# AETP: イベントの関連づけによるWebアプリケーションに対する有効な攻撃の検知手法

鐘　揚[1]　佐藤　徹[1]　谷川　真樹[1]

**概要**：Web アプリケーションの脆弱性に対する攻撃は日々発生しており，対策である WAF は日々大量のアラートを出力するため，真に影響あるアラートの判断が課題となる．本稿では，HTTP リクエストと，システムイベント（ファイルアクセス，コマンド実行，DB アクセス等）を関連付け，一連のイベントの異常性を判断することで，脆弱性情報に依存しない，Web アプリケーションに対する攻撃の有効性を検知する手法 AETP を提案する．評価の結果，新たな攻撃も検知可能であり，さらに既存の IDS に比べアラートを約 50%削減することができ，攻撃検知における効率性を向上させたことを示す．

**キーワード**：Web セキュリティ，攻撃検知，相関分析

# AETP: A Host based Intrusion Detection Method for Identifying Effective Web Attacks

Yang ZHONG[1]　Tohru SATO[1]　Masaki TANIKAWA[1]

***Abstract:*** Attackers launch exploits to web applications persistently for checking their vulnerabilities. WAFs are triggered a large amount of alerts by these attacks, but which is not effective in most times. In this paper, we present a novel approach (AETP) leverage event correlation techniques to detect effectiveness of web attacks. Evaluation results show AETP can detect new attacks and it reduce 50% of false positive alerts.

***Keywords:*** Web Security, Attack Detection, Correlation Analysis

## 1. はじめに

Web アプリケーションは外部に公開されているため，日々大量の脆弱性スキャンや攻撃の脅威に晒されている．脆弱性が公開された場合，攻撃者は継続的に Web 空間に対して脆弱性スキャンを行い，Web サイトにその脆弱性が存在するか調査する．Web アプリケーションに対する攻撃の対策である WAF はこうした脆弱性を悪用する攻撃を検知するため，攻撃が行われる度アラートを出力する事となる．そのため，攻撃が日々行われている現在では WAF は日々，大量にアラートを出力し，それらのアラートの調査に要する人的コストも大きい．

この課題に対して WAF などの IDS のアラートの情報とアプリケーションの脆弱性の情報を関連付けることで，攻撃の有効性を確認する Kruegel らの手法 [1] が提案されている．より具体的には WAF などの IDS から出力されるアラート情報と脆弱性スキャナなどの結果と関連付けることで，攻撃によるアラートがあった場合，そのアラートが示す攻撃対象のアプリケーションが対象の攻撃に対する脆弱性を含むバージョンである場合のみ，有効な攻撃として判断する手法である．そのため，脆弱性のないバージョンのアプリケーションに対する攻撃は無効と判断できるため，真に調査が必要なアラートの件数を削減することが可能である．しかし，既存手法では既知の攻撃に関する情報を用いているため，有効な攻撃を検知可能な範囲が限定的である．さらに独自アプリケーションにも対応できないという

[1] NTT セキュアプラットフォーム研究所
NTT Secure Platform Laboratories

課題も存在する．

本稿では，既知の攻撃情報及び脆弱性情報等を利用しない Web アプリケーションへの攻撃の有効性を判定する手法を提案する．提案手法では HTTP リクエストと攻撃による影響が現れるイベント（ファイルアクセス，コマンド実行や DB アクセス等）を関連付け，それらのイベントに異常性が現れるかどうかで攻撃の有効性を判定する．実際の Web アプリケーションに本手法適応した評価の結果，新たな攻撃も検知可能であり，さらに既存の IDS に比べアラートを約 50%削減することができ，攻撃検知における効率性を向上させたことを示す．

2 章では既存の Web アプリケーションに対する攻撃の対策についてその本研究で考える要件と既存対策の課題について述べる．3 章では有効性ある攻撃を検知する提案手法について詳しく解説する．4 章では提案手法に対して行った評価結果について説明する．本提案手法における制約を 5 章にて述べ，6 章にて本稿をまとめる．

## 2. Web アプリケーションに対する攻撃への対策

2 章では著者らが考える Web アプリケーションに対する攻撃検知における要件，及びその要件に照らし，既存方式の課題を説明する．

### 2.1 要件

- **有効な攻撃の判定：**Web アプリケーションに対する攻撃が日々行われている現在では，攻撃を検知したことを表すアラートが大量に発生してしまい，アラートを調査する人的コストが高い．その攻撃が有効だったかどうか判断することによって有効でない攻撃によるアラートを排除し，真に対処すべきアラートに効率的に対処する必要がある．
- **新たな攻撃への耐性：**様々な Web アプリケーションが広く開発され，利用される現在では新たな脆弱性が発見され，その脆弱性を悪用した新たな攻撃が行われる．新たな攻撃に対しても検知・判定である可能である必要がある．
- **高精度な検知：**外部に公開されている及び不特定多数のユーザから利用される特性上，大量のアクセスがありかつ一定のサービス品質の担保が必要とされる．つまり，正常の通信を誤って検知してしまう誤検知が少ないことが要求される．

