Linuxによるセキュリティ入門（4）
― ログの管理（syslog基本編）―

西　村　竜　一

Ⅰ．ログをチェックしよう

今回はログの話をしましょう。ログ（log）というのは，みなさんご存じだと思いますがサーバの状態や通信内容などの記録のことです。稼働している計算機は，おそらく常になんらかのログを出力しており，それらのチェックはシステムを安定に運用するうえで重要な仕事の一つです。システムにトラブルが生じたときにログをチェックして問題を解決した人も多いでしょう。トラブル発生時のみではなく，日頃からログをチェックして，トラブルの発生を未然に防ぐことができるようになれば管理者として一人前です。

もちろんセキュリティの対策としてのログチェックは非常に重要な業務です。特にサーバプログラムへのアクセスを記録したアクセスログは，不正アクセスの検出に必須です。例えば，これはWebサーバApacheへのアクセスログの例です（本物ではなく少し加工しています）。

```
192.000.000.000 - - [25/May/2003:07:03:59 +0900] ``GET /default.ida?XXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX%u9090%u6858%ucbd3%u780
90%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%
u8b00%u531b%u53ff%u0078%u0000%u00=a  HTTP/1.0'' 404 277
```

実は，これは有名なNimdaウィルスがWebサーバへ攻撃してきたときに記録されるアクセスログです。おそらく同じようなログをWebサーバのアクセスログを見たことがある人なら何度も目にしていることでしょう。Nimdaは過去のウィルス，すでに対策されているのが普通であり，いまとなってはNimdaに感染することもないと思いますが，いまだに私の管理しているサーバにも一日何度も，この攻撃があり（つまりNimdaに感染しているホストが世の中にはまだ多数存在する），なぜそんな不用心な計算機が残っているのか不思議で仕方がありません。しかし，例は過去のウィルスNimdaのものでしたが，今後も似たような手段を用いたウィルスが登場してくる可能性があります。あきらかに不自然なアクセスログ（Nimdaのログは他の正常なアクセスログより異常に一行が長くなる）を発見することができれば，不正なアクセスを発見することができます。

攻撃を検知できても，その不正アクセスに対する適切な対処がされてなければ攻撃を防ぐことはできません。残念ながらログで攻撃を発見した時点では不正アクセスを許してしまった後かも

しれません。しかしながら，発見できなければ，あなたの計算機はいつまでも不正利用を許してしまうことになり，また今後の攻撃に対する対策をすることもできません。ログをこまめにチェックして，より強固なシステムを構築できるようあなたの管理スキルを研いてください。

それではここからは実際にログを読んでみましょう。Debianの場合，ログが記録されたファイルは/var/log/ディレクトリに保存されます。ファイルの中身を見るにはページャプログラムを用いますが，今回はDebianでは標準でインストールされているmoreを用いて説明します。less（jless）やlvなどの高機能のページャプログラムを使いたいときは，apt-getでインストールして使ってください。

/var/log/syslogというファイルを読んでみましょう。このファイルには，syslogdというログ管理デーモンが出力したログが記録されています。

```
% sudo more /var/log/syslog
```

/var/log/syslogファイルは，標準では一般ユーザでは読み込み不可に設定されています。読むときは，sudoコマンドを用いましょう。

また，tailコマンドに-fオプションを付けて，ログファイルを読み込ませることでどんどん追加されるログを表示し続けることができます。例えば，

```
% sudo tail -f /var/log/syslog
```

