セキュリティのための段階的通信制限システムの評価と改良

2004MT012 福井 麻美　　2004MT097 末吉 昭仁
指導教員　後藤 邦夫

1　はじめに

インターネットが普及している近年，その安全性が大きな問題となっている．特に DoS 攻撃 (Denial of Service attack) などの不正アクセスについては，掲示板や検索サイトなどがターゲットとして狙われ，その被害について報道される場合が多い [3].

本研究では，この対策手段のひとつである「段階的通信制限を実現するゲートキーパーの提案と試作」(以下，既存手法とする)[2] の改良に重点を置いて進めていく．なお，待ち行列での処理には GINE 論文の Queue 処理の手法 [1] を用いる．

主な改良点はフィルタリングルールの管理方法である．既存手法では，既存のネットワーク型 IDS が発するアラートを元にルールをフィルタリングするという方法をとっていたが，本研究では IDS，メイルサーバ等の外部アプリケーションからいつでも自由にフィルタリングルールを追加，削除出来るマニュアル操作機能を追加する．マニュアル操作機能を実現するため，IDS，メイルサーバ等の複数の外部アプリケーションからの要求を受け付ける機能を付け加える．メールサーバや IDS などの外部アプリケーションからフィルタをコントロールするために，疑似アタックのテストによる効果，基本性能の向上を目指す．

GK 本体の作成と実験は共同で行い，末吉は主に実験環境構築と疑似アタックを担当し，福井は主に GK と外部アプリケーションとの通信機能，ルールのエントリとフィルタ操作を担当した．

2　システムの概要

この節では，本研究で提案するシステムの概要を既存手法と比較して，GK の基本的なネットワーク構成，フィルタリングルールの追加，更新，削除について述べる．

2.1　既存システムの概要

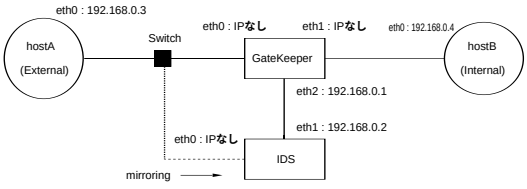

図 1　ネットワーク構成

既存システムでは，GK を外部ネットワークと内部ネットワーク間で IP アドレスをつけずにブリッジとして動作させ，フレーム通過時に通信を制限する (図 1 参照)．IDS はスイッチでミラーリングされたネットワーク上を流れるパケットを監視し，検知した攻撃の種類，攻撃下の IP アドレスなどの情報を直結した GK に渡す．GK はこの情報に基づいてリアルタイムに通信制限を行う．

GK は主に，Receiver，Queue，Sender，Real Time Clock Timer(以下，RTCTimer とする)，Filter，FilterManager，RuleUpdator から成り立っている (図 2 参照)．

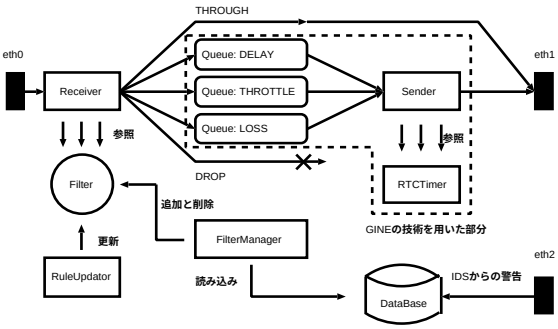

図 2　GK の構成

Receiver は制限対象リストの Filter をチェックし，受信フレームをどの経路に振り分けるか判断する．THROUGH の場合は直接送信し，DELAY，THROTTLE，LOSS の場合は各 Queue に入れる．また DROP の場合は全てのフレームをその場で破棄する．各 Queue に入れられたフレームは，障害を発生される処理を行った後，RTCTimer と連動した Sender によって送信される．Filter の情報は FilterManager と RuleUpdator によって管理されている．FilterManager はデータベースに出力された IDS からの警告を読み込み，Filter に書き加える．また同時に古くなった情報の削除も行う．RuleUpdator は，FilterManager が書き込んだ制限対象をどの経路に振り分けるかという情報を管理する．

