コンピュータセキュリティ 15－4
（２００１．１２．２１）

Webサーバ防御ツールの実用性評価

三友 仁史*、鳥居 悟*、小野 越夫*

*㈱富士通研究所

我々は以前より、サーバに対する不正なアクセス要求をフィルタリングする技術の研究開発に取り組んできた。それに関して、我々は以前、不正なHTTPリクエストをフィルタリングするWebサーバ防御ツールについて、その判定ロジック部分の検討／試作／評価を行った。その目的は、この技術の実現可能性を判断するための材料を得ることにあったが、結果は十分に満足できるものであった。

本研究では、前記の結果を受け、以前我々が提案したWebサーバ防御ツールに対して、より実運用に近い観点から機能検討、評価実験を行った結果について報告する。まず、前ツールを実運用に耐えるレベルにまでブラッシュアップするために、その基本要件を高めるための機能検討を行った結果を示す。次に、実運用に必要な処理のメインパスをほぼ実装し、実運用を想定した環境においてスループットを評価した結果を示す。

A practical evaluation of the filtering tool for Web servers

Masashi Mitomo*,　Satoru Torii*,　Etsuo Ono*

*Fujitsu Laboratories Ltd.

We have been studying the filtering technology for malicious requests for important servers. On the problem, we investigated a web server protection system, and studied the feasibility of it.

In this paper, we discuss above system, in the view of more “practical use”. This is defined at first. Then, we investigate the new functions which satisfy the practical use. Finally, we construct the main path of processing, and evaluate its filtering performance in the practical use.

1. はじめに

近年、ネットワークを介した不正アクセスの増加が問題になっている。特に、Web サーバはその社会的重要性から、悪意あるクラッカーから不正アクセス対象として選ばれやすい。また、最近、従来のWebサーバに動的な処理能力を付与するCGI(Common Gateway Interface)、ASP(Active Server Page)、PHP、DB連携技術が話題になっているが、同時にこれらのプログラム

の脆弱性による不正アクセス被害も多数報告されている。Web アプリケーションに対する不正アクセスは、ホームページの不正な書き換えやサーバが踏み台にされるなどの脅威に繋がり、最終的にはサイトの社会的信用が失墜する。

以上のような理由から、Web サーバに対する不正アクセスを防御したいという欲求は、このところ高まる一方である。我々は以前、このような要求を解決する為、Web サーバを対象としたリアルタイムフィルタリングツールについて提案し、その一部を実装評価した[1]。本稿では、この成果の流れを汲み、リアルタイムフィルタをまず実運用の観点から考えることを試みた。具体的には、まず、前ツールを実運用に耐えるレベルにまでブラッシュアップすることを目的に、その基本要件を高めるための機能検討を行った結果について論ずる。次に、実運用に必要な処理のメインパスをほぼ実装し、実運用を想定した環境において評価した結果を示す。

以下では、2章にて以前に報告したリアルタイムフィルタについて簡単に説明する。次に 3 章にて、リアルタイムフィルタが運用される環境と、実用化に向けての課題について述べる。4 章にて、判定の網羅性と正確性を向上させるための手段について論ずる。5 章にて、リアルタイムフィルタの処理におけるメインパスを実装した上で、実際の運用面から見た評価を行った結果について示し、6章にて、リアルタイムフィルタの汎用性について検討する。最後に7章で、まとめと今後の課題について述べる。

2. リアルタイムフィルタ

我々は以前、Web サーバに対する不正リクエストをフィルタリングするツール（リアルタイムフィルタ）について検討を行い、一部のモジュールを実装した。これは、既存の Web サーバが受け取る全てのリクエストにおける攻撃性の有無を、Web サーバの前段で判断し、攻撃性が認められたリクエストは Web サーバに渡さない(フィルタリング)ようにしたものである。これによって、Web サーバに対する任意のサイトによる不正アクセスを一度目から回避でき、且つ IP Spoofing 攻撃も正しく防御できる。以下は、その実施例図である。

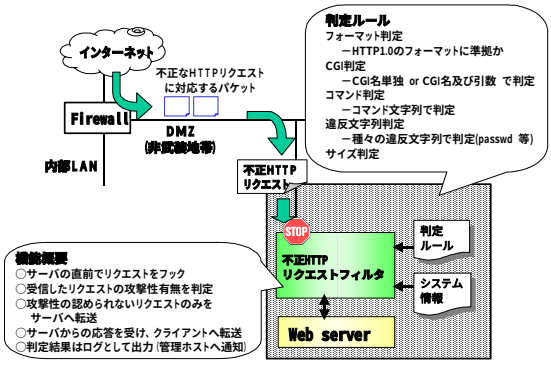

図 1:リアルタイムフィルタ

リクエストの正当性の判断には、正確性と即時性が要求される。リアルタイムフィルタにおいては、アプリレベルの判定及びリクエストに対する構文解析等により高い正確性を、ルール構造の2分木化及び画像リクエストに対する判定スキップ等によって高い即時性を確保することに成功した。以下に、リアルタイムフィルタが行うことが出来る判定を示す。

