Global Information Assurance Certification Paper

Syslog and Netsaint: How to Integrate Centralized Logging with Centralized Monitoring

7/27/2001

SANS Security Essentials
GSEC Practical Assignment
Version 1.2e

Richard Murphy

Abstract

In today's organization where there are NT and Linux servers popping up alongside midrange UNIX servers, there is a growing need for centralized management. Many commercial products attempt to solve this problem by providing software that integrates centralized host management, single sign-on, SNMP monitoring, and remote control tools. These packages can be very costly and difficult to implement successfully across the enterprise.

It is possible to achieve the same tasks with open-source software and the built-in utilities that most OS's provide. This paper will address three aspects of centralized management: 1) Centralized log management 2) Centralized monitoring and 3) The integration of the two technologies. The integration of these two technologies will give the overworked systems administrator more time to proactively manage his/her systems by virtually eliminating time spent poring over log files and constantly checking system status.

Introduction

Centralized system logging is an essential part of a good security policy and centralized network monitoring is an essential part of a good network operations policy. Integrating these two technologies can enhance monitoring in an organization as well as enhance the ability of an organization to react to a critical situation.

Centralized logging provides easier management and administration of an organization's system logs by compiling the logs from various systems into one place. This makes it easier to run automated tools that scan those log files looking for suspicious activity.

> Failure to enable the necessary data collection mechanisms will greatly weaken or eliminate your ability to detect suspicious behavior and intrusion attempts and to determine whether or not such attempts succeeded.
>
> Failure to configure and secure the volume of data produced by these mechanisms will place the data at risk of compromise and make subsequent review and analysis difficult, if not impossible. (CERT: Manage logging and other data collection mechanisms, 2000)

Centralized network and host monitoring is typically done to ensure that all hosts are up and running. Most network monitors can monitor specific services as well as monitor subnets and network latency. They are essential in providing the system administrator with timely alerts to system or network outages.

In this project these two technologies will be integrated to form a centralized monitoring system based on Netsaint.

Tools

The following tools will be used to implement the solution:

- Syslogd
- Perl 5 – http://www.perl.org
- Netsaint 0.0.6 http://www.netsaint.org
- Ntsyslog - http://www.sabernet.net/software/ntsyslog.html
- NSCA - NetSaint Service Check Acceptor http://www.netsaint.org/download/
- Logcheck – http://www.sabernet.net/software/

Setting up the Syslogd logging server

According to Joshua Weinberg, syslog first appeared as part of BSD UNIX from Berkeley. Eric Allman of sendmail fame wrote Syslog, and it eventually became the standard for system logging. Before syslog came on the scene, logging was done to flat files that were stored across the system depending on what application was doing the

logging. With syslog, the administrator had control of where and how much information was being logged. Syslog has been integrated into all modern UNIX operating systems and it implements a priority scheme that makes it easy for the administrator to separate the wheat from the chaff. The best feature of syslog is that it can log to files, send alerts to users or the system console, or send the message itself off to another syslog server for remote logging. This remote logging ability is what allows us to configure a central logging server to handle messages from all of our UNIX and NT hosts. (Weinberg, 2001)

The first step in setting up the syslog server was to start with a fresh install of Red Hat 7.1 Linux. The workstation option was picked during setup in order to minimize the software installed and the firewall was set to the High Security option with one open incoming port being syslog, port 514/UDP.

The second step was to reconfigure syslog to according to CERT recommendations available at http://www.cert.org/security-improvement/implementations/i041.08.html. The syslog.conf file recommended by CERT allows for local logging to the console, logging all messages to the messages file for use with log analysis tools like SWATCH, and finally logging to separate files based on system for archival purposes. This particular configuration is very complete, but it uses a lot of disk space, by duplicating every message that is logged. A good log rotation policy and plenty of disk space should be prerequisites for this project. Depending on the number of hosts logging to the server, and the level of messages being generated anywhere between 20MB and 1GB a day can be logged to the central server.