### 2.2 方式

- **NIDS:** NIDS (Network based Intrusion Detection System) 方式はネットワーク上の通信内容に基づいて攻撃を検知する．Web サーバに対する攻撃の検知に特化したものに WAF が存在する．古くからよく利用されているものに Snort[*1] や ModSecurity[*2] などが存在する．
- **HIDS:** HIDS (Host based Intrusion Detection System) 方式は OS やアプリケーションが出力する情報に基づいて攻撃検知を行う．この分野に関する研究はすでに古くから行われている．例えば，STIDE はシステムコールの順序性を特徴量として攻撃検知を行う手法 [2] である．学習したシステムコールの順序と異なる順序のシステムコールが現れた場合，攻撃として検知する．
- **Alert Correlation:** Alert Correlation 方式では前述の NIDS，HIDS や脆弱性スキャナなどのアラートを関連づけることによってより確度が高いアラートを出力することを目的としている．関連付けの方法はアラートに含まれる URL あるいは CVE 番号が同一かどうかで判断している．例えば，Kruegel らの手法 [1] では脆弱性スキャナで一旦，Web アプリケーションや Web サーバに脆弱性が存在するバージョンを利用していないか確認し，その脆弱性に対する攻撃を検知した場合は有効な攻撃と判断する手法である．近年の SIEM (Security Information and Event Management) 製品でも同様の判定アルゴリズムを実装しているものも存在する．

### 2.3 課題

2.2 節で述べた方式では 2.1 節で述べた要件に対して様々な課題が存在する．既存方式を各要件で比較したときの課題点を表 1 に示す．NIDS 方式ではネットワーク通信のみを特徴として扱うため，Web サーバ内の動作を観測することができず，その攻撃が有効かどうかを判断することは困難である．HIDS 方式ではホスト内の挙動を観測できるため，攻撃の有効性を検知することは可能だが，提案されている既存手法では利用する情報量が限られているため学習が正確でなく，誤検知が多いという課題が存在する．この点については 4.1 節で詳しく述べる．Alert Correlation 方式では NIDS のアラート及び脆弱性スキャナや HIDS によるのアラートを関連づけて分析するが，Web アプリケーションあるいは Web サーバに脆弱性があるという情報を予め収集しておく必要がある．そのため，予め脆弱性の情報がない新たな攻撃や独自の Web アプリケーションで脆弱性が存在するかどうか把握できない場合にには対応できないという課題が存在する．つまり，3 つの方式それぞれに目標とする攻撃検知の要件に対して課題が存在する．

## 3. 提案手法

本稿では 2.1 節で述べた要件を満たす，新たな攻撃にも対応可能な攻撃の有効性を高精度に検知する手法 AETP (Application Event Tracking & Profiling) を提案する．AETP では Web アクセスに対してそのアクセスを処理する際の

[*1] https://www.snort.org
[*2] https://www.modsecurity.org

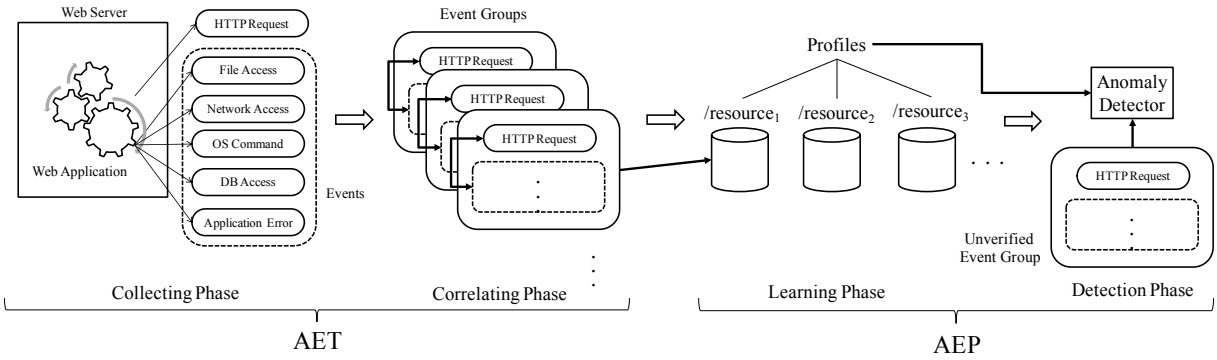


**図 1** AETP の各処理の概要
**Fig. 1** Overview of AETP

**表 1** 要件に基づく既存手法の評価
**Table 1** Conventional methodology evaluation

| | NIDS | HIDS | Alert Correlation |
|---|---|---|---|
| 有効な攻撃の判定 | ✗ | ✓ | ✓ |
| 新たな攻撃への耐性 | ✓ | ✓ | ✗ |
| 高精度な検知 | ✓ | ✗ | ✓ |

