Computer Security Symposium 2017
23 - 25 October 2017

Webアプリケーションのための攻撃検出と防御

野見山 賢人[1,a)]　小出　洋[2,b)]

概要：近年，情報システムに対しての攻撃手法が変化しており特定の組織を対象とした標的型攻撃が主流となっている． その一方で Web アプリケーション等の情報システムはクラウド環境等にも用いられるようになり，より大規模かつ複雑になっているため，運用されている情報システムのすべてを把握することはより難しい．そこで，運用されている Web アプリケーションを変更せず，Web アプリケーションと密に連携したセキュリティ機能を持つことが必須である．本研究は既存の Web アプリケーションをハニーポット化することで，Web アプリケーションを攻撃から保護しながら攻撃の検出と情報収集を可能にするシステムを提案する．

キーワード：ハニーポット, Web アプリケーション

Attack Detection and Defense for Web Applications

Kento Nomiyama[1,a)]　Hiroshi Koide[2,b)]

Abstract: Recent cyber attacks to information systems are more sophisticated and more complex. And attacks to organizations are becoming the mainstream. On the other hand, it is difficult to understand all complicated systems, because information systems consisted on web applications use cloud computing and other information technologies so that they become larger scale and more complex. Therefore, it is required to web application frameworks themselves have cyber security functions, because it is difficult to defend all complicated information systems from attacks. In this reseach, we propose a new attack detection method which makes existing web applications honeypots by appending security functions.

Keywords: HoneyPot, Web Application

1. はじめに

近年，情報システムに対しての攻撃手法が変化しており特定の組織を対象とした標的型攻撃が主流となっている．標的型攻撃は，メールの添付ファイルの実行や改竄されたWeb サイトの訪問等を通じてマルウェアを感染させる手法を始め，対象の情報システムの内部に侵入するために巧妙な攻撃手法が用いられている．特に Web サイトの改竄においてはクロスサイトスクリプティングや SQL インジェクションなどを用いて改竄を行い，改竄された Web サイトを利用者が訪問することでマルウェアを配布する悪性 Web サイトに誘導し，マルウェアを利用者端末にダウンロードするドライブ・バイ・ダウンロード攻撃 [1] に発展する場合もある．

その一方で Web アプリケーション等の情報システムはオンプレミス環境以外にも，一部クラウド環境等にも用いられるようになっている．一部をクラウド環境に移行することでオンプレミス環境のみと比較し，情報システムのシステム構成やネットワーク構成が，より大規模かつ複雑になっていることから，運用されている情報システムのすべてを把握することはより難しい．そのため，運用されている Web アプリケーションを変更せず，Web アプリケーショ

[1] 九州工業大学大学院情報工学府
Graduate School of Computer Science and System Engineering, Kyushu Institute of Technology
[2] 九州大学情報基盤研究開発センター
Reseach Institute for Information Technology, Kyushu University
[a)] nonoca@klab.ai.kyutech.ac.jp
[b)] koide@cc.kyushu-u.ac.jp

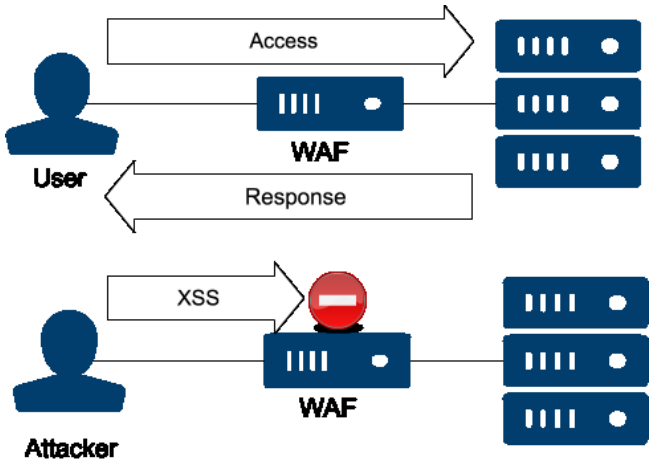

図 1　WAF の動作概要.

