機械学習によるログ解析を利用した WAF Mod Security 用チューニングツールの開発

工藤 千寛[1,a)]　山森 一人[2,b)]　相川 勝[3,c)]　井上 健太郎[2,d)]　齊藤　燎[4,e)]

概要：Web Application Firewall（WAF）運用の要である異常通信検知ルールの設定は難解であり、ログ解析や検知ルール記述言語の理解といった専門的な知識を要する。学習コスト面や金銭面の問題を解決できない中小企業や個人の技術者では、WAF を用いたセキュリティ対策を施すことが難しいという課題がある。本研究では、機械学習を利用した自動ログ解析を用いて WAF のログから偽陽性検知を検出し、誤検知全体の発生抑制が期待できる検知ルールの修正を支援する手法を提案する。

キーワード：Web Application Firewall, Character Level Convolutional Neural Networks, ログ解析, 自然言語処理

Development of WAF Mod Security tuning tool supported by machine learning for log analysis

Chihiro Kudo[1,a)]　Kunihito Yamamori[2,b)]　Masaru Aikawa[3,c)]　Kentaro Inoue[2,d)]　Ryo Saito[4,e)]

Abstract: Setting of detection rules, which is the key to the operation of WAF, is a difficult and complex task, it requires specialized knowledge such as log analysis and understanding for correction of detection rule description language. Therefore, it is difficult task for small and medium enterprises and individual engineers who cannot pay learning and financial cost. In this research, we propose a method to detect false positives from WAF logs by automatic log analysis using machine learning, and to support the correction of detection rules that can be expected to reduce the total number of misdetections.

Keywords: Web Application Firewall, Character Level Convolutional Neural Networks, Log analysis, Natural Language Processing

1. 序論

近年、Web 技術の発展と Web サービスの普及に伴い、我々の生活の中における Web サービスへの依存度が高まり、同サービス周辺に起こり得る脅威も増大している。Web サービスに対する攻撃は、サービス運用への妨害行為やサービスに附随する個人情報等の重要情報の搾取を目的としており、情報漏洩やサービスの可用性が損なわれるといったインシデントが発生する可能性がある [1]。これらの攻撃が発生する原因の一つに、サービスを提供するアプリケーションに潜む脆弱性がある。アプリケーションに脆弱性が含まれないように開発者は注意しているが、サービス

[1] 宮崎大学 工学研究科
Graduate School of Engineering, University of Miyazaki, Japan
[2] 宮崎大学 工学教育研究部
Faculty of Engineering, University of Miyazaki, Japan
[3] 宮崎大学 工学部 教育研究支援技術センター
Technical Center, Faculty of Engineering, University of Miyazaki, Japan
[4] 宮崎大学 農学工学総合研究科
Interdisciplinary Graduate school of Agriculture and Engineering, University of Miyazaki, Japan
a) kudo@taurus.cs.miyazaki-u.ac.jp
b) yamamori@cs.miyazaki-u.ac.jp
c) aikawa@cs.miyazaki-u.ac.jp
d) inoue@cs.miyazaki-u.ac.jp
e) saitou@taurus.cs.miyazaki-u.ac.jp

のリリース時にすべての脆弱性が取り除かれている保証はない。また発覚した脆弱性を何らかの理由ですぐには修正できないこともありうる。そのような状況で、アプリケーションの脆弱性対策に有効とされているのが、WAF（Web Application Firewall）[2] である。WAF はファイアウォール（F/W）と Web サーバの間に配置され、予め設定された検知ルールに基づき、アプリケーションに対する攻撃と思われる通信の検知や遮断などを行うセキュリティツールである。WAF は、F/W や IDS/IPS（Intrusion Detection System/Intrusion Prevention System）と異なり、HTTP リクエストや HTTP レスポンスの内容を検査するため、OSI 参照モデルにおける主として第 6、7 層でのアクセス制御では防げない攻撃への対策に有用である。

しかし、WAF 運用の要である検知ルールの設定は難解であり、ログ解析や検知ルールの修正といった専門的な知識を要する。Web サービスを運用する際、予算の余裕がある場合は WAF の運用を外部委託するか、外部の WAF サービスベンダを利用することが多い。

一方、新規サービスを容易に開発するためのフレームワークやツールが近年では発展しており、コスト面でセキュリティ人材の雇用が難しい中小企業やセキュリティの知識に乏しい個人の技術者でも Web サービスを一般に公開している例は多い。しかし、セキュリティ対策の技術までは簡易化されておらず、技術力や知識が不足した技術者が開発した Web サービスでは、セキュリティ対策を十分に施されていない場合がある。Web サービスに対する攻撃は、セキュリティ対策が乏しい、中小企業や個人が運用する Web サービスに対して行われることが多い。

WAF の運用を始める際には、誤検知を削減するためチューニングと呼ばれる検知ルールの修正を繰り返す必要がある。この過程ではログ解析などの専門的な知識技術を要する。よって学習コスト面や金銭面の問題を解決できない中小企業や個人の技術者では、WAF を用いたセキュリティ対策を施すことは難しい。