挙動を観測し，異常な挙動を示した場合，攻撃によってホストに影響を与え，攻撃が有効だったと判断する HIDS 方式の手法である．本稿では対象アプリケーション／サーバに対して有効性を示す攻撃を「有効攻撃」，有効性を示さない攻撃を「無効攻撃」と呼ぶ．AETP は大きく AET (Application Event Tracking) と AEP (Application Event Profiling) の 2 つの部分に分かれている．各処理の概要を図 1 に示す．AET では OS 及びアプリケーションからイベントを収集し，イベントの関連付けを行う．HTTP リクエストに関連づけられたイベントをイベントグループと呼ぶ．AEP ではまず，イベントグループを入力として Web アプリケーションの機能単位でプロファイルを作成する．そして，学習したプロファイルに対してアノマリ検知を適用することで，攻撃による有効性判定を行う．以下の各節で AET，AEP についてそれぞれ詳しく説明する．

### 3.1 Application Event Tracking

AET でのキーアイデアは，HTTP リクエストを Web アプリケーションが処理する時にホストで発生する様々なイベント収集する部分 (Collecting) と，収集したイベントを関連付ける部分 (Correlating) にある (図 1)．イベントの関連付けでは HTTP リクエストを処理した開始時刻，終了時刻，プロセスのスレッド ID (TID) やプロセス ID (PID) 等とファイルアクセスやコマンド実行のシステムコールのイベントから取得できる TID，PID や親 PID の関連性を用いて関連付けを行う．詳細についてはすでに鐘らが提案している手法 [3] を利用している．

### 3.2 Application Event Profiling

AEP は学習と検知の処理に分かれている．各節にてそれぞれの詳細を説明する．

#### 3.2.1 学習フェーズ

学習ではイベントを正規化し，正規化されたイベントへのタグ付け及びプロファイル作成の 2 つのステップが存在する．イベントの正規化を行う理由は誤検知を減らすためである．アノマリ検知では複雑なアプリケーションになればなるほど，学習時に全ての正常パターンを網羅することは困難となり，誤検知の多さが問題となる．そのため，イベントを正規化することで正常パターンの網羅性を高め，誤検知を減らすことが必要である．

イベントへのタグ付けとはそのイベントを表すタグを付与することであり，異なる種類のイベントを一意的扱うための手法である．この手法はログ分析の分野で広く利用されている方法 [4] である．以下にて，4 種類のイベント [3] に対して正規化及びタグ付け処理について詳細に説明する．

- **ファイルアクセス：**一般的に，Web アプリケーションがアクセスするファイルはライブラリや設定ファイルなどであるため，その傾向は大きく変化しない．そのため（アクセスタイプ，ファイルの絶対パス，アクセス権限）のタプルをイベントタグとする．以下にファイルアクセスのイベントタグのを示す．

$$\begin{aligned} & (\texttt{open}, \texttt{/var/www/index.php}, \texttt{read}) \\ \longrightarrow\ & \texttt{file}_x \end{aligned}$$

しかし，セッション変数／ユーザ設定を保持するファイルはユーザ毎に異なるため，それらのファイルパスについては正しく抽象化する必要がある．一般的にはセッション変数／ユーザ設定を保持するファイルのディレクトリ名は変化しないかつ少量であるため，ディレクトリ名で正規化するディレクトリは予め指定しておく．例えば PHP のセッション変数を保持するファイルについて以下のようなルールで抽象化する．

(open, /tmp/session/sess_1234, create)
⟶ (open, /tmp/session/*, create)
⟶ $\texttt{file}_x$

● **ネットワークアクセス：**プロキシ等のような Web アプリケーションでなければ，通信先は限定的であると考えられる．そのため（接続の種類，IP アドレス，通信ポート番号）のタプルをイベントタグとする．接続の種類は connect, accept などのネットワークアクセスに関係するシステムコールの名前である．以下に例を示す．

(connect, 127.0.0.1, 3306)
⟶ $\texttt{net}_x$

● **コマンド実行：**ネットワークアクセスと同様に，Web アプリケーションから利用するコマンドも限定的である．そのため （コマンド名）のタプルをイベントタグとする．以下に例を示す．

ping -c 4 192.168.1.1
⟶ (ping)
⟶ $\texttt{command}_x$

● **DB アクセス：**多くの Web アプリケーションは DB を利用してデータの保存を行っている．DB を利用する際は SQL と呼ばれる言語を通じて操作を行うため，SQL クエリの実行を監視することで DB への攻撃を観測できる．アプリケーションによって操作する DB のテーブルやカラムは様々であるため，SQL クエリのトークン文字列をイベントタグとする．この方法は既存の SQL クエリに対して異常検知を行う手法ら [5, 6] でもよく用いられる正規化方法である．以下に SQL クエリ発行に対するイベントタグの例を示す．

SELECT * FROM users WHERE LIMIT 1;
⟶ (SELECT FROM WHERE LIMIT)
⟶ $\texttt{db.query}_x$