ンと密に連携したセキュリティ機能を持つことが必須である [2]．Web アプリケーションを変更せず既存のシステム以上にセキュリティ機能を満たすシステムとして，Web Application Firewall（WAF）[3] が挙げられる．WAF は Web アプリケーションをクロスサイトスクリプティングなどから保護するシステムである．WAF は POST メソッドや GET メソッドなどから得られるパラメータなどの情報から，攻撃の検出を可能にしており，攻撃を検出すると通信の遮断などの対応を行う．しかし他人のユーザになりすます不正ログインを行うことは，WAF では正常な通信として処理されてしまうため，全ての攻撃から Web アプリケーションを保護することは難しい．

本研究では，既存の Web アプリケーションを用いて攻撃者を誘導することにより，Web アプリケーションに対する攻撃手法の収集・分析を可能にすることを目的とする．その目的を達成するために，Web アプリケーションに疑似 API の機能を付加し，ハニーポット化することで攻撃検出を行う手法を提案する．ハニーポットはアプリケーションの脆弱性により攻撃者を誘い出し不正アクセスを促すシステムである．ハニーポットを利用することにより攻撃者の攻撃行動の観察や分析を行うことが可能となる．

2. Web Application Firewall

Web Application Firewall（WAF）[3] は，クロスサイトスクリプティングや SQL インジェクションなどといった，Web アプリケーションの脆弱性を悪用する攻撃から保護するシステムである．WAF を導入することで Web アプリケーションの保護及び攻撃の検出・遮断が可能となる．図 1 に WAF の動作概要を示す．

WAF を用いた攻撃の検出には，検出パターンに基づき通信のパラメータ値を比較するため，目視で判定する場合と WAF での判定の場合とでは異なった結果が生じる場合がある．そのため，悪意のある攻撃の検出が行われない場合や，正規のユーザが接続された際に誤検知され通信遮断される場合もある．そのような誤検知を防ぐために WAF ではブラックリストやホワイトリストを用いることで誤検知率を下げることができる．

ブラックリストは，通信のおける攻撃パターンを定義したリストである．ブラックリストを利用して検出を行うと，通信内容がリスト内のパターンと一致した場合に対象の通信を不正通信として判断する．最新の攻撃を検出する場合には随時リストの更新が必須となる．

ホワイトリストは，通信における正規パターンを定義したリストである．ホワイトリストを利用して検出を行うと，通信内容がリスト内のバターンに一致しない場合に対象の通信を不正通信として判断し，検出する．ホワイトリストの作成には，Web アプリケーションにおけるパラメータの設計に基づき作成し依存するため，Web アプリケーション毎にリストの作成を行う必要がある．

しかし，特定の利用者のみが利用可能な機能を全ての利用者に対して使用可能な状態になっている場合や，辞書攻撃やパスワードリスト攻撃といったブルートフォース攻撃を用いて，他人になりすまし不正にログインするような攻撃に対しては WAF では正常な通信として判断される．このため，すべての攻撃を WAF を用いて検出，保護することは難しい．

3. ハニーポット

ハニーポットはアプリケーションや OS に存在する脆弱性を利用することで，攻撃者を誘導し不正アクセスを促すシステムである．ハニーポットを用いることで，攻撃者の行動や攻撃手法といった情報を収集することが可能である．攻撃者からの接続により初めてハニーポットとしての役割を持つ対話型のハニーポットは，低対話型ハニーポットと高対話型ハニーポットの 2 種類に分類される．

3.1 低対話型ハニーポット

低対話型ハニーポットは特定のプロトコルや脆弱性を持ったアプリケーションをエミュレートして動作することで攻撃者をエミュレート環境に誘導し攻撃活動を促すハニーポットである [4]．低対話型ハニーポットの特徴として，攻撃者が接続したシステム上で行っている動作やコマンドはシステム側で擬似的にエミュレートされているものである．疑似コマンドを利用するため，攻撃者側の画面上では攻撃が成功しているように見えているがホストマシンへの影響はない．代表例として SSH ハニーポットの“cowrie”[5] や HTTP ハニーポットの“glastopf”[6] などが挙げられ，それらを統合したハニーポットとして“T-Pot”[7] が存在する．

