プラグインを用いたWordPressに対する攻撃検知と分析手法の提案

松本 悦宜[1]　寺岡 良真[1]　満永 拓邦[2]

概要：WordPress はオープンソースの Contents Management System であり，多数の Web サイトで利用されている．その反面，WordPress を狙う攻撃が継続的に発生しており，Web サイト改ざんなどの被害が多数報告されている．WordPress は運用の容易さやユーザ数の多さなどから共有サーバ上に展開されることがあるが，これらの環境ではサービスの特性上，一般的な Web サイトのセキュリティ対策として用いられる Web Application Firewall や IPS などを導入することは難しい．本稿では，共有サーバでも利用できる WordPress のプラグインを利用して，Web Application Firewall や IPS と同等な機能をもつ防御手法を提案する．また，被害の未然防止を目的に，プラグインにて検知した情報を共有し分析するプラットフォームも構築し有効性を評価する．

キーワード：Web, CMS, WordPress

Proposal for a Method to Detect and Analyze Attacks using a WordPress Plugin

Yoshinori Matsumoto[1]　Ryoma Teraoka[1]　Takuho Mitsunaga[2]

Abstract: WordPress is a popular CMS (Contents Management System), used by a large number of websites. WordPress has many users, and the number of attacks against the platform is increasing. Mitigations such as WAF (Web Application Firewall) and IPS products are useful to prevent attacks. We developed a flexible WordPress plugin to prevent attacks similar to the methods used by WAFs and IPSes because a WordPress plugin can be installed on WordPress hosted on shared servers. Furthermore, we developed a platform system to analyze the logs generated by our WordPress plugin. Finally, we evaluated the plugin and system.

Keywords: Web, CMS, WordPress

1. 序論

近年，社会的需要から Web サイトの高機能化や大規模化が進み，開発や運用に必要なリソースが増加している．そのため Web サイトの コンテンツ作成や管理を効率化を目的とした様々な手法が導入されている．このような状況の中，Web サイトを構築するうえで，それらの手法の一つに，CMS (Contents Management System) がある．CMS を使用することにより，開発時や更新時のコストを抑えることができる．また，他のベンダーや開発者が作成したものを拡張機能として追加することができ，自由度の高い Web サイトを開発することができる．

CMS は Web サイトを構築する際に用いるプログラミング言語ごとに，様々な製品が開発されている．中でも，最もユーザ数が多い CMS の一つとして　WordPress [*1] がある．WordPress はオープンソースライセンスで配布されて

[1] 株式会社神戸デジタル・ラボ
Kobe Digital Lab Inc.
[2] 東京大学
The University of Tokyo

[*1] WordPress
https://wordpress.org/

いる CMS であり，W3Tech の調査によれば，CMS の中で最もユーザ数が多いものとされている [1] ．WordPress は 2003 年に配布が開始され，ブログサイト用の CMS として使用されていた．その後，ブログのような動的に更新するコンテンツだけではなく，固定ページと呼ばれる静的なコンテンツの作成も実装され，企業のコーポレートサイトなどにも使用されるようになった．また，サードパーティー製のプラグインやサイトデザイン（テーマ）も多数公開されており，これらを用いることで，EC サイトやコミュニティサイトなど広い用途での普及も進んだ．

ユーザ数が多い反面，WordPress を使用して構築した Web サイトに対する攻撃も多くみられる．2017 年 2 月には，WordPress の REST API に深刻な脆弱性が発見され，攻撃が容易だった点も重なり，多くの Web サイトが改ざんされる事例があった．[2]

一般的にこれらの攻撃に対しては WAF（Web Application Firewall） を使用する場合がある．WAF は Web アプリケーションの攻撃に特化したファイヤーウォールで，Web サイトに対するリクエスト内容を評価し，攻撃と判断するとアクセスを遮断する機能をもつ．WAF を使用することで，これらの攻撃を遮断することができる．しかしながら，WordPress を使用した Web サイトでは，費用や運用の容易さから，共有サーバ上で構築・運用されることがある．この場合には共用サーバ上のアカウントの権限やネットワーク管理などの理由により WAF の導入が容易ではない．共有サーバによっては，企業がサービスの一環として WAF を導入しているところもある．しかしながら，企業側で導入している WAF はユーザの管理外にあり，効果の評価など関しては管理することはできない．