● **実行時エラー：**攻撃によっては実行時にエラーを起こすものも存在する．実行時エラーイベントは（正規化されたエラーメッセージ）のタプルをイベントタグとする．エラーメッセージには PHP Warning のメッセージや Web アプリケーション独自のログ出力となるため，定まったパターンは存在しない．そのため，ヒューリスティックにエラーメッセージに含まれる数値，文字列等のマスキングを実施した．以下の例はアクセスされたファイルが存在しなかったことを示すエラーメッセージである．

File not exists: /var/www/favion.ico
⟶ (File not exists: *)
⟶ $\texttt{http.err}_x$

2 つ目のステップではプロファイルの作成を行う．1 つ目のステップにより，各 HTTP リクエストに関連するシステムイベントにタグが付与された状態になる．提案手法ではプロファイルを Web アプリケーション機能単位に作成する．一般的には Web アプリケーションの機能は URL のパス部分で表現されるが，近年の傾向として MVC モデルを利用した Web アプリケーションの場合，URL マッピング機能により URL のパス部分で機能単位が正しく表現されないため，ヒューリスティックに指定を行う必要がある．プロファイルを機能単位に作成する理由は検知精度の向上を期待できるためである．一般的に，アノマリ検知におけるプロファイル（学習モデル）は適切な粒度で作成される必要があり，粒度が粗く，複雑過ぎるプロファイルでは検知漏れが発生する可能性が高くなり，逆に粒度が低く，個別の事象のみを表すプロファイルでは誤検知を発生させる原因となる．そのため，過去の検知手法でも機能単位でプロファイルを作成するという仕組みはよく用いられる [7]．

ここでいうプロファイルは機能単位に集計されたイベントグループに含まれるイベントタグの重複無し和集合である．まず，各イベントグループ $G = \{e_i | 0 \leq i \leq n\}$ は機能 $p$ 毎に集計され，イベントグループの集合 $GS_p = \{G_{pj} | 0 \leq j \leq m\}$ と表現でされる．次に，$GS_p$ に含まれるユニークなイベントの集合 $E_p = \{e_k | 0 \leq k \leq n'\}$ を求める．このイベント集合 $E_p$ のヒストグラム $EH_p = \{h(e_k) | 0 \leq k \leq n'\}$ を機能 $p$ に対するプロファイルとする．以下にプロファイル $EH$ の例を示す．プロファイルには機能 /index.php と /edit.php について学習がされており，各機能を実装した際に起きるイベントのヒストグラムが含まれている．

$$EH = \begin{cases} \texttt{/index.php} = \begin{cases} \texttt{file}_1 & : 0.5 \\ \texttt{db}_1 & : 0.6 \end{cases} \\ \texttt{/edit.php} = \begin{cases} \texttt{file}_2 & : 0.3 \\ \texttt{db}_2 & : 0.4 \end{cases} \end{cases}$$

### 3.2.2 検知フェーズ

プロファイル作成時と同様にまず，HTTP リクエストに関連するシステムイベントを収集し関連付け，正規化，タグ付けを行う．さらにイベントグループから重複するイベントを取り除き，各イベントに対してプロファイルに含まれるそのイベントの出現確率を求める．イベントグループに含まれるイベントの内，プロファイルにおける出現確率が閾値 $\alpha$ 以下，イベントの個数が閾値 $\beta$ 以上の場合，攻撃による影響があり，攻撃が有効であると判定する．例えば，$\alpha = 0.0$，$\beta = 1$ と設定した場合（プロファイルに現れないイベントが 1 以上現れた場合），ある/index.php へのアクセス時のイベントグループに出現するイベントの出現確率が以下の場合，

$$\begin{cases} \texttt{file}_1 & : 0.5 \\ \texttt{command}_1 & : 0.0 \end{cases}$$

出現確率が閾値 $\alpha$ 以下のイベントが 1 つあり，閾値 $\beta$ 以上のため，攻撃が有効であったと検知する．

### 3.3 実装

AETP はイベントを取得する部分 AET と取得したイベントを関連付けして有効性判定を行う部分 AEP に分かれている．AETP システムの全体像を図 2 に示す．

AET はアプリケーション及び OS から情報を取得する必要があるため，ホストにカーネルモジュールを組み込むことで実現する．このカーネルモジュールの作成には Web アプリケーション及び OS から関数等の実行時の情報を一元的に取得し，扱うことができる SystemTap [8,9] による実装を採用した．

AEP の部分はサービスを行う Web サーバに影響を与えないよう別サーバとして実装した．AET によって出力されたイベントは syslog 等を通じてこのサーバに送られ，プロファイルを作成する．詳細な実装方法は，鐘らの論文 [3] から参照できる．