3.2 高対話型ハニーポット

高対話型ハニーポットは実際の物理マシンや仮想マシン

上に脆弱性を持ったアプリケーションや OS を配置し動作させることにより，攻撃者に接続させ攻撃活動を誘導させるハニーポットである [4]．高対話型ハニーポットは低対話型ハニーポットとは異なり，実際のマシン上でアプリケーションや OS の動作を行う．そのため，より高度な攻撃手法の取得が可能になる一方，OS コマンドやアプリケーションの全機能を利用できることから，ファイルシステムの破壊やハニーポット自体が踏み台となるリスクもある．このことから高対話型ハニーポットを設置する上で攻撃者が他のシステムに影響を与えないようにネットワーク周りを考慮することが必須であるため，低対話型ハニーポットが主に利用されている．

4. 提案手法

4.1 概要

本研究においては攻撃検出を行うシステムを設置し，そのシステムがユーザからの HTTP リクエストを受け取った際にユーザ属性の状態に応じ，コンテンツ側で用意した API を実運用と疑似用に切り替えることで，攻撃手法の取得を行うシステム Hoppin[*1] を提案する．各システム及びコンテンツはすべて Linux コンテナ技術である Docker[4] を用いてシステムの設計を行っている．Docker を用いることにより，各コンテナ内で実行されているプロセスは他のコンテナに対しては影響を与えない．そのため，本研究で提案するシステムでは高対話型ハニーポットの機能を持っているが，低対話型ハニーポットと同等の安全性を担保している．

4.2 システム構成

Hoppin では，攻撃検知を行うリバースプロキシと収集した情報を閲覧，分析を行える管理システムの 2 種類である．なお管理システムに関しては攻撃者からの攻撃を受けないように接続可能な箇所を限定する．ハニーポットとして用意するコンテンツは複数個用意することで，そのうちの一つがハニーポットであると攻撃者が判断しても，接続しなおすことで他のコンテンツに接続され，引き続き攻撃活動を促すことが可能である．攻撃検出システム及びコンテンツ内で操作された SQL 文などの情報はデータベース上で記録され管理システムで確認することが可能である．図 2 に Hoppin のシステム構成を示す．

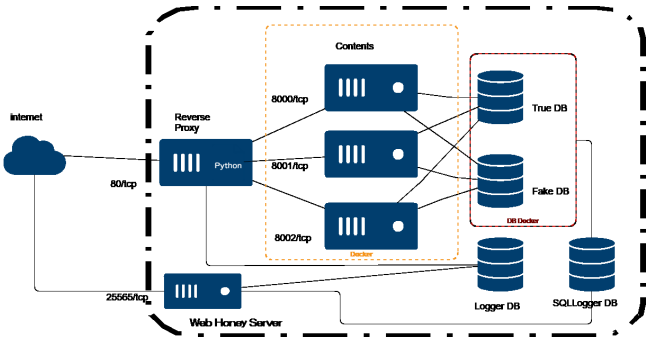

図 2 システム構成．

4.3 攻撃検出システム

攻撃者が Hoppin に接続を行う際に攻撃検出システムを介して各コンテンツに接続を行う．攻撃検出システムは接続時に接続元の IP アドレスをキーとして，ランダムに決定されたコンテンツを紐付けしセッション情報とともに

[*1] Honeypot in web application framework

```
{
  "Session": "da275c8d4ffd4d6a9eebfc1900e911ee
      ",
  "Port": 8000
}
```

図 3 紐付けたコンテンツのキャッシュ情報.

キャッシュを行う．キャッシュされる情報を図 3 に示す．

これにより，初回に接続した IP アドレスでシステムにより選択されたコンテンツに一定時間接続することが可能になる．ユーザ属性の状態は 2 種類あり，通常モードとユーザがシステムに対して攻撃を続けた際に，切り替えが生じる記録モードが存在する．コンテンツ接続時には，HTTP リクエストとレスポンスを確認し，認証が必要なページに認証なしで接続する場合に返されるステータスコード 403 やファイルが存在しない場合に返されるステータスコード 404 などが生じた場合，攻撃の疑いがあると推測し当該セッションのモード切替の値を増加する．この値は閾値を超えると通常モードから記録モードに変更するための値である．攻撃者が各コンテンツに用意した認証ページに対してブルートフォース攻撃により複数回認証に失敗する場合も同様にモード切替の値を増加させ，一定値を超えた場合にセッションに対して記録モードに移行し通信情報の記録を行う．