本研究では、機械学習を利用した自動ログ解析により WAF のログから偽陽性検知を検出し、誤検知全体の発生数低減が期待できる検知ルールの修正案を運用者に提案する、オープンソースの WAF を中心に構成したツールを開発する。OSS（Open Source Software）の WAF である Mod Security [3] は、Apache Web サーバ（Apache）のモジュールとして実装されている。Mod Security が異常と判断した Apache への通信は、Mod Security のログとして記録される。この Mod Security のログの中には、真陽性検知と偽陽性検知の二つが含まれる。提案手法では、Mod Security のログを学習したモデルを使用して、真陽性検知か偽陽性検知かを二値分類する。偽陽性検知と判別された通信ログをもとに、検知ルールの修正案を作成する。

本論文の構成は，以下の通りである。第 2 章で WAF の概要を述べ、第 3 章で提案手法を説明する。第 4 章で提案手法を評価する。第 5 章は本研究のまとめである。

2. Web Application Firewall

2.1 Web セキュリティ

Web セキュリティでは、OSI（International Organization for Standardization）参照モデルにおけるプレゼンテーション層（第 6 層）やアプリケーション層（第 7 層）を対象とした対策がメインである。これらの層におけるセキュリティ対策は、一般的なソフトウェアの脆弱性対策に加えて、XSS（cross site scripting）や SQL injection に代表される Web アプリケーションプログラムの脆弱性対策が主なものである。これらの脆弱性は、開発時には想定されていないアプリーケーションへの入力が要因となり発生する攻撃である。

Web アプリケーションは OS やサーバプログラムのような、プラットフォームとして共通で利用されることが前提となる階層のプログラムと異なり、さまざまな主体が各々の目的のためにプログラムを作成する。そのため、アプリケーションの開発段階で脆弱性が入り込みやすく、開発者の知識だけでは脆弱性を完全に排除することは難しい。一方で、これらへの攻撃が招く被害は実に大きく、例えばデータベースへの不正なアクセスを許してしまうと、個人情報などの高い機密性が求められる情報が流出する可能性があり、適切なプログラム設計をするなど対策を施すことは非常に重要である。

2.2 WAF とは

2.2.1 WAF の有効性

開発者は Web アプリケーションへの攻撃を防ぐために安全なプログラムの記述と保守を心がけているが、Web アプリケーションのプログラムに脆弱性が含まれていないことを保証することは不可能である。脆弱性が見つかった後、それが改修されるまでは脅威にさらされたままとなる。こうした場合の対策に有用であるのが、図 1 に示す WAF と呼ばれるセキュリティソフトウェアである。WAF は Web サーバに対して送信された HTTP/HTTPS 通信の内容を検査し、あらかじめ定義された検知ルールにマッチすれば通信の記録や遮断を行う。WAF の性能は、この検知ルールに依存する。

2.2.2 IPS、IDS や F/W との役割の違い

ネットワークやサーバ、アプリケーションを外部から守る手段の代表的なものとして、F/W、IDS/IPS、WAF がある。それぞれのセキュリティ対策機器やソフトウェアにとって得意分野があり、防ぐべき攻撃を明確にすることで効果的な対策が可能となる。

ファイアーウォール（F/W）とは、ポリシーあるいはフィルタリングルールに基づいて、パケットの通過、拒否、破

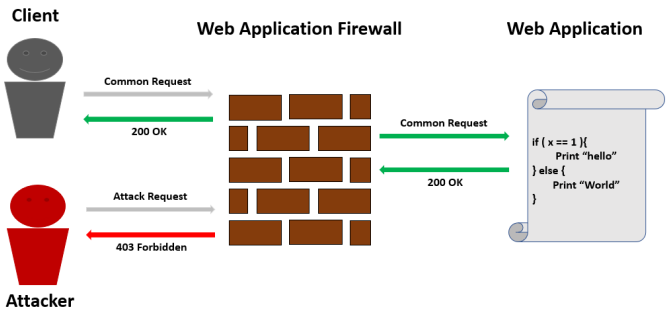

図 1　一般的な WAF の役割.

棄を行うネットワーク機器、あるいはソフトウェアのことである [4]。OSI 参照モデルにおけるネットワーク層（第 3 層）やトランスポート層（第 4 層）において通信を防御する。フィルタリングでは、宛先及び送信元の IP アドレス、プロトコルの種類やポート番号を用いてパケットを選別する。F/W は通信の内容を検証しないため、近年増加している Web アプリケーションを狙った SQL injection や XSS のような攻撃は検知できない。

通信路を監視し、ネットワーク上の攻撃を検知するシステムのうち、検知のみを行うシステムを IDS（侵入検知システム）、検知した通信を遮断するシステムを IPS（侵入防御システム）という [5]。検知方法としては、異常通信を抽象化したシグネチャを格納したデータベースとのパターンマッチングによる方式と、通常とは異なる状態を検知するアノーマリ検知による方式がある。IDS/IPS も F/W と同様に通信の内容までは検証しないため、近年増加している Web アプリケーションを狙った SQL injection や XSS のような攻撃は検知できない。