The syslog.conf is included below:

```
# syslog configuration file (loghost)
#
# output to console

*.err;mail,kern.notice;daemon,auth.debug;user.info/dev/console

# output to local file "messages" for automatic log file
# analysis (logsurfer)

*.err;auth,daemon,mark,kern.debug;mail,user.notice/var/log/messages

# output to local files for archiving messages of potential interest

auth.debug   /var/log/auth.log
daemon.debug        /var/log/daemon.log
lpr.debug     /var/log/lpr.log
mail.debug   /var/log/mail.log
news.debug              /var/log/news.log
uucp.debug              /var/log/uucp.log
user.debug              /var/log/user.log

# end of /etc/syslog.conf
```

After making the above changes to the syslog.conf file the syslog service will need to be

restarted with the commands:

```
# kill  <PID>

# syslogd –rm 0
```

where <PID> is the process id of the syslog service. The –r option sets syslog up for "remote reception". You now have a central syslog server waiting on incoming messages.

The final step is to edit the syslog configuration file to have syslog start up in "remote reception" mode upon system boot. This is done by editing the file */etc/sysconfig/syslog* to add –r to the syslog options. If your system doesn't have the above syslog config file, you can also manually edit the init.d startup script to have syslog start with the –r option. Depending on your system, this file can be found in of the rc.d directories.

Setting up the UNIX syslog clients

Setting up the UNIX clients is the easy part of this project. Each UNIX OS has a standard syslog.conf file that logs important messages locally. In order to minimize administration, the current configuration will not be changed except to add a line that forwards all messages to the central logging server or the "loghost". Taking this approach will allow administrators to continue to deal with familiar logging on their own systems while at the same time allowing log checkers and Netsaint to monitor all hosts.

The line added to the syslog.conf file follows:

```
# forward to loghost

mark.debug;*.debug   @loghost
```

An entry in the /etc/hosts file will also need to be made that defines the IP address for the loghost, or alternatively, an entry can be made into the DNS tables to resolve *loghost* to the proper IP address.

Stop and restart the syslog daemon by issuing it a SIGHUP command and you now have your UNIX client forwarding all of its messages to the syslog server.

Setting up Windows NT to log to the syslog server

Windows NT/2000 comes with a similar logging utility called the Eventlog that logs messages in much the same way as syslog. The main problem with the Eventlog, however, is that it cannot log to a syslog server. This problem has been solved by many utilities that can read the event log and then forward messages to a logging host running syslog. Many of these can be found at http://www.loop-back.com/syslog.htm.

The utility we are going to use is NTsyslog, which is available from the Sabernet web site: http://www.sabernet.net/software/ntsyslog.html. This utility is free and is distributed under the GNU General Public License. This program runs as a service under Windows NT 4.0 and Windows 2000. It formats all System, Security, and Application events into a single line and sends them to a syslog host. The syslog host is set by creating a key in the registry that contains the DNS name or IP address of the syslog server.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\SaberNet]
"Syslog"="loghost.example.com"
```

NTsyslog can also be configured to send some or all event messages to the syslog server through registry entries as shown below. The example below enables all event types for each log.

```
[HKEY_LOCAL_MACHINE\SOFTWARE\SaberNet\Syslog\System]
"Information"=dword:00000001
"Warning"=dword:00000001
"Error"=dword:00000001
"Audit Success"=dword:00000001
"Audit Failure"=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\SaberNet\Syslog\Security]
"Information"=dword:00000001
"Warning"=dword:00000001
"Error"=dword:00000001
"Audit Success"=dword:00000001
"Audit Failure"=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\SaberNet\Syslog\Application]
"Information"=dword:00000001
"Warning"=dword:00000001
"Error"=dword:00000001
"Audit Success"=dword:00000001
"Audit Failure"=dword:00000001
```

A file can be downloaded from the NTsyslog web page that can make these registry entries for you.

The one limitation with NTsyslog is that it only sends messages with the *user.alert* priority. This makes it difficult to manually browse through the logs on the syslog server. However, by sending all messages to the *messages* file, a log-checking program will have no problem sorting through the messages.

After the registry entries are made, then the service needs to be installed. This is done through the NTsyslog command with the *–install* option.