1) HTTP1.1 のフォーマットに合わないリクエストを却下
2) CGI 名及びその引数によって、リクエストの可否を判定
3) コマンド文字列を含むリクエストを却下
4) リクエスト全体を見渡して、違反文字列を含むリクエストを却下
5) 長すぎるリクエストを却下

我々は、このようなフィルタリングツールの判定ロジック部を実装した。そのスループットを計測した結果、20Mbps のスループットを得た。また、判定ロジックの網羅性を確認するため、2000 年 3 月までに報告された Web 関連のセキュリティホール 204 件の素性を調査し、それぞれに対して本ツールの判定項目への対応付け

を行った。その結果から、ルールを記述する形式言語は、判定方法が既知であるセキュリティホールの 93.2%に対するルールを記述できることが示された。

これらから、リアルタイムフィルタを実用化するための足掛かりが、性能的に得られたと判断された。以下では、このことを踏まえ、リアルタイムフィルタをより実運用面からフォーカスして検討していく。

3. 実用に向けての課題

先にも触れたように、我々が以前報告したリアルタイムフィルタは、技術の実現性を判定するためといった意味合いが強かった。これを実用レベルにするために、クリアしなければならない課題を以下に列挙する。

・ 判定の網羅性と正確性の向上

実用を考えると、リアルタイムフィルタには判定の正確性と網羅性の向上が課題である。これらを解決するために、必要な機能を追加していくための検討を行う必要がある。

・ 実用化を踏まえた性能評価

評価が判定ロジックのスループットに偏っていた。実際に運用される際には、通信や Web サーバとの連携と切り離すことは出来ず、それらを含めた評価が為される必要がある。

・ 汎用性に対する検討

現状ではリアルタイムフィルタは Web サーバを念頭において作られていたが、Mail サーバや DNS サーバにおいても有効であるような、汎用性を有していると有り難い。なぜならば、これらのサーバも公開サーバであり、Web サーバと同様に攻撃の対象とされるからである。したがって、リアルタイムフィルタがこれらのサーバにおける実現性について検討する必要がある。

次章以降では、これらの課題を一つずつ順に検討していく。

4. 網羅性と正確性を向上させる手段

我々は、[1]の発表以降、リアルタイムフィルタの機能面から、新たに追加すべき項目を洗い出すための検討を進めてきた。その結果、以前検討／試作したフィルタリングツールには、機能的な観点から、いくつかの問題点があることが明らかになった。以下ではそれらについて、問題点と共に述べる。

● 判定ロジックの充実

以前検討／試作したフィルタリングツールにおいては、リクエストの正当性の判定ロジックは、構文解析も行っていたものの、基本的にはパターンマッチングによるものに偏っていた。既知の脆弱性について調査した結果、これでは正確性、網羅性について十分でない場合もあるとの観点から、新たな判定ロジックについて検討した。

● (特に外部との連携について考慮した) 新たな必要機能についての検討

以前の検討においては、Web サーバとの通信機能等の必要最低限な基本機能は網羅していたが、実用を考えた場合に、必要となってくる機能が新たに判明した。本研究では、それらについて検討した結果について報告する。

4.1. ＣＧＩ 変数の解析を強化

CGI はしばしばセキュリティホールが発見され、これらは攻撃者によって Web サーバに侵入されるための道具としてよく用いられる。CGI は一般に、クライアントによってリクエストに含められたパラメータを用いてレスポンスを返すが、このパラメータの与え方によって、CGI に予期せぬ挙動を引き起こし、引いては Web サーバにもその影響を与えることもある。

このような攻撃を防ぐためには、CGI パラメータがクライアントによって正しく与えられていることを実証する必要がある。

ここではそのような攻撃に対する防御の一例として、ディレクトリトラバーサルを含むリクエストの拒絶を挙げる。

ディレクトリトラバーサルとは、有名な攻撃のクラスであり、公開ルートディレクトリより上位のディレクトリにアクセスしようとするものである。この攻撃は、CGI のパス変数に対して行われる場合も多い。この攻撃に対処するために、ネットワーク IDS では、単純に文字列 "/.."が含まれているリクエストを違反とすることで検知することがほとんどである。しかし、この方法では、文字列 "/.." を含む正当リクエストまで切られてしまうことになる。