2.3　WAF 運用の流れ

2.3.1　WAF 導入の事前検討と運用計画

WAF を導入する前に、効率的に運用を進めるためにいくつかの項目について事前に検討しておくことが望ましい。例えば、下記のような点についてである。

- 誰が運用の中心となるのか
- 予算や人材はどの程度掛けられるのか
- 既存のネットワーク構成やサーバの設置場所はどこか
- サーバの性能はどの程度か
- どのようなタイプの WAF を運用するのか
- 脆弱性が発見された場合に開発者に Web アプリケーションの改修を依頼できるか

WAF を運用する者はサーバ・ネットワーク管理者と、WAF 導入によってサーバに及ぼす処理能力面での影響やネットワーク構成の変更が必要かどうかなどを協議・調整しておく必要がある。サービス運営者には、偽陽性検知発生時にサービスの利用に支障をきたす可能性や、導入時にサービスが一時停止することなど運用におけるリスクを説明しておく必要がある。Web アプリケーション開発者とは、保守サポートの契約について確認し、WAF を導入することでサポートの範囲外にならないかを確認しておく必要がある。WAF サービスベンダが提供する WAF を導入する場合には、WAF サービスベンダの担当者を交えて打ち合わせしておく必要がある。また運用面では、どのような基準でログを出力するかの設定や、有効にする検知ルールの選定などについて協議しておく必要がある。

2.3.2　チューニングと運用テスト

WAF を運用する際、検知基準となる検知ルールのチューニング作業が必要である。WAF を運用し始めた初期段階では、検知ルールが個々の Web アプリケーションに最適化されていないため誤検知が多く発生する。実運用の前に実運用環境とは別のテスト環境を構築し、テスト用の通信データを用いて検知ルールをあらかじめチューニングしておくことが望ましい。実際の運用開始後も、図 2 のように継続的に WAF から出力されたログを解析し、誤検知を減らすように検知ルールの修正が必要である。

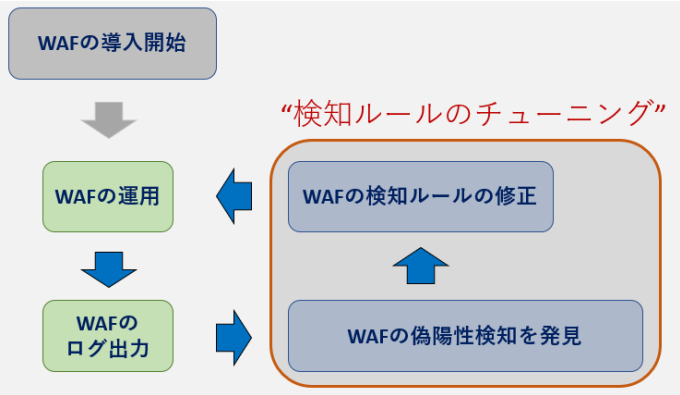

図 2　検知ルールのチューニング.

2.4　Web アプリケーションの脆弱性対策の難しさ

脆弱性を発見した場合、まずはその脆弱性を改修できるかどうかを確認する必要がある。複雑巧妙化した攻撃を防御するためのプログラムの改修は難易度が高く、時間もかかり、またサービスの運用が続く限り継続的に脆弱性が発見されるため、膨大なコストを要する。改修に掛かるコストが予算を超えてしまったり、保守の対象外となった Web アプリケーションなどで開発者のサポートを受けられない場合など、脆弱性の改修が難しい場合もある。このようなケースも WAF が存在意義を発揮する一つのシチュエーションではあるが、脆弱性そのものが取り除かれていないことに留意が必要である。

2.5 WAF 運用の課題

Web アプリケーションのセキュリティ管理には専門的な知識や技術が必要である。近年、クラウドコンピューティングサービス [6] や Web アプリケーションフレームワーク [7] の発達に伴い、個人や中小企業の技術者でも Web アプリケーションを開発し、公開することが容易になっている。一方でセキュリティ管理手法は簡易化されておらず、知識や技術が備わっていないままでの対策導入は難しい。Web アプリケーションの脆弱性対策に有用とされる WAF を運用するためには、ログ解析や検知ルールの修正など、セキュリティ知識の浅い技術者にとっては難しい作業が必要である [8]。

この問題を解決するには、外部の専門技術者にコストをかけて依頼するか、クラウド型の WAF サービスを利用する、自社で人材を育成するなどの方法がある。しかし、中小企業や個人の技術者のほとんどは、導入と運用コストという大きな障壁に直面する。この障壁は、個人や中小企業の技術者が Web アプリケーションの適切な脆弱性対策を施すことを阻害する理由になりうる。

2.6 偽陰性発見時の対応

偽陰性検知となる通信を運用後のログ解析で発見することは難しい。そのため実運用前のテスト段階で、通常通信と異常通信でそれぞれラベル付されたテストデータを用いて検知ルールをチューニングしておくことが望ましい。しかし、運用開始後にテストデータとは異なる実際の通信で誤検知が発生してしまう可能性は十分に考えられる。特に、偽陰性検知となる通信は WAF のログには記録されないため、攻撃による被害を実際に確認した段階で初めて発覚することになる。そのため偽陰性検知の発生は Web アプリケーションの運用者や利用者にとって非常に脅威であり、検知ルールの適切な設定は非常に重要である。