2.2　新システムの概要

本研究では，外部アプリケーションからフィルタ変更要求を受け付けることで，より正確で効率の良いフィルタリングを実現できると考えた．

本研究の主な改良点は，以下の 3 点である．

- 外部からのルール追加，削除，表示要求を受け付けるコマンド受付・返信スレッドの実装
- ホスト管理表の実装
- Filter のルール処理機能の追加

本研究では，フィルタ変更をする際の必要情報は，IDS を含めたその他の外部アプリケーション (メイルサー

バ，IDS 等) から送信される．そのためには，外部アプリケーションから必要情報を受け付けるためのコマンド受付・返信スレッドを新たに実装する必要があると考えた．

次に，ホスト管理表の実装である．ホスト管理表は外部アプリケーションと GK 内のコマンド受付・返信スレッドの通信の際に参照される．GK の信頼性を高めるため，通信を行う相手ホストの情報はホスト管理表で管理し，一覧に無いホストからの要求は受け付けないことにする．そうすることで悪意のあるホストからの接続が大幅に軽減されると考えた．

最後に，Filter のルール処理機能の追加である．外部アプリケーションからルールの追加，削除，表示要求を受け付けるようにしたことで，ルールの処理方法にも変更を加えた．詳しくは次節で述べる．

3 システムの実現

本研究で提案するシステムを実現するためには，外部アプリケーションからフィルタリングルールの追加・削除要求を受け付けるコマンド受付・返信スレッドの実装が必要である．

プログラムのクラスは主に，Commander，Filter，DatalinkSocket，Reciver，Sender，Timer によって構成される (図 3 参照)． Commander は本研究で新しく

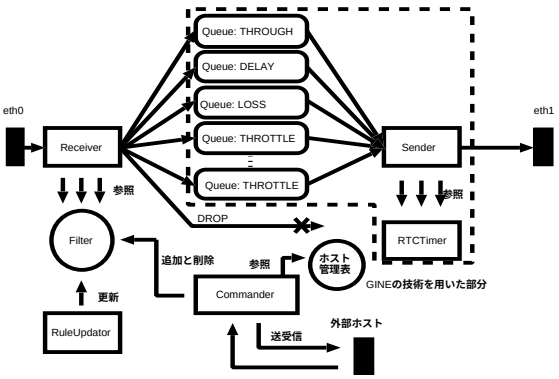

図 3 提案するシステムの構成

加えた外部ホストとの通信部である．ここでフィルタ操作の振り分けを行う．Filter では，外部ホストからの要求でルールの追加，削除，表示等の処理を行う．DatalinkSocket では，経路の設定を行う．

なお，Commander，Receiver，Sender，Timer はプログラム内でそれぞれスレッドとして動作し，その実現には GK と同様にオブジェクト指向の C++ 言語を用いた．

本研究では，通信制限のアクションが THROUGH の場合も Queue を作成し，他のアクションの Queue と同様に扱えるように改良した．また，THROTTLE はルールによって設定する帯域幅が異なるので，ルール毎に Queue を作成した．

3.1 コマンド受付・返信スレッド (Commander) の実装

GK 内に実装したコマンド受付・返信スレッド (以下，Commander) でおこなう主な処理は，外部アプリケーションとの通信，外部アプリケーションからの要求に応じて，フィルタリングルールの追加や削除，表示などの処理を行うことである．また，Commander は外部アプリケーションからの依頼を設定する時の設定内容の確認通知や，依頼された処理が成功したか失敗したかの結果の返信も行うことにする．