しかし、リアルタイムフィルタにおいては、CGI に対してパスを値として取る変数を認識しており、まずその値を抽出し、その値の正当性を正しく評価する。典型例は以下のようなリクエストである。

```
GET  /msadc/Samples/SELECTOR/showcode.
asp?source=/msadc/Samples/../../../target
```

この例では、公開範囲を越えた場所に置かれたファイル target にアクセスしようとしている。このような攻撃を防ぐにはパス引数の値を正しく構文解析し、その正当性をチェックすることが有効であると考える。このような解析は通常のネットワーク IDS 等には不可能なものである。

また、CGI 変数のチェックは、限定された範囲の値をとるべきパラメータに対し規定外の値を与えることによる攻撃を、全般的に有効に防御することができる。これは次の「サーバ（サイト）に関する情報を判定に反映」において詳しく説明する。

4.2. サーバ（サイト）に関する情報を判定に反映

ネットワーク IDS は、サーバ、ネットワーク構成に関係なく、一様の判定を行うことが多い。リアルタイムフィルタは、サーバの構成、インストールされている CGI の種類、など、その運用環境に応じた判定を行うことができ、これにより判定の精度、速度を高める事ができる。

これは具体的には以下のように行われる。

- Webサーバの種類、OS などを自動認識し、そのことを自動的にルールに反映させる。すなわち、それらに応じてルールを選択し、必要最小限なルールによる判定を行う。これによって判定の速度が高まる効果が得られる。
- クライアントからパラメータを受け取る Web アプリケーションなどにおいて、それが受け取るべきパラメータの型を（半）自動的にルール化する。このような攻撃には以下がある。
  - 決まった数値(日にち等)を取るべきパラメータに、決まりに反した値を適用することによる攻撃
  - 数値を取るべきパラメータに、スクリプトなどの文字列を適用することによる攻撃
  - （アンケート集計 CGI など）パラメータの値がいくつかの選択肢から選ばれるべきなのに、選択肢以外の回答を適用する攻撃

これによって、単なるマッチングに比して高精度の判定を実施することが可能となる。

4.3. レスポンスを基にした不正検出

有名な攻撃の一つに、サーバ上に存在しないファイルを要求するリクエストによって、サーバ上の有用な情報を獲得できるというものがある。この攻撃を防ぐためには、サーバにおいて、リクエストが要求するファイルが存在するものか否かを一つ一つチェックする方法があるが、それはコストがかかりすぎる問題がある。もう一つの方法として、サーバのレスポンスを見て、クライアントに返したくない内容であれば、その内容を変更してしまうという方法もある。この方法は、不正リクエストが、Webサーバからのレスポンスに乗じて情報取得を試みるだけである場合において有効な方法であり、判定の網羅性を高めるものといえる。

4.4. 動的なルール更新

現状のリアルタイムフィルタは、静的なルールを用いて判定をしているが、それらを動的に更

新することで、よりきめ細かな判定が可能となる。この機能は、判定の精度、網羅性のアップに繋がる。

例えば、攻撃とみなされるリクエストを送ってきたサイトがあれば、当該サイトからの以後のリクエストを落とすようなルールを動的に作ることや、当該サイトからの以後のリクエストから攻撃とみなされるパターンを抽出し、新たなルールに加えるという例が考えられる。

また、動的なルール更新は、ルールの有効期限或いは範囲といった問題を提起することになる。詳細は今後の課題とする。

5. 性能評価

以前試作したフィルタは、判定ロジックの部分のみを実装し、その部分におけるスループットのみを計測した。しかし、実際の運用においては、この判定ロジックはフィルタリングツールに組み込まれることになる。すなわち、判定ロジックはWeb サーバがフィルタと連携して不正なリクエストを排除するフローの中の一部を占めているに過ぎない。

このような観点から、プロキシ型のリアルタイムフィルタリングツールについて、Web サーバとの連携部分と判定ロジックを実装することにした。すなわち、HTTP リクエストの処理にかかる部分を全て実装した。さらに、実運用を想定したプロキシ型フィルタリングツールの評価を行う必要がある。

