(T_{A_1}) = 0.2233$
  - $e(T_{N_1}) = 0.3929$
- l_2
  - $e(T_{A_2}) = 0.0592$
  - $e(T_{N_2}) = 0.2616$
- l_3 :
  - $e(T_{A_3}) = 0.1931$
  - $e(T_{N_3}) = 0.1576$

残差の計算結果から，$i = 1, 2, 3$ のとき $e(T_{A_i}) < e(T_{N_i})$ であるから，l_1, l_2, l_3 は攻撃として検知に成功したことがわかる．

5.2.5 正常データ

SQL インジェクション攻撃には多くの記号が含まれる．したがって，記号を多く含む入力文字列は，SQL インジェクション攻撃と似たような特徴をもっている可能性がある．そこで，提案モデルの有用性を検証するために，本研究では記号が多用されている正常なパラメータを含むデータを 50 個収集・作成し，正常データとして使用した．以下にこのような正常データのサンプルの一部を示す．

- l_4
  - `https://www.google.co.jp/search?q=%22%3Balert(docu ment.domain)%2F%2F&oq=%22%3Balert(document.d omain)%2F%2F&aqs=chrom e..69i57&sourceid=chrome&es_sm=122&ie=UTF-8`
- l_5
  - `http://msdn.microsoft.com/ja-jp/library/kk6xf663(v=vs.80).aspx`

これらのデータから得られる座標の組はそれぞれ以下のように表現される．

- l_4
  - $T_{A_4} = \{(4, 0.5, 1), (5, 0.5, 1)\}$
  - $T_{N_4} = \{(4, 0.5, 0), (5, 0.5, 0)\}$
- l_5
  - $T_{A_5} = \{(4, 0.5, 1), (5, 0.5, 1)\}$
  - $T_{N_5} = \{(4, 0.5, 0), (5, 0.5, 0)\}$

この例では，l_4, l_5 ともに同じ座標の組となっている．このように，異なるデータでも特徴を表現する座標は同じものになる場合が存在する．これらの座標の組と，特徴抽出アルゴリズムによって得られた超平面との残差はそれぞれ以下の通りである．

- l_4
  - $e(T_{A_4}) = 0.2057$
  - $e(T_{N_4}) = 0.0568$
- l_5
  - $e(T_{A_5}) = 0.2057$
  - $e(T_{N_5}) = 0.0568$

残差の計算結果から，$i = 4, 5$ のとき $e(T_{A_i}) \geq e(T_{N_i})$ であるから，l_4, l_5 は正常として検知に成功したことがわかる．

5.2.6　検知実験結果

表 5.3 に提案手法を用いて検知実験を行った結果をまとめた．特徴抽出アルゴリズムから得られたモデルは，

$$y = (b_{00} + b_{10}t) + (b_{01} + b_{11}t)x + (b_{02} + b_{12}t)x^2, \tag{5.17}$$
$$b_{00} = 0.368, \tag{5.18}$$
$$b_{10} = 0.521, \tag{5.19}$$
$$b_{01} = 0.009, \tag{5.20}$$
$$b_{11} = -0.428, \tag{5.21}$$
$$b_{02} = -0.004, \tag{5.22}$$
$$b_{12} = 0.061, \tag{5.23}$$

である．このモデルは，$t = 1$ のときは攻撃の特徴を，$t = 0$ のときは正常の特徴を表している．

表 5.3　検知実験結果 (提案手法)

真のラベル	攻撃	正常
学習用データ	2779 個	444 個
テスト用データ	200 個	50 個
検知成功データ	188 個	50 個
検知成功率	94.0%	100.0%

第 6 章

従来研究との比較と考察

5 章では，潜在曲線モデルを応用することで SQL インジェクション攻撃の特徴抽出と攻撃検知を行うアルゴリズムを提案した．SQL インジェクション攻撃を検知する手法としては，WAF を Web サーバに導入して攻撃をブロックするものが一般的であるが，機械学習を用いて攻撃を検知する手法も研究・開発されている．本章では，オープンソースソフトウェア (OSS) の Web サーバーである Apache[17] に組み込んで利用できる WAF として有名な ModSecurity[22] と，汎化性能が高いことで有名な Support Vector Machine (SVM)[9] および SoftCondence-Weighted (SCW) アルゴリズム [16] を用いた攻撃検知手法と，本研究で提案した潜在曲線モデルによる攻撃検知手法とについて検知実験を行った性能を比較し，それぞれの特性について考察を行う．