3. 提案手法

3.1 提案手法の概要

本研究では、低コストな WAF の運用支援手法を提案する。本研究で提案する手法のターゲットは、ノウハウと予算不足の観点から、WAF の導入と運用に予算を確保することが難しい中小企業と個人の技術者である。

OSS の WAF である Mod Security を採用することで、導入コストの問題を解決する。Mod Security はソフトウェア型の WAF であり、物理サーバやクラウドサーバのどちらでも導入できるため環境依存性が低く、他の形態の WAF に比べて環境構築が容易である。また、機械学習による自動ログ解析 [9] の結果を利用して、WAF の運用上で困難な要素の一つである検知ルールのチューニングの問題を解決することを目指す。

提案手法のアーキテクチャを図 3 に示す。Web サーバが受信したリクエストを WAF が検証し、その結果は検知ログに出力される。検知結果には WAF によって異常と判別された通信ログが記録されているが、中には正常な通信が誤検知された検知結果を含む。機械学習による自動ログ解析によって、ログから誤検知の検知結果を検出し、その情報をもとに検知ルールセットの修正案を作成する機能を提供することでチューニング作業を自動化する。

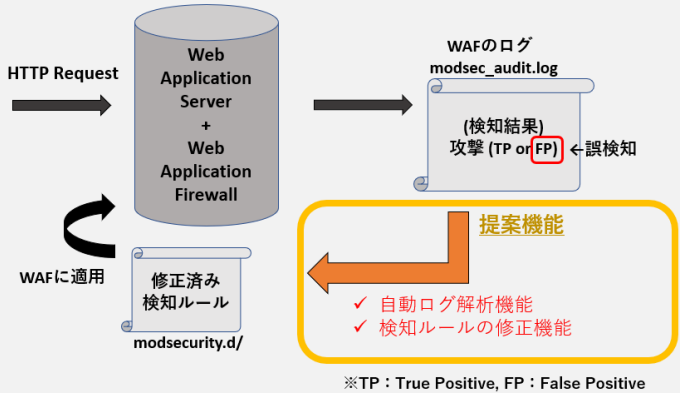

図 3 提案手法の概要.

3.2 HTTP DATASET CSIC 2010

HTTP DATASET CSIC 2010（CSIC 2010）[11] は、スペイン研究高等評議会（CSIC）の情報セキュリティ研究所が開発した、HTTP 通信の DATASET である。36,000 個の正常な通信と 25,000 個を超える異常な通信を含む。これらはそれぞれ、正常な通信と異常な通信でファイルを分け提供されている。異常通信の中には、SQL injection、バッファオーバーフロー、情報収集、ファイル開示、CRLF injection、XSS、サーバサイドインクルード、パラメーター改ざんなどの一般的な Web アプリケーションに対する攻撃を含む。

3.3 Mod Security

Mod Security は、Information-technology Promotion Agency, Japan（IPA）も利用を推奨している、Trustwave SpiderLabs が GPLv2 ライセンスの下提供している OSS の WAF である。ソフトウェア型やインストール型、ホスト型と呼ばれる種類の WAF であり、Nginx や Apache、IIS といった著名な Web サーバと組み合わせて利用でき、ハードウェアの購入コストが生じないメリットがある。また WAF サービスベンダ提供のサービスではなく OSS であるため、自らがその運用を行えばかかるコストはゼロである。

3.4 OWASP Core Rule Set

Mod Security は多くの場合、OWASP（Open Web Application Security Project）が GPLv2 ライセンスのもと提供している Core Rule Set（CRS）[10] とセットで運用される。CRS とは、Mod Security の SecRules 言語で記述されたオー

プンソースの検知ルールセットであり、これを利用することで 1 から検知ルールを設定する必要がないというメリットがある。

検知ルールは CRS を構成する各ファイルの中で記述されている。図 4 は、SQL injection に関する検知ルール記述の例である。CRS の検知ルールは、新規に追加することも可能である。

CRS は広く公開された汎用的な検知ルールセットであるため、個々の環境に応じてチューニングが必要である。検知結果は、modsec_audit.log に出力されるため、同ログファイルを解析して検知ルールをチューニングするのが一般的である。

デフォルトの CRS を適用した場合の WAF の検知精度を測定したときの、偽陽性検知と偽陰性検知の発生率を図 5 に表す。図 5 は、正常と異常な通信のそれぞれ 500 個の異なるデータセットを使用して実験を 6 回繰り返した時の値である。

偽陽性率が 40%と高く、偽陰性率が 0%であることから、デフォルトでの CRS の検知ルールは厳し目の設定となっていることがわかる。

```
SecRule \
  REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* \
  "(/\*!?|\*/|[';]--|--[\s\r\n\v\f]|(?:--[^-]*?-)|([^\-&])#.*?[\s\r\n\v\f]|;?\\x00)" \
  "phase:2, \
  rev:'2', \
ver:'OWASP_CRS/2.2.6', \
maturity:'8', \
accuracy:'8', \
id:'981231', \
t:none, \
t:urlDecodeUni, \
block, \
msg:'SQL Comment Sequence Detected.', \
severity:'2', \
capture, \
logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}', \
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION', \
tag:'WASCTC/WASC-19', \
tag:'OWASP_TOP_10/A1', \
tag:'OWASP_AppSensor/CIE1', \
tag:'PCI/6.5.2', \
setvar:tx.anomaly_score=+%{tx.critical_anomaly_score}, \
setvar:tx.sql_injection_score=+1, \
setvar:'tx.msg=%{rule.msg}', \
setvar:tx.%{rule.id}-OWASP_CRS/WEB_ATTACK/SQL_INJECTION-%{matched_var_name}=%{tx.0}"
```

図 4　CRS の検知ルールの記述例.

