DebianでWAF

ModSecurity-nginx 1.0.1による
Web Application Firewall

Kentaro Hayashi

ClearCode Inc.

2021年2月 東京エリア・関西合同Debian勉強会

スライドは公開済みです

- この資料はRabbit Slide Showで公開済み
  - DebianでWAF - ModSecurity-nginx 1.0.1による Web Application Firewall
    - https://slide.rabbit-shocker.org/authors/kenhys/-tokyodebian-modsecurity-20210220/

プロフィール

- ひよこDebian Developer
  - 2020/09になったばかり
  - twitter: @kenhys
- トラックポイント(ソフトドーム派)
- わさビーフ(わさっち派)

本日の内容

- WAFとは
- 自由なWAF
- ModSecurityを試すには
- **注:誤りについては指摘してもらえると助かります**

WAFとは

- Web Application Firewall
  - Webサイトへの攻撃の影響を低減するためのもの
  - 攻撃と判断した通信を遮断する

WAFの提供形態

- アプライアンス型(専用機器設置)
- サービス型(外部提供のWAFサービスを利用)
- **ソフトウェア型** (今回はコレ!)

WAFへの理解を深めるには

- 「Web Application Firewallの導入に向けた検討項目」
  - https://www.ipa.go.jp/files/000072484.pdf
- 「Web Application Firewall 読本」
  - https://www.ipa.go.jp/files/000017312.pdf

自由なWAF

- Shadow Daemon - GPL-2.0 or later
  - https://shadowd.zecure.org/
  - 言語ごとのconnectorをインストールする
- NAXSI(Nginx) - GPL-3.0
  - https://github.com/nbs-system/naxsi
- **ModSecurity(Apache/Nginx)** Apache-2.0
  - https://modsecurity.org/

ModSecurity in Debian

- For Apache
  - libapache2-mod-security2
  - ModSecurity 2.x
- For Nginx
  - パッケージはない
  - https://github.com/SpiderLabs/ModSecurity-nginx
    - 最新版は1.0.1 (ModSecurity 3.0.3以降)

ModSecurity For Apache

```
$ sudo apt install apache2 modsecurity-crs \
            libapache2-mod-security2
$ sudo a2enmod security2
$ sudo cp /etc/modsecurity/modsecurity.conf-recommended \
     /etc/modsecurity/modsecurity.conf
$ sudo systemctl restart apache2
```

- DetectionOnlyで有効になる
- /var/log/apache2/modsec_audit.log にログ

ModSecurity For Nginx

- モジュールをビルドする
- ビルドしたモジュールを配置する
- Nginxの設定ファイルを書く

モジュールをビルドする準備(unstable)

```
$ sudo apt build-dep nginx
$ apt source nginx
$ git clone https://github.com/SpiderLabs/ModSecurity-nginx.git
$ vi nginx-1.18.0/debian/rules
```

- xxx_configure_flagsにオプション追加する
  - --add-dynamic-module=$(CURDIR)/../ModSecurity-nginx

モジュールをビルド

```
$ cd nginx-1.18.0
$ debuild -us -uc
$ find nginx-1.18.0 -name '*security*.so'
nginx-1.18.0/debian/build-xxx/objs/ngx_http_modsecurity_module.so
```

- xxxはcoreやlight、extrasのパッケージに対応
  - nginx-core
  - nginx-light
  - nginx-extras

モジュールをコピー

- ngx_http_modsecurity_module.so
  - コピー先は/usr/lib/nginx/modules

Nginxで有効にするには (nginx.conf)

```
load_module "modules/ngx_http_modsecurity_module.so";



modsecurity on;
modsecurity_rules_file /etc/nginx/modsecurity.conf;
```

- モジュールを読み込む
- ModSecurityを有効にする
  - 3.x向けの記述方法
  - ModSecurityEnabled onは2.x向けの記述

Firewallのルールを指定する

- ModSecurityはFirewallの枠組み
- ルールは追加で指定する必要がある
  - ModSecurity Core Rule Set

modsecurity-crs

- OWASP ModSecurity Core Rule Set
  - https://github.com/coreruleset/coreruleset
  - コミュニティベースでメンテナンス(になった)
  - ApacheとNginxとで共通で使えることになっている
- /usr/share/modsecurity-crs/rules/*.conf

設定の注意 for Nginx

- /etc/modsecurity/modsecurity.conf-recommended
  - libapache2-mod-security2向け
  - SecRequestBodyInMemoryLimit は3.xでは使えないやつ
- ログを別に出力する設定をおすすめ
  - SecAuditLog /var/log/nginx/modsec_audit.log
  - 既定だとエラーログに吐かれる