本稿では，WordPress プラグインを使用して，WordPress に対する攻撃を検知し遮断する手法を提案する．また，これらを Elasticsearch を使用して分析するシステムを構築した．これにより，新しい攻撃の傾向分析などに応用が期待される．

2. 関連研究

従来より，Web サイトに対する攻撃手法としては，クロス・サイト・スクリプティング攻撃などがあげられるが，これらの攻撃は，Web サイトに対するリクエストに攻撃用のコードが含まれることで発生する [3] ．

これらの攻撃に備えるために，具体的な攻撃情報を把握することが必要になる．従来では，攻撃情報を把握するために，ハニーポットのようなシステムをインターネット上に設置し，システムが受け取るパケットを収集し分析する方式がよく用いられてきた [4] ．

しかし，WordPress に対する攻撃を収集する場合は，特定の WordPress に対する攻撃は十分に収集できないと考えられる．これは，特定の WordPress に対する攻撃は，攻撃者が攻撃対象サイトを WordPress を使用した Web サイトであると判断していると考えられるためである．

松本らは，これらの攻撃を収集するために，ログの収集を行うための Web サーバとして，3 つのサーバを設置した [5] ．これらのサーバは WordPress をインストールした．サーバに対するアクセスを増やすために，コンテンツを定期的に自動更新するようにした．ここで更新するコンテンツは，外部の Web サイトから情報収集を行い，これらの情報を元にコンテンツを作成するスクリプトを開発した．この時，それぞれのサーバごとに異なるコンテンツが作成されるようにした．作成したコンテンツは，WordPress の XMLRPC を取り扱う機能を使用して更新した．

攻撃の各種検証に関しては，リクエスト情報が必要であるため [6] ，それぞれのサーバにはリバースプロキシを設置し，アクセスログを収集した．ここでは，アクセス元の IP アドレス，タイムスタンプ，アクセス先 URL，POST リクエストのボディ部分に関して収集した．

収集の結果，WordPress に対する攻撃は主として以下の項目に分類されることがわかった．

- WordPress のログイン画面に対する総当たり攻撃
- 拡張機能（テーマ・プラグイン）の脆弱性を使用する攻撃

2.1 WordPress のログイン画面に対する総当たり攻撃

WordPress には，記事の編集や拡張機能の管理などを行うため，ユーザを認証するログイン画面が用意されている．初期設定において，ログイン画面の構造および URL パターンは WordPress の仕様で決められている．この構造は，主に ID とパスワードを POST リクエストで送るだけであり，リクエスト構造は非常に単純なものになっている．また，初期設定においてトークンなどによる送信元の確認や，ログイン試行回数の上限がないことから，ログイン時の POST リクエストを大量に試行する総当たり攻撃が行われる可能性が高い．

2.2 拡張機能（テーマ・プラグイン）の脆弱性を使用する攻撃

WordPress には，トップページ，各記事，その他の静的ページの外観を設定するテーマや，機能を追加するプラグインなど，Web サイトごとに自由度の高い開発が可能になる拡張機能がサポートされる．また，これらのテーマやプラグインなどを配布または販売する場合もあり，1 つの Web サイトで複数のサードパーティ製の製品が使用されていることがある．しかしながら，セキュリティ対策はそれぞれの拡張機能の開発者に委ねられており，一部の製品では，深刻な脆弱性を作り込んだまま配布され続けているものも見られる．このような状況から，テーマ・プラグインの脆弱性を使用する攻撃や，脆弱性のあるプラグインの

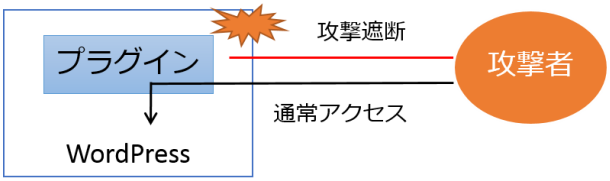

図 1　プラグイン概要
Fig. 1　Overview of the plugin.

```
if (strpos($namespaces, 'wp/v2/posts/') === 1)  {
  $request_body = $request->get_body();
  $request_arr = json_decode($request_body);
  $blog_version  = get_bloginfo('version');
  $vulnerable = array(
      "4.7", "4.7.1"
  );
  if (in_array($blog_version, $vulnerable))  {
     die('rest_api_attack');
  }
}
```