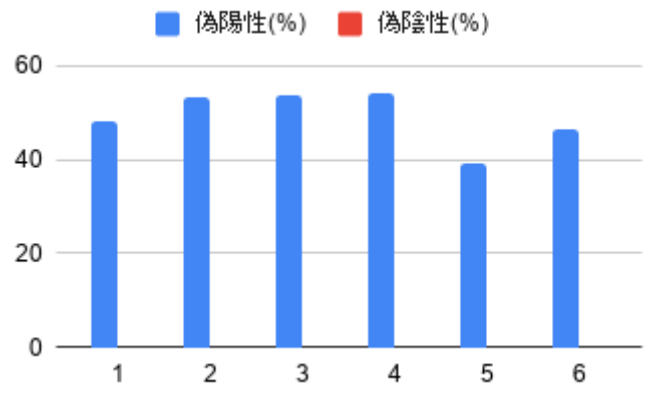

図 5　デフォルトの CRS を適用した場合の Mod Security の誤検知率.

3.5　自動ログ解析機能

3.5.1　Character-level CNN

Convolutional Neural Network（CNN）は、画像や動画認識の分野で広く使われるモデルである。典型的な CNN では、畳み込み層とプーリング層を繰り返し積層し、最後の結合層（FC）を経て入力を分類する。

畳み込み層は複数のフィルタから構成され、これらのフィルタを使って畳み込みを行う。フィルタのサイズをカーネルサイズと呼ぶ。畳み込み処理では、点ではなく領域で特徴を抽出できるため、特徴を大きく損なわないまま情報量を小さくできる。出力サイズは、畳み込み層への入力を 0 パディングすることで調整できる。

プーリングは、特定の領域を一つの要素に集約する演算であり、入力データのサイズを小さくする演算である。最大プーリングは最も一般的なプーリング方法の 1 つであり、特定の領域における最大値を選択する。これによって計算量の削減が見込まれるため、学習時間の短縮や計算リソースの節約が期待できる。

文章データを CNN を用いて分類するためには、文章データを数値データに変換する必要がある。文章データを数値データに変換する手法はいくつかあり、その一つに one-hot エンコーディング [12] がある。one-hot エンコーディングとは、ベクトルで単語を表現する手法の一つである。

前処理を行ったうえでベクトル化した文章データを、学習のために CNN に入力する。入力データが文章データである CNN を、一般に Character-level CNN（CLCNN）と表現する。提案手法では、Mod Security が出力したログを CNN に入力し、検知結果の一つ一つが偽陽性検知か真陽性検知かに分類する。

3.5.2　学習用データセットの準備

提案手法では、ログに記録された検知結果が真陽性検知か偽陽性検知かを CLCNN により分類する。学習と評価のために、真陽性検知と偽陽性検知のログを等量抽出する。

デフォルトの CRS を適用した場合の誤検知率は、3.4 節の計測より、偽陽性検知率は最大で 55%、偽陰性検知率は 0%である。よって、デフォルトの CRS 適用時における偽陽性検知のログを 16,000 個集めるためには、正常通信が約 30,000 個、真陽性検知のログを 16,000 個集めるためには異常通信が約 16,500 個あれば十分である。実際には、プログラムで自動的に CISC 2010 に記録された HTTP リクエストを読み込み Apache へ送信するため、プログラムのループ処理の中で送信データ数をそれぞれ設定する。

CSIC 2010 では、HTTP リクエストが正常通信と異常通信に予め分類されている。したがって、出力されたログは、図 6 のように HTTP リクエストが正常か異常かに応じて、偽陽性検知のログまたは真陽性検知のログとして簡単にラベル付けできる。

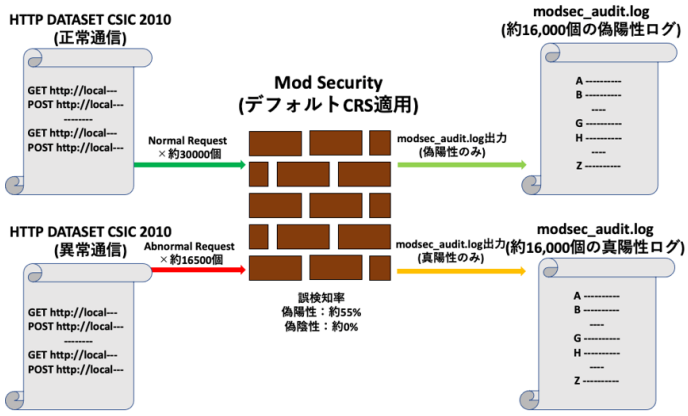

図 6　データセットの準備手順.