6.1 従来研究との比較

6.1.1 ModSecurity との比較

提案手法の有効性を確認するため，Apache のモジュールである ModSecurity に対して 5.2.6 節の検知実験と同様のテスト用データを適用し，検知実験を行った．ModSecurity は OWASP ModSecurity Core Rule Set [29] に基づいて攻撃を検知し，攻撃から Web アプリケーションを守るために広く利用されているだけでなく，攻撃検知のためのルール更新が頻繁に行われているという特徴をもつ．表 6.1 は ModSecurity による検知実験の結果である．

なお，なお，ModSecurity はデフォルトの状態で使用し，以下の環境のもとで検知実験を行った．

- Web サーバ OS: CentOS 6.6
- Web サーバソフトウェア: Apache/2.2.15

表 6.1　検知実験結果 (ModSecurity)

真のラベル	攻撃	正常
テスト用データ	200 個	50 個
検知成功データ	200 個	19 個
検知成功率	100.0%	38.0%

- Web サーバ側スクリプト言語: PHP 5.5.18
- Web サーバ側データベース: mysql Ver 14.14 Distrib 5.5.40, for Linux (x86_64)
- WAF ソフトウェア: mod_security-2.7.3-3.el6.x86_64
- 自動検知ルールセット: Core ModSecurity Rule Set ver.2.2.6
- クライアント側ブラウザ: FirefoxESR 31.2.0

6.1.2　機械学習による攻撃検知手法との比較

筆者らの研究 [30] では，機械学習のアルゴリズムである SVM を用いた攻撃検知について検討している．本研究では提案手法と機械学習を用いて攻撃検知する手法を比較するため，表 5.1 にある 5 つの記号を特徴記号とし，SVM を用いた攻撃検知について検知実験を行った．結果は表 6.2 の通りである．

表 6.2　検知実験結果 (SVM)

真のラベル	攻撃	正常
学習用データ	2,779 個	444 個
テスト用データ	200 個	50 個
検知成功データ	178 個	48 個
検知成功率	89.0%	96.0%

なお，計算には OSS のプログラミング言語である Python[31] 2.7.8 の機械学習ライブラリ scikit-learn[32][33] 0.15.2 の関数 SVC を利用した．チューニングの設定は，

```
C=100.000000, kernel=rbf, sigma=10.000000, gamma=0.005000
```

とした．

最後に，SCW を用いた攻撃検知実験の結果を表 6.3 に示す．計算は同じく Python に

3 章で述べたアルゴリズムを実装して実行した．

表 6.3 検知実験結果 (SCW)

真のラベル	攻撃	正常
学習用データ	2,779 個	444 個
テスト用データ	200 個	50 個
検知成功データ	199 個	25 個
検知成功率	99.5%	50.0%

6.2 考察

本節では，6.1.2 節の検知実験の結果について考察する．この検知実験では，提案手法を用いて開発した Apache モジュールと，Apache のモジュールである ModSecurity を用いて攻撃データと正常データの検知を行った．以下，攻撃検知と正常検知それぞれの実験結果について考察を行う．

6.2.1 攻撃検知について

表 6.1 より，ModSecurity は用意した真のラベルが攻撃であるテスト用データ 200 個すべてについて正しく検知し，攻撃検知率は 100.0% であった．一方，表 6.2 より SVM の攻撃検知成功率は 89.0% とやや下がり，表 6.3 より SCW の攻撃検知成功率は 99.5% と ModSecurity に次いで高い成功率となった．一方，表 5.3 より提案手法の攻撃検知成功率 94.0% となり，200 個のテスト用データのうち 188 個を正しく攻撃と検知し，残りの 12 個のデータを誤検知した．提案手法が誤検知したデータを 1 つ紹介する．

```
|SELECT current_setting('data_directory');
```

表 5.2 にある記号に付与される重みの情報を見ると，この文字列に含まれる記号の重みは攻撃・正常ともに大きくなると考えられる．したがって，提案手法はこのデータを正常データとして誤検知したと考えられる．この例は攻撃データであるから正しくは攻撃データと検知すべきであるが，ModSecurity では記号を多く含む文字列は攻撃として検知する可能性が非常に高いと考えられる．例えば，URL のパラメータに記号が多用されるデータを攻撃と検知する可能性が高い．一方，提案手法は ModSecurity や機械学習を用いた検知法と比較すると，入力文字列における記号の扱い方が柔軟であると考えられる．その