ブロックさせるサンプル

```
SecRuleEngine On
Include owasp-modsecurity-crs/crs-setup.conf
Include owasp-modsecurity-crs/rules/REQUEST-901-INITIALIZATION.conf
Include owasp-modsecurity-crs/rules/REQUEST-942-APPLICATION-ATTACK-SQLI.conf
Include owasp-modsecurity-crs/rules/REQUEST-949-BLOCKING-EVALUATION.conf
```

- /etc/nginx/modsecurity.conf(パスは任意)に記述する
- curl http://localhost/?union+select が403になる

まるっと適用するなら

```
SecRuleEngine On
Include /etc/modsecurity/crs/crs-setup.conf
Include /etc/modsecurity/crs/REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf
Include /usr/share/modsecurity-crs/rules/*.conf
Include /etc/modsecurity/crs/RESPONSE-999-EXCLUSION-RULES-AFTER-CRS.conf
```

- /etc/nginx/modsecurity.conf(パスは任意)に記述する

有効・無効の切り替え

- SecRuleEngine
  - On: ModSecurityを有効にする
  - Off: ModSecurityを無効にする
  - DetectionOnly: 検知のみ行う

既定の挙動を設定

```
#SecAction \
#  "id:900000,\
#   phase:1,\
#   nolog,\
#   pass,\
#   t:none,\
#   setvar:tx.paranoia_level=1"
```

- crs-setup.conf
  - SecActionで挙動に影響する変数を設定

リクエストに関する設定

- REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf
  - 既定では全部コメントアウト
  - preフィルタ的感じでつかう
    - ctl:ruleEngine
    - ctl:ruleRemoveByXXX
    - ctl:ruleRemoveTargetByXXX

レスポンスに関する設定

- RESPONSE-999-EXCLUSION-RULES-AFTER-CRS.conf
  - 既定では全部コメントアウト
  - postフィルタ的感じでつかう
    - SecRuleRemoveByXXX
    - SecRuleUpdateActionByXXX
    - SecRuleUpdateTargetByXXX

どんなルールセットがあるか？

- REQUEST-903.*
  - Drupal
  - Wordpress
  - NextCloud
  - DokuWiki
  - CPanel
  - Xenford

ルールが読み込めているか?

- /var/log/nginx/error.log参照
  - ルールをリモートから取得もできる

```
[notice] 16026#16026: ModSecurity-nginx v1.0.1 \
(rules loaded inline/local/remote: 0/911/0)
```

ルールがどのように適用されたか?

- /var/log/nginx/modsec_audit.log を参照
  - Hセクションをみるべし

```
--622ca252-H--
Message: Access denied with code 403 (phase 1). \
Pattern match "/phpmyadmin" at REQUEST_FILENAME. \
[file "/etc/httpd/conf.d/mod_security.conf"] \
[line "94"] \
[id "10000"] \
[msg "Blocking access to /phpmyadmin/index.php."] \
[tag "Blacklist Rules"]
```

ひっかかったのを検出するには

https://github.com/molu8bits/modsecurity-parser

- ModSecurity 2.xと3.xのログに対応している
- png,json,xlsxでレポートを生成

```
$ python modsecurity-parser.py --version3 \
 -f modsec_audit.log
```

ここまで話をしたけれど

- 本物のウェブアクセスログを使用した、機械学習による異常検知(全データ/ソースコード公開)
  - https://www.scutum.jp/information/waf_tech_blog/-2021/01/waf-blog-077.html
  - Javaのサンプルあり(バッチ処理)

ざっくりいうと

- アクセスログを1行ごとに特徴ベクトルにする
  - クエリーの特定の記号の出現回数とか
- Isolation Forestでモデルを構築する
- モデルを使って評価したスコアの外れ値をはじく

ModSecurity + Isolation Forestで遊ぶには(Ruby編)

- https://github.com/david-cortes/isotree
  - Isolation Forestを実装したライブラリー
  - https://github.com/ankane/isotree (Ruby)
    - 最近リクエストしてモデルのエクスポートに対応してもらった

ModSecurity + Isolation Forestで遊ぶには

- https://github.com/matsumotory/ngx_mruby
  - ngx_mrubyからリクエストを渡して評価

```
location / {
  mruby_set_code $backend '
    r = Nginx::Request.new
    ...ここでIsolationForestのモデルでr.uriのスコアを評価
    return score > 0.9 ? "" : "http://127.0.0.1:3000/"
  ';
  if ($backend = "") {
    return 403;
  }
  proxy_pass $backend;
}
```

debian

まとめ

- 自由なWAFはいくつかある
- DebianでModSecurity-nginxを使うには自分でビルドする必要がある
- とっかかりにはmodsecurity-crsのルールセットを使うとよい
- まずは検知モードで動かしてみるのがおすすめ
- さらなる強化は別の手法の併用を考えるとよいかも