3.5.3　modsec_audit.log の出力

Apache が HTTP リクエストを受信すると、Mod Security のログである modsec_audit.log が出力される。正常通信を受信した際の modsec_audit.log の出力は偽陽性検知を意味する。一方で、異常通信を受信した際の modsec_audit.log の出力は、真陽性検知を意味する。これらはラベル付けのために別々のファイルで管理する。

3.5.4　データの前処理

modsec_audit.log をそのまま CLCNN に入力すると、学習時間の増大や計算機リソースの浪費が過大となり、モデルの性能面でも期待する性能を発揮できない可能性がある。そのため CLCNN の性能が高くなるような特徴を modsec_audit.log から抽出するプロセスが必要である。Web アプリケーションに対する攻撃が行われる際には、HTTP/HTTPS リクエストのリクエスト部に攻撃に関連した文字列が含まれている。そこで素の modsec_audit.log から、HTTP リクエストが記録された文字列を抽出する。

本研究で取り扱っているデータセットには、HTML タグや Javascript プログラムなどの Web テキストを含まれているため、これらはノイズとして CLCNN の学習を阻害する恐れがある。そのため CLCNN に入力するデータを前処理して、HTML タグやハッシュタグなどのノイズを取り除いておく。

抽出したデータは、tsv ファイル形式で一つのファイルにまとめる。ただし二つのログに記録された通信の数は、3.5.2 節で述べた通り、偽陽性検知と真陽性検知のログそれぞれ 16,000 個を目指し、余裕を持った数の HTTP 通信を送信しているため、正確に 16,000 個とは限らない。そのため、一つの tsv ファイルにまとめる際に、真陽性検知と偽陽性検知が各 16,000 個ずつとなるように調整する。modsec_audit.log の内容はすでに真陽性検知と偽陽性検知の二つのファイルに分類されているため、これを利用して真陽性検知は 0、偽陽性検知は 1 とラベル付けしながら tsv ファイルを作成する。

文章データを CNN で分類するためには、入力する文章データを数値データに変換する必要がある。文章データ中の文字を一つずつ unicode の 2 進数形式に変換し、8 ビットの数値文字列で表現する。さらに、one-hot エンコーディングでベクトル形式に変換する。one-hot エンコーディングで表現されるベクトルの次元数は、単語の数と同じ数となる。CLCNN への入力は、Ito [13] らの研究に基づき 1,000 文字とした。1,000 文字に満たない文字列については、不足分を Unicode において null を表す「0」でパディングした。

3.5.5　CLCNN の学習と分類

本研究で構築した CLCNN のアーキテクチャは、Ito らの研究に基づき、図 7 [13] のように畳み込み層とプーリング層は各 2 層ずつとし、2 層の畳み込み層と 1 層目のプーリング層のカーネルサイズは 1 × 7 となっている。2 層目のプーリング層におけるプーリングサイズは、畳み込み層の出力次元と同じである。活性化関数はすべて ReLU を用いている。

モデルを用いて、3.5.2 節で準備した真陽性検知ログ 16,000 個、偽陽性検知ログ 16,000 個の合計 32,000 個のデータセットを学習・評価する。学習後のモデルを用いて、Mod Security が検知した通信が真陽性検知か偽陽性検知かを分類する。

3.6　検知ルール修正機能

偽陽性検知が見つかった場合は、CRS の検知ルールを修正し、偽陽性検知を削減する必要がある。

Mod Security が出力した modsec_audit.log には、検知した通信のそれぞれについて、A から Z までの項目に分けて詳細な情報が記録されている。その一つである H 属性には、検知の根拠となった検知ルールを特定できる id と呼ばれる情報が含まれている。この id を利用して当該検知ルールを無効化する。具体的には、無効化したい検知ルールの id を Mod Security の設定ファイル、/etc/httpd/conf.d/mod_security.conf に書き込む。id ”981231”の検知ルールを無効化する設定の例を図 8 に示す。

4.　実験

4.1　実験環境

本研究で開発したツールの性能を検証するために、二つの観点からツールを評価する。一つ目の観点は、CLCNN を用いた自動ログ解析で Mod Security の偽陽性検知を発見する精度である。二つ目の観点は、開発ツールによる修正を受けた検知ルールセットを適用した場合の、Mod Security の検知精度である。実験環境のシステム要件と概要を図 9 に示す。