3.2 フィルタの管理

ここでは，フィルタリングルールのエントリと，追加，削除，表示の各処理について説明する．

フィルタリングルールのエントリ

外部アプリケーションからの要求でルールが追加される．ルールをエントリするために必要な情報は，パケットの配信元 IP アドレス・マスク，宛先 IP アドレス・マスク，プロトコル番号, 配信元ポート番号, 宛先ポート番号，設定する通信制限の種類 (DELAY, THROTTLE, LOSS)，通信制限のパラメータ，ルールの有効期限，攻撃のタイプである．これらの情報に加え，そのルールの追加要求をしてきた外部ホストの IP アドレスもエントリの際に取得しておく．また，ルールの管理はルール番号で行うため，ルール番号を格納する変数も用意しておく．

フィルタの各処理

フィルタの処理には，フィルタリングルールの追加，削除，表示，挿入がある．追加，挿入はどちらも新たにルールを作成する処理であるが，その相異点は，ルールを格納する位置である (図 4 参照)． ルールの追加の場

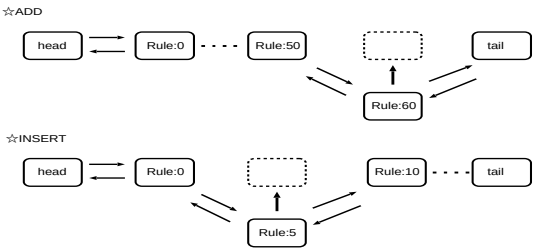

図 4 ルールの追加と挿入

合，ルールは常にフィルタの最後尾に格納する．挿入処理では，ルールを追加する位置を Host 側で決めることが出来る．また，挿入と削除と併せて利用することで，ルールの更新を実現できると考えた．また，ルールの削除は，ルール番号を使用するため，表示と併せて利用するものとする．ルールの削除を希望する Host は，まずルールの表示依頼をする．ルールの表示は，その Host が設定しているルールの一覧が表示される．Host は，一覧のルール番号を元にルールの削除依頼を行う．

4 実験による評価

本研究で試作したシステム (以下，GK2 とする) を用いて実験を行い，ネットワーク上での GK2 の有効性に

ついて述べる．

評価項目は以下の通りである．

- 制限なしの状態で，GK2 がブリッジとして高速に動作するか．通信自体を妨害しないか．
- 各制限が設定値通りの挙動を示すか．
- フィルタ追加，削除，表示のフィルタ操作が正確にできるか．
- TCP や UDP に対して，各制限はどのような効果が期待できるか．
- 疑似アタックに対して，GK2 の有効性が示せるか．

上記の点に注目して実験を進めた．

4.1 実験ネットワークの構成

図 5 は実験環境について示した図である．実験環境は端末として Linux マシンを 2 台用意し，それぞれを ExternalHost A(攻撃ホスト)，Internal Host B(保護されるホスト) に見立て，その間に GK2 をインストールした PC を挟むような形で LAN ケーブルをつなぎ構築した．PC のスペックは表 1 の通りである． また，全ての

表 1　機器のスペック

	GK2	hostA，B
CPU	Intel(R)Xeon(R)1.86GHz	Intel(R)Celeron(R) M processor 1200MHz
メモリ	512MB	1GB
NIC - eth0	Broadcom BCM5712 Gigabit Ethernet	Intel Corp.82801BD PRO/100VE(MOB) Ethernet
NIC - eth1	Broadcom BCM5754 Gigabit Ethernet	−

PC の OS に VineLinux を用い，ルールの追加は telnet コマンドを用いて GK2 内のコマンド受付・返信スレッドに接続して行った．

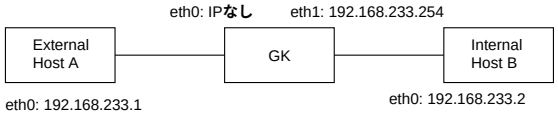

図 5　実験環境

4.2 試行する疑似アタックの種類

試行する疑似アタックは以下の 3 通りである．

- WWW サーバに対する DoS 攻撃．
- パスワード解析系攻撃．
- 大量のメールを勝手に送り付けるスパムメール．

各疑似アタックを行うための Linux コマンドは，WWW サーバに対する DoS 攻撃は「ab」，パスワード解析系攻撃は「expect」，スパムメールは「sendmail」で疑似することにした．