図 2　REST API 防御コード
Fig. 2　Protectio REST API

使用有無を調査するようなスキャンが確認されている．

3.　方法

3.1　WordPress プラグイン

Web サイトへの攻撃を検知し遮断する WordPress プラグインの開発を行った．作成したプラグインの概要は図 1 で示す．プラグインは WordPress と同様に PHP で作成する．基本的には一般的な PHP のシステムとして使用するが，WordPress が定義するのプラグイン API，アクションフックなどの機能を使用することで，WordPress の機能を一部使用することができ，これらを使用してリクエストを取得した [7]．

本システムでは，以下の攻撃を想定して実装した．

- WordPress の脆弱性を使用する攻撃
- WordPress のログイン画面に対する総当たり攻撃
- 拡張機能（テーマ・プラグイン）の脆弱性を使用する攻撃

WordPress の脆弱性を使用する攻撃では，「WordPress の REST API の wp-includes/rest-api/endpoints/class-wp-rest-posts-controller.php における任意のページを変更される脆弱性」（CVE-2017-1001000，以下 REST API の脆弱性と表記する）[8] に対する攻撃を検知し，アクセスを遮断するようにした．ここで実装したコードの概要を図 2 に示す．まず URL から REST API の機能があるかどうか調べ，現在使用している WordPress のバージョンと比較し，脆弱性のあるバージョンを使用している場合はアクセスを遮断する．

WordPress のログイン画面に対する総当たり攻撃では，単位時間内のログイン試行回数が閾値を超えた場合，総当たり攻撃とみなし，アクセスを遮断するようにした．ここでは wp_authenticate と呼ばれるフックを使用しており，ログイン時に呼び出す関数を追加することができる．このフックを使用すると，ログイン試行が行われた ID とパスワードを取得することができる．遮断したアクセスは，後述する分析システムに送信するようにした．

テーマ・プラグインに関しては，URL からどのテーマ・プラグインに対してのアクセスか確認し，使用しているプラグインであれば，プラグインのバージョンを確認し，公式リポジトリと比較して古い場合は攻撃とみなしアクセスを遮断するようにした．また，使用していないプラグインに対してもスキャンが行われる可能性があるため遮断するようにした．ここでは parse_request のと呼ばれるフックを使用しており，リクエストが送られたタイミングで，リクエスト内容が評価できるため，テーマ・プラグインの URL かどうかを調べることができる．遮断したアクセスは同様に，後述する分析システムに送信するようにした．

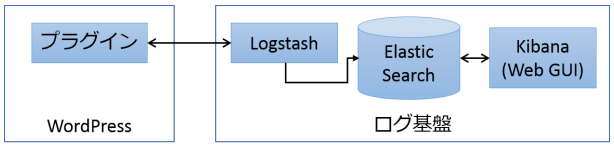

図 3　分析システム概要
Fig. 3　Overview of analysis system.

3.2　分析システム

収集した攻撃を分析するシステムを構築した．システムの概要を図 3 に示す．Web サイトから得られた攻撃を Logstash [*2] を用いて Elasticsearch [*3] に保存するシステムを構築した．Elasticsearch の内容は GUI で確認できるようにした．Elasticsearch はオープンソースライセンスで開発されている検索エンジンである．大規模なログの検索などを行うことができる．ログの検索などは Elasticsearch が提供する REST API を使用することで検索などの操作を行うことができる．Logstash はオープンソースライセンスで開発されているログ収集管理ツールである．ログの変換を行い Elasitcsearch へ送信する役割をもつ．

4.　評価

本システムの評価手法について解説する．

4.1　WordPress のインストール

評価では，WordPress を使用した Web サイトを使用するが，ここでの Web サイトは原則として WordPress インストール直後のものを使用する．WordPress のバージョ

[*2] Logstash
https://www.elastic.co/jp/products/logstash
[*3] Elasticsearch
https://www.elastic.co/jp/products/elasticsearch

```
POST http://domain/wp-json/wp/v2/posts/1/?id=1abc HTTP/1.1
Content-Length: 21
Content-Type: application/json
Connection: close
User-Agent: Python-urllib/2.7
Host: domain

{"content": "test\n"}
```

図 4　REST API 検証コード
Fig. 4　PoC code for the REST API vulnerability.

```
POST http://domain/wp-login.php HTTP/1.1
Host: domain
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0)\
Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml+xml,\
application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 103

log=admin&pwd=admin
```

図 5　ログイン時のリクエスト
Fig. 5　Request for login.