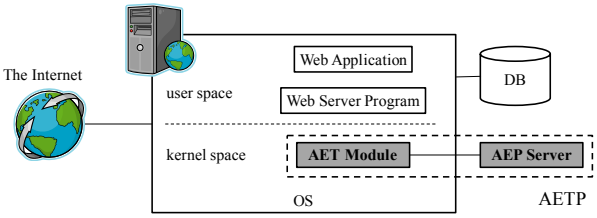


**図 2** AETP システムアーキテクチャ
**Fig. 2** AETP system architecture

## 4. 評価

AETP に対して判定精度及び処理性能の観点から評価を行った．

### 4.1 精度評価

まず最初に，AETP による攻撃の有効性判定の精度に関する評価を行った．精度評価に用いたデータセットを表 2 に示す．OpenCart はオープンソースのショッピングカートシステムであり，EC サイトを構築する際の一連の機能を有した Web アプリケーションである．2016 年 7 月現在，約 30 万の Web サイトで利用されている [*3]．精度評価には 4 種類の脆弱性を用意した (RCE：リモートコード／コマンド実行，SQLI：SQL インジェクション，DT：ディレクトリトラバーサル，FM：任意のファイルの操作)．学習に使用するデータは Web アプリケーションを正常に利用した際に，発生した HTTP リクエスト及びそれに関連づけられているイベントから収集した．機能をより網羅的にするため，人間によるアクセス以外にも，正常な URL をクロールしてデータを収集した．

検査に使用するデータには 3 種類あり，正常アクセス，有効攻撃及び無効攻撃の場合にそれぞれ分けられる．正常データは学習時と同様に生成した．有効攻撃のデータは実際に攻撃が成立する攻撃コードを Web アプリケーションに対して送信し，発生するイベントから収集した．実験に使用した有効攻撃は脆弱性のタイプにより攻撃シナリオが異なり（表 3），いずれも実際の攻撃者が過去に利用した攻撃コードから作成した．

**表 3** 有効な攻撃の攻撃シナリオ
**Table 3** Effective attack scenarios

| 脆弱性 | 攻撃シナリオ | 攻撃コード例 |
|---|---|---|
| RCE | ユーザ一覧の奪取 | `cat /etc/passwd` |
| SQLi | ユーザ一覧の奪取 | `UNION SELECT load_file('/etc/passwd');#` |
| DT | 設定ファイル一覧の奪取 | `../../../../etc/` |
| FM | ファイルの消去 | |
| | ファイルのコピー | |

無効攻撃のデータは Web アプリケーションの仕様あるいは脆弱でない箇所であることから攻撃が成立しない攻撃コードを送信し，発生するイベントから収集した．より多くの攻撃を包含するため，無効攻撃データの生成には脆弱性スキャナである w3af[*4] を使用した．w3af による脆弱性スキャンでは OS コマンドインジェクション，LFI，SQL インジェクション，eval 関数の悪用の 4 つの機能を有効にし，スキャンを行った．これらの攻撃が成功しないことは事前に確認ずみである．

精度評価に利用する指標は検知率及び誤検知率であり，有効と検知した有効攻撃数，無効攻撃数，正常アクセス数をそれぞれ $N_{de}$，$N_{di}$，$N_{dl}$ とし，データセットに含まれるの有効攻撃数，無効攻撃数，正常アクセス数をそれぞれ，$N_e$，$N_i$，$N_l$ とすると，検知率 $(DR)$，誤検知率 $(FPR_i, FPR_l)$ はそれぞれ $DR = \frac{N_{de}}{N_e}$，$FPR_i = \frac{N_{di}}{N_i}$ 及び $FPR_l = \frac{N_{dl}}{N_l}$ と定義される．誤検知率が 2 種類あるのは，無効攻撃と正常アクセスをそれぞれ検知してしまった場合が存在するからである．

AEP には検知時に設定する閾値 $\alpha$ と $\beta$ が 2 つあり，それぞれプロファイルにおける出現確率の閾値，及び閾値 $\alpha$ 以下のイベントの個数の閾値である．今回の評価では $\alpha = 0.0$，$\beta = 1$ と設定した．つまり，学習時に存在しなかったイベントが 1 つ以上発生した場合，有効な攻撃と判定する．

検知精度の評価結果を表 4 に示す．平均はデータの個数に依存しない加重平均を用いている．比較としてシステムコールの順序性から攻撃を判定する HIDS である STIDE [2] 及び，NIDS と脆弱性スキャナが出力するアラートを関連づけて判定する Kruegel らの手法 [1] についても評価を行った．STIDE はプログラムが公開されており，ニューメキシ

[*3] http://trends.builtwith.com/

[*4] http://w3af.org/

**表 2** データセット概要
**Table 2** Dataset overview

| アプリケーション名 | バージョン | 脆弱性 | 学習リクエスト数 | 検査リクエスト数 | | |
|---|---|---|---|---|---|---|
| | | | | 正常 | 無効攻撃 | 有効攻撃 |
| OpenCart | 2.2.0.0 | RCE | 3222 | 187 | 381 | 1 |
| | 1.5.6.1 | SQLI | 3134 | 201 | 381 | 1 |
| | 1.5.5.1 | DT,FM | 2784 | 257 | 381 | 3 |