4.3 GK2 の性能評価

hostA から hostB に向かって，ping と iperf を用いて GK2 の性能評価のための実験を行った．hostA から hostB に向かう通信に対して，DELAY，THROTTLE，LOSS，DROP の各通信制限のルールを設定した．設定値は表の通りである．

実験結果は表 2 のようになる．DELAY の RTT と LOSS の損失率を見ると，設定値がそれぞれ反映されていることがわかる．通信の遮断である DROP も正常に動作している．また，フレーム損失が起きる LOSS と DROP 以外では損失が起きていない点，遅延を発生させない設定遅延 0msec の DELAY と LOSS や設定帯域幅 0.1Mbps，1.0Mbps の THROTTLE の RTT が THROUGH の RTT と比較してほとんど差がない点から，GK がブリッジとして高速に動作し，期待通りの通信制限が行われていることがわかる．設定帯域幅 0.05Mbps の THROTTLE では ICMP パケットの通過に影響をもたらす結果となった． 実験結果より，外部ホストから追加要求をしたルールがきちんと反映されていること，また，設定したパラメータどおりに通信制限が行われていることがわかった．

表 2　RTT とスループット，フレーム損失率の測定結果

	設定値			測定値		
制限	遅延 (msec)	帯域幅 (Mbps)	損失率 (%)	RTT(msec)	スループット (Mbps)	損失率 (%)
THROUGH	−	−	−	0.52	367.80	0
DELAY	0	−	−	0.50	232.00	0
DELAY	300	−	−	300.58	0.86	0
DELAY	500	−	−	500.55	0.49	0
DELAY	1000	−	−	1000.52	0.22	0
THROTTLE	−	0	−	0.52	232.00	0
THROTTLE	−	0.05	−	1.92	0.047	0
THROTTLE	−	0.1	−	0.53	0.095	0
THROTTLE	−	1.0	−	0.51	0.95	0
LOSS	−	−	0	0.52	−	0
LOSS	−	−	25	0.53	−	26
LOSS	−	−	50	0.57	−	49
LOSS	−	−	75	0.61	−	75
LOSS	−	−	100	−	−	100
DROP	−	−	100	−	−	100

4.4 WWW DoS 疑似アタック

WWW サーバに対する疑似アタックは ab コマンドを使用した．この実験において重要と言える項目は以下の 5 つである．

- Complete Requests(CR)：総リクエスト数
- Failed Requests(FR)：失敗したリクエスト数
- Requests per second(R/s)：1 秒当たりの平均処理リクエスト数
- Time per Request(T/R)：1 リクエスト当たりの平均処理時間
- Transfer Rate(TR)：1 秒当たりに受け取った byte 数

実験結果を表 3 に示す．表からわかるように，全ての制限において THROUGH の時より，各項目がサーバーにとって良い測定値を示している．特に LOSS による制限が最も効果的であると言える．LOSS の制限は 50 %，75 %とまだまだ重くできるので，より激しい攻撃に対しても効果が期待できる．

表 3　WWW DoS 疑似アタックの測定結果

	測定値				
制限	CR	FR	R/s(#/sec)	T/R(msec)	TR(kbytes/sec)
THROUGH	1000	0	538.54	18.57	1622.09
DELAY(300)			16.10	621.17	48.49
DELAY(500)			9.66	1035.38	29.12
THROTTLE(0.1)			25.52	391.86	76.86
LOSS(25 %)			5.95	1681.74	17.91
LOSS(100 %)	−	−	−	−	−

4.5　パスワード解析系疑似アタック

今回の実験においては expect というプログラムを用いて，パスワード解析系攻撃を疑似した．expect は他の対話形式プログラムとプログラムされた問答を行なうプログラムで，通信の流れを自動化するものである．