ため，攻撃を正常と誤検知する危険性がある一方，記号が多用されているデータをいつでも攻撃と検知しないことから正常データを攻撃と誤検知する危険性は緩和されるものと考えられる．

6.2.2　正常検知について

表 5.3 より，提案手法は真のラベルが正常であるテストデータ 50 個すべての正しく検知した．一方，表 6.2 より SVM の正常検知の成功率は 96.0%，であったが，表 6.3 より SCW のそれは 50.0% となった．SVM は SCM は機械学習分野で代表的なアルゴリズムであるが，提案手法と比較すると正常検知の成功率の観点で必ずしも優位性があるとはいえない結果となった．提案手法では，記号の特徴を攻撃と正常の両者の視点からとらえており，攻撃の特徴となる記号が含まれていないものは正常に振り分けられやすいようにモデル化されていることが，提案手法が正常検知に有用である理由になっていると考えられる．

一方，表 6.1 より ModSecurity の正常検知の成功率は 38.0% で，50 個のテスト用データのうち 19 個の正常データを正しく検知し，残りの 31 個のデータを誤検知した．ModSecurity が誤検知したデータを 1 つ紹介する．

```
https://www.google.co.jp/?gws_rd=ssl#q=
(%EF%BE%9F%E2%88%87%5E*)%EF%BD%B5
%EF%BE%8A%EF%BE%96%E2%99%AA
```

この例が示す通り，SQL インジェクション攻撃にはなり得ない，記号を多く含むデータに対しても ModSecurity(デフォルト設定) は攻撃として検知することがわかる．それに対して，提案手法では記号それぞれに対して攻撃の重みと正常の重みを付与し，それらの情報を抽象化（数学モデル化）して攻撃を検知するため，攻撃検知のパフォーマンスは ModSecurity より劣る可能性は高い一方，正常データを攻撃と誤検知する可能性は低くできるものと考えられる．

しかしながら提案モデルを用いて検知を行う場合はいくつか注意すべき点がある．記号の重みは攻撃データ集合と正常データ集合それぞれの集合における記号の出現頻度から計算されるため，サンプルとなるデータの集め方に依存してしまう．したがって，特徴抽出にはデータの収集方法が重要となるが，一方で，記号を多く含む正常データを許容する場合，そのようなデータを収集・作成することで記号を多く含む入力文字列に対して柔軟な攻撃検知を実現できるものと考えられる．

6.2.3 提案モデルについて

本研究では，潜在曲線モデルの考え方を応用することで SQL インジェクション攻撃の特徴を記号の重みという形で抽出した．提案モデルのパラメータは，筆者らの研究 [30] の公式を用いることで計算され，攻撃検知も単純な四則演算のみで行うことができる．そのため，提案モデルを用いて攻撃検知を行う場合でも，ModSecurity を用いて攻撃検知を行う場合でも，攻撃検知に要する処理時間の差はほとんど感じることはない．さらに，データに欠損がある場合に潜在曲線モデルはデータに対して当てはまりが良くなる例がある [34] ことが指摘されている．提案手法ではデータを 2 次関数に当てはめる方法を採用したため，用意したテストデータに対する検知性能を高めるために 5 つの記号にしか重みを付与することができなかった．モデルを改良することで入力可能なすべての記号に対して重みを付与することができると考えられるため，この点は今後の課題である．

第 7 章

議論

本研究では，SQL インジェクション攻撃の文字列に含まれる記号に着目し，それらを特徴とした攻撃検知手法の提案を攻撃検知システムの開発を行った．SQL インジェクション攻撃の対策は，2 章でも紹介しているが，保険的な対策である WAF の重要性について議論されることもある．本章では，攻撃検知システムの必要性と数理的手法を用いる意義について議論する．

7.1　攻撃検知の必要性

本研究では，SQL インジェクション攻撃を検知するための数理的手法を提案し，そのアルゴリズムを Apache のモジュールに実装することで Web アプリケーションファイアウォール (WAF) を開発した．WAF は Web サーバー側でユーザーからの入力をチェックして，攻撃と判断される文字列をブロックする．図 7.1 はこれをクライアント側のブラウザからみた様子である．

図 7.1　WAF が通信をブロックする様子

SQL インジェクション攻撃は，入力に攻撃の目的を達成するための SQL 文が挿入されるため，ユーザーの入力から出来るだけ SQL 文を組み立てないようにすることが重要である．そのため，