**表 4** 判定精度（単位：%）
**Table 4** Detection accuracy

| | | | AETP | | | STIDE [2] | | | Kruegel et al. [1] | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| アプリケーション名 | バージョン | 脆弱性 | $FPR_l$ | $FPR_i$ | $DR$ | $FPR_l$ | $FPR_i$ | $DR$ | $FPR_l$ | $FPR_i$ | $DR$ |
| OpenCart | 2.2.0.0 | RCE | 0.00 | 16.1 | 100 | 4.81 | 40.2 | 100 | 0.00 | 0.00 | 0.00 |
| | 1.5.6.1 | SQLI | 1.00 | 16.4 | 100 | 16.6 | 93.7 | 100 | 0.00 | 0.00 | 0.00 |
| | 1.5.5.1 | DT, FM | 1.17 | 15.0 | 100 | 3.83 | 42.7 | 100 | 0.00 | 0.00 | 0.00 |
| 平均 | | | **1.01** | **15.8** | **100** | 8.41 | 58.8 | 100 | 0.00 | 0.00 | 0.00 |

コ大学の Web サイト [*5] からダウンロードできる．Kruegel らの論文では NIDS である Snort と脆弱性スキャナである Nessus を利用しているが，今回の実験ではオープンソースソフトである ModSecurity と OpenVAS[*6] を利用した．この理由は Web アプリケーションに対する攻撃検知に特化した ModSecurity の方がより攻撃を検知できる可能性を有し，コミュニティベースで対応アプリケーション数が多い OpenVAS の方が適切な評価に適していると考えたからである．

AETP は STIDE に比べ，誤検知率が低い．特に無効攻撃に関しては STIDE では 40%以上の誤検知率を示しているのに対して，AETP では約 15%の誤検知率であるため，著しく誤検知率を削減できている．この理由は，STIDE ではアプリケーションの処理単位を正しく識別できず，正確なプロファイルを表現出来ていないが，AETP ではアプリケーションの処理単位でプロファイルを作成する分，より適切なプロファイルを作成可能だからである．

また, Kruelgel らの手法では有効攻撃を全く検知できていない．この理由は OpenVAS では今回利用した OpenCart の脆弱性を判定出来なかったことにある．OpenVAS には複数の OpenCart の脆弱性有無判定プラグインがあり，実験では一つの脆弱性が OpenVAS によって検出されていた．表 5 に ModSecurity と OpenVAS によるアラート数を示す．ModSecurity では 1 つの HTTP リクエストに対して複数のアラートが発生するため，1 つの HTTP リクエストに対する複数アラートは 1 つと数える．ModSecurity では今回の実験で使用した有効攻撃を全て検知していた．また，OpenVAS の脆弱性判定では 1 つの脆弱性が発見されていた．しかし，この脆弱性は機微情報を公開してしまう脆弱性であり，アラートに含まれる URL あるいは CVE 番号によって ModSecurity のアラートと正しく関連づけることができなく，攻撃の有効性を判定出来なかった．そのため，有効攻撃に対して全てを見逃してしまうという結果となった．これは 2.3 節でも述べたように，Kruegel らの手法では既知の脆弱性に制約されてしまうという課題に起因するものである．

**表 5** ModSecurity と OpenVAS のアラート数
**Table 5** # of alerts of ModSecurity and OpenVAS

| バージョン | ModSecurity | | | OpenVAS |
|---|---|---|---|---|
| | 正常 | 無効攻撃 | 有効攻撃 | 脆弱性判定 |
| 2.2.0.0 | 0 | 140 | 1 | 0 |
| 1.5.6.1 | 0 | 140 | 1 | 1 |
| 1.5.5.1 | 0 | 140 | 3 | 1 |

以上のことから AETP は新たな攻撃に対しても正しく有効性を検知でき，さらに既存の HIDS と比較して平均，約 50% [*7] のアラートを削減できることを示した．

### 4.2 分析

次に，正しく有効性を検知できた場合及び，誤検知した場合の理由について分析する．

#### 4.2.1 正しく有効性を検知できた攻撃の分析

正しく攻撃による影響を検知できた例を 2 つ挙げ，検知できた理由について分析する．

- **OpenCart 2.2.0.0 RCE：** 一般的にコード実行の脆弱性では，通常発行されない OS コマンドが発行されるため，そのイベントが異常を引き起こす．OpenCart の場

[*5] http://www.cs.unm.edu/~immsec/systemcalls.htm
[*6] http://www.openvas.org/
[*7] $FP_l$ については $8.41 - 1.01 = 7.40\%$のアラートを削減，$FP_i$ については $58.8 - 15.8 = 43.0\%$のアラート削減した．