実験環境を以下に示す。

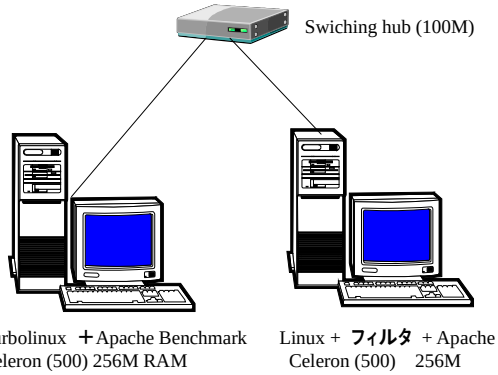

図 2:実験環境

今回は、特に Web サーバに要求される性能として、同時接続ユーザ数に対する、スループットの低下にフォーカスして検証を行った。以下では、Apache に標準で添付されるベンチマークツールである Apache Bench (AB) を用い、リアルタイムフィルタに負荷をかけることにした。このとき、以下のパラメータは固定とした。

・ ルール数(100 個)
・ リクエスト(リクエスト長 30byte)

また、Apache とリアルタイムフィルタにおける、初期プロセス数を同じにした。

このような仮定の下、同時接続ユーザ数を変化させ、リアルタイムフィルタを介在させたケースとそうでないケースにおける、スループットをプロットしたのが以下のグラフである。

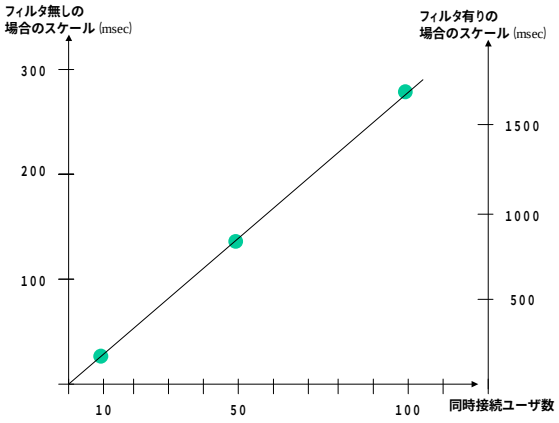

図 3: スループット比較図

このように、スループットは同時接続ユーザ数に比例しており、またフィルタがある場合と無い場合には、スケールにしておよそ6倍の差があることが判明した。この差が発生する原因については、今後検討する。

6. 汎用性の検討

現状のリアルタイムフィルタは HTTP に対するものとなっているが、ルールを変更するだけで、Bind や Mail サーバに対してもこれを適用することができるはずである。

著名なセキュリティ・ポータルサイトである SecurityFocus[8] において2000年4月以降に公開された脆弱性のうち、Bind 関連、Sendmail 関連はそれぞれ14件、23件であった。これらについて調査したところ、サービス不能攻撃(DoS: Denial of Service)以外の攻撃に対しては、技術流用が可能である、すなわち、(判定ロジックには手を入れないで)ルールを入れ替えるだけで使いまわせる見通しが得られた。

これは、これらのプロトコルも構文解析＋パターンマッチングによって、悪意あるリクエストを判別できることを意味している。

7. まとめ

本稿では、我々が以前に提案したWebサーバ防御ツールを、より実運用に近い観点から、機能検討、評価実験を行った。機能検討の結果、ツールに対する諸要件を引き上げるための新たな機能を挙げる事ができた。また、性能評価の結果、実装したリアルタイムフィルタには、いくつかの問題があることが判明した。これについては今後の課題としたい。

さらに、今後は、ツールが想定される実運用に十分耐える性能を有していること、を実証することを当面の課題とする。

参考文献

[1]三友 他、“ Web サーバ向け不正アクセスフィルタ ”、情報処理学会 第 62 回全国大会、Mar. 2001

[2] Hypertext Transfer Protocol – HTTP/1.1 (RFC2068)

[3] A look at whiskers anti-IDS tactics **<http://www.wiretrip.neyt/rfp/pageis/whitepapers/whiskerids.html>**

[4] 武田 他, “ネットワーク侵入検知 －不正浸入の検出と対策－”, ソフトバンクパブリッシング刊, ISBN:479731253X, Jun. 2000

[5] Robert Graham, “NIDS-FAQ,” **<http://www.robertgraham.com/pubs/network-intrusion-detection.html>**

[6] SCMagazine, TestCenter, ”IntrusionDetection”, Jun. 2000, **<http://www.scmagazine.com/scmagazine/2000_06/testc/testc.html>**

[7] S.Boran, “Personal Firewalls / Intrusion Detection Systems”, SecurityPortal, **<http://securityportal.com/articles/pf_main20001023.html>**

[8] SecurityFocus, **<http://www.securityfocus.com/>**

[9] Honeynet Project, “Know Your Enemy”, **<http://project.honeynet.org/papers/>**