ンは評価時の最新版である「4.8.1」 のものを使用したが，「WordPress の脆弱性を使用する攻撃」では「4.7.1」を使用した．

4.2　WordPress の脆弱性を使用する攻撃

WordPress の REST API の脆弱性を使用した攻撃で行われるリクエストを再現した．ここでの PoC コード（検証用コード）は Sucuri 社の分析結果 [9] を基に作成した．リクエストの送信先の WordPress のバージョンは，本脆弱性の影響を受ける「4.7.1」を使用した．また，動作の安定性を確認するため，影響を受けないバージョン「4.7.2」を使用し，両バージョンにおいて，プラグインを有効化した場合，無効化した場合のそれぞれについて検証した．検証用コードは，図 4 のようにリクエストを送信した．

4.3　WordPress のログイン画面に対する総当たり攻撃

WordPress に対してログインを行う際には図 5 のようなリクエストを送信する．POST パラメータにある「log」にログイン用の ID，「pwd」にログイン用のパスワードを指定する．開発したプラグインが，ログイン試行のリクエストを攻撃と判断する閾値として，「1 秒間に 10 回以上のログイン試行」と設定した．このログイン試行を超えるものは総当たり攻撃と判断して，アクセスを遮断するように行った．

本検証では，図 5 で示すようなリクエストを効率的に送信するために WPScan というツールを使用した．WPScan[*4] は Sucuri 社が開発しているオープンソースのスキャンツールで，スキャンの内容が WordPress に特化しているのが特徴である．WordPress のセキュリティチェック時の疑似攻撃などに使用される．WPScan は主に以下の機能を持つ．

- ユーザ名の推測
- バージョンの推測
- WordPress のアカウントに対する総当たり攻撃
- テーマに対するスキャン
- プラグインに対するスキャン
- 特定のファイル（timthumb）のスキャン

この中から，WordPress のアカウントに対する総当たり攻撃機能を使用した．総当たり攻撃のパスワードリストは WPScan では搭載されていないため，パスワード解析ツールなどを配布している Openwall Project が作成したパスワードリスト [*5] に記載されているパスワード 3,546 件を使用した．

4.4　拡張機能（テーマ・プラグイン）の脆弱性を使用する攻撃

テーマ・プラグインに対する攻撃に関する検知機能の評価方法として，WPScan を使用してスキャンを行い検証した．本項目の検証は，WPScan のテーマ・プラグインへのスキャン機能を用いた．対象となるテーマ・プラグインは，2017 年 8 月 22 日現在において WPScan に搭載されているテーマ・プラグインのリストから，過去に脆弱性が発表されているもの，テーマ 281 種類，プラグイン 1,545 種類に対して行った．

5.　結果

5.1　WordPress の脆弱性を使用する攻撃

WordPress の REST API の脆弱性の攻撃を行った結果，影響のあるバージョンは図 6 のようなレスポンスが送信された．一方で影響のないバージョンでは図 7 のようなレスポンスが表示された．影響のあるバージョンおよび影響のないバージョンに関して，プラグイン有り・無しの場合をそれぞれ検証した結果，表 1 のようになった．

表 1　REST API の脆弱性の検証結果
Table 1　Result of the REST API vulnerability.

	Plugin なし	Plugin あり
4.7.1	影響あり	影響なし
4.7.2	影響なし	影響なし

[*4] WPScan
https://wpscan.org/
[*5] 使用したパスワードリスト
http://download.openwall.net/pub/wordlists/passwords/password.gz

```
HTTP/1.1 200 OK
Date: Wed, 16 Aug 2017 02:33:04 GMT
Server: Apache/2.4.10 (Debian)
X-Powered-By: PHP/5.6.30
X-Robots-Tag: noindex
Link: <http://domain/wp-json/>;\
rel="https://api.w.org/"
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total,\
X-WP-TotalPages
Access-Control-Allow-Headers: Authorization,\
Content-Type
Allow: POST, PUT, PATCH, DELETE
Content-Length: 1624
Connection: close
Content-Type: application/json; charset=UTF-8
```