記録モードにシステムが移行した場合，接続を行う度に HTTP リクエスト情報の取得を行う．取得する情報としてはパス，接続メソッドを収集する．POST メソッドで接続した際にはパラメータの情報も合わせて収集を行う．収集した情報はデータベースに記録を行うが，パラメータに関しては POST メソッドで受け取るパラメータが接続するパス毎に異なるため，別途システム側で生成した一意の ID をキーとして定義し，NoSQL に格納することで異なるパラメータへの対応が容易となる．また API に接続を行う場合，コンテンツにリクエストを送信する前にコンテンツ側で用意されている，疑似用 API にパスの情報を書き換えて送信を行う．これにより攻撃者には通常の API に接続していると錯覚させることが可能となり，より高度な攻撃を誘導することが可能になる．

以上のことから Web Application Firewall では対応できないなりすまし攻撃の検出を行うことが可能になる．

4.4 コンテンツ

Hoppin で使用するコンテンツは管理者のみ接続が可能なエリアと API を用意する．コンテンツに接続されるデータベースは，通常運用中の場合に用いるデータベースと疑似運用中の場合に用いるデータベースの 2 種類が接続されており，テーブル構造はどちらも同一の構造が格納されている．これは攻撃者が SQL インジェクションを行った場合にシステム側に記録モードに移行になった際にテーブル構造の変化によりハニーポットだと判断されることを防止する役割がある．

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class APIHomeController extends Controller
{
    protected function ret_msg （）{
        return 'true API for Home';
    }

    public function index （ Request $req ）{
        return response （）->
        json （['msg'=>$this->ret_msg （）],
        200 ）;
    }
}
```

図 4 通常運用の Home API.

コンテンツ API の作成時に攻撃検出システムと連動を行うために通常運用用と疑似運用用の 2 種類の API の作成を行う必要がある．その際に設定するルーティング情報は攻撃検出システム側でパス情報が変更された状態でコンテンツに送信されるため統一を行う必要がある．以下は home という API を実行する場合の一例である．

通常運用用

/api/home

疑似運用用

/fake/api/home

通常運用用と疑似運用用の API は同一動作を行うため，同じデータ構造を用意する必要がある． 疑似運用用の API 作成時にはオブジェクト指向の手法を用いて，通常運用の API を継承することにより，作成を容易にすることが可能である．データベースを用いる場合に関してはデータベースへの接続を行うメソッドを作成した後にオーバーライドする形で実装することで通常運用のデータベースと疑似運用のデータベースに接続することが可能である．

図 4 に通常運用時の HomeAPI を，図 5 に疑似運用時の HomeAPI を示す．

これらの Home API を実際に実行を行った際のレスポンスとして，図 6 に通常運用時を，図 7 に疑似運用時を示す．

図 6，図 7 のように通常運用時と疑似運用時で同一のレスポンスにすることにより，API に直接接続する以外にフ

表 1　WAF との比較.

	Hoppin	WAF
クロスサイトスクリプティング	Possible	Yes
SQL インジェクション	Possible	Yes
なりすまし攻撃	Yes	No

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class FakeAPIHomeController extends
    APIHomeController
{
    protected function ret_msg  () {
        return 'Fake API Home';
    }
}
```

図 5　疑似運用の Home API.

```
{
  "msg": "true API for Home"
}
```

図 6　通常運用の Home API の実行例.

```
{
  "msg": "Fake API Home"
}
```

図 7　疑似運用の Home API の実行例.

ロントエンドからの Ajax による間接的な接続に対応することが可能である．また，フロントエンド側で呼び出した際の処理は通常運用も疑似運用も共通で利用可能であるためユーザインターフェースを崩さずに運用を続けることが可能となる．

5. 考察

5.1 レスポンス時間

コンテンツ API のレスポンス時間を本研究で提案している Hoppin を用いた場合と用いない場合の比較を行った．応答時間の計測には cURL を用いてエンドノードにて行い，Hoppin を用いた場合と用いない場合それぞれ 200 回 API に接続しその平均値の値を取った．計測した結果 Hoppin を用いた場合は 2.43705[s]，未使用の場合が 0.48455[s] となった．２つの場合を比較した際， Hoppin を用いた場合の計測値が用いない場合より約５倍の遅延が発生していることが分かった．遅延が生じた主な要因としては HTTP リクエスト及びレスポンスの書き換えが考えられる．Hoppin では HTTP リクエストを受け取った際にユーザ属性の現時点の状態に関係なく Host 情報の書き換えを行って送信している．また，レスポンスの際には受け取った際にシステム側で接続元のポート番号に書き換えを行っている．その際，リクエスト及びレスポンスの全文すべてを正規表現によるマッチングで書き換え位置を決定しており，その書き換えを終えた後ユーザに送信しているため遅延が生じていることが考えられる．