となります。tailを終了するには，コントロールキーを押しながらCを押してください。

ログの中身は以下のような感じになっていると思います。

```
Jun  1 06:18:49 debian syslogd 1.4.1#10: restart.
Jun  1 06:18:50 debian kernel: klogd 1.4.1#10, log source = /proc/kmsg started.
Jun  1 06:18:50 debian kernel: Inspecting /boot/System.map-2.4.20
Jun  1 06:18:50 debian kernel: Loaded 14581 symbols from /boot/System.map-2.4.20.
Jun  1 06:18:50 debian kernel: Symbols match kernel version 2.4.20.
Jun  1 06:18:50 debian kernel: Loaded 74 symbols from 7 modules.
Jun  1 06:18:50 debian kernel: Linux version 2.4.20 (nisimura@debian) (gcc version
2.95.4 20011002 (Debian prerelease)) #1 Sun Mar 9 12:30:59 JST 2003
（途中省略）
Jun  1 06:38:49 debian -- MARK --
Jun  1 06:58:49 debian -- MARK --
Jun  1 07:18:49 debian -- MARK --
```

このログの前半は，システムが起動したときにLinuxカーネルが出力したものです。システムが認識したCPUや各種デバイスの構成が出力されています。ログには，出力されたときの時間が記録されています。この例では，“Jun 1 06:18:49”つまり，６月１日の６時18分にシステムが起動した（ログの記録が開始された）ことがわかります。“debian”というのはこのシステムのホストネームでみなさんが計算機に付けた名前が記録されていると思います。なお，少し余談ですがLinuxカーネルが出力したログを確認する手段としてはsyslogファイルを読む以外にdmesgコマンドを用いて確認することもできます。

```
% dmesg
```

先ほどの例の後半では“-- MARK --”という行が記録されています。これは定期的にシステムが記録するものであり，この行をチェックすることでシステムが稼働していることを確認できます。システムがクラッシュしたとき，システムはいつまで稼働していたのかを調べるときなどに利用します。

システムにメールサーバなどのサーバプログラムがインストールされている場合は同じく/var/log/syslogファイルにそのログも出力されているかもしれません。以下の例はメールサーバプログラム（MTA）のpostfixが出力するログです。

```
May 31 16:45:39 debian postfix/smtpd[6197]: connect from localhost[127.0.0.1]
May 31 16:45:39 debian postfix/smtpd[6197]: 4F9B053E4F: client=localhost[127.0.0.1]
May 31 16:45:39 debian postfix/cleanup[6198]: 4F9B053E4F: message-id=<0000@example.ac.jp>
May 31 16:45:39 debian postfix/smtpd[6197]: disconnect from localhost[127.0.0.1]
```

このようにサーバプログラムにはログの出力にsyslogdを利用するものがあります。

また，逆にログの出力にsyslogdを利用できない（しない）プログラムもあります。例えば，有名なWebサーバプログラムであるApacheはsyslogdを利用しません。Debianの標準設定のApacheでは，ログは/var/log/apache/ディレクトリ以下のファイルにプログラムが自分自身で（syslogdを経由しないで）保存します。

このようにLinuxでは大きく分類すると２通りのログ保存方法が提供されています。つまり，

- syslogdを利用する
- syslogdを利用しない

です。syslogdは，UNIX系OSでは古くから利用されているロギング（ログ収集）デーモンです。Linux以外のFreeBSDやSolarisでも使われており，ネットワークを利用したサーバ・クライアント機能により複数のクライアントのログをサーバで集中的に管理できるようになっています。

先ほど，ログの管理方法にはsyslogdを利用するものと利用しないものの二種類あると書きましたが，サーバプログラムによっては，syslogdを利用するようにも，しないようにも設定でき

るものもあります。また，場合によっては，syslogdが利用可能なプログラムでもsyslogdを利用しないように設定した方が良い場合もあります。これは一つでも余分なデーモンは動かさない方が良いためです。syslogdもデーモンプログラムの一つであり，過去にはセキュリティホールが発見されたこともあります。また，syslogd以外の同様の機能を提供する新しいプログラムも登場しています。しかし，syslogdは，現在，最も良く利用されているロギングデーモンであり，利用する機会も多いと思いますので，今回はこのsyslogdの設定方法について説明をすることにしましょう。