合では，独自実装の json_decode 関数に脆弱性があり，入力値を適切にサニタイズしないために，任意のコードを実行される脆弱性が存在する．実験ではこの脆弱性を悪用して system 関数を介して，攻撃を行った．そのときに検知されたイベントを表 6 に示す．番号 1 から 4 は system 関数を経由したときに発生するファイルアクセスであり，コマンド実行のための準備として様々なライブラリを読み込んでいることを表している．番号 5 はコマンド実行の内容が cat /etc/passwd であったことを示す．その後コマンド実行により目的とするファイルにアクセスがあったことがわかる（番号 6）

**表 6** OpenCart に対する RCE 攻撃によって発生した異常イベント
**Table 6** Anomaly events detected from OpenCart remote code execution attack

| 番号 | イベント種類 | イベント内容 |
|---|---|---|
| 1 | file | open, /lib64/libtinfo.so.5, O_RDONLY |
| 2 | file | open, /lib64/libdl.so.2, O_RDONLY |
| 3 | file | open, /lib64/libc.so.6, O_RDONLY |
| 4 | net | connect, AF_UNIX, /var/run/nscd/socket |
| 5 | command | cat /etc/passwd |
| 6 | file | open, /etc/passwd, O_RDONLY |

● **OpenCart 1.5.6.1 SQLI：** SQL インジェクションによる攻撃の場合，SQL クエリ文に異常，DB アクセスエラーが発生する可能性が高い．OpenCart 1.5.6.1 に対する SQL インジェクションは典型的な脆弱性であり，入力文字列をサニタイズしないため，SQL 文を挿入できてしまう．表 7 に AETP によって検知したイベントを示す．SQL 文のトークン並びに異常性が見られ，通常現れない UNION SELECT load_file() の部分が挿入されているため，検知された．

### 4.2.2 誤検知の分析

誤検知の主な原因は，学習時に存在しないイベントの発生である．影響なし攻撃を正常リクエストよりも多く誤検知する理由としては，正常な入力を行っていないため，より学習時に観測できなかったイベントが発生するからである．AETP では正常アクセスを誤検知する確率より無効攻撃を誤検知する確率の方が高い（表 4）この理由の原因は無効攻撃によるアプリケーションでのエラーの発生である．無効攻撃によって発生した一連の異常イベントを表 8 に示す．番号 1 と 2 のイベントはどちらもアクセスしている Web ページが見つけられないことによってのみ発生するファイルアクセスである．学習時には発生しないため，異常として検知されてしまう．

## 4.3 性能評価

性能評価では HTTP リクエストのスループット及びハードウェアリソースの使用量の評価を行った．AEP の部分は別ハードウェアで Web サーバからオフロードできるため，AET の部分による Web アプリケーションへの性能影響のみを計測した．性能評価に利用したマシンのスペックは 1 Intel Xeon 2.40GHz CPU，8GB RAM，128GB HDD，CentOS 7.1 である．

● **HTTP リクエストのスループット：**スループットは Apache JMeter[*8] を使って計測した．多重接続がある環境を想定して，各 Web アプリケーションのトップページに対して並列に 10 アクセス行いながら合計 1000 アクセスを 5 セット繰り返し，その平均値を求めた．表 9 にイベント取得モジュールを有効にした場合の性能低下率を計測した結果を示す．性能低下率 $D$ はイベント取得モジュールを有効にしていない場合の HTTP リクエストのスループットを $T_d$，イベント取得モジュールを有効にした場合の HTTP リクエストのスループットを $T_e$ として，$D = \frac{T_e - T_d}{T_d}$ で求めている．OpenCart の各機能とも遅延時間が 5ms/req 以内であり，約 10%程度の低下率である．例えば OpenCart の場合，トップページ表示には約 30 個程度の HTTP リクエストを行うため，ページが表示が完了するまでには約 $5 \times 30 = 150$ ミリ秒，つまり 0.15 秒の遅延しか発生しないことを意味している．Nielsen らの調査 [*9] によれば，1 秒程度の遅延まではユーザビリティに大きな影響を与えないことから AETP はユーザビリティの観点において実用的に問題ないと言える．

**表 9** AETP を使用した際の OpenCart の各機能の遅延時間及びスループット低下率
**Table 9** Performance overhead of request throughput

| 機能 | 遅延時間 (ms/req) | 低下率 (%) |
|---|---|---|
| トップページ表示 | 4.6 | 14.7 |
| 商品ページ表示 | 4.2 | 13.3 |
| ユーザログイン | 3.5 | 8.18 |
| 商品登録 | 4.7 | 11.6 |

● **ハードウェアリソース消費量：**最大メモリ使用量は，各 Web アプリケーションの 1000 アクセスを行う時に 1 秒毎に AET が利用している SystemTap のプロセスの実メモリ使用量を測定し，その最大値を求めた．最大 CPU の使用率は，1000 アクセスを行う時に 1 秒毎に AET が利用している SystemTap のプロセスの CPU 使用率を測定し，その最大値を求めた．平均ディスク使用量は，1000 アクセスを行う時に AET から出力されたログの圧縮後の容量の機能毎の平均で求めた．表 10 に AET のハードウェアリソースの消費量の一覧を示す．最大メモリ使用量は 53.5MB 程度，また CPU 使用率が 4%程度であるので現代の Web サーバのリソース状況から鑑みれば，実用可能

