2-2-17__初 04.1.13 9:02 PM ページ 188

ネットワークセキュリティ 最新 自己防衛ガイドライン

2 悪者を探し出せ！ 高機能IDS Snort活用のツボ

本章では，Snortに関して基本的なインストール方法から，Webサーバやデータベースなどのほかソフトウェアと連携して使用するための拡張機能の解説や設定のTIPS，また実際にコンパイルする際や機能を使用するための補足情報などについて解説します．

awacs@hawkeye ■ awacs@hawkeye.ac

はじめに

2003年1月，皆さんもよくご存知と思いますが，SQL Slammerが世界中を駆けめぐりました．皆さんが管理しているコンピュータにもワームがノックした跡があったのではないでしょうか．しかし，中には「そんなことはなかった」とおっしゃる方もいらっしゃるのではないでしょうか？ しかし，そうとは言いきれません[注1]．

そんな方にぜひ使ってみてほしいソフトウェアがあります．それがSnortです[注2]．Snortは，Martin Roeschが中心となって開発しているフリーの侵入検知システムです．フリーとは言っても，その実力は商用システムに引けを取りません．

本誌2002年1月号にて，このソフトウェアのインストールについて解説した記事を執筆させていただきました[注3]．今回は古くなってしまった部分などについて補足説明を行うとともに，Snortの拡張機能の説明と使用方法，さらにSnortを活用するためのツールなどを紹介したいと思います．

なお，ここではRed Hat Linux 8.0にインストールする場合について説明します．

現在のバージョン

現在，Snortの最新公式リリースは1.9.1です．

実は，この原稿を書き始めたとたんにニュースが飛び込んできました[注4]．Snortにセキュリティホールが存在していたのです．このため現状でセキュリティホールの見つかっていない，唯一安全なリリースバージョンが1.9.1となります．もしほかのバージョンを使用している方がいらっしゃいましたら，すぐにバージョンアップされることをおすすめします[注5]．

なお，次期バージョンとされているSnort 2.0は，性能が大幅に向上するという発言が公式ページにも掲載されており，公開が期待されています．

インストールまでの流れ

本稿ではソースからコンパイルし，インストールする場合について説明します．ソースのRPMパッケージも存在するのですが，specファイルの修正が必要となります．なお，RPMパッケージからのバイナリのインストールについては，後述の理由より動作に責任が持てないので割愛します．

注1：筆者の56Kモデムのダイアルアップ用回線ですら，多くの痕跡を確認していました．か細い帯域もしっかり埋めてくれましたし:-(.
注2：http://www.snort.org/
注3：編集部のご厚意によりこの記事のPDFをご提供いただきました．http://www.yk.rim.or.jp/~shikap/security/awacs/snort191.htmlにて公開しております．
注4：さらに脱稿直後，1.9.1には検出不可能なパケットパターンが存在することが確認されました．こういった類のソフトウェアにとって致命的です:-(．この問題は2.0rc1でFixされています．なお，新しいバージョンのインストールにも，きっとこの記事が役に立つと信じています:-)．
注5：現在活用中でバージョンアップしたくないと言う方は，rpc Decoderというプリプロセッサを無効にして再起動しておけばこのセキュリティホールが無効になります．具体的にはsnort.confにある，「preprocessor rpc_decode: 111 32771」という行の先頭に「#」を入れてコメントアウトします．その後，再起動するだけです．

●ライブラリなどのインストール

まず，インストールする前に必ず行う必要があることについて説明しておきましょう．それはlibpcapのインストールです．ネットワーク系ツールの多くがこのライブラリを使用していますからすでにインストールされているとは思いますが，libpcapの最新版，0.7.2に入れ替えておきましょう[注6]．

libpcapのインストール自体は非常に簡単な部類に含まれると思います．./configureしてmake，どちらも正常に終了したらroot権限でmake installするだけです．ただし，Red Hat Linux 8.0では古いバージョンがRPMパッケージからインストールされているため，これをアンインストールしてからのほうが無難です．

次に，後ほど説明する拡張機能を使用する場合は，さらに必要なライブラリなどをインストールする必要があります[注7]．

インストールの基本

Snortは各種プログラムと同様に，tar ballを展開し，できたディレクトリ（snort-1.9.1/となります）に移動（以降断りない限りここをカレントディレクトリとします），./configureを実行し，エラーがなければmakeでコンパイルします．そして，ここでもエラーが出なければrootユーザとなり，make installでバイナリファイルなどをインストールする，という一般的な方法でインストールできます．

インストール後の作業

通常の拡張機能を用いない場合には上記の方法でインストールをして，設定すれば，Snortが実行可能となります．以下，簡単にインストール後の流れをまとめます．

●ルールセット，設定ファイルのコピー

make installまで実行しインストールが終わった時点では，Snort本体とmanファイルが，/usr/local/bin/と/usr/local/man/にそれぞれインストールされているだけです．Snortを実行するためには，これら以外にルールセット，設定ファイルが必要となります．これらのファイルはrules/とetc/にそれぞれ入っているので，これらをコピーします．

まずroot権限で/etc/snort/を作成し，etc/のファイルを/etc/snort/にコピーします．次に/etc/snort/rules/を作成し，そこにrules/のファイルをコピーしましょう．もちろん，これらのファイルのパーミッションには十分に注意してください[注8]．