- プリペアドステートメントを適切に利用すること
- シングルクォートやセミコロンを SQL として特殊な意味を持たない記号に置換す

ること

を徹底することが重要である．しかしながら，予期せぬことを全て想定して攻撃を防ぐことは一般的には困難なことである．このような想定し得ない事態が怒った場合，WAF を用いれば攻撃をブロック出来る可能性を否定することはできない．そのためにも，入力された文字列が攻撃であるかどうかを判断する手法を開発することは重要な問題であると言える．

7.2　数理的手法を応用した WAF の意義

本研究では，SQL インジェクション攻撃の特徴を，攻撃文字列に含まれる記号に着目することで抽出し，それに基づいた WAF を開発した．2016 年からマイナンバー制度が導入されたことにより，組織が抱える個人情報の保護のための対策がより重要な問題となっている．このような社会背景に伴ってクラウド上で提供される WAF が増えており，WAF の導入のハードルはかなり低くなったと言える．フリーで使用できる WAF として有名な ModSecurity[22] は，SQL インジェクション攻撃以外の攻撃も検知することができる高性能な WAF として知られている．検知ルールも頻繁に更新され，攻撃検知には有用であるが，適切に運用するための設定は簡単ではない．クラウドを利用した WAF[21] は，このような面倒な WAF の設定を解消するために開発・販売されているが，その導入には金銭面や設置の準備など様々なコストが発生する．

数理的手法を応用した WAF では，攻撃の特徴を文字列レベルで抽出するため，適切に特徴抽出を行うことができれば，WAF の細かい設定をする必要はない．さらに，抽出した特徴が，攻撃に必要な要素を包含しているなら，想定外の入力にも対応できる可能性もある．特徴抽出が適切であるということが大前提であるが，以上のことが数理的手法を応用した WAF の大きなメリットである．

以上のことを踏まえて，5, 6 章の実験結果について考察を行う．本研究の手法による攻撃検知の実験では，攻撃文字列を正常文字列と誤検知してしまうサンプルがいくつか存在したが，これは，提案手法では記号に基づいて攻撃検知を行うため，必然的に攻撃特徴として使用する記号を含む正常な文字列が存在するためである．一方で，ModSecurity の Core ルールをデフォルトのまま使用すると，記号を含む文字列は正常文字列であっても攻撃文字列であると誤検知する可能性が高いということが分かる．ModSecurity では，検知ルールを調整することで正常を攻撃と誤検知する割合を減少させることも可能であると考えられるが，この調整にはかなりの専門的知識が必要でコストがかかることと，攻撃を正しく検知する割合も減少する可能性が生じる．以上より提案法のような数理的手法を応用した WAF には，一定の有効性があるものと考えられる．

7.3 機械学習による攻撃検知

想定外の攻撃に対して有用であると考えられる WAF であるが，機械学習を応用した攻撃検知に関する研究も進められている．多くの機械学習のアルゴリズムもライブラリとして無償で公開されているため，R[35] や Python[31] などを利用することで簡単に機械学習のアルゴリズムを試すことができる．6 章では，汎化能力の高いことで知られている Support Vector Machine (SVM)[9]，SCW[16] と提案手法，ModSecurity との攻撃検知精度の比較を行った．機械学習における汎化能力とは，未知のデータに対応する能力のことであり，このような能力は攻撃検知においても期待される．しかしながら，6 章の実験結果が示す通り，本研究における実験では，全ての文字列を攻撃文字列として検知してしまうという結果が得られた．これは，提案手法と SVM の学習のためのデータには記号を含んだものを利用しているが，テストデータに記号をほとんど用いない攻撃文字列を含めたり，学習データとはパターンの異なる正常文字列を含めたためであると考えられる．SVM では，ガウシアンカーネルを利用しパラメータを色々と調整しても得られる結果に変化は見られなかった．この結果は，機械学習では理論と現実世界において大きな乖離があることを示しているが，上述のような特徴を持ったデータを与えての考察を行っていることに注意したい．本研究では，提案手法と同じ特徴空間上で SVM を学習させているが，異なる特徴空間上では SVM の精度向上を期待することもできる．SQL インジェクション攻撃の検知においては，適切な特徴抽出を行うだけでなく，WAF の実装や誤検知の分析を容易に行えること，特徴の選択を容易に行えることも非常に重要な問題であると言えるため，機械学習を攻撃検知に使用する際は初期段階にかなり大きなコストが必要になると考えられる．