図 6 影響のあるバージョンでのレスポンス（ヘッダ部分のみ）

Fig. 6 HTTP response (only header) from an affected WordPress version.

```
HTTP/1.1 400 Bad Request
Date: Wed, 16 Aug 2017 02:52:20 GMT
Server: Apache/2.4.10 (Debian)
X-Powered-By: PHP/5.6.30
X-Robots-Tag: noindex
Link: <http://192.168.99.100/wp-json/>;\
rel="https://api.w.org/"
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total,\
X-WP-TotalPages
Access-Control-Allow-Headers: Authorization,\
Content-Type
Allow: GET
Content-Length: 133
Connection: close
Content-Type: application/json; charset=UTF-8
```

図 7 影響のないバージョンでのレスポンス（ヘッダ部分のみ）

Fig. 7 HTTP response (only header) from an NOT affected WordPress version.

5.2 WordPress のログイン画面に対する総当たり攻撃

WordPress のログイン画面に対して総当たり攻撃を行った結果を表 2 に示す．ここで検知したものはログ分析システムに送信を行った．

表 2 ログイン試行の検証結果

Table 2 Result of login attempt requests.

レスポンス	ログイン試行回数
ログイン成功	0
ログイン失敗	13
アクセス遮断	3,533

5.3 拡張機能（テーマ・プラグイン）の脆弱性を使用する攻撃

WordPress のテーマ・プラグインに関してスキャンを行った結果を表 4 に示す．ここで検知したものは同様に，ログ分析システムに送信を行った．

表 3 スキャン結果

Table 3 Result of scan.

	送信件数	検知数
テーマ	281	281
プラグイン	1,545	1,545

5.4 ログ分析システム

ログを分析するシステムを作成した．ログの収集範囲は，5.2 および 5.3 の検知した内容に関して行った．上記スキャン後，収集したデータをログ分析システムに送信した．この結果，データ数は以下の通りになった．

表 4 スキャン結果

Table 4 Result of scan.

	検知数	ログ
ログイン試行	3,546	3,546
テーマ・プラグイン試行	1,826	1,826

6. 考察

6.1 WordPress の脆弱性を使用する攻撃

結果から，REST API の脆弱性を使用する攻撃に対して，検知および当該アクセスの遮断を行うことができた．しかし，本脆弱性以外の WordPress の脆弱性は現状のシステムでは検知できないため，今後はこれらの脆弱性に対応する手法が必要になる．

6.2 WordPress のログイン画面に対する総当たり攻撃

結果から，総当たり攻撃を検知および遮断することができた．「1 秒間に 10 回以上のログイン試行」としていたが実際 13 回行われていた．これは，リクエストを送信する際の速度や，サーバでの処理の関係によるものと考えられる．攻撃と検知したパスワードに関してはログの分析システムに送信することができた．しかし，実環境で使用する場合は，攻撃かどうかの判別手法を慎重に検討することが必要である．また，ログ分析システムにログイン試行された際に使用されたパスワードを送っているが，正規ユーザのミスタイプや，ログイン ID と同一のものなどが含まれている可能性があるため，複数の Web サイトを想定しているログ分析システムへ送信する場合は，送信可否を詳細に設定する機能が必要になる．

6.3 拡張機能（テーマ・プラグイン）の脆弱性を使用する攻撃

結果から，脆弱性のあるテーマ・プラグインに対するスキャンを検知・遮断することができた．このことから，既

知の脆弱性を探索するリクエストを遮断することができたと考えられる．実際の攻撃もスキャンに類似することが多いため [5]，本提案により，攻撃も検知・遮断すると推測できる．しかしながら，一部の製品は脆弱性が公開されても修正されないこともあり，この場合は製品個々に対策が必要になる．また，実環境で使用する場合は，従来の WAF 製品と同様に，使用している製品ごとに正常動作が行えるかどうかの確認が必要になる．検知したものはログ分析システムへの送信を行うことが可能になった．このことから，攻撃やスキャンの傾向を確認することができ，今後，攻撃者に狙われやすい製品などを早期発見することが期待できる．

プラグインが検知したリクエストを分析するためのログ分析システムを作成した．本システムは複数の Web サイトからのログ送信も想定しているため，今後は複数サイトからログを集めることを目指す．ログ収集の観点では，従来は Web サーバに市バースプロキシなどの専用の設定を行い攻撃を収集していたが，プラグインによる収集手法を確立することで，様々な環境で動作する WordPress の攻撃情報の収集が期待できる．