● snort.confの修正

/etc/snort/snort.confを必要に応じて修正します．おもな修正部分を表1に挙げました．

●ログディレクトリの作成

ログ出力を行うディレクトリ（/var/log/snort）を作成します．

● Snort専用ユーザの作成

最後に，万が一Snortがセキュリティホールを持っていた場合に備え，Snort専用ユーザを作成し，ログイン不可に設定しておきます．これで最低限実行するのに必要な設定が終了したことになります．

● Snortの起動

図1①のようなオプションを付けて実行して，エラーがでなければ最低限の設定は終了です[注9]．実際に動かす場合には，-Dオプションを付加し，ユーザも変えた上でデーモンモードで動かすと良いでしょう（図1②）．あとは必要に応じて/etc/rc.localにsnortの

表1　/etc/snort/snort.confのおもな修正部分

修正前の行番号	修正前の行	説明
47	var HOME_NET any	検知対象のIPアドレス（レンジ）を記述，anyはすべてのアドレス．複数の場合は[]でくくる．区切り文字は「,」でスペースは加えない
52	var EXTERNAL_NET any	攻撃元の可能性があるアドレス（レンジ）を指定，anyはすべてのアドレス
102	var RULE_PATH ../rules	ルールファイルの保存箇所を絶対パスまたはsnort.confからの相対パスで指定

注6：多くのディストリビューションでは，古いバージョンがインストールされています．しかし古いバージョンを使用した場合，Snort実行時にワーニングメッセージが出てしまいます．また，古いバージョンにはセキュリティホールが（tcpdumpに）見つかっている関係からも，0.7.2を使うことをすすめます．RPMのエラーを防ぎたい方はRPMパッケージを探してインストールしてください．

注7：もちろんコンパイル作業を行いますので，開発環境（GCCなど）は必須となります．インストールされていない場合にはインストールしてください．

注8：Makefileなどいらないファイルもありますが，誌面の都合のため説明は省略します．ご了承ください．またパーミッションですが，一般的にはrootユーザ読み書き可で，Snort専用ユーザは読み込みのみ可，それ以外は読み書き不可，というのが良さそうですね．後ほど紹介するツールを使用する場合は，rules/のファイルはツール実行ユーザが修正可能なようにしておいたほうが良いでしょう．

注9：SnortはNICをプロミスキャスモードに切り替える必要があるため，実行時権限は必ずrootでなければなりません．まちがってもSnortのバイナリをsetuidしないようにしましょう:-)．

2-2-17__初 04.1.13 9:02 PM ページ 190

起動コマンドの付加などを行えば良いでしょう．

また，**contrib/**には**S99snort**というファイルがあります．これは**/etc/init.d**に置く起動スクリプトになっているので，図**1**③のようにすればマルチユーザモードで**snort**が自動起動するようになります[注10]．

なおコマンドライン引数の多くは**snort.conf**に記述できます．詳しくは**doc/**にある，**SnortUsersManual.pdf**の「**2.1.3 Config**」を参照してください．

拡張機能

続いて**Snort**の拡張機能について説明します．

flexresp

●flexrespとは

まずはじめは**flexresp**を取り上げます．**flexresp**はルールにマッチした通信を自動的に遮断する拡張機能です．通信遮断の方法は，**TCP**での通信の場合は**RST**フラグ付きのパケットを，**UDP**の場合は**ICMP**をサーバ，クライアント，またはその双方に送信することで行います[注11]．

●インストール

通信遮断に使用するこれらのパケットを生成するため，**libnet**というライブラリが必要になります．そのため，この機能を使う場合にはこのライブラリがインストールされている必要があります．なお**libnet**の最新バージョンは**1.1.0**なのですが，**Snort**の**configure**は**libnet 1.0.2a**以外のバージョンではエラーを返すようになっているので，**1.0.2a**を用いましょう[注12]．前述の「インストールまでの流れ」にある，**libpcap**と同様の方法でインストール可能です．

flexrespを利用できるようにするには**Snort**インストール時に，**./configure --enabe_flexresp**を実行してから**make**します．それ以外は先に示したインストール方法と変わりません．

●実行

frexrespは**Snort**のルールを拡張する形で実装されているため，この機能を使用するためにはルールの修正／作成が必要となります．ルールの記述方法については後述します．

SNMPサポート

●SNMPサポートとは

SNMPサポートはルールにマッチした通信が発生したときに，**SNMP**の**trap**と呼ばれる機能を用いて，**SNMP**マネージャにアラートの発生を通知する機能です．**SNMP**を用いてネットワーク管理をされている方にとっては便利な機能でしょう[注13]．

●インストール

インストールに必要なものは，**NET-SNMP**のインクルードファイル／ヘッダファイルです．**Red Hat Linux 8.0**ならば，**net-snmp-devel**パッケージをインストールする必要があります．

なお，**configure**実行時にエラーが生じてしまいます．これはスクリプト実行時，前述のヘッダファイルの有無を確認するところで**gcc**のワーニング出力を評価してしまい，ヘッダファイルがないと誤認識するためです．これを回避するために**configure**ファイルを修正します[注14]．そして，**./configure --with-snmp**を実行します．

次に**make**ですが，これもこのままでは通りません．コンパイルを通すために無理矢理ではあるのですが，**src/output_plugin/spo_SnmpTrap.c**を修正しました[注15]．これでコンパイルが通ります．

●設定

この機能を有効にするためには，**snort.conf**の修正が必要です．**snort.conf**の（無変更時であれば）**457**行目以降に**snmp**サポートの設定サンプルが記されて

図1　snortの起動

```
①設定確認時（最低限の確認）※
# snort -i lo -c /etc/snort/snort.conf

②実際に起動する場合
# snort -D -i eth0 -u snort -g snort -c /etc/snort/snort.conf

③S99snortを使用する場合の例
# cp contrib/S99snort /etc/init.d/snortd
# ln -s /etc/init.d/snortd /etc/rc3.d/S99snort
# ln -s /etc/init.d/snortd /etc/rc5.d/S99snort
```

※：snort -?で簡単なヘルプが表示されます．なお，コマンド引数でsyslogへアラート出力（-s）を指定すると，snort.confのプロセッサがすべて使用できなくなる場合がありますので，気を付けてください．また，本文の注16と同様の理由により，-cオプションは必須です．

注10：なお，ファイルの一部（パスなど）を修正する必要があります．修正個所はS99snortの10，11，14，20行目です．
注11：誤解を恐れず言うなら，偽装パケット送信ツールです．どれだけ過激なものかわかりますね;-)．
注12：最新の1.1.0でのコンパイルに挑戦したのですが，configureに相当手を入れなければならないようだったので途中で止めました．
注13：とはいえ，trapを理解できるsnmpマネージャを使ってないと無意味ですね．
注14：場当たりconfigureをhttp://www.yk.rim.or.jp/~shikap/security/awacs/snort191.htmlに置いておきますので，参考にしてください．
注15：同じく場当たりパッチをhttp://www.yk.rim.or.jp/~shikap/security/awacs/snort191.htmlに置いておきますので，自己責任で使ってください．

います．たとえば**snmpv2c**を用いる場合，リスト**1**のような行を追加すると**trap**が送信されます．**ID**は任意の正数です．オプションは「**[c],[p[m|s]]**」という書式を取り，それぞれの意味は表**2**のようになります．表**2**以外の引数は**SNMP**の機能ですので，ここでの説明は割愛します．

●実行

当初，動作確認をしていたところ，「そのようなプラグインはない」と出力されうまく動かず途方に暮れていましたが，検証の結果，正常に動作することを確認しました[注16]．ぜひ試してみてください．

SMBサポート

●SMBサポートとは

SMBサポートは，**SNMP**サポートと同様に別マシンにアラートを送る拡張機能で，その出力に**SMB**を用います[注17]．このためこの機能は，監視を**Windows**で行いたい方のためのサポートとして考えられています．

●インストール

この機能を利用するためには，**Samba**がインストールされている必要があります．今回はテストと言うことで**Red Hat Linux 8.0**標準の**Samba 2.2.5**を使用してみました[注18]．

コンパイルについては，**SNMP**サポートと異なり，**./configure --enable-smbalerts**も**make**も素直に通ります．

●設定

この機能も**snort.conf**に設定を書き込むことで使用できますが，**snort.conf**内にはどこにも記述がありませんので，**SNMP**サポートの近傍に追加します．書式はリスト**2**のとおりです．

表2 オプション「[c],[p[m|s]]」の意味

オプション	説明
c	簡単な出力を行う
p	アラートの元になったパケットを出力
m，s	それぞれMD5，SHA1を用いてパケットを出力する

リスト1 SNMPサポート使用時のsnort.confへの記述方法

```
output trap_snmp: alert, snortのID, オプション, {trap|inform} -v 2c -p ポート番号 trapを受け取るマシンのIPアドレス コミュニティストリング
```

●実行

SNMP同様に当初はうまく動きませんでしたが，その後正常に組み込まれたことを確認しました[注16]．適当な**SMB**クライアントがなく完全にはテストできていませんが，**NetBIOS**の名前解決を行っていることは確認しています．

データベースサポート

●データベースサポートとは

データベースサポートを使うことで，各種データベースに**Snort**の出力を記録することができるようになります．この機能を使えばログの閲覧が非常に楽になります．のちほど**ACID**というログ管理ツールについて説明しますが，このツールもデータベースを利用しているので，**ACID**を使う場合にはこのサポートが必須となります．

ここでは**PostgreSQL**サポートについて説明を行いますが，そのほかのデータベース（**MySQL**，**unix ODBC**，**SQL Server**，**Oracle**）についても同様の流れでインストールできると思います．なお，データベースのインストール／設定については割愛します[注19]．

●インストール

まず，**postgresql-devel**がインストールされている必要があります．あとは**./configure**時に**--with-postgresql**オプションを付けることを除けば，通常のインストールと同じです．

●設定

snort.confを修正します．**snort.conf**の**421**行目以降がデータベースサポートの設定に関する部分です．書式はリスト**3**①のとおりです．パラメータ部には，**user**，**password**，**dbname**，**host**などを記述します．一般的なパターンはリスト**3**②のものでしょう．この場合，**localhost**で実行されている**PostgreSQL**のデータベース**snort**に，ユーザ名**snort**，パスワード**test**で接続，アラートを記録する，ということになります．

リスト2 SMBサポート使用時のsnort.confへの記述方法

```
output alert_smb: 送信マシンのリストを列記したファイル名
```

注16：なぜかRed Hat Linux 8.0，Vine Linux 2.6r1ともアウトプットのプロセッサが全滅してしまいました（本稿執筆時）．snort-usersMLにも同様の投稿がありましたが，解決法が投稿される気配がないようです．ソースを見たところ，snort.cでプロセッサのイニシャライズを行う順番が間違っていることが原因のようです．-cオプションを付けた場合は正常に動作します．
注17：Windowsはほとんど使わないので，今回初めて試してみました．
注18：Sambaは比較的頻繁にセキュリティ問題が確認されていますので，情報の収集を欠かさないようにしましょう．
注19：インストール手段だけでなく，使用するデータベースそのものの使い方にも慣れておいたほうが良いと思います．経験者が言っているのですから本当ですよ;-)．

2-2-17__初 04.1.13 9:02 PM ページ 192

●データベースの作成

あらかじめアラートを記録するデータベースを作成しておく必要があります．まず，postgresユーザの権限を取得し，createuserコマンドでsnort.confで設定したユーザ名（前述の「インストール後の作業」で説明したSnort専用ユーザが良いでしょう）を登録しましょう．このとき，新しいデータベースを作成することはできても，新しいユーザは登録できないように設定したほうが良いと思います．

次に設定したユーザの権限を取得し，createdb -Wデータベース名（snort.confで設定した名前）と実行し，snort.confで設定したパスワードを入力します．これでデータベースの「元」が作成されます．次にcontrib/に移動し，psql < create_postgresqlと実行してデータベースを初期化します．

●実行

これで，データベースを使用してログ管理を行うことができます．こちらも-cオプションがないとうまく動作しません[注16]．注意して運用してください．

ここまで説明したプロセッサについては，当然すべての機能を含めてコンパイルすることも可能ですが，Red Hat Linux 8.0ではSNMPの時と同様の症状がlibnetのチェック時にも生じてしまいました．configureを修正すればコンパイルが可能です[注14]．

デフォルトで有効なプロセッサ

前述の機能以外にも多くの機能が存在します．また，それらはデフォルトコンパイルでも使用可能となっていますので，ぜひとも活用したいところです．

プリプロセッサ

まず，プリプロセッサについてです．これはパケットを検査する際に使用されるプロセッサです（表3）．これらプリプロセッサについては，デフォルトでの運用でも問題ないと思います．ただしstream4_reassemblyについては，後述するマルチコンテンツルールと相性が悪いので注意しましょう[注20]．

アウトプットプロセッサ

次に，アウトプットプロセッサについてです[注16]（表4）．これはアラートなどの出力の際に使用されます．これらについては自分にとって便利な方法のみ有効，それ以外は無効とし，CPU負荷を下げるようにしたほうが良いでしょう．

snortを便利にするツールの紹介

snortをより便利にするためのツールとして，ログ閲覧ツールとルールアップデートツールを紹介します．

ACID

●ACIDとは

データベースサポートを使用するなら，ぜひとも使ってほしいのがACIDです．これを使用すれば，Web経由でsnortが出力したログを閲覧することができます．一度使うとこれ無しではいられなくなります[注21]．

●インストール，設定，実行

このソフトウェアは，Snortが使用しているデータベースが格納されているマシン上で動作させます．必要になるソフトウェアは多く，Apache，GD，PHP（データベース用とGD用のパッケージも必要です），PHPlot，ADODBです．PHPlot，ADODBを除けば，すべてRPMパッケージからインストール可能だと思います．残りに関しても，Apacheのdocument rootに展開するだけです[注22]．

次にACIDをインストールします．ACIDはcontrib/にあるので，これをApacheのdocument rootに展開しておけば動作します．

リスト3　データベースサポート使用時のsnort.confへの記述方法

①書式
```
output database: [log | alert], [データベースの種類], [パラメータ]
```
②一般的なパターン
```
output database:alert,postgresql,user=snort,password=test,dbname=snort,host=localhost
```

注20：その名のとおり，ストリームを再構築しますので，複数のコンテンツにヒットする可能性が高くなり，誤検知が増えることになります．これを防ぐにはルールの組み合わせ（本文で後述）を考える必要があります．

注21：ただし，SnortのアラートをWeb経由で出力したがために，それがルールに引っかかり，それがアラートとなり……という無限地獄が生じる場合がありますので，気を付けてください．

注22：WebサイトはそれぞれPHP：http://www.php.net/，Apache：http://www.apache.org/，ADODB：http://www.weblogs.com/adodb/，PHPlot：http://www.phplot.com/，GD：http://www.boutell.com/gd/です．使用するパッケージが多いため，それだけセキュリティにも注意する必要があることは言うまでもありません．

2-2-17__初 04.1.13 9:02 PM ページ 193

展開時に作成されたディレクトリ内の設定ファイル，**acid_conf.php**を修正します（表5）．

あとはデータベースサーバと**Apache**が起動しているのを確認後，**Web**ブラウザで`http://ACID`が動作しているマシンのIPアドレス`/acid/`にアクセスします．エラーが出なければ，右上付近にあるボタン

表3　プリプロセッサ

プロセッサ名	オプション名	説明
frag2（パケット再構築を行う）	timeout	タイムアウトの時間を指定
	memcap	frag2が用いるメモリサイズを指定
	min_ttl	最小TTLを指定
	ttl_limit	TTLの許容ずれ量を指定
stream4（ステートフルインスペクション／ストリーム再構築）	detect_scans	ステルスポートスキャンの検知
	detect_state_problems	TCPの変移状態が異常かどうか検知
	disable_evasion_alerts	シーケンスの重複などのアラートを停止
	ttl_limit	TTLの許容ずれ量を指定
	keepstats	セッションの統計を保存する．フォーマットも指定可能
	noinspect	ステートフルインスペクション機能を停止
	timeout	タイムアウト時間を指定
	memcap	stream4が使用するメモリ量を指定
	log_flushed_streams	イベント検知時，バッファ内のパケットをディスクにフラッシュする
stream4_reassemble（ストリーム再構築）	clientonly	セッションのクライアント側からの通信のみ再構築する
	serveronly	セッションのサーバ側からの通信のみ再構築する
	both	クライアント側，サーバ側通信双方を再構築する
	noalerts	ここでのアラートを停止する
	ports	再構築を行うポート番号を指定する
http_decode（HTTP通信の正規化）	ポート番号	対象とするポートを指定
	unicode	Unicodeを正規化する
	iis_alt_unicode	IIS用オプション
	double_encode	2重エンコードをアラート対象にする
	iis_flip_slash	￥を\に正規化する
	fll_whitespace	￥tをスペースとして処理する（Apache用）
rpc_decode（RPC通信の正規化）	ポート番号	対象とするポート番号を指定
	alert_fragments	フラグメント化されたTCP通信をアラート対象にする
	no_alert_multiple_requests	パケット中に複数のRPCクエリがある場合をアラート対象にしない
	no_alert_large_flagments	サイズを越えたパケットをアラート対象にしない
	no_alert_incomplete	1つのセグメントがサイズを超えている場合をアラート対象にしない
bo（Back Orifice検知）	-nobrute	ブルートフォースを検知しない
	キー番号	トラフィック解読のデフォルトキー番号を指定
telnet_decode（telnetのネゴシエーション文字列の正規化）	なし	
portscan※（ポートスキャンを検知）	IPアドレス（レンジ）	検知対象のIPアドレス（レンジ）を指定
	2組の数字	ポートスキャンと判断するしきい値を指定
	ファイル名	ログファイルを指定
portscan_ignorehosts（portscanプロセッサが無視するIPアドレスを指定）	IPアドレス（レンジ）	無視するアドレス（レンジ）を指定
arpspoof（ARP偽装やARP一覧作成などを検知する）	-unicast	ユニキャストでのARPリクエストを検知する
	arpspoof-detect-host	検知するホストの一覧を指定
	IPアドレス MACアドレス	検知するホストのアドレス対応を記述
asn1_decode（ASN.1プロトコルの不正使用を検知）	なし	
fnord（ポリモーフィックシェルコード検知）	なし	
conversation（TCP，UDP，ICMP通信のトラッキング．portscan2に必要）	allowed_ip_protocols	許可するプロトコルを指定
	timeout	タイムアウト時間を指定
	max_conversations	一度に対応する通信の最大値を指定
	alert_odd_protocols	allowed_ip_protocolsで指定しなかったプロトコルを指定
portscan2（ポートスキャン検知）	scanners_max	検知するスキャナの最大値を指定
	targets_max	スキャン対象の最大値を指定
	target_limit	スキャン対象かどうかのしきい値（ホスト）を指定
	port_limit	スキャン対象かどうかのしきい値（ポート）の最大値を指定
	timeout	タイムアウト時間を指定
	log	ログディレクトリを指定
perfmonitor（パフォーマンスモニタ）	なし	

※引数はこの順序でなければいけません．

2-2-17__初 04.1.13 9:02 PM ページ 194

をクリックして下さい．もう一度同じ**URL**にアクセスすれば，**ACID**のメイン画面が表示されます．これで設定は終了です．

ただし，このツールには**1**つ難点があります．それはセキュリティ機能が皆無と言って良いことです．**SSL**を使用するなどして認証を用いることでセキュリティの確保を図ってください．

表4　アウトプットプロセッサ

プロセッサ名	オプション名	説明	
alert_syslog （syslogにアラートを転送）	ファシリティ	syslogのファシリティを指定	
log_tcpdump （tcpdump形式でログを出力）	ファイル名	ログファイル名を指定	
database （データベースにログを出力）	log|alert	データベースに出力するファシリティを指定	
	mysql|postgresql|odbc|mssql|oracle	データベースの種類を指定	
	パラメータ	以下の種類のパラメータを指定（パラメータ=値）	
		dbname	データベース名
		host	データベースのあるホストを指定
		port	データベースが使用するポート番号を指定
		user	データベースに接続するユーザ名を指定
		password	接続時に使用するパスワードを指定
		sensor_name	snortセンサーの識別名を指定
		encoding	ペイロードのエンコードを指定 （hex，base64，asciiのいずれか）
		detail	データの出力レベルを指定（full，fastのいずれか）
		ignore_bpf	BPFフィルタを毎時生成するかどうか （no，0，yes，1のいずれか）
xml （XML形式でログを出力）	log|alert	ログ出力のファシリティを指定	
	パラメータ	以下の種類のパラメータを指定（パラメータ=値）	
		file	ログを出力するファイル名を指定
		protocol	ログを出力するプロトコルを指定 （http，https，tcp，iapのいずれか）
		host	ログを出力するホストを指定
		port	ログを出力するポートを指定
		cert	X.509証明を指定
		key	秘密鍵を指定
		ca	CAを指定
		server	正当なサーバリストのファイルを指定
		sanitize	削除したいIPアドレス（レンジ）を指定
		encoding	ペイロードのエンコードを指定 （hex，base64，asciiのいずれか）
		detail	データの出力レベルを指定（full，fastのいずれか）
unified （snort unifiedバイナリでログを出力）	filename	出力ファイル名を指定	
	limit	最大ファイルサイズをMバイト単位で指定	
trap_snmp SNMPトラップを生成）	log|alert	出力のファシリティを指定	
	パラメータ	以下の種類のパラメータを指定（snmpv2cの場合，v3は省略）	
		ID番号	snortのID番号
		出力オプション	通知レベルを指定（[c],[p[m|s]]）
		trap|inform	出力形態を指定
		-vバージョン	SNMPバージョンを指定
		-pポート番号	ポート番号を指定
		-cコミュニティ名	コミュニティ名を指定
		ホスト名	ホスト名を指定
alert_smb （Windowsにメッセージを送信）	ファイル名	アラートを送信するマシンのNetBIOS名を記したファイルを指定	

表5　acid_conf.phpの変更点

行番号	変更前の記述	修正部分
12	$DBlib_path = ""; $DBtype = "mysql"; $alert_dbname = "snort_log"; $alert_host = "localhost"; $alert_port = ""; $alert_user = "root"; $alert_password = "mypassword"; $ChartLib_path = ""	""内にADODBのパスを記述 ""内にデータベースの種類を記述 ""内にデータベース名を記述 ""内にデータベースホストを記述 ""内にデータベース用ポート番号を記述 ""内にデータベースアクセスユーザ名を記述 ""内にパスワードを記述 ""内にPHPlotのパスを記述
20		
32		
33		
34		
35		
36		
54		

注23：ただし，ApacheをIISに偽装させて運用し，スクリプトキディの行動を確認するなど，時には使ったほうが良い場合もあります．
注24：筆者は自宅／会社ともにトラフィックが少ない箇所で運用していますので，あえて誤検知を減らさず運用し，ルールの癖を確認したりしています．これも1つのポリシーかと:-P．

oinkmaster

●oinkmasterとは

Snortを使ううえで最も面倒な作業は，ルールセットの保守です．日々増え続ける攻撃手法に対し，snort.orgでは定期的にルールをアップデートしています．本当に頭が下がる思いです．これらのルールを自分のSnortセンサに登録するのは，作業そのものは簡単なのですが，非常に手間がかかります．

その登録作業を自動的に行ってくれるツールがあります．それがoinkmasterです．これはPerlで作成されたスクリプトです．

●インストール

Perlスクリプトですからインストールといっても，ツール実行用ユーザを作成し，oinkmaster.plとoinkmaster.confを適当なディレクトリ（たとえば，それぞれ/usr/local/bin/と/usr/local/etc/）にコピーし，oinkmaster.confを修正するだけです．

●設定，実行

ただ使うだけならば，図2のコマンドで実行可能です．ただしこのようにすると，変更してほしくないルールまで変更されてしまいます．このため，confファイルで変更してほしくないルールのSIDを記述したり，変更してほしくないルールファイルを記述することで，これらの変更を停止することができます．詳しくは後述をご覧ください．

ルールの記述とTips

それでは，ルールセットの記述に関して，さらに詳しく解説します．なおSnortのルールは，ルールファイルをエディタで開いてみるとわかるとおりテキスト形式です．このためユーザが必要に応じて修正することが簡単にできます．

ルールセットの取捨選択

修正について説明する前に，ルールセットの取捨選択について述べておきます．snort.orgのルールセットは，検知対象や種別によってルールファイルが分かれています．この中にはたとえば，web-iis.rulesのように単一のプロダクトに的を絞ったルールファイルや，icmp-info.rulesのようにデフォルトでは使用しないルールファイルなどがあります．

ルール取捨選択の第一歩は，これらをうまく選ぶことです．一般的には，IISを使っていないのにweb-iis.rulesを使う必要はないと思います[注23]．また，icmp-info.rulesのようにネットワークの状況判断には役立つかも知れないけれど，頻繁にアラートが出るため実運用に支障が出るようなルールファイルもあります．これらのルールセットをどう取捨選択するかは結局のところ，そのネットワークの運用状況によって大きく異なります．

月並みかもしれませんが，まずは自ネットワークで運用しているものに該当するルールファイルをすべて利用してみて，必要に応じてファイルを取捨選択する，ということになると思います[注24]．

個々のルール

次に，個々のルールについてみていきましょう．ルールの基本構造とオプションの意味は図3，表6のとおりです．これらを組み合わせて，攻撃時に発生するパケットがマッチするルールを記述していきます．また前述のとおりflexrespを使う場合には，ルール内に記述する必要があります．リスト4①に簡単なルールの例を示します．

このように，ルールの記述そのものはあまり難しくはありません．難しいのは，やはりTCP/IPパケットの構造を理解していなければならないことと，検知したいパケットがどのようなものであるかをあらかじめ知っていなければならない，ということでしょうか．

●攻撃パケットと安全なパケットの両方にマッチしてしまう例①

最も気を付ける必要があるのは，攻撃パケットにマッチするものの，安全なパケットにもマッチしてしまうようなルールを記述しないようにすることです．

たとえば，バッファオーバーフロー攻撃が成功したかどうかを試すとき，攻撃者はidコマンドを発行しますから，その返答が送信されたらalertを出力する，というルールを作るとします（リスト4②）．ところが，BugtraqなどでExploitの説明をするときによくこのコマンドのコマンドの結果が書かれていたりしますか

図2 oinkmaster実行方法

```
oinkmaster.pl -o アップデートを行うディレクトリへのパス -b バックアップを行うディレクトリへのパス
```

2-2-17__初 04.1.13 9:02 PM ページ 196

ら，メール受信の際にこのルールが合致し，誤検知となってしまうことがあります．このような場合，そのルールを使わない，**POP3**のポートではこれを検知しない，場合によっては**flow**オプションを用いることでトラフィックの流れを明示して誤検知を防ぐ，などの工夫が必要となります．

●攻撃パケットと安全なパケットの両方にマッチしてしまう例②

これもメールでの例ですが，ウィルスが送信されてきたことを知らせるルールをリスト**4**③のように書いたとします．旬のウィルスの場合はいろいろなところで話題になるので，当然**ML**などで話題に出るたび誤検知することになります．この場合，ウィルスの話題ではヘッダ構造まで書くところは少ない，という点に着目すれば，少しだけ誤検知が減ることになります．

表6　ルールのオプション

ルール名	説明
msg	ログ出力の際使用するメッセージを指定
logto	パケットを引数で指定したファイルに保存
ttl	IPヘッダのTTLを評価
tos	IPヘッダのTOSフィールドを評価
id	IPヘッダのフラグメントIDを評価
ipoption	IPオプションフィールドを評価
fragbits	IPヘッダのフラグメントビットを評価
dsize	パケットペイロードサイズを評価
flags	TCPフラグを評価
seq	TCPシーケンス番号を評価
ack	TCP ACKフィールドを評価
itype	ICMPタイプフィールドを評価
icode	ICMPコードフィールドを評価
icmp_id	ICMPエコーIDを評価
icmp_seq	ICMPエコーシーケンス番号を評価
content	ペイロード中のパターンを検索
content-list	ペイロード中のパターンリストを評価
offset	contentオプションで評価するペイロードの開始位置を指定
depth	contentオプションで評価するペイロードの最終位置を指定
nocase	大文字小文字を区別しない
session	セッションのレイヤ情報を出力
rpc	RPCサービスを監視
resp	flexrespで通信を遮断
react	flexrespで通信を遮断
reference	外部リファレンスデータベースを参照
sid	SnortのルールIDを指定
rev	ルールのリビジョンナンバーを指定
classtype	ルールの分類を指定
priority	ルールの重要度を指定
uricontent	URI内のパターンを指定（URI専用content）
tag	ルールに一致したパケットを送信したソースアドレスの通信にタグを付ける（さらに詳細なログを作成するため）
ip_proto	IPヘッダ中のプロトコル値を指定
sameip	ソースアドレスとディスティネーションアドレスが一致するパケットを選択
stateless	TCPの変移状態を検知（flowオプションに移行した）
regex	パターンの正規表現を使用することを宣言
flow	TCPの変移状態を定義
fragoffset	フラグメントオフセットを定義
rawbyte	正規化されていないデータを記述

●誤検知が生じる例

ストリーム再構築を行っている場合，**content**を複数使用していると誤検知が生じてしまうことがあります．これはしくみ上生じてしまうため，**pass**ルールをうまく使う必要があります．これは，マッチしたらこのパケットは監視しない，というルールです．

ただし，デフォルトでは**alert**ルールを調べてから**pass**ルールを評価するため，不便な場合もあります．この順序は**-o**オプションで入れ替えることができます．そうすれば**alert**ルールにマッチしてしまう本当は無害なパケットを，あらかじめ**pass**ルールに記述しておくことにより誤検知を防ぐことができます．

リスト**4**④の場合はストリーム再構築が入ってしまうと，ストリーム内には必ず文字列“**USER**”が入るため，改行のない長い文字列が送信されているパケットを受け取るときに問題が生じてしまいます．そこでリスト**4**⑤のようにしておけば，一般のログイン処理を行った場合に評価が行われなくなりますから，誤検知が減ることになります[注25]．

このように，誤検知のないルールを書くのはなかなか骨の折れる作業です．検知したいパターンと誤検知となるパターンの違いをきちんと理解したうえで記述すれば誤検知は減りますから，ぜひ知的パズルと思って挑戦してみてください．

misc

Snortはツールの性格上，使いこなすには**TCP/IP**などのプロトコルに関する知識が必須です．また，ネットワークパケットから見た攻撃手法をどれだけ

図3　Snortのルールの書式

アクション（①）　プロトコル（②）　IPアドレス（③）　ポート番号（④）　通信方向（⑤）　IPアドレス（③）　ポート番号（④）　（ルールオプション;ルールオプション…）（⑥）（⑥）

①アクション：alert，log，pass，activate，dynamicのいずれか
②プロトコル：tcp，udp，icmp，ipのいずれか
③IPアドレス：アドレス/CIDRで指定（例：192.168.0.0/24）．～以外は！～と指定
④ポート番号：ポート番号を指定，カンマ区切りで複数指定．レンジ指定はA:Bの形式で指定．～以外は！～と指定
⑤通信方向：-> または <> で指定
⑥ルールオプション：オプション名:引数と指定

注25：この部分については，みっきー氏にヒントをいただきました．みっきー氏のWebサイト（http://www.hawkeye.ac/micky/）はSnortユーザ必見です．

2-2-17__初 04.1.13 9:02 PM ページ 197

知り，理解しているかがより誤検知の少ないルール作成の鍵となります．ですから**TCP/IP**の教科書を熟読するのみならず，必要に応じて実際に攻撃ツールを使用し，そのときに発生するパケットを調査することも時には必要となります．

また，通常時のパケットがどのようなものであるかを理解していないと，どのアラートが誤検知でどれが本物のアラートか区別できない，ということにもなります．社内のシステムでどのようなものが動いているかを理解していないと，大量のルールを取捨選択することすらできません．

厳しい言い方になりますが，このツールは侵入検知のみならず，システム管理者の質そのものを問いただすツール，ともいえるのです．狼少年にならないようにするためには，まず足場を固めなくてはいけない，ということになりますね．

おわりに

Snortの機能をできる限り説明しようと思うと，どうしても駆け足での説明しかできなくなってしまいますね．それくらい**Snort**は高機能であり，奥も深いのです．しかも，うまく動かないおまけ付きでした**:-(**．ただ，これもある意味ではチャンスかもしれません．今こそオープンソースに貢献できるいいタイミングなのだ，と自分に言い聞かせていたりします**:-)**．ともあれ，これらの機能をフル活用すれば，守る側にとってこれくらい頼もしいツールもなかなかないのではないでしょうか．

まだまだ発展し続ける**snort**．ぜひ必須ツールの**1**つに数えてみてください．SD

リスト4　Snortのルールの例

```
①簡単なルールの例
●ユーザhogeのFTPログインを検知
alert tcp any any -> 192.168.0.0/24 21 (content:"USER hoge";nocase;msg:"FTP:USER hoge login attempt";)
●DNSサーバに/bin/shという文字列が送信されたことを検知
alert udp any any -> $HOME_NET 53 (content:"/bin/sh";msg:"Why send "/bin/sh" to DNS?";)
●ポート1524に仕掛けられたバックドアに通信開始しようとする通信を遮断
alert tcp any any -> 192.168.0.0/24 1524 (flags:S; resp:rst_all; msg:"Root shell backdoor attempt";)

②攻撃パケットと安全なパケットの両方にマッチしてしまう例①
alert ip any any -> any any (msg:"ATTACK RESPONSES id check returned root"; content: "uid=0(root)";)

③攻撃パケットと安全なパケットの両方にマッチしてしまう例②
alert tcp $POP3server 110 ->$MY_CLIENT any (content:"big@boss.com";msg:"Virus from big@boss.com :-)";)

④誤検知が生じる例
alert tcp $EXTERNAL_NET any -> any 110 (msg:"POP3 USER overflow attempt"; content:"USER "; nocase; content:!"|0a|";
within:50;)                                                                                    （実際は1行）

⑤passルールの使用例
pass tcp any any -> any 110(content:"USER"; content:"PASS";)
```

hogwash

本文のとおり，Snortには通信切断の機能が含まれています．この機能をうまく使えばファイアウォールの代わりができないか，と考えた方もいらっしゃるかもしれません．筆者もそう考えた1人でした．しかし，やはり世界は広く，すでにそのようなプロジェクトが活動しているのでした．

それがhogwashです．最新版は0.4pre1であり，本家Snortをしのぐような機能も有しています．このツールは，Snortと同様のルールによって通信の許可／不許可を設定します．ポート番号だけのルールならば一般的なフィルタリングになりますし，「ユーザhogeにはFTPを使わせない」などというルールも作成可能です．ここまではSnortでもできるでしょう．

なによりも素晴らしい機能は，replace機能でしょう．パケットペイロードの文字列を違う文字列に変換したうえでルーティングすることが可能です．たとえば，バッファオーバーフロー攻撃を受けた場合，そのペイロードにはたいてい/bin/shという文字が含まれているのですが，これを/pin/shと書き換えてルーティングさせてしまえば攻撃は失敗します．しかも，攻撃者には「なぜ攻撃が失敗したのかわからない」ということになります．

この機能は，最近いろいろなところで話題になりつつあるハニーポットの構築にとても役立ちます．攻撃されているが攻撃されていない，という状況を作り出すことができるわけです．実際，ハニーポットを用いて攻撃者の行動を研究しているhoneynet.orgでもこのツールを紹介しています．

ただし，Snortを含むすべてのIDSと同様にfalse positiveが生じますし，ふつうのIDSと異なり，通信を遮断したりパケットペイロードを改竄したりと機能がけっこう過激ですので，運用には細心の注意が必要であることは間違いありません．

こういったマイナス面をふくめても，このプロジェクトは非常に技術者心をくすぐるものですので，ぜひ皆さんにも試してほしいと思います．