4.2　実験結果

4.2.1　偽陽性検知精度の検証

本研究で構築した CLCNN のアーキテクチャは、3.5.5 節

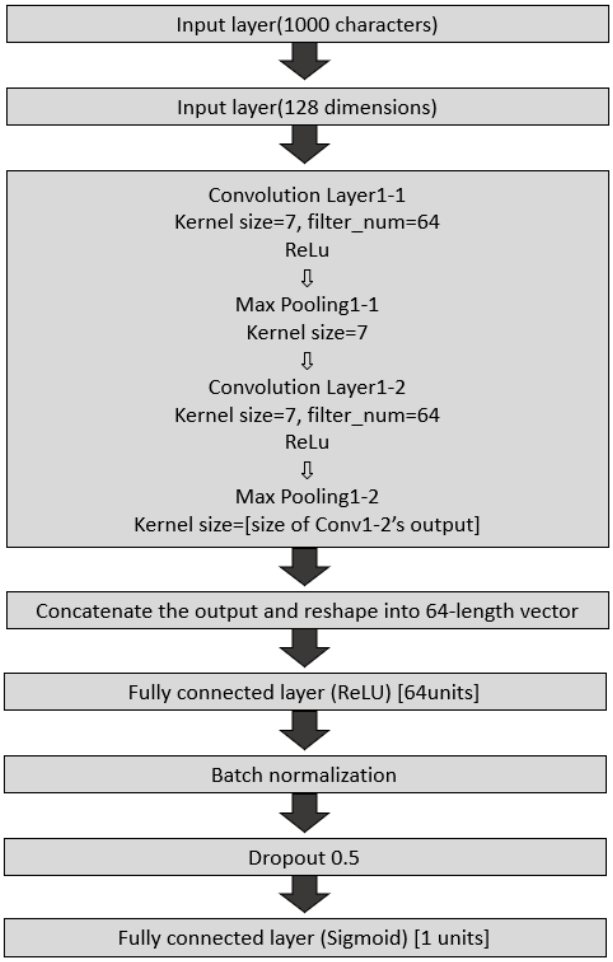

図 7　提案手法における CLCNN のアーキテクチャ.

```
SecRuleRemoveById  981231
```

図 8　OWASP CRS の検知ルール無効化の例.

で述べた通りである。表 1 は、異なる組み合わせのデータセットを使用し、6 回の試行における CLCNN の分類精度を示す。なお学習に用いたデータ数を表 2 に示す。

表 1　CLCNN の分類精度.

回数	1	2	3	4	5	6
検知率（%）	93.44	93.87	94.26	93.64	93.95	93.66

4.2.2　カスタム CRS での偽陽性/偽陰性検知精度

正常通信と異常通信それぞれ 500 個を含む、異なる通信ログをもとに、本ツール提案による検知ルールを適用したカスタム CRS での誤検知率を測定した。図 10 は、ツールによって変更された CRS を適用した場合の、ModSecurity

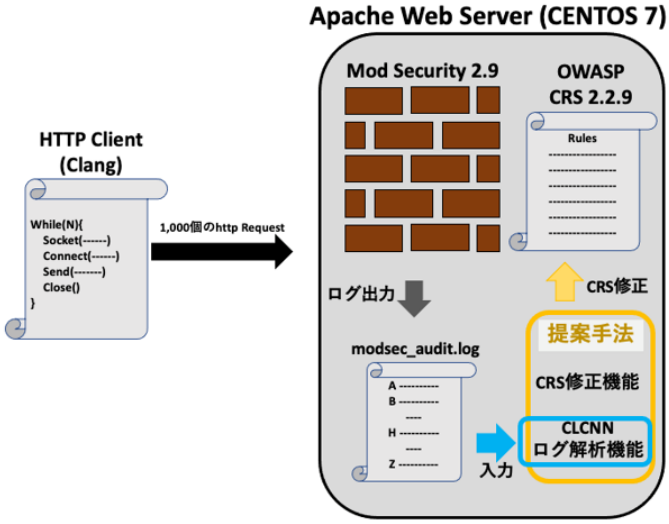

図 9　実験環境の概要.

表 2　CLCNN の学習・評価に用いたデータ数.

データ名	データ数（個）	ラベル
真陽性ログ	16,000	0
偽陽性ログ	16,000	1
合計	32,000	-

による偽陽性検知率と偽陰性検知率を示している。3.4 節の測定と同じデータセットを使用して、実験を 6 回試行している。デフォルトの CRS を適用した場合と比較して、偽陽性検知は大幅に削減できた一方で、偽陰性検知が新たに生じている。偽陰性検知の発生は、誤ったログ解析結果に基づいて無効化された検知ルールの存在があるためと考えられる。

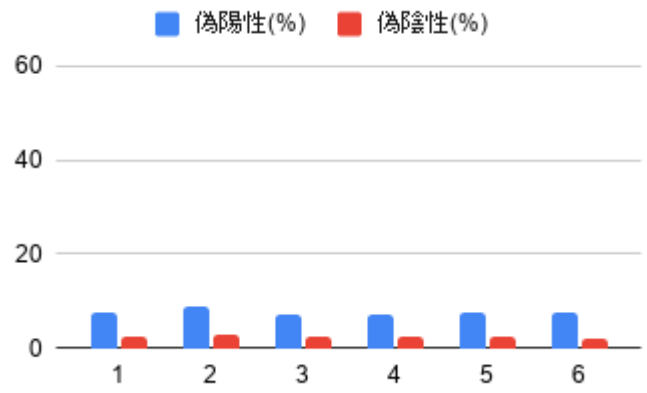

図 10　カスタム CRS を適用した場合の Mod Security の誤検知率.

