ULOGD - the Userspace Logging Daemon

Harald Welte <laforge@gnumonks.org>

Revision $Revision: 5846 $, $Date: 2005-07-12 17:33:23 +0200 (Tue, 12 Jul 2005) $

This is the documentation for `ulogd`, the Userspace logging daemon. ulogd makes use of the Linux >= 2.4.x packet filter subsystem (iptables) and the ULOG target for iptables.

Contents

1 DESIGN

1.1 CONCEPT

I want to provide a flexible, almost universal logging daemon for my netfilter ULOG target. It is not optimized in any way, the goal is to keep as simple as possible. These are my thoughts about how the architecture which is most capable of doing that:

Interpreter plugins

It should be possible to add plugins / runtime modules for new protocols, etc. For example the standard logging daemon provides source-ip, dest-ip, source-port, dest-port, etc. Logging for various other protocols (GRE, IPsec, ...) may be implemented as modules.

Output plugins

... describe how and where to put the information gained by logging plugins. The easiest way is to build a line per packet and fprint it to a file. Some people might want to log into a SQL database or want an output conforming to the intrusion detection systems communication draft from the IETF.

1.2 DETAILS

The major clue is providing a framework which is as flexible as possible. Nobody knows what strange network protocols are out there :) Flexibility depends on the communication between the output of the logging plugins and input of the output plugins.

Rusty advised me to use some kind of type-key-value triples, which is in fact what I implemented.

One issue is, of course, performance. Up to ulogd 0.3, ulogd did several linked list iterations and about 30 malloc() calls _per packet_. This changed with the new >= 0.9 revisions:

- Not a single dynamic allocation in the core during runtime. Everything is pre-allocated at start of ulogd to provide the highest possible throughput.
- Hash tables in addition to the linked lists. Linked lists are only traversed if we really want to access each element of the list.

2 INSTALLATION

2.1 Linux kernel

First you will need a recent 2.4.x kernel. If you have a kernel >= 2.4.18-pre8, it already has the kernel support for ULOG (ipt_ULOG.o).

If you have an older kernel version (between 2.4.0 and 2.4.18-pre6), you can use the patch-o-matic system of netfilter/iptables, as described in the following section.

2.2 ipt_ULOG from netfilter/iptables patch-o-matic

You only need to read this chapter if you have a 2.4.x kernel <= 2.4.18-pre6.

In order to put the ipt_ULOG module into your kernel source,you need the latest iptables package, or even better: the latest CVS snapshot. A description how to obtain this is provided on the netfilter homepage .

To run patch-o-matic, just type

```
make patch-o-matic
```

in the userspace directory of netfilter CVS.

2.3 ulogd

2.3.1 Recompiling the source

Download the ulogd package from and untar it.

If you want to build ulogd with MySQL support, type './configure –with-mysql'. You may also have to specify the path of the mysql libraries using '–with-mysql=path'. To build ulogd without MySQL support, just use './configure'.

To compile and install the program, call 'make install'.

2.3.2 Using a precompiled package

I also provide a SRPM, which should compile on almost any rpm-based distribution. It is available at

Just download the package and do the usual 'rpm –rebuild <file>'.

3 Configuration

3.1 iptables ULOG target

3.1.1 Quick Setup

Just add rules using the ULOG target to your firewalling chain. A very basic example:

```
iptables -A FORWARD -j ULOG --ulog-nlgroup 32 --ulog-prefix foo
```

To increase logging performance, try to use the

```
--ulog-qthreshold N
```

option (where 1 < N <= 50). The number you specify is the amount of packets batched together in one multipart netlink message. If you set this to 20, the kernel schedules ulogd only once every 20 packets. All 20 packets are then processed by ulogd. This reduces the number of context switches between kernel and userspace.

Of course you can combine the ULOG target with the different netfilter match modules. For a more detailed description, have a look at the netfilter HOWTO's, available on the netfilter homepage.

3.1.2 ULOG target reference

–ulog-nlgroup N

The number of the netlink multicast group to which ULOG'ed packets are sent. You will have to use the same group number in the ULOG target and ulogd in order to make logging work.

–ulog-cprange N

Copyrange. This works like the 'snaplen' parameter of tcpdump. You can specify a number of bytes up to which the packet is copied. If you say '40', you will receive the first fourty bytes of every packet. Leave it to `0`

–ulog-qthreshold N

Queue threshold. If a packet is matched by the iptables rule, and already N packets are in the queue, the queue is flushed to userspace. You can use this to implement a policy like: Use a big queue in order to gain high performance, but still have certain packets logged immediately to userspace.

–ulog-prefix STRING

A string that is associated with every packet logged by this rule. You can use this option to later tell from which rule the packet was logged.

3.1.3 ipt_ULOG module parameters

The ipt_ULOG kernel module has a couple of module loadtime parameters which can (and should) be tuned to accomodate the needs of the application:

nlbufsiz N

> Netlink buffer size. A buffer of the specified size N is allocated for every netlink group that is used. Please note that due to restrictions of the kernel memory allocator, we cannot have a buffer size > 128kBytes. Larger buffer sizes increase the performance, since less kernel/userspace context switches are needed for the same amount of packets. The backside of this performance gain is a potentially larger delay. The default value is 4096 bytes, which is quite small.

flushtimeout N

> The flushtimeout determines, after how many clock ticks (on alpha: 1ms, on x86 and most other platforms: 10ms time units) the buffer/queue is to be flushed, even if it is not full. This can be used to have the advantage of a large buffer, but still a finite maximum delay introduced. The default value is set to 10 seconds.

Example:

```
modprobe ipt_ULOG nlbufsiz=65535 flushtimeout=100
```

This would use a buffer size of 64k and a flushtimeout of 100 clockticks (1 second on x86).

3.2 ulogd

ulogd is what this is all about, so let's describe it's configuration...

3.2.1 ulogd configfile syntax reference

All configurable parameters of ulogd are in the configfile, typically located at '/etc/ulogd.conf'.

The following configuration parameters are available:

nlgroup

> The netlink multicast group, which ulgogd should bind to. This is the same as given with the '–ulog-nlgroup' option to iptables.

logfile

> The main logfile, where ulogd reports any errors, warnings and other unexpected conditions. Apart from a regular filename, the following special values can be used; “syslog” to log via the unix syslog(3) mechanism. “stdout” to log to stdout.

loglevel

> This specifies, how verbose the logging to logfile is. Currently defined loglevels are: 1=debug information, 3=informational messages, 5=noticable exceptional conditions, 7=error conditions, 8=fatal errors, program abort.

plugin

> This option is followed by a filename of a ulogd plugin, which ulogd shold load upon initialization. This option may appear more than once.

rmem

Size of the netlink socket receive memory. You should set this to at least the size of the kernel buffer (nlbufsiz parameter of the ipt_ULOG module). Please note that there is a maximum limit in /proc/sys/net/core/rmem_max which you cannot exceed by increasing the “rmem” parameter. You may need to raise the system-wide maximum limit before.

bufsize

Size of the receive buffer. You should set this to at least the socket receive buffer (rmem).

3.2.2 ulogd commandline option reference

Apart from the configfile, there are a couple of commandline options to ulogd:

-h –help

Print a help message about the commandline options.

-V –version

Print version information about ulogd.

-d –daemon

For off into daemon mode. Unless you are debugging, you will want to use this most of the time.

-c –configfile

Using this commandline option, an alternate config file can be used. This is important if multiple instances of ulogd are to be run on a single machine.

4 Available plugins

It is important to understand that ulogd without plugins does nothing. It will receive packets, and do nothing with them.

There are two kinds of plugins, interpreter and output plugins. Interpreter plugins parse the packet, output plugins write the interpreted information to some logfile/database/...

4.1 Interpreter plugins

ulogd comes with the following interpreter plugins:

4.1.1 ulogd_BASE.so

Basic interpreter plugin for nfmark, timestamp, mac address, ip header, tcp header, udp header, icmp header, ah/esp header... Most people will want to load this very important plugin.

4.1.2 ulogd_PWSNIFF.so

Example interpreter plugin to log plaintext passwords as used with FTP and POP3. Don’t blame me for writing this plugin! The protocols are inherently insecure, and there are a lot of other tools for sniffing passwords... it’s just an example.

4.1.3 ulogd_LOCAL.so

This is a 'virtual interpreter'. It doesn't really return any information on the packet itself, rather the local system time and hostname. Please note that the time is the time at the time of logging, not the packets receive time.

4.2 Output plugins

ulogd comes with the following output plugins:

4.2.1 ulogd_OPRINT.so

A very simple output module, dumping all packets in the format

```
===>PACKET BOUNDARY
key=value
key=value
...
===>PACKET BOUNDARY
...
```

to a file. The only useful application is debugging.

The module defines the following configuration directives:

dumpfile

The filename where it should log to. The default is `/var/log/ulogd.pktlog`

4.2.2 ulogd_LOGEMU.so

An output module which tries to emulate the old syslog-based LOG targed as far as possible. Logging is done to a seperate textfile instead of syslog, though.

The module defines the following configuration directives:

file

The filename where it should log to. The default is `/var/log/ulogd.syslogemu`

sync

Set this to 1 if you want to have your logfile written synchronously. This may reduce performance, but makes your log-lines appear immediately. The default is `0`

4.2.3 ulogd_MYSQL.so

An output plugin for logging into a mysql database. This is only compiled if you have the mysql libraries installed, and the configure script was able to detect them. (that is: –with-mysql was specified for ./configure)

The plugin automagically inserts the data into the configured table; It connects to mysql during the startup phase of ulogd and obtains a list of the columns in the table. Then it tries to resolve the column names against keys of interpreter plugins. This way you can easily select which information you want to log - just by the layout of the table.

If, for example, your table contains a field called 'ip_saddr', ulogd will resolve this against the key 'ip.saddr' and put the ip address as 32bit unsigned integer into the table.

You may want to have a look at the file '`doc/mysql.table`' as an example table including fields to log all keys from ulogd_BASE.so. Just delete the fields you are not interested in, and create the table.

The module defines the following configuration directives:

table

Name of the table to which ulogd should log.

ldb

Name of the mysql database.

host

Name of the mysql database host.

port

TCP port number of mysql database server.

user

Name of the mysql user.

pass

Password for mysql.

4.2.4 ulogd_PGSQL.so

An output plugin for logging into a postgresql database. This is only compiled if you have the mysql libraries installed, and the configure script was able to detect them. (that is: –with-pgsql was specified for ./configure)

The plugin automagically inserts the data into the configured table; It connects to pgsql during the startup phase of ulogd and obtains a list of the columns in the table. Then it tries to resolve the column names against keys of interpreter plugins. This way you can easily select which information you want to log - just by the layout of the table.

If, for example, your table contains a field called 'ip_saddr', ulogd will resolve this against the key 'ip.saddr' and put the ip address as 32bit unsigned integer into the table.

You may want to have a look at the file '`doc/mysql.table`' as an example table including fields to log all keys from ulogd_BASE.so. Just delete the fields you are not interested in, and create the table.

The module defines the following configuration directives:

table

Name of the table to which ulogd should log.

db

Name of the database.

host

Name of the mysql database host.

port

TCP port number of database server.

user

Name of the sql user.

pass

Password for sql user.

4.2.5 ulogd_PCAP.so

An output plugin that can be used to generate libpcap-style packet logfiles. This can be useful for later analysing the packet log with tools like tcpdump or ethereal.

The module defines the following configuration directives:

file

The filename where it should log to. The default is: `/var/log/ulogd.pcap`

sync

Set this to `1` if you want to have your pcap logfile written synchronously. This may reduce performance, but makes your packets appear immediately in the file on disk. The default is `0`

4.2.6 ulogd_SQLITE3.so

An output plugin for logging into a SQLITE v3 database. This is only compiled if you have the sqlite libraries installed, and the configure script was able to detect them. (that is: –with-sqlite3 was specified for ./configure)

The plugin automagically inserts the data into the configured table; It opens the sqlite db during the startup phase of ulogd and obtains a list of the columns in the table. Then it tries to resolve the column names against keys of interpreter plugins. This way you can easily select which information you want to log - just by the layout of the table.

If, for example, your table contains a field called 'ip_saddr', ulogd will resolve this against the key 'ip.saddr' and put the ip address as 32bit unsigned integer into the table.

You may want to have a look at the file '`doc/sqlite3.table`' as an example table including fields to log all keys from ulogd_BASE.so. Just delete the fields you are not interested in, and create the table.

The module defines the following configuration directives:

table

Name of the table to which ulogd should log.

db

Name of the database.

buffer

Size of the sqlite buffer.

4.2.7 ulogd_SYSLOG.so

An output plugin that really logs via syslogd. Lines will look exactly like printed with traditional LOG target.

The module defines the following configuration directives:

facility

The syslog facility (LOG_DAEMON, LOG_KERN, LOG_LOCAL0 .. LOG_LOCAL7, LOG_USER)

level

The syslog level (LOG_EMERG, LOG_ALERT, LOG_CRIT, LOG_ERR, LOG_WARNING, LOG_NOTICE, LOG_INFO, LOG_DEBUG)

5 QUESTIONS / COMMENTS

All comments / questions / ... are appreciated.

Just drop me a note to laforge@gnumonks.org

Please note also that there is now a mailinglist, ulogd@lists.gnumonks.org. You can subscribe at