第 8 章

結論

これまでに述べてきた通り，SQL インジェクション攻撃は Web アプリケーションのデータベースに不正にアクセスするために使われる非常に深刻なサイバー攻撃である．サイバー攻撃にはシステムに障害を与えるものや，機密性の高い情報を盗むものもあれば，なりすましやデータの改竄など様々な目的がある．

本研究では、多くの攻撃手法の中から SQL インジェクション攻撃に焦点をしぼって攻撃検知手法を開発したが，その理由の大部分は以下の 2 つ文章としてまとめることができる．

1 つは SQL インジェクションの脆弱性を狙った攻撃が頻発しているため．もう 1 つは SQL インジェクション攻撃は攻撃の構造が比較的簡単であり，他のサイバー攻撃の対策の足掛かりになると考えられるためである．

本研究の結果から，SQL インジェクション攻撃の構造は比較的単純であるとはいえ，攻撃文字列であるか正常文字列であるかを完全に分類することは容易でない場合も存在することが分かった．SQL インジェクション攻撃の基本的な対策はプリペアドステートメントを適切に利用し，SQL として意味をもつ記号のエスケープ処理を組み合わせることである．しかしながら，アプリケーションへの入力は様々なパターンを想定すべきであり，SQL としても意味をもちながら，他の意図として意味をなす記号も存在するため，開発時にエスケープ処理に頼らざるを得ない場面でもアプリケーションとして正常なサービスを提供することを考えると，攻撃文字列か正常文字列かを精度良く判別できる手法を確立することは重要な問題であるといえる．

攻撃文字列と正常文字列の判別を正確に行うためには，それぞれの特徴のオーバーラップ部分をなくすような手法を見出さなければならず，その場合は，本研究で着目した記号以外の様々な情報が必要となる．攻撃検知のために必要な情報を増やしすぎると，現状の WAF の課題のような肥大化の問題が発生するだけでなく，過学習による誤検知を引き起こすことも考えられる．過学習を引き起こさないように攻撃文字列と正常文字列のオー

バッラップを解消する手法を研究することは，サイバー攻撃を適切に防御するための重要な課題であり，これらは本研究の今後の課題でもある．

謝辞

はじめに，中央大学理工学部情報工学科情報通信工学研究室の趙 晋輝 教授からは，多大なるご指導とご支援を賜り，社会人学生として研究活動と社会活動を両立し，限られた時間の中で学位論文をまとめられましたことを深謝いたします．

また，本論文作成にあたり，審査委員として多くのご助言を頂きました中央大学理工学部電気電子情報通信工学科の白井 宏 教授，同学部情報工学科の牧野 光則 教授，情報セキュリティ大学院大学情報セキュリティ研究科の佐藤 直 教授にも深く感謝いたします．

さらに，博士課程への進学および研究全般にわたる多大なご支援，ご指導を賜りました中央大学研究開発機構 辻井 重男 フェロー，早稲田大学の平澤 茂一 名誉教授，長崎県立大学情報システム学部情報セキュリティ学科の松田 健 准教授，小樽商科大学商学部社会情報学科の小泉 大城 准教授，合同会社 binary lab の大谷 康介氏にも大変お世話になりました．

本研究を遂行する上で，多くの先生方に時には厳しい指導を，時には優しい励ましを頂いたことで，研究はもとより日常の業務に関連することまで幅広い知見を得ることができたこともまた私自身の大きな成果となりました．

また，中央大学大学院理工学研究科情報工学専攻 趙研究室 修士課程の梅原 章宏君，南後 吉秀君，および同研究室 学部生の牛込 龍太郎君，秋本 悠一郎君には共に研究するだけでなく，研究の打ち合わせの様々な調整に至るまでたいへんお世話になりました．心より感謝いたします．

最後に，いつも暖かく見守ってくださり心の支えとなっている家族に深い感謝の意を表して謝辞といたします．

2016 年 12 月

園田　道夫

参考文献

[1] 文部科学省, 「AIP プロジェクト」に係る平成 28 年度戦略目標の決定について, http://www.mext.go.jp/b_menu/houdou/28/05/1371147.htm, 2016 年 5 月.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2010.

[3] The Open Web Application Security Project (OWASP) Foundation, OWASP Top Ten Project, https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project .

[4] Phrack Magazine, http://www.phrack.org/ .

[5] rain.forest.puppy (rfp), "NT Web Technology Vulnerabilities," http://phrack.org/issues/54/8.html, Phrack Magazine, vol. 8, Issue 54, Dec. 1998.

[6] The Open Web Application Security Project (OWASP) Foundation, The Open Web Application Security Project (OWASP), https://www.owasp.org/ .