```
# NTsyslog [ -install ] [ -remove ]
```

After this last step is done, reboot your computer to make sure the service starts upon boot, and check the syslog server to make sure it received the messages related to the Windows NT system booting up. (Sabernet, 2000)

Checking the log for problems

Now that we have our hosts forwarding log information to our central loghost, we need a way to monitor all of those messages. There are many tools freely available to perform the task of monitoring logs and they range from the simple to the very complex. They have varying options and are typically user configurable. A good list of these tools can be found at Talisker's Network Security Tools http://www.networkintrusion.co.uk/HIDS.htm.

Most of the available tools can be 'hacked' to integrate with Netsaint, but a simple tool written in Perl will be chosen for simplicity. Perl is a natural choice for this type of tool because of its rich support for regular expressions. Any number of signatures or strings can be searched on or matched. If a match is found then the information can be forwarded to Netsaint.

Sabernet's logcheck tool (http://www.sabernet.net/software/) is a great example of a Perl program that can be configured to search for many types of strings or expressions in the log files. It will only need to be edited slightly in order to forward alerts to Netsaint. A simple description listed at the beginning of the program is shown below: (Sabernet, 2000)

```
#!/usr/local/bin/perl
#--------------------------------------------------------------------------
# SaberNet.net
#--------------------------------------------------------------------------
#
# logcheck.pl - Log file checker
# Jason Rhoads <jason.rhoads@sabernet.net>
#
# This script checks system log files for unusual activity and emails
# the result to administrators for review.
#
# Any log entry that is not expressly ignored will be reported.
#
# Add the following line to your root crontab to run logcheck every
# ten minutes:
#
#    0,10,20,30,40,50 * * * * /usr/local/etc/logcheck.pl
#
#
# Based on logcheck.sh by Craig Rowland <crowland@psionic.com>
```

```
#
#
# Revision history:
#
#  1.00  09-Jan-97  First release
#  1.01  18-Oct-97  Added check to verify syslog is running
#
#--------------------------------------------------------------------------
```

Since the logcheck program is a Perl program, we can easily edit it to communicate to Netsaint with the appropriate information when it finds something of alert status. The following line is added to logcheck in the *check_log* subroutine:

```
exec("echo -e \'gislog\tCHECK_LOG\t2\t$ALERT\n\' |send_nsca 192.168.0.10 -c
/etc/send_nsca.cfg");
```

This line is placed inside of the *alert* conditional statement after the code that emails the alert to the user. The reference to *send_nsca* will be discussed below in the Netsaint section. The entire code of this script can be found in Appendix A.

Netsaint

NetSaint is a program that will monitor hosts and services on your network. It has the ability to email or page you when a problem arises and when it gets resolved. Netsaint is typically run as a daemon that periodically checks hosts and services that are specified in its configuration file. The actual service checks are performed by external "plugins" which return service information to NetSaint. It is the plugin feature of Netsaint that makes it so expandable. There are downloadable plugins that enable Netsaint to monitor a myriad of hosts and services. Below are just a few of the services Netsaint can monitor:

- Ping- check if any network device is alive
- http – check to see if your web browser is alive
- ftp – check to see if your ftp server is alive
- snmp – receive snmp traps
- disk, processor load – check system health
- printers – check printer status

Besides the downloadable plugins available from the Netsaint website (http://www.netsaint.org) plugins can be written in any language to check any sort of service. Perl is a great language for writing plugins. Because it is interpreted instead of compiled, it can be quickly written and debugged.

Usually all plugins reside on the Netsaint host and actively poll the specified service to determine its status on a periodic basis. Passive service checks, one of Netsaint's newer features, will allow Netsaint to receive status information asynchronously from a remote

host only when a change in status occurs. We will take advantage of this new feature to allow Netsaint to receive status information from our syslog logcheck program.

Making the Final Connection

The Netsaint Service Check Acceptor (NSCA) plugin will be used to send passive service check results from our syslog server to our Netsaint monitor (A description of Netsaint's passive service checks can be found here http://www.netsaint.org/docs/0_0_6/passivechecks.html). The NSCA plugin is a great utility that allows Netsaint to be very flexible in receiving service check results from remote hosts with a minimum of effort. Below is a short description from the Netsaint web site:

> This program is designed to accept passive service check results from clients that use the send_nsca utility (also included in this package) and pass them along to the NetSaint process by using the external command interface. The program can be run either as a standalone daemon or as a service under inetd.

The first step to get NSCA going is to have a working Netsaint installation. NSCA requires NetSaint version 0.0.6 or newer. The second step is to enable Netsaint to check for external commands. This is done by editing the netsaint.cfg file to look like the following:

```
# EXTERNAL COMMAND OPTION
# This option allows you to specify whether or not NetSaint should check
# for external commands (in the command file defined below).  By default
# NetSaint will *not* check for external commands, just to be on the
# cautious side.  If you want to be able to use the CGI command interface
# you will have to enable this.  Setting this value to 0 disables command
# checking (the default), other values enable it.

check_external_commands=1


# EXTERNAL COMMAND CHECK INTERVAL
# This is the interval at which NetSaint should check for external commands.
# This value works of the interval_length you specify later.  If you leave
# that at its default value of 60 (seconds), a value of 1 here will cause
# NetSaint to check for external commands every minute.  Note: In addition
# to reading the external command file at regularly scheduled intervals,
# NetSaint will also check for external commands after event handlers are
# executed.

command_check_interval=1


# EXTERNAL COMMAND FILE
# This is the file that NetSaint checks for external command requests.
# It is also where the command CGI will write commands that are submitted
```

```
# by users, so it must be writeable by the user that the web server
# is running as (usually 'nobody').  Permissions should be set at the
# directory level instead of on the file, as the file is deleted every
# time its contents are processed.

command_file=@localstatedir@/rw/netsaint.cmd
```

The external command check interval is set to 60 seconds to be consistent with the active service check interval already in use. The third entry that refers to the external command file is the default entry. This file is a named pipe that gets created each time an external command is written to it; it is then piped to Netsaint.

The next step is to make the appropriate entries into Netsaint's hosts.cfg file. Entries will need to be made in the HOSTS, HOSTGROUP, and SERVICE section of this file that refer to our syslog host. The actual service command needs to be configured as a volatile service with a nonexistent time interval so that Netsaint doesn't try to actively check that service (These terms are described in the Netsaint documentation http://www.netsaint.org/docs/0_0_6/). The service command should look like the following:

```
service[rosie]=PING;1;none;3;5;1;nt-admins;120;24x7;1;1;0;;check_ping
```

After these changes have been made, restart the Netsaint daemon and it will be ready to receive passive service checks via the NSCA daemon.

The final steps are to download, compile, and install the NSCA plugin. The plugin will need to be compiled and installed on the Netsaint host as well as the syslog host. After compiling the plugin, the two programs *nsca* and *send_nsca* will need to be copied to a directory that is in the *path*, i.e. */usr/local/bin*. The *nsca.cfg* and *send_nsca.cfg* files need to be copied into a directory that is accessible by the programs; typically */etc*. These files then need to be edited on both the Netsaint host and the syslog host to have the same password.

```
# DECRYPTION PASSWORD
# This is the password/passphrase that should be used to decrypt the
# incoming packets.  Note that all clients must encrypt the packets
# they send using the same password!
password= zyx8k23
```

The default encryption for NSCA is a simple hash, but stronger encryption options are available depending on your installation. More information can be found in the SECURITY file found in the NSCA distribution. In short, if you have libmcrypt installed on your systems, you can choose from multiple crypto algorithms (DES, 3DES, CAST, xTEA, Twofish, LOKI97, RJINDAEL, SERPENT, GOST, SAFER/SAFER+, etc.) for encrypting the traffic between the client and the server.

After you have edited the *nsca*.cfg file on the Netsaint host, you need to start the nsca daemon. This is done by the command:

```
# nsca –d /etc/nsca.cfg
```

The nsca daemon can also be run under inetd, and if you have tcpwrappers installed this would be the preferred method. Details for running under inetd can be found in the README file in the NSCA package.

Once you have NSCA installed and configured on both hosts, and the nsca daemon is running on the Netsaint host, you are ready to run the modified logcheck program from your syslog host and begin forwarding your alerts to Netsaint. In order to automate the logcheck program, you need to add an entry to your crontab file to have it run periodically. Below is an example crontab entry that will run logcheck every 10 minutes:

```
0,10,20,30,40,50 * * * * /usr/local/etc/logcheck.pl
```

Security Implications

There are a few security implications in using this integrated monitoring system that should be discussed. They are listed below with a brief discussion for each:

- Netsaint – the Netsaint host will have to be running a web server in order for Netsaint to be able to show its visual alerts. Apache should be used because of its good security record, but as with all things, it should be battened down and all unnecessary modules should be commented out of the config file.
- NSCA – because NSCA transmits service check results across the network, it is susceptible to interception or spoofing. The libmcrypt module should be installed and one of the encryption methods implemented for a secure transmission of the data.
- Syslogd – standard syslog daemons use UDP, and are susceptible to spoofing and accepting bogus messages. There are more secure versions available, and if possible, these alternatives should be investigated. Syslogd from Sabernet (http://www.sabernet.net/software/syslogd.html) can use tcpwrappers, and Nsyslogd (http://cheops.anu.edu.au/~avalon/nsyslog.html) supports TCP connections and SSL for encrypted delivery of syslog messages over the network.

Summary

The integrated Netsaint/syslog monitoring system supports the three basic categories of administrative security:

1. Overall security planning and administration
2. Day-to-day security administration
3. Day-to-day system administration

This system is more involved in the third category because system administrators are usually the ones who actively monitor the systems and network that run the organization.

System administrators are usually the first to be notified by monitoring systems if anything is amiss, and they are the first ones to react. This integrated system can eliminate hours of poring over log files or manually running scripts to search through the logs. This system helps system administrators fulfill one of computer security's key goals: "availability". It keeps them abreast of any changes in system status or improper access with almost immediate alerting capabilities and web-based monitoring consoles (Russell, 1992).

Implementing this solution can be tedious and not for the faint of heart, but it can be a godsend for an overworked system administrator. Having network resources and system logs being monitored all from a single console will greatly enhance an organizations ability to keep its critical systems "available".

References

CERT, *Manage logging and other data collection mechanisms* (2000). CERT [Online]. Available: http://www.cert.org/security-improvement/practices/p092.html

Joshua Weinberg, *Centralized Logging with syslog* (2001). [Online]. Available: http://www.topsecret.net/sysadmin/html/v07/i10/a2.htm

Sabernet, *NTsyslog*, (2000). [Online]. Available: http://www.sabernet.net/software/ntsyslog.html

Loopback, *Windows Syslog Daemons* (1998). [Online]. Available: http://www.loop-back.com/syslog.htm

Netsaint, *Netsaint* (2001) [Online]. Available: http://www.netsaint.org

Sabernet, *logcheck*, (2000). [Online]. Available: http://www.sabernet.net/software/

Deborah Russell, G.T. Gangemi Sr. *Computer Security Basics*. O'Reilly & Associates, Inc., 1992

© SANS Institute 2000 - 2005 Author retains full rights.

Appendix A

```
#!/usr/local/bin/perl
#--------------------------------------------------------------------------
# SaberNet.net
#--------------------------------------------------------------------------
#
# logcheck.pl - Log file checker
# Jason Rhoads <jason.rhoads@sabernet.net>
#
# This script checks system log files for unusual activity and emails
# the result to administrators for review.
#
# Any log entry that is not expressly ignored will be reported.
#
# Add the following line to your root crontab to run logcheck every
# ten minutes:
#
#    0,10,20,30,40,50 * * * * /usr/local/etc/logcheck.pl
#
#
# Based on logcheck.sh by Craig Rowland <crowland@psionic.com>
#
#
# Revision history:
#
#  1.00  09-Jan-97  First release
#  1.01  18-Oct-97  Added check to verify syslog is running
#
#--------------------------------------------------------------------------

#
# User configurable settings:
#
# ADMIN         : System administrator to send general notices to.
#                 Example: sysadmins@some.com
#
# ALERT         : Incident response team to send security violations to
#                 Example: sysadmins@some.com infosec@some.com
#
# MAIL_CMD      : Local mail command.
#                 Example: /usr/bin/mailx
#
# LOG_FILE      : System log file to process
#                 Example: /var/adm/messages
#
# IGNORE_FILE   : File that contains regular expressions that match normal
#                 log entries that you are *not* interested in.
#                 Example: /usr/local/etc/logcheck.ignore
#
# VIOLATION_FILE : File that contains regular expressions that match
#                 security violations.  These log entries will also be sent
#                 to the ALERT alias.
#                 Example: /usr/local/etc/logcheck.violation
#
#
# SYSLOG_CMD    : The syslog command.  Define this if you want syslog
#                 automatically started if not already running.
#                 Example: /usr/sbin/syslogd
#
# PS_CMD        : The ps command.  This is used to check for syslogd.
```

