WAFのパフォーマンス計測について

斉藤 宏明† 森 皓生† 笠原 卓也†† 島崎 聡史†† 中澤 昌史†† 齋藤 孝道†
†明治大学理工学部 ††明治大学大学院

1 はじめに

Webアプリケーション(以降，Webアプリと呼ぶ)に対するXSS(Cross Site Scripting)[1]やSQLインジェクション(SQL Injection)[2]等のHTTPリクエストを用いた攻撃が非常に増えてきており，その対策の一つとしてWAF(Web Application Firewall)が広く利用されつつある．WAFには検査内容をルールとして設定し動作するものがあり，様々な攻撃に対処するためには多くのルール数を設定しなければならない．しかしルールの増加によってマシンへの負荷が増加すると予想される．そこで，本論文では，WAFのひとつであるGuardian@JUMPERZ.NET[3](以降，Guardianと呼ぶ)を用いて実際にどのくらい負荷がかかるか計測し，評価を行った．

2 通信に関する基礎知識

2.1 WAF

WAFとは，ファイアーウォールの中でもWebアプリとの通信を監視するファイアーウォールである．WAFはルールで設定されている文字列をHTTPリクエストやHTTPレスポンスと照らし合わせ，合致した場合，ルールに従って処理する．攻撃が含まれていなければ検査後Webアプリへ送信する．また，WAFはWebサーバからクライアントへのレスポンスも検査できる．例えば，レスポンスに個人情報が含まれていたら，情報漏えいを阻止する目的でクライアントへの送信を遮断するという処理も可能である．

2.2 Guardian@JUMPERZ.NET

本論文で対象とするGuardianはJavaで実装されているフリーソフトのWAFである．Webサーバの一部として組み込まれるWAFではなく，プロクシ型のWAFである．起動させる前に検査に適用するルールを設定し，起動後はそれに従いHTTPメッセージの検査を行う．また，Guardianの管理者が独自のルールを追加できる．
Guardian用の設定ファイルの中に，ruleと呼ばれるファイルがある．ここにGuardianがリクエストやレスポンスの検査をする際に参照する文字列や処理が書かれており，複数の項目によって構成されている．
中でも以下に示す4項目がHTTPメッセージに対する検査の規則を構成している:

・type
HTTPメッセージの，どの箇所を検査するか設定する．

・pattern
検査対象とする文字列を正規表現で設定する．

・log
HTTPメッセージがpatternと一致した際に，内容をログに書き込むか否か設定する．内容は検査対象がHTTPリクエストならHTTPリクエスト，HTTPレスポンスならHTTPレスポンスのみ，両方なら両方保存する．

・action
ルールが適用された際のHTTPメッセージに対する処理を設定する．HTTPメッセージの送信を止めたり，そのまま送信することが可能である．

3 計測内容

Guardianが稼働している際のマシンに与える負荷を計測した．今回はネットワーク機器の性能評価を行うavalanche2700[4](以後，avalancheと呼ぶ)を使用した．Webサーバはapache2.2.14(以後，apacheと呼ぶ)を用いた．apacheとGuardianは同一のマシンに導入するのではなく，別々のマシンにそれぞれ導入した．

3.1 計測方針

Guardianのルール数を初期設定の61個に対して，約1.5倍の93個，半分の30個，更に1個のみの計4つの異なるルール数で計測した．

3.2 ネットワーク構成

今回の計測では他のマシンのトラフィックによる計測への影響を無くすために，独立したネットワークを構成した．avalancheとGuardian，apacheはスイッチを使って接続させている．Guardianはapacheのリバースプロクシサーバとして動作する．

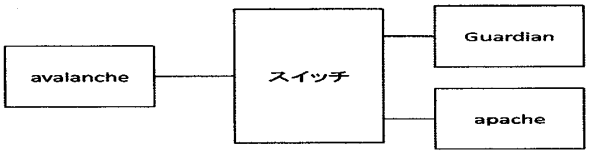

図1: ネットワーク構成図

3.3 使用したマシンの仕様

apacheとGuardianを導入したマシンとavalancheのシステム構成を以下に示す．

表1:マシンのシステム構成

	apache/Guardian	avalanche
CPU:	Xeon5050 3GHz	Xeon 3.6GHz
Memory:	3GB	8GB
OS:	Fedora9(kernel2.6.25)	
NIC:	1000Mbps	1000Mbps

3.4 計測前の設定

Guardianを起動する時にJavaの最大ヒープサイズを2000Mbyteに設定し，Fedora9の使用できるファイルディスクリプタ数の最大値を30000に設定した．

3.5 計測方法

apacheには，掲示板*のWebアプリを用意する．avalancheにはXSS攻撃を含ませたHTTPリクエストを徐々に増やしながら最大毎秒約340個ずつ15分間，合計で約135,000個送信させる．
Guardianには XSS攻撃を検知した場合，リクエストの拒絶はせず，ログを採取するよう設定しておく．avalancheとapacheによるHTTPメッセージの流れは以下に示す通りである:

1. avalancheからGuardianに向かってHTTPリクエストを送信する．

† Hiroaki SAITO, Akio MORI
†† Takuya KASAHARA ,Satoshi SHIMAZAKI , Masashi NAKAZAWA
† Takamichi SAITO