[7] 小菅 祐史, 花岡 美幸, 河野 健二, 「SQL インジェクション攻撃の脆弱性の効果的な自動検出手法」, 情報処理学会 研究報告コンピュータセキュリティ (CSEC), vol. 2008-CSEC-41, no. 45, pp. 103–108, 2008 年 5 月.

[8] 独立行政法人 情報処理推進機構 セキュリティセンター, 「安全な SQL の呼び出し方」, https://www.ipa.go.jp/security/vuln/press/201003_websecurity_sql.html, 2010 年 3 月.

[9] C. Cortes and V. Vapnik, "Support-Vector Networks," Machine Learning, Vol. 20, Issue 3, pp. 273–297, Sep. 1995.

[10] J. Pearl, *Probabilistic Reasoning in Intelligent Systems,* Morgan Kaufmann, 1988.

[11] H. Takahashi, H. F. Ahmad, and K. Mori, "Application for Autonomous Decentralized Multi Layer Cache System to Web Application Firewall," Proc. of 2011 10th International Symposium on Autonomous Decentralized Systems (ISADS), pp.113–120, Mar. 2011.

[12] 伊波 靖, 高良 富夫, 「サポートベクタマシンを用いた WAF への異常検知機能の実装と評価」, 情報処理学会論文誌, コンピューティングシステム, vol. 7, no. 1, pp.

1–13, 2014 年 3 月.

[13] Y. Wang, Z. Li, “SQL Injection Detection with Composite Kernel in Support Vector Machine,” International Journal of Security & Its Applications, vol. 6 Issue 2, pp. 191–196, Apr. 2012.

[14] R. Komiya, I. Paik, and M. Hisada, “Classification of Malicious Web Code by Machine Learning,” Proc. of 2011 3rd International Conference on Awareness Science and Technology (iCAST), pp. 406–411, Sep. 2011.

[15] 豊田 秀樹, 「共分散構造分析［入門編］ —構造方程式モデリング—」, 朝倉書店, 1998 年.

[16] Jialei Wang, Peilin Zhao and Steven C. Hoi, “Exact Soft Confidence-Weighted Learning,” Proc. of the 29th International Conference on Machine Learning (ICML-12), pp. 121–128, 2012.

[17] The Apache Software Foundation, The Apache HTTP Server Project, https://httpd.apache.org/ .

[18] A. Sadeghian, M. Zamani, and S. Ibrahim, “SQL Injection Is Still Alive: A Study on SQL Injection Signature Evasion Techniques,” Proc. of 2013 International Conference on Informatics and Creative Multimedia (ICICM), pp. 265–268, Sep. 2013.

[19] T. Matsuda, D. Koizumi, M. Sonoda, and S. Hirasawa, “On Predictive Errors of SQL Injection Attack Detection by the Feature of the Single Character,” Proc. of 2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC2011), pp. 1722–1727, Nov. 2011.

[20] T. Matsuda, “Feature Extraction of Web Application Attacks based on Zeta Distributions, ” Proc. of 2013 World Congress on Internet Security (WorldCIS), pp. 119–121, Sep. 2013.

[21] Securesky Technology Inc. and Bitforest Co. , Ltd. , Scutum, http://www.scutum.jp/ .

[22] Trustwave Inc. , ModSecurity, https://www.modsecurity.org/ .

[23] 園田 道夫, 松田 健, 小泉 大城, 平澤 茂一, 辻井 重男, 「文字単位の特徴抽出による SQL インジェクション攻撃検出法について」, 情報処理学会 研究報告コンピュータセキュリティ (CSEC), vol. 2011-CSEC-52, no. 49, pp. 1–7, 2011 年 3 月.

[24] M. Sonoda, T. Matusda, D. Koizumi, and S. Hirasawa, “On Automatic Detection of SQL Injection Attacks by the Feature Extraction of the Single Character,” Proceedings of the 4th International Conference on Security of Information and Networks (SIN2011), pp. 81–86, Nov. 2011.

[25] M. Sonoda, T. Matsuda, and D. Koizumi,“On the Approximate Maximum Likelihood Estimation in Stochastic Model of SQL Injection Attacks,” Proceeding of 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC2016)(Accepted to appear), Oct. 2016.

[26] D. Koizumi, T. Matsuda, M. Sonoda, and S. Hirasawa, “A Learning Algorithm of Threshold Value on the Automatic Detection of SQL Injection Attack,” Proceedings of the 2012 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA’12), pp. 933–937, Aug. 2012.