```
#               Example: ps -ef (SVR4) or ps -ax (BSD)
#
# HOSTNAME      : Command that prints the machine's hostname.
#               Example: uname -n
#
$ADMIN          = "root";
$ALERT          = "root";
$MAIL_CMD       = "/usr/bin/mail";
$LOG_FILE       = "/var/adm/messages";
$IGNORE_FILE    = "/usr/local/etc/logcheck.ignore";
$VIOLATION_FILE = "/usr/local/etc/logcheck.violations";
$SYSLOG_CMD     = "/usr/sbin/syslogd";
$PS_CMD         = "/usr/bin/ps -ef";
$HOSTNAME       = "/usr/bin/uname -n";

#
# Main
#
checkSyslog();
openLog();
setHostname();
checkLog();
closeLog();
exit(1);

#
# openLog : Opens the log file and seeks to the last known
#           file position.
#
sub openLog
{
            local($last_pos) = `cat $LOG_FILE.pos 2> /dev/null`;
            chop($last_pos);

            open(LOG, "<$LOG_FILE") or die("Unable to open log file: $LOG_FILE\n");
            seek(LOG, 0, 2);
            if ($last_pos == tell(LOG))
            {
                        exit(0);
            }
            elsif ($last_pos < tell(LOG))
            {
                        seek(LOG, $last_pos, 0);
            }
            else
            {
                        seek(LOG, 0, 0);
            }
}

#
# closeLog : Closes the log file and saves the current position.
#
sub closeLog
{
            local($last_pos) = tell(LOG);
            `echo $last_pos > $LOG_FILE.pos`;
            close(LOG);
}

#
# checkSyslog : Verifys that syslogd is running
#
sub checkSyslog
```

```
{
        local($grep) = `$PS_CMD | grep $SYSLOG_CMD | grep -v grep`;
        chop($grep);
        if ("$grep" eq "")
        {
                `$SYSLOG_CMD`;
        }
}

#
# setHostname : Set the HOSTNAME variable
#
sub setHostname
{
        # Get hostname
        $HOSTNAME = `$HOSTNAME`;
        chop($HOSTNAME);

        # Get domain name
        open(IN, "</etc/resolv.conf");
        while (<IN>)
        {
                chop;
                $_ = lc($_);
                if (lc($_) =~ /domain\s*(.+)/)
                {
                        $HOSTNAME .= ".".$1;
                        break;
                }
        }
}

#
# checkLog : Checks the log for interesting entries
#
sub checkLog
{
        local(@ignore, @violations, $gen, $alert, $i);

        # Load ignore file
        open(IN, "<$IGNORE_FILE") or
                die("Unable to open ignore file: $IGNORE_FILE\n");
        $i = 0;
        while (<IN>)
        {
                chop;
                $ignore[$i++] = $_;
        }
        close(IN);

        # Load violations file
        open(IN, "<$VIOLATION_FILE") or
                die("Unable to open violation file: $VIOLATION_FILE\n");

        $i = 0;
        while (<IN>)
        {
                chop;
                $violations[$i++] = $_;
        }
        close(IN);

        # Process log file
        while (<LOG>)
```

```
		{
			if (!matches($_, @ignore))
			{
				if (matches($_, @violations))
				{
					$alert .= sprintf("%s", $_);
				}
				else
				{
					$gen .= sprintf("%s", $_);
				}
			}
		}

		# Mail results
		if ("$gen" ne "")
		{
			open(GEN, "| $MAIL_CMD $ADMIN");
			print GEN "Subject: Logcheck : $HOSTNAME\n\n";
			print GEN "$gen\n";
			close(GEN);
		}

		if ("$alert" ne "")
		{
			open(ALERT, "| $MAIL_CMD $ALERT");
			print ALERT "Subject: Security Alert: $HOSTNAME\n\n";
			print ALERT "$alert\n";
			close(ALERT);
			exec("echo -e \'gislog\tCHECK_LOG\t2\t$ALERT\n\' |send_nsca 158.111.33.22 -c
/etc/send_nsca.cfg");
		}
}

#
# matches : Checks each expression against test.  Returns 1 if
#	a match is found or 0 if not.
#
sub matches
{
		local($test, @exps) = @_;
		local($i) = 0;

		while ($i < @exps)
		{
			if ($_ =~ /$exps[$i++]/) { return(1); }
		}

		return(0);
}
```