Ⅱ．syslogd設定のキホン

1　syslog.confの基本書式

syslogdの設定を行いましょう。とは言っても，Debianの場合，標準の設定をそのまま使っても問題ありません。まずは標準の設定がどのような設定になっているかを確認しましょう。syslogdの設定は/etc/syslog.confファイルで行います。ファイルの中身を見てみます（図１）。

各行は，タブで区切られたセレクタフィールドとアクションフィールドで構成されています。つまり，

> セレクタフィールド<TAB>アクションフィールド

という形になっており，図１中の１行目では，auth,authpriv.*がセレクタフィールド，/var/log/auth.logがアクションフィールドです。セレクタフィールドは送られてくるログの分類を定義したものです。アクションフィールドは，その分類のログが送られてきたとき，syslogdがどのような動作をするかを定めたものになっています。

/etc/syslog.confの中では“＃”の後はコメントなので無視されます。

2　セレクタフィールド

まずは，セレクタフィールドの詳細を見てみましょう。セレクタフィールドはさらにピリオド（“．”）で区切られるファシリティとプライオリティの二つの部分で構成されます。図１の最終行を例に挙げると，mailがファシリティ，errがプライオリティです。

```
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
#cron.*                         /var/log/cron.log
daemon.*                        -/var/log/daemon.log
kern.*                          -/var/log/kern.log
lpr.*                           -/var/log/lpr.log
mail.*                          -/var/log/mail.log
user.*                          -/var/log/user.log
uucp.*                          /var/log/uucp.log
mail.info                       -/var/log/mail.info
mail.warn                       -/var/log/mail.warn
mail.err                        /var/log/mail.err
```

図１　/etc/syslog.confファイルの例

ファシリティとはそのログの種類を示したものです。例えば，mailはメール関連のプログラムが出力したログのことを指します。Linuxで利用できるファシリティのリストを表１に示します（OSによって利用できるファシリティには違いがあります。また特殊なファシリティは除外しています）。ログのタイプをこのようにわけることで，ログが送られてきたときの動作をタイプに応じて分離できるようになっているわけです。

syslogdを用いてログを出力する場合，各デーモンプログラムは，自分自身がどのファシリティに属すべきなのかを知っていなければなりません。すこしわかりにくいと思いますので，ここでは，ソフトウェア開発者の視点に立って，具体的な説明をしたいと思います。開発者は，プログラム中syslogdでログ出力させたい場合，openlog()とsyslog()関数（C言語の場合です）を用います。このopenlog()関数では引数（パラメータ）として，先ほどのファシリティを指定する必要があります。つまり，メール関係のソフトウェアを作った場合，mailファシリティを使用するのだと開発者が正しく定義する必要があります。これが間違っているとメール関係ソフトウェアなのにプリンタ関係のログファイルにログが出力されてしまうことになります。既存の世の中に出回っているソフトウェアでは，ファシリティは開発者によって正しく定義されていると思います。そのソフトウェアがどのファシリティを利用するかを知るには付属のドキュメントを参照するか，ドキュメント内に説明がない場合はソースコードを直接参照してください。

プログラムによっては，ファシリティが固定ではなく設定ファイルによって自由に変更できるものがあります。また，私たちが新しいサーバプログラムを開発してsyslogdでログ出力させたい場合は，どのファシリティを使うように設定するのが適切なのでしょう。メールプログラムのような表１の分類中にあきらかにあてはまる場合は，そのファシリティを設定すべきなのはわかります。もし，適切な分類が存在しないなら，他のファシリティに属さないデーモンのファシリ

ティであるdaemonを設定することになります。しかし，このままだと他のdaemonファシリティを使用するプログラムのログもまじって出力されてしまいます。それを避ける場合は，local1からlocal7の７個のファシリティが用意されているのでそれらを利用することになります。これらlocal1からlocal7は，みなさんのシステムにおいて独自に利用できるファシリティとして用意されているものなので自由に使って構いません。