7. 結論

本提案では，WordPress に対する攻撃の検知および収集の手法として，共用サーバのサービスなど，様々な環境で使用できるように WordPress のプラグインを採用した．

想定する攻撃は，「WordPress の脆弱性を使用する攻撃」，「WordPress のログイン画面に対する総当たり攻撃」，「拡張機能（テーマ・プラグイン）の脆弱性を使用する攻撃」を対象として開発した．開発したシステムと WordPress の検証用 Web サイトを使用して，性能評価を行った．

「WordPress の脆弱性を使用する攻撃」では，WordPress で発見された REST API の脆弱性を使用した攻撃を検知して遮断するようにした．評価では，同脆弱性の検証コードを用いてリクエストを送信し，遮断することを確認できた．

「WordPress のログイン画面に対する総当たり攻撃」では，WordPress のログインフォームに対して単位時間内のログイン試行回数が閾値を超えた場合，総当たり攻撃とみなし検知・遮断するようにした．評価では，スキャンツールを用いて，同様のリクエストを送信し，遮断することを確認した．また遮断したリクエストはログとして保存し，ログ分析システムに送信することを確認した．

「拡張機能（テーマ・プラグイン）の脆弱性を使用する攻撃」では，URL からどのテーマ・プラグインに対してのアクセスか確認し，使用しているプラグインであれば，プラグインのバージョンを確認し，バージョンが最新ではないと確認された場合は，攻撃とみなしアクセスを遮断するようにした．評価では，スキャンツールを用いて，同様のリクエストを送信し，遮断することを確認した．また遮断したリクエストはログとして保存し，ログ分析システムに送信することを確認した．

以上の性能評価から，本提案で開発したシステムは，WordPress に対する攻撃の検知および遮断に有用であると考えられる．今後は，これらの検知精度の向上や，実際に運用している WordPress でも使用できるようにログ出力のレベルや送信の有無を設定する機能などを実装する必要がある．

また，プラグインで検知したものはログとして保存し，これらのログを分析するシステムを使用した．ログの分析を通じて，WordPress の攻撃傾向などを検討できるシステムを構築した．今後は複数の Web サイトを使用して，大規模な攻撃収集システムを構築し，攻撃傾向の早期把握，WordPress 運用者やセキュリティ研究者などへの情報提供に活用したい．

参考文献

[1] W3Techs : Usage of content management systems for websites,
入手先 ⟨https://w3techs.com/technologies/overview/content_management/all⟩ (2017.08.15).

[2] JPCERT/CC : JPCERT/CC 活動概要 [2017 年 1 月 1 日〜2017 年 3 月 31 日]
入手先 ⟨https://www.jpcert.or.jp/pr/2017/PR20170413.pdf⟩ (2017.08.15).

[3] 林昌吾, 松浦幹太スクリプト言語によるオブジェクト指向の *WEB* アプリケーションにおける *XSS* 攻撃脆弱性に対するクラスキャッシュを用いた静的解析, 暗号と情報セキュリティシンポジウム (2017).

[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).

[5] 松本悦宜, 三木剛, 力宗幸男 *CMS* に対する攻撃情報の収集方式の実装と評価, コンピュータセキュリティシンポジウム (2014).

[6] 鐘揚, 朝倉浩志, 高倉弘喜, 大嶋嘉人 *Web* アプリケーションのパラメタを悪用する攻撃のアノマリ検知手法, コンピュータセキュリティシンポジウム (2014).

[7] WordPress Codex 日本語版 : プラグイン API/アクションフック一覧
入手先 ⟨https://wpdocs.osdn.jp/プラグイン_API/アクションフック一覧⟩(2017.08.15).

[8] JVN iPedia :
JVNDB-2017-002318
WordPress の REST API の wp-includes/rest-api/endpoints/class-wp-rest-posts-controller.php における任意のページを変更される脆弱性
入手先 ⟨http://jvndb.jvn.jp/ja/contents/2017/JVNDB-2017-002318.html⟩ (2017.08.15).

[9] Sucuri :
Content Injection Vulnerability in WordPress 入手先 ⟨https://blog.sucuri.net/2017/02/content-injection-vulnerability-wordpress-rest-api.html⟩ (2017.08.15).