[27] T. Oosawa and T. Matsuda,“SQL Injection Attack Detection Method using the Approximation Function of Zeta Distribution,” Proc. of 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC2014), pp. 819–824, Oct. 2014.

[28] The Open Web Application Security Project (OWASP), Testing for SQL Injection (OTG-INPVAL-005), https://www.owasp.org/index.php/Testing_for_SQL_Injection_(OTG-INPVAL-005) .

[29] Trustwave Inc. et al. OWASP ModSecurity Core Rule Set (CRS), https://modsecurity.org/crs/ .

[30] 園田 道夫，松田 健，“攻撃特徴記号に基づく WAF 開発,” 情報処理学会論文誌, vol. 56, no. 9, pp. 1826–1833, 2015 年 9 月.

[31] The Python Software Foundation, The Official Home of the Python Programming Language, https://www.python.org/ .

[32] Pedregosa et al., “Scikit-learn: Machine Learning in Python,” Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[33] scikit-learn, http://scikit-learn.org/ .

[34] 松田 健,「潜在曲線モデルを用いた能動的学習態度の推定」, 情報処理学会 研究報告 数理モデル化と問題解決 (MPS), vol. 2014-MPS-100, no. 26, pp. 1–4, 2014 年 9 月.

[35] The R Foundation, The R Project for Statistical Computing, http://www.r-project.org/ .

研究業績

分類	題目，発表・発行掲載誌名，発表・発行年月，著者
1. 論文 (査読付)	攻撃特徴記号に基づく WAF 開発 情報処理学会論文誌, vol. 56, no. 9, pp. 1826–1833, 2015 年 9 月 園田 道夫，松田 健
2. 国際会議 (査読付)	On the Approximate Maximum Likelihood Estimation in Stochastic Model of SQL Injection Attacks Proceeding of 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC2016)(Accepted to appear), Oct. 2016 M. Sonoda, T. Matsuda, and D. Koizumi
3. 国際会議 (査読付)	Markov Chain Monte Carlo Method Simulation of SQL Injection Attack Detection Proceeding of the 11th World Meeting for the International Society for Bayesian Analysis (ISBA2012) , p. 155, Jul. 2012 M. Sonoda, T. Matsuda, D. Koizumi, and S. Hirasawa
4. 国際会議 (査読付)	Predictive Distribution of SQL Injection Attacks Detection Model Proceeding of the 11th World Meeting for the International Society for Bayesian Analysis (ISBA2012), p. 133, Jul. 2012 T. Matsuda, D. Koizumi, M. Sonoda, and S. Hirasawa
5. 国際会議 (査読付)	On the Automatic Detection Algorithm of Cross Site Scripting (XSS) with the Non-stationary Bernoulli Distribution Proceeding of the 5th International Conference on Communications, Computers and Applications (MIC-CCA2012), pp. 131–135, Oct. 2012 D. Koizumi, T. Matsuda, and M. Sonoda

分類	題目，発表・発行掲載誌名，発表・発行年月，著者
6. 国際会議 (査読付)	Cross Site Scripting Attacks Detection Algorithm based on the Appearance Position of Characters Proceeding of the 5th International Conference on Communications, Computers and Applications (MIC-CCA2012), pp. 65–70, Oct. 2012 T. Matsuda, D. Koizumi, and M. Sonoda
7. 国際会議 (査読付)	A Learning Algorithm of Threshold Value on the Automatic Detection of SQL Injection Attack Proceedings of the 2012 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'12), pp. 933–937, Aug. 2012 D. Koizumi, T. Matsuda, M. Sonoda, and S. Hirasawa
8. 国際会議 (査読付)	On Automatic Detection of SQL Injection Attacks by the Feature Extraction of the Single Character Proceedings of the 4th International Conference on Security of Information and Networks (SIN2011), pp. 81–86, Nov. 2011 M. Sonoda, T. Matusda, D. Koizumi, and S. Hirasawa
9. 国際会議 (査読付)	On Predictive Errors of SQL Injection Attack Detection by the Feature of the Single Character Proceeding of 2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC2011), pp. 1722–1727, Oct. 2011 T. Matsuda, D. Koizumi, M. Sonoda, and S. Hirasawa
10. 講演	クロスサイトスクリプティング (XSS) 攻撃の検知におけるバッチ学習とオンライン学習の比較実験 情報処理学会 第 78 回全国大会講演論文集, vol. 2016, no. 1, pp. 463–464, 2016 年 3 月 梅原 章宏, 松田 健, 園田 道夫, 水野 信也, 趙 晋輝
11. 講演	文字の出現頻度とそれらの関連性による SQL インジェクション攻撃検出法 情報処理学会 研究報告コンピュータセキュリティ (CSEC), vol. 2015-CSEC-71, no. 16, pp. 1–6, 2015 年 11 月 佐野 綾子, 松田 健, 園田 道夫, 趙 晋輝