表１　ファシリティのリスト

	分類内容
auth	セキュリティや認証（ログインなど）関係
authpriv	セキュリティや認証（ログインなど）関係（プライベート）
cron	時間によって動作するデーモン（cron と at）
daemon	他のファシリティに属さないデーモン
kern	カーネルが出力したシステムログ
lpr	プリンタ関係
mail	メール関係
news	NetNews関係（innなど）
syslog	syslogd内部で利用
user	一般的なユーザレベルメッセージ（デフォルト）
uucp	UUCP関係
local0 ～ local７	その他（ローカル使用）

表２　プライオリティのリスト

	意味
debug	デバッグ用ログ（正常動作）
info	一般的な報告に関するログ（正常動作）
notice	重要なログ（正常動作）
warning（warn）	警告ログ（警告）
err（error）	エラーログ（異常）
crit	重大なエラーログ（重大な異常）
alert	すぐに何らかの対処が必要なログ（かなり重大な異常）
emerg（panic）	緊急の対処を必要とするログ（システムに致命的な異常）

つぎにプライオリティの説明です。プライオリティはログの重要性を示したものです。プライオリティとして使えるキーワードは，表２になります。プライオリティの意味を見るとリストの下に行くほど重要性が高いとわかります。なお，warningとwarn，errとerror，emergとpanicは，同じものです。

プライオリティを決めるのもソフトウェアの開発者です。開発者は，syslogd経由でログ出力する際には，そのログの重要性を判断しログが適切なプライオリティで出力されるように定義しなければなりません。

セレクタフィールドの具体例を見てみましょう。図１の最後３行，“ mail.info ”，“ mail.warn ”，“ mail.err ”はそのファシリティからわかるようにメール関係のログを示すセレクタです。プライオリティにはinfo，warn，errが記述されています。このようにプライオリティだけを記述した場合は，その指定されたプライオリティ以上のすべてのプライオリティを指定したことになりますので注意してください。つまり，mail.errは，err，crit，alert，emergのプライオリティを指定したことになります。errプライオリティのみを指定したい場合はmail.=errとプライオリティの前にイコール（“ = ”）を付けて記述します。

複数のセレクタを併記するときはセミコロン（“ ; ”）を使います。図１の２行目，*.*;auth,authpriv.noneは，*.*とauth,authpriv.noneの２つのセレクタを併記したものです。それぞれのセレクタの指す条件が重なる場合，後者のセレクタが前者のセレクタの範囲を上書きすることになります。

もう少し細かく見てみましょう。先ほどの例で，前者のセレクタは*.*です。アスタリスク（“ * ”）はすべてのファシリティまたはプライオリティを指定するための特殊な記号です。つまり，*.*はすべてのファシリティのすべてのプライオリティのログを指定したことになります。後者のセレクタは，auth,authpriv.noneです。コンマ（,）は同じプライオリティを示す一文の中に複数のファシリティを指定するための記号です。auth.noneとauthpriv.noneを併記したことになります。noneはこれも特殊な記号で，それが与えられたファシリティについては，すべてのプライオリティを除外することを意味します。よって，auth,authpriv.noneは認証関係のファシリティにおいては，すべてのプライオリティを除外することになります。

この結果，“ ; ”で結合されていますので，*.*;auth,authpriv.noneはすべてのファシリティのすべてのプライオリティのログ，ただし認証関係のものは除外する，という意味になります。

ここまでの説明を少し応用すると，認証関係のものは除外した全ファシリティの全プライオリティのログ，ただし，infoプライオリティのみは認証関係もログも含める。といった複雑な指定もできるようになります。その答えは*.*;auth,authpriv.none;auth,authpriv.=infoです。

また，セレクタにはもう一つ特殊な記号としてエクスクラメーションマーク（“ ! ”）を用いることができます。これはプライオリティに対して使える記号であり，そのプライオリティとそれよりも高いプライオリティのログのすべてを無視するときに用います。例えば，newsファシリティにおいて，infoプライオリティより高く，alertプライオリティより低い範囲のログを指定する場合のセレクタはnews.info;news.!alertとなります。

３　アクションフィールド

つぎにアクションフィールドの説明をします。アクションフィールドは，セレクタフィールドによって分類されたログをsyslogdがどうするかの動作を指示するためのフィールドです。

再び図１の例を見てください。すべての行のアクションフィールドには/var/logディレクトリ以下のファイル名が指定されています。つまり，それらのログはここで指定されたファイルに保存せよということになります。例えば，先頭の２行の

```
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
```

は，認証関係のすべてのログ（auth,authpriv.*）は/var/log/auth.logに保存する指示になり，認証関係を除くすべてのログ（*.*;auth,authpriv.none）は/var/log/syslogに保存という意味になります。ファイル名は“/”で始まるフルパスで指定する必要があります。

また，保存先のファイル名の前にハイフン（“-”）を付けるとファイルをディスクに書き出す際にバッファリングをするようになります。バッファリングを行わないとハードディスクに細かいデータを頻繁に書き出すため，ディスク関連の負荷が上昇し，システムパフォーマンスの低下が生じることがあります。バッファリング（ある程度の量のデータをためてから書き出す）を行うことにより，ある程度，負荷の上昇を抑えることができます。しかし，バッファリングをしている最中にシステムがクラッシュしてしまうと，たまっていたデータはディスクに書き出されずに消えてしまいます。そのため，重要なログの保存にはバッファリングを行わないように設定するのが良いでしょう。先ほどの例では，特に重要な認証関係のログの保存にはバッファリングを行わない（ハイフンなし），それ以外のログの保存にはバッファリングを行う（ハイフン付き）ように切り分けられています。

アクションにはファイルへの保存の他に表３に挙げる動作を指定することができます。順にそれぞれを見ていきます。

表３　アクションのリスト

	動作の説明
ファイル	ログをファイルに保存
ターミナル	ターミナル（tty）へログを送る
コンソール	/dev/consoleへログを送る
リモートコンピュータ	他のコンピュータへログを送る
ユーザ	指定されたユーザに対して通知する
全ユーザ	システムにログインしている全ユーザに対して通知する
名前付きパイプ	mkfifoによって作られたパイプにログを送る

まず，「ターミナル」と「コンソール」です。アクションにターミナルを指定することで，そのターミナルにログを表示することができます。「ターミナル」とは何でしょう？細かい説明は省略しますが簡単に説明します。昔，UNIXシステムにログインするとき，テレタイプライターという

端末を用いてユーザはホストコンピュータを利用していました。現在はテレタイプライターを使うことはないと思いますが，このテレタイプライターとホストコンピュータとの接続をソフトウェア的に再現することでリモートログインやX Windowで用いるxtermやktermなどのターミナルソフトを実現しています。説明が悪いですが接続における窓みたいなものだと思ってください。例えば，Linuxの場合，xtermを一つ開くと/dev/ttyp?（?には数字が入ります）と呼ばれるソフトウェア的な仮想ターミナルを開きます。そして，そのターミナルを通じてbashやcshなどのシェルを利用することができます。「コンソール」とは，ターミナルの特殊なものでシステムにおいてメインとなるターミナルです。もともとはホストコンピュータに直接つながっているシステム管理に用いるターミナルのことでした。syslogdでは，これらターミナルやコンソールにログを出力することができます。少し前まで，高性能なホストコンピュータが一台あって，それをさまざまなターミナルを通じて複数の人が共有して利用するというのがUNIXの一般的な使い方でした。ホストコンピュータにはコンソールが接続されています。そして，そのコンソールにsyslogを通じてシステムのログを常に表示するのが一般的でした。コンソールを見れば，管理者はホストコンピュータの状態を確認することができたため広く利用されていた方法です。今では，ログをコンソールに表示する使われ方は減ったように思います。もし，コンソールに常にログを表示したい場合は/etc/syslogd.confのアクションフィールドに/dev/consoleと指定してください。しかし，それよりもターミナルでのログの表示には冒頭で紹介したtailコマンドの利用をおすすめします。

少し余談が長くなってしまいました。話を本筋に戻します。つぎのアクションフィールドは「リモートコンピュータ」です。syslogdでは，TCP/IPを用いてログをネットワーク上の他のコンピュータに送信することができるようになっています。こうすることで多くのコンピュータのログを一台のサーバで集中的に管理することができます。リモートコンピュータの/etc/syslog.confでの設定は以下のようになります。

```
mail.*                          @loghost
```

この設定ではmailファシリティのログをloghostという名前のコンピュータに送信します。なお，loghost上ではsyslogdがログをTCP/IPで受信できるモードで動いていなければなりません。その設定に関しては，後の「設定の変更とsyslogdの再起動」で説明します。

つぎのアクションフィールドは，「ユーザ」です。以下の例のようにアクションフィールドにログを通知したいユーザのアカウントを列挙します。

```
*.alert                         root,nisimura
```

そのアカウントのユーザがログインしている場合，そのユーザのターミナルにログを出力して注意を促すことができます。

さらに強力な動作でログインしている全ユーザのターミナルにログを出力するのが,「全ユーザ」アクションです。このアクションでは，wallコマンドを利用して全ログインユーザのターミナルにログを表示します。ログをすべてのユーザに通知するかなりインパクトのあるアクションです。設定は，以下のようにアクションフィールドをアスタリスク（“ * ”）にします。

```
*.=emerg                        *
```

最後のアクションは,「名前付きパイプ」です。このアクションを利用することでログを他のプログラムの標準入力へ送ることができます。mkfifoコマンドによって作られた特殊なファイルであるFIFO（名前付きのパイプ）を用いてパイプを実現します。申し訳ありませんが，この機能の説明は少し複雑になってしまうので今回は説明を省略します。詳しくは，以下のsyslog.confとmkfifoのオンラインマニュアルを参照してください。

http://www.linux.or.jp/JM/html/sysklogd/man5/syslog.conf.5.html

http://www.linux.or.jp/JM/html/gnumaniak/man1/mkfifo.1.html

さて，ここまでのおさらいとして/etc/syslog.confの設定例を２つ紹介しましょう。

```
mail.*;mail.!=info          /var/log/mail.log
```

この設定は，infoプライオリティを除くmailファシリティのログを/var/log/mail.logファイルに保存します。

```
*.=info;*.=notice;\
        mail,news.none  @loghost
```

mailとnewsを除くinfoとnoticeプライオリティのログをloghostに送ります。なお，/etc/syslog.confでは，改行の前にバックスラッシュ（“ \ ”）を入れることでひとつのルールを複数行に分けて記述することができます。

Ⅲ．設定の変更とsyslogdの再起動

/etc/syslog.confを編集して設定を変更したらsyslogdを再起動して設定を有効にしましょう。Debianでは，/etc/init.d/sysklogdを用いてsyslogdの再起動ができます。

```
% sudo /etc/init.d/sysklogd restart
```

次頁のように表示されたら，設定変更は終了です。

```
Stopping system log daemon: syslogd.
Starting system log daemon: syslogd.
```

先ほどアクションの説明の中でリモートコンピュータ（@loghost）にログを送信するには，受信側のsyslogdでの設定が必要と述べました。受信可能モードでsyslogdを動かすには，-rオプションをつけてsyslogdを立ち上げます。Debianでは，この設定は/etc/init.d/sysklogdの以下の箇所をエディタで編集することで可能です。

```
# Options for start/restart the daemons
#   For remote UDP logging use SYSLOGD="-r"
#
SYSLOGD=""
```

このうち最後の行を

```
SYSLOGD="-r"
```

とします。編集できたら

```
% sudo /etc/init.d/sysklogd restart
```

でsyslogdを立ち上げ直して完了です。気をつけてほしいのは，受信可能モードのsyslogdは，認証などをせずにすべてのリモートホストからのログを受信してしまうということです。開発がすすめられている新しいsyslogdでは，認証やアクセス制限ができるようになっていますが，現在の標準的なsyslogdではできません。このため，syslogdを受信可能モードで動かすコンピュータでは，ファイヤウォールや前回説明したパケットフィルタを用いたsyslogdのポートのアクセス制限をしてください。syslogdはUDPの514番ポートを使いますから，本連載で前回とりあげたiptablesによるパケットフィルタでは，

```
# iptables -A block -p udp --dport 514 -j DROP
# iptables -A block -p udp -s 192.168.1.10/32 --dport 514 -j ACCEPT
```

のように設定することになります。これで192.168.1.10からのみログを受信することができます。iptablesの設定の詳細は本連載の前回を参照ください。

Ⅳ．ログファイルのローテーション

システムを運用していると多くのログが出力されます。今回は，syslogdを経由して出力するログをファイルに保存する方法を紹介しましたが，syslogdを使わないプログラムも同様に大量にログを出力します。どんどんログファイルは肥大化し，このままではディスク容量の不足が生じます。もしディスクの空き容量がなくなったらシステムは不安定になり，そのままでは運用を続けることができません。

これを防ぐためには，ある程度古くなったログを捨てることが必要になってきます。例えば，以下のような操作でログファイル（/var/log/syslog）を捨てることができます。

```
% sudo cp /dev/null /var/log/syslog
% sudo /etc/init.d/sysklogd restart
```

この操作では，まずログファイルを削除（中身を空に）し，その後syslogdを再起動することで再びログの記録を開始しています。この方法は簡単ですが，最近出力されたログも同時に消去されてしまい，いざ必要というときに読むことができません。そこでログのローテーションが必要になってきます。

ログのローテーションでは，ある一定期間，例えば一週間ごとにログファイルを改名して，ログを保存します。例えば，/var/log/syslogファイルは/var/log/syslog.0に改名され，その後，ログの記録は/var/log/syslogに再開されます。同時に/var/log/syslog.0は/var/log/syslog.1に，/var/log/syslog.1は/var/log/syslog.2に改名されます。こうすることで，現在記録中のログは/var/log/syslogに，一週間前のログは/var/log/syslog.0に，二週間前のログは/var/log/syslog.1，そして三週間前のログは/var/log/syslog.2に保存されていることになります。/var/log/syslog.2を他の名前に改名しないとそれ以前のログは削除されます。

このように古くなったログファイルをどんどん改名していくことで，期間が過ぎた古いログのみを削除することができます。また期間ごとのファイルに分けて保存されているので，あとからログを読むときに目的の時間のログを探しやすくなります。

Debianでは，このログのローテーションを自動的に行ってくれるスクリプトが提供されています。早速利用してみましょう。

```
% sudo apt-get install cron logrotate
```

ここでは，cronとlogrotateパッケージをインストールしました。決められた時間にプログラムを動かすためのデーモンがcronです。ログのローテーションスクリプトはcronから決められた時間ごとに呼び出されます。このとき，メールサーバプログラムも同時にインストールされま

すが，必要ないならいまのところは動かさないに設定しましょう。もしメールサーバプログラムがなにも入ってないなら，apt-getはeximというメールサーバをインストールします。しばらくするとeximの設定がはじまりますが，とりあえずいまは“(5) No configuration”を選択してください。ログを管理者にメールで送るときには設定が必要になりますが，このメールサーバプログラムの設定に関しては次回に説明する予定です。

さて，cronから呼び出されて実際にログのローテーションを行うスクリプトは，/etc/cron.weekly/sysklogdと/etc/cron.daily/sysklogd，/usr/sbin/syslogd-listfiles，そして/usr/bin/savelogです。今回，これらの詳細の説明は長くなってしまうのでできませんが，中身は簡単なスクリプトです。設定を変更するときは直接エディタで編集してください。現在の標準設定では，/var/log/syslogファイルは毎日更新で７日間分，それ以外のsyslogdが出力するログファイルは一週間ごとの更新で4週間分保存するようになっています。保存する期間を変更したいときは，/etc/cron.weekly/sysklogdと/etc/cron.daily/sysklogdの中のsavelogのオプション-cの数字を変更することになります。

先ほどcronと同時にインストールしたlogrotateは，syslogd出力ではないログをローテーションするためのソフトウェアです。Apacheのログなどをローテーションしてくれます。Debianの場合，多くのパッケージはインストールすると同時にログのローテーションの設定もしてくれます。それらの設定は，/etc/logrotate.d/ディレクトリの下にパッケージごとのファイルに分けて記述されています。ログの保存期間の変更などはこれらファイルを編集してください。

また，標準ではローテーションしてくれないログのローテーション設定をする際には，/etc/logrotate.d/の既存のファイルを別名でコピーして設定に利用すれば良いでしょう。

Ⅴ．ログの例

最後にsyslogd経由で出力されたいつかのログの例を見てみましょう。

このログはsshでユーザnisimuraがログインしたときに出力されるものです。認証関係のログなのでDebianの標準設定では，/var/log/auth.logに記録されます（以下の例ではログに記録される時間情報とホストネームは省略します）。

```
PAM_unix[14689]: (ssh) session opened for user nisimura by (uid=1000)
```

一方でsshでログインを試みたがパスワードでの認証ができずにログインに失敗した場合のログは以下になります。

```
PAM_unix[14694]: 1 more authentication failure; (uid=0) -> nisimura for ssh service
PAM_unix[14696]: authentication failure; (uid=0) -> nisimura for ssh service
sshd[14696]: Failed password for nisimura from 192.168.24.102 port 32778 ssh2
```

192.168.24.102からnisimuraというユーザでログインを試みたがログインできなかった場合のログです。このようなログが大量に出力されていた場合は，sshに対する不正アクセスがあった可能性があります。

```
sudo: nisimura : TTY=pts/0 ; PWD=/home/nisimura ; USER=root ; COMMAND=/usr/bin/apt-get update
```

この例はsudoを使ったときに記録されるログです。sshと同じく/var/log/auto.logに出力されます。

telnetでアクセスがあった場合，/var/log/syslogに

```
in.telnetd[14751]: connect from 192.168.1.100
```

のように出力されます。また，ftpサーバにアクセスがあった場合，

```
Jun 1 03:25:23 wu-ftpd[14773]: connect from 192.168.1.100
Jun 1 03:25:27 wu-ftpd[14773]: FTP LOGIN FROM dhcp100.example.ac.jp [192.168.1.100], nisimura
Jun 1 03:25:43 wu-ftpd[14773]: FTP session closed
```

とログが残ります。見てわかるようにアクセス元のIPアドレス（ホストネーム）とユーザ名の他にセッションが終了した時刻も記録されています。この例では，ftpサーバプログラムにwu-ftpdを使用しています。同じサービスを提供するプログラムでもプログラムによって出力されるログのフォーマットは違いますので注意してください。

VI．おわりに

今回はログの基本のsyslogdの説明をしました。ログのチェックはそれ自体は不正アクセスを防ぎませんが，攻撃の検知やシステムの脆弱性を発見するうえで重要な役割を果たします。

次回は，もうすこし高度なログの設定を紹介します。やはりログに目を通すのはなかなか大変ですよね。楽にチェックできるようにしたいところです。そこでメールを利用したログの送付サービスや不正アクセスを検知できるログチェックツールの導入をします。

（にしむら りゅういち ：奈良先端科学技術大学院大学情報科学研究科）
（nisimura@linux.or.jp）