制限なしの状態ではジョブ接続後，パスワードの問答が繰り返され，loop.sh で指定した回数だけ expect スクリプトを実行して終了した．全体的にスムーズに通信を行ったという感覚だった．次に DELAY(300) を追加して行ったところ，全体的に動作が鈍くなった．しかしながら，遅いながらも通信自体は成立しておりジョブがパスワード解析攻撃を受け続けていた．expect スクリプトのタイムアウトのデフォルトが 20 秒なので，GK2 の DELAY で 20000(msec) の遅延を追加したところ，ジョブの応答よりも早くパスワードを送信して，接続をすることができなかった．これによりパスワード解析系攻撃においても GK2 の通信制限が有効的であると考えられる．

4.6　スパムメール疑似アタック

スパムメールを疑似するために，Linux コマンドの sendmail を用いた．sendmail とは Linux に標準的に用意されているコマンドで，以下の入力方法で使用できる．

- /usr/lib/sendmail
  -i 04mt012@[192.168.233.2]<test.txt

まず相手に送るためのメール内容を記述したファイルをテキスト形式で用意する．端末のコマンドラインから上記の入力で 1 通のメールが 04mt012@[192.168.233.2] 宛に送信できる．上記のコマンドをシェルスクリプトに 100 回繰り返し記述し，それを実行することで大量送信を実現した．今回送るメールのサイズは平均的なメールサイズである 12KB とした．なお，表 4 の送信時間の目安は次の 4 段階とした．

- 1：非常に遅い
- 2：遅い
- 3：どちらかというと遅い
- 4：普通

実験結果は表 4 の通りである．まず全ての制限においてメール到着時間にかなりの遅延が発生し，もしスパムメール送信者がメールの到着率が悪いとすぐに他の標的に変更するという古いタイプのスパムメールソフトを使用した場合は効果が期待できる．DELAY の 50000 では，タイムアウトしてしまい，メールを送信することができなかった．LOSS が設定値通りにいかなかったのは，postfix の再送機能によるものだが，それでも LOSS の 75 %に至っては 100 通中 90 通送り終わったところで 1 時間近くかかった． これらの実験結果より，スパムメールに対して最も有効であるのは DELAY であることがわかった．また，LOSS も有効であることがわかった．

表 4　メールの送信時間と損失率の測定結果

	測定値			測定値	
制限	遅延 (msec)	帯域幅 (Mbps)	損失率 (%)	送信時間	メール損失率 (%)
THROUGH	−	−	−	4	0
DELAY	300	−	−	3	0
DELAY	500	−	−	3	0
DELAY	1000	−	−	2	0
DELAY	10000	−	−	1	0
DELAY	50000	−	−	−	100
THROTTLE	−	0.1	−	2	0
THROTTLE	−	1.0	−	2	0
LOSS	−	−	25	2	0
LOSS	−	−	50	1	5
LOSS	−	−	75	1	10
DROP	−	−	100	−	100

5　おわりに

本研究では，外部アプリケーションからの要求を受け付けるために，コマンド受付・返信スレッドを実装した．コマンド受付・返信スレッドを実装したことで，外部アプリケーションの要求に柔軟に対応することが可能となった．

また，運用レベルでの効果測定として疑似アタックを試行することで評価を行った．実験結果から，本研究で提案したシステムが実ネットワーク環境下で設置した場合にも効果的に作用するだろうと考えられる．

今後の課題としては以下の 3 点が挙げられる．

- セキュリティ強化のためのホスト管理表の実装
- インターネット環境下での運用テスト
- 攻撃に対しての情報を収集するハニーポットへの切り替え機能の追加

参考文献

[1] Ihara, A., Murase, S., and Goto, K.: IPv4/v6 Network Emulator using Divert Socket, *Proc. of 18th InternationalConference on Systems Engineering (ICSE2006),* Coventry, UK, pp. 159-166(Sep.2006)

[2] 青山 正樹, 小島正和: “段階的通信制限を実現するゲートキーパーの提案と試作”, 卒業論文, 南山大学数理情報学部情報通信学科 (2007)

[3] 警察庁セキュリティポータルサイト@police: “我が国におけるインターネット治安情勢の分析について”, 警察庁 (2007). (http://www.cyberpolice.go.jp/detect/pdf/20070726.pdf)