分類	題目，発表・発行掲載誌名，発表・発行年月，著者
12. 講演	機械学習を用いたクロスサイトスクリプティング (XSS) 攻撃の検知に関する考察 情報処理学会 研究報告コンピュータセキュリティ (CSEC), vol. 2015-CSEC-71, no. 14, pp. 1–4, 2015 年 11 月 梅原 章宏, 松田 健, 園田 道夫, 水野 信也, 趙 晋輝
13. 講演	線形分類器によるクロスサイトスクリプティング (XSS) 攻撃の検知に関する考察 第 14 回情報科学技術フォーラム (FIT2015) 講演論文集, no. 1, pp. 155–156, 2015 年 8 月 梅原 章宏, 松田 健, 園田 道夫, 水野 信也, 趙 晋輝
14. 講演	16 進数コードの出現状況に着目したバッファオーバーフロー攻撃の特徴抽出 第 14 回情報科学技術フォーラム (FIT2015) 講演論文集, no. 1, pp. 153–154, 2015 年 8 月 南後 吉秀, 松田 健, 園田 道夫, 趙 晋輝
15. 講演	潜在曲線を用いた着色による SQL インジェクション攻撃の特徴の可視化 情報処理学会 第 77 回全国大会講演論文集, vol. 2015, no. 1, pp. 435–436, 2015 年 3 月 藤岡 あやか, 松田 健, 園田 道夫, 趙 晋輝
16. 講演	線形分類器によるクロスサイトスクリプティング (XSS) の検知に関する考察 情報処理学会 第 77 回全国大会講演論文集, vol. 2015, no. 1, pp. 429–430, 2015 年 3 月 梅原 章宏, 松田 健, 園田 道夫, 趙 晋輝
17. 講演	URL 埋め込み型クロスサイトスクリプティング攻撃の特徴検出 情報処理学会 第 77 回全国大会講演論文集, vol. 2015, no. 1, pp. 427–428, 2015 年 3 月 海寶 貴人, 松田 健, 園田 道夫, 趙 晋輝
18. 講演	SQL インジェクション攻撃に含まれる文字の出現頻度とその関連性の解析による攻撃検出方法の提案 情報処理学会 第 76 回全国大会講演論文集, vol. 2014, no. 1, pp. 295–296, 2014 年 3 月

分類	題目，発表・発行掲載誌名，発表・発行年月，著者
	佐野 綾子, 松田 健, 園田 道夫, 趙 晋輝
19. 講演	ゼータ分布を用いた SQL インジェクション攻撃を検知する方法について 情報処理学会 第 76 回全国大会講演論文集, vol. 2014, no. 1, pp. 293–294, 2014 年 3 月 前田 すみれ, 園田 道夫，松田 健, 趙 晋輝
20. 講演	主成分分析を用いた分類器による SQL インジェクション攻撃の自動検出法 第 12 回情報科学技術フォーラム (FIT2013) 講演論文集, no. 4, pp. 187–192, 2013 年 8 月 園田 道夫, 松田 健, 小泉 大城, 趙 晋輝
21. 講演	文字集合からの特徴抽出による SQL インジェクション攻撃の自動検出における閾値学習アルゴリズム 情報処理学会 第 74 回全国大会講演論文集, vol. 2012, no. 1, pp. 559–560, 2012 年 3 月 小泉 大城, 松田 健, 園田 道夫, 平澤 茂一
22. 講演	攻撃文字列の特徴抽出と Web アプリケーションの自動検出へのアプローチ 情報処理学会 第 74 回全国大会講演論文集, vol. 2012, no. 1, pp. 557–558, 2012 年 3 月 園田 道夫, 松田 健, 小泉 大城, 平澤 茂一, 辻井 重男
23. 講演	文字単位の特徴抽出による SQL インジェクション攻撃検出法について 情報処理学会 研究報告コンピュータセキュリティ (CSEC), vol. 2011-CSEC-52, no. 49, pp. 1–7, 2011 年 3 月 園田 道夫, 松田 健, 小泉 大城, 平澤 茂一