5.　結論

本研究では、難解な WAF のチューニングを支援するツールを開発した。提案方法は、CSIC 2010 データセットに対する Mod Security のログを用いて学習させた CLCNN によってログを解析し、Mod Security の誤検知を減らすため

に CRS の検知ルールを修正する。

CLCNN は、約 93.8%の精度で Mod Security に記録された検知結果が真陽性検知か偽陽性検知かを分類できる。500 個の正常通信と異常通信それぞれの検知結果をもとにカスタマイズされた検知ルールを適用した Mod Security は、偽陽性検知の発生をデフォルトの CRS より約 41.6%減少させることができた。一方、偽陰性検知が新たに約 2.4%発生した。偽陽性誤検知を削減することで、WAF 運用者の負担を引き下げつつサービスの可用性を向上させた。偽陰性検知の発生は、Web アプリケーションを運用する上で大きな問題となる可能性がある。偽陽性検知の発生原因として、CLCNN が異常通信を正常と誤分類したことで、誤って無効化された検知ルールの存在があると考えられる。ログ解析の精度向上と、検知ルールの無効化に一定の基準を設けることが求められる。また現在の実装では、偽陽性検知したすべての検知ルールを一括して無効化する。危険度の低い通信を検知するためにある検知ルール X を無効化したところ、以後検知ルール X が有効であれば検知できた危険度の高い攻撃を防げなくなる場合が想定される。解決の方法として、検知ルールそれぞれの検知実績をもとに検知ルールとしての有効度を評価し、これをもとに検知ルールの無効化の是非を決定することが有効であると考える。

開発したツールでは、検知ルールの無効化を自動ログ解析によって得られた情報をもとに行っている。偽陰性検知となる通信がログに記録されることはなく、その発生をログ解析で直接的に確認することはできない。しかし偽陰性検知となる通信はインシデントにつながる可能性が高く、偽陽性検知となる通信と比較して特に無視できない。そのため、偽陽性検知の発見をもとに検知ルールを変更しつつ、その副作用で偽陰性検知が増えないような対策を講じることが今後の課題である。

謝辞

本研究は JSPS 科研費 19K12139 の助成を受けたものです。

参考文献

[1] Piyush A. Sonewar and Sonali D. Thosar, Detection of SQL injection and XSS attacks in three tier web applications, 2016 International Conference on Computing Communication Control and automation（ICCUBEA）, pp.1-4, Aug 2016.

[2] Victor Clincy and Hossain Shahriar, Web Application Firewall: Network Security Models and Configuration. 2018 IEEE 42nd Annual Computer Software and Applications Conference（COMPSAC）, pp.835-836, June 2018.

[3] Imrana Abdullahi Yari, Babangida Abdullahi and Steve A. Adeshina, Towards a Framework of Configuring and Evaluating ModSecurity WAF on Tomcat and Apache Web Servers, 2019 15th International Conference on Electronics, Computer and Computation（ICECCO）, pp.1-7, Dec 2019.

[4] Alex X. Liu and Mohamed G. Gouda, Firewall Policy Queries, IEEE Transactions on Parallel and Distributed Systems, vol. 20, no. 6, pp. 766-777, June 2009.

[5] Filip Hock and Peter Kortiš, Commercial and open-source based Intrusion Detection System and Intrusion Prevention System（IDS/IPS） design for an IP networks, 2015 13th International Conference on Emerging eLearning Technologies and Applications（ICETA）, pp.1-4, Nov 2015.

[6] Shahryar Shafique Qureshi, Toufeeq Ahmad, Khalid Rafique and Shuja-ul-islam, Mobile cloud computing as future for mobile applications - Implementation methods and challenging issues, 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, pp.467-471, Sep 2011.

[7] Stenly Ibrahim Adam and Stevani Andolo, A New PHP Web Application Development Framework Based on MVC Architectural Pattern and Ajax Technology, 2019 1st International Conference on Cybernetics and Intelligent System（ICORIS）, pp.45-50, Aug 2019.

[8] Dennis Appelt, Annibale Panichella, and Lionel Briand, Automatically Repairing Web Application Firewalls Based on Successful SQL Injection Attacks, 2017 IEEE 28th International Symposium on Software Reliability Engineering（ISSRE）, pp.339-350, Oct 2017

[9] Weiliang Ji, Shihui Duan, Renai Chen, Song Wang and Qiang Ling, A CNN-based network failure prediction method with logs, 2018 Chinese Control And Decision Conference（CCDC）, pp.4087-409, Jun 2018

[10] OWASP ModSecurity Core Rule Set, https://owasp.org/www-project-modsecurity-core-rule-set/.

[11] HTTP DATASET CSIC 2010, [online], Available:https://www.isi.csic.es/dataset/.[accessed:3-Sep-2020].

[12] Mohammed A El Affendi and K H S Al Rajhi, Text encoding for deep learning neural networks: A reversible base 64（Tetrasexagesimal）Integer Transformation（RIT64）alternative to one hot encoding with applications to Arabic morphology, 2018 Sixth International Conference on Digital Information, Networking, and Wireless Communications（DINWC）, pp.70-74, Apr 2018.

[13] Michiaki Ito and Hitoshi Iyatomi, Web Application Firewall using Character-level Convolutional Neural Network, 2018 IEEE International Colloquium on Signal Processing & its Applications, vol.4, pp.103–106, March 2018.