[*8] http://jmeter.apache.org/
[*9] https://www.nngroup.com/articles/website-response-times/

**表 7** OpenCart に対する SQLI 攻撃によって発生した異常イベント
**Table 7** Anomaly events detected from OpenCart SQL injection attack

| 番号 | イベント種類 | イベント内容 |
|---|---|---|
| 1 | db.query | root, opencart_1561, SELECT FROM WHERE UNION SELECT load_file() |

**表 8** OpenCart に対する無効攻撃によって発生した異常イベント
**Table 8** Detected events of ineffective attack on OpenCart

| 番号 | イベント種類 | イベント内容 |
|---|---|---|
| 1 | file | open, /catalog/controller/error/not_found.php, O_RDONLY |
| 2 | file | open, /catalog/language/english/error/not_found.php", O_RDONLY |

であると考える．ディスクの容量は Apache の通常のログフォーマットの Web アクセスログと比較すると約 500 倍になるが，今後，画像や CSS の取得といった攻撃となり得ないリクエストの除外や，イベントログに保持期間を設けることによって，ディスクを圧迫しないように工夫することが可能である．

**表 10** 最大メモリ使用量，最大 CPU 使用率及び平均ディスク使用量
**Table 10** Maximum memory usage, maximum CPU usage and average disk usage

| Mem (MB) | CPU(%) | Disk (MB/1000 reqs) |
|---|---|---|
| 53.5 | 9.3 | 4.3 |

## 5. 制約

AETP は以下の 3 つの観点において制約が存在する．

- **クライアントサイドへの攻撃：**AETP は攻撃によってホスト（サーバ）に発生する異常なイベントを検知する仕組みであるため，攻撃を行ってもサーバ側に影響を及ぼさないクライアントサイドへの攻撃（例えば，XSS や CSRF）は検知することができない．
- **ホストへの侵入：**攻撃によって攻撃者がシステム管理権限を取得出来た場合，AETP の機能を無効にされたり，イベントが取得できないように設定することが可能である．AETP カーネルモジュールに実行時の整合性チェックを行うことによって今後解決できると考える．
- **検知システムの脆弱性：**AETP の実装では SystemTap を利用しているため，攻撃者は SystemTap の脆弱性を悪用することで攻撃を行うことも可能である．そのため，脆弱性を予めパッチ等によって対処しておく必要がある．今回利用した SystemTap に関しては，最新の脆弱性は 2012 年に発見されたものであり，すでに安定したソフトウェアとなっている．

## 6. おわりに

Web アプリケーションの脆弱性に対する攻撃は日々発生しており，真に影響あるアラートの判断が課題となる．この課題に対して，既存手法では新たな攻撃に対応できない，あるいは検知精度が低いといった課題が存在する．本稿では，HTTP リクエストと，システムイベントを関連付け，一連のイベントの異常性を判断することで，攻撃の有効性を判断する AETP を提案した．さらにアノマリ検知を適用することで既存の攻撃情報に依存しない，新たな攻撃にも対応できるという強みがある．評価の結果，既存の HIDS の手法に比べ約 50%の正常アクセス及び無効攻撃によるアラートを削減できた．今後は更なる精度改良及びイベントログ量の削減を目指す．

### 参考文献


[1] Christopher Kruegel and William Robertson. Alert verification - determining the success of intrusion attempts. In *DIMVA*, 2004.

[2] Steven A Hofmeyr, Stephanie Forrest, and Anil Somayaji. Intrusion detection using sequences of system calls. *Computer Security*, 1998.

[3] 鐘揚, 谷川真樹, 大嶋嘉人. Web サーバへの攻撃影響特定のための高精度なイベント関連付け手法. In *SCIS*, 2016.

[4] T. Kimura, A. Watanabe, T. Toyono, and K. Ishibashi. Proactive failure detection learning generation patterns of large-scale network logs. In *Proceedings of the 11th International Conference on Network and Service Management (CNSM2015)*, 2015.

[5] William G. J. Halfond and Alessandro Orso. AMNESIA: Analysis and monitoring for neutralizing sql-injection attacks. In *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, 2005.

[6] Fredrik Valeur, Darren Mutz, and Giovanni Vigna. A learning-based approach to the detection of sql attacks. In *DIMVA*, 2005.

[7] Christopher Kruegel, Giovanni Vigna, and William Robertson. A multi-model approach to the detection of web-based attacks. *Computer Networks*, 2005.

[8] Vara Prasad, William Cohen, FC Eigler, Martin Hunt, Jim Keniston, and J Chen. Locating system problems using dynamic instrumentation. In *Ottawa Linux Symposium*, 2005.

[9] Frank Ch Eigler. Problem solving with systemtap. In *Ottawa Linux Symposium*, 2006.