* クライアントが任意で作成した文章を投稿し，その文章を順々に表示させておくWebアプリ

2. Guardian が HTTP リクエストを受信し，内容を検査する．
3. Guardian で設定したルールを順々に適用し HTTP リクエストを検査し，XSS を検知する．そして当該 HTTP リクエストを保存する．
4. 残りの全てルールを適用して検査を行った後，JUMPERZ は apache へ HTTP リクエストを送信する．
5. apache は HTTP リクエストを受信し，処理を行う．その後，apache は Guardian へ HTTP レスポンスを送信する．
6. Guardian がそれを受信して，設定したルールを適用しレスポンスを検査をした後，avalanche へ送信する．
7. HTTP レスポンスを avalanche が受信する．

計測中に sar(System Admin Reporter) を用いて Guardian が稼動しているマシンのディスク I/O，メモリ使用量，CPU 使用率，ネットワークトラフィックの情報を 5 秒ごとに取得した．

3.6 計測結果

sar で取得した CPU 使用率のグラフを図 2，3，4，5 に示し，1 秒あたりに処理した HTTP リクエスト数の最大値と平均値を表 2 に示す．各折れ線グラフの項目は図の下に示す．

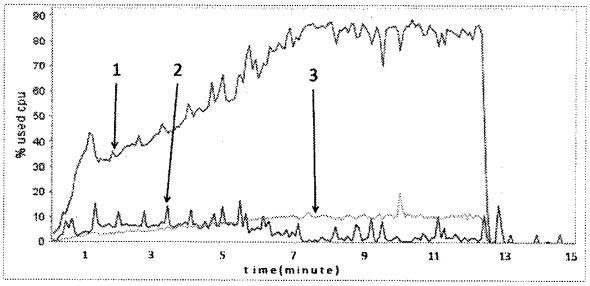

図 2:ポリシー 93 個時のマシンの CPU 使用率

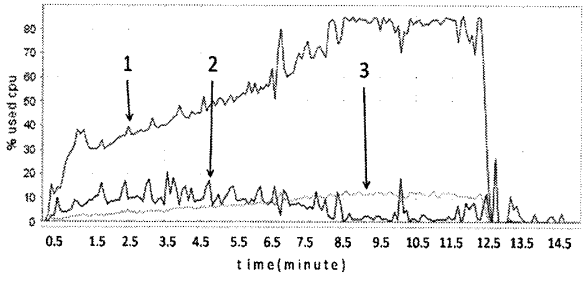

図 3:ポリシー 61 個時のマシンの CPU 使用率

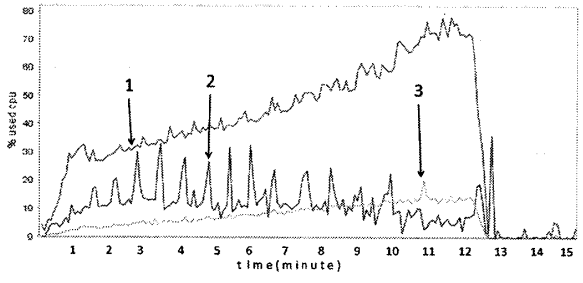

図 4:ポリシー 30 個時のマシンの CPU 使用率

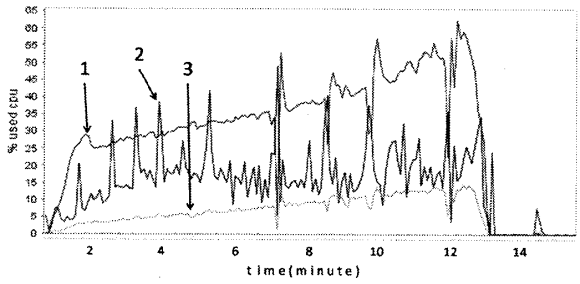

図 5:ポリシー 1 個時のマシンの CPU 使用率

1. ユーザープロセスの CPU 使用率
2. I/O 待ち時間の割合
3. システムプロセスの CPU 使用率

表 2:1 秒毎に処理した HTTP リクエスト数の最大値と平均値 [リクエスト数/秒]

ルール数	1	30	61	93
最大値	352	355	299	252
平均値	288.6	296.6	248.4	197.9

Guardian のルール数が 93 個の時はユーザープロセスの CPU 使用率が最大約 90 %になり，ルール数が 1 個のみだと CPU 使用率が最大約 65 %となり，マシンに与える負荷が少ない．また，1 秒あたりに処理した HTTP リクエスト数の平均値は，ルール数を増やすにつれ減少し，1 個の場合は 93 個の場合と比べて平均値が約 1.5 倍高い．

4 まとめ

本論文では，WAF の 1 つである Guardian が実際にマシンに及ぼす負荷を，ルール数ごとに計測した．ルール数を増加させると負荷も増加し，1 秒あたりに処理できる HTTP リクエスト数は減少することが分かった．今後の課題として，サーバと Guardian を同じマシンで稼働させ，マシンに与える負荷を計測し，apache モジュールの WAF である Mod_Security[5] がマシンに与える負荷を計測，前者と比較することが挙げられる．

参考文献

[1] "Cross-siteScripting"OWASP
http://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

[2] "SQLInjection"OWASP
http://www.owasp.org/index.php/SQL_Injection

[3] "Guardian@JUMPERZ.NET"
http://guardian.jumperz.net/index.html

[4] "Avalanche2700,Reflector2700"東陽テクニカ
http://www.toyo.co.jp/spirent/avalanche/2700series.html

[5] "ModSecurity"
http://www.modsecurity.org/