5.2 Web Application Firewall との併用

Hoppin と Web Application Firewall（WAF）はそれぞれで対策できる攻撃が異なる．表 1 に Hoppin と WAF との対策可能な攻撃の比較を示す．

表 1 より WAF ではクロスサイトスクリプティング及び SQL インジェクションなどのフォームやパラメータに対しての攻撃に有効ではあるが，現状の Hoppin では WAF で可能な攻撃の検出を行うことは不可能である．一方 Hoppin では，不正ログインを行うことで他人になりすまし各コンテンツを利用する，なりすまし攻撃の検出が可能である．そのため，既存の WAF と Hoppin は併用して使用することが可能であり，併用することでより多くの攻撃の検出を行うことが可能である．

6. おわりに

本研究では既存の Web アプリケーションに疑似用の API を付加することによりハニーポット化するシステム Hoppin を提案した．Hoppin を用いることにより Web Application Firewall では検出することができないなりすまし攻撃の検出を可能にした．また，疑似用の API を利用することによりコンテンツに対して影響を与えることなく攻撃者に攻撃活動を行わることが可能になった．

Hoppin は高対話型のハニーポットではあるが，Linux コンテナである Docker を用いることにより，コンテナ内で攻撃者の活動により稼働し続けることが難しい場合，容易に破棄，再起動が可能である．また，コンテナ間は独立しており他のコンテナに対して影響を与えないため，低対話型ハニーポットと同等の安全性を担保することが可能となった．

今後の課題として，次の 2 点が挙げられる．

6.1 実環境への配置・攻撃の収集

Hoppin は現在外部ネットワークから接続可能な環境に設置されていない．そのため，攻撃者が実際に接続を行い，攻撃を行うことは不可能である．またレスポンスの応答時間に関しても通常接続の約 5 倍の遅延が生じている．レスポンスの応答時間を通常接続並の時間に最適化を行い，実際のクラウド環境に Hoppin を設置を行う．設置後は攻撃者の攻撃手法を収集しつつ，記録モードに移行する値の調整を行うことで，一般ユーザの誤検知を防ぐことを行う．

6.2 Hoppin の有用性について評価

実環境に設置を行った後に Hoppin がハニーポットとして有用なのかどうかについて評価を行う．検討段階であるが評価指標としては，ハニーポットとして必要な機能が提供されているか，収集した情報に不足している点は存在しないか，既存の低対話型 HTTP ハニーポットである“glastopf”[6] と比較を行い攻撃活動が継続して行われているかの 3 点で評価を行うことを考えている．

参考文献

[1] 笠間 貴弘， 井上 大介， 衛藤 将史， 中里 純二， 中尾 康二，“ドライブ・バイ・ダウンロード攻撃対策フレームワークの提案.”，コンピュータセキュリティシンポジウム 2011 論文集，2011，3，pp.780 - 785 (2011)

[2] T.Ishikawa and K.Sakurai, “Parameter Manipulation Attack Prevention and Detection by Using Web Application Deception Proxy.”, Proc. IMCOM’17, pp.74:1-74:9 (2017)

[3] 独立行政法人情報処理推進機構,“Web Application Firewall（WAF）読本 改訂第 2 版第 3 刷.”, オンライン，(２０１７年７月１０日参照),
https://www.ipa.go.jp/files/000017312.pdf

[4] 山本 健太，齊藤 泰一，“Linux コンテナ技術を利用した SSH ハニーポットの提案と評価.”，コンピュータセキュリティシンポジウム 2016 論文集,2016，pp.770 - 776 (2016)

[5] cowrie，リポジトリ，（２０１６年１０月３日参照),
https://github.com/cowrie/cowrie

[6] glastopf，リポジトリ，（２０１７年５月３日参照),
https://github.com/mushorg/glastopf

[7] T-Pot: A Multi-Honeypot Platform，オンライン，（２０１６年５月３日参照),
http://dtag-dev-sec.github.io/mediator/feature/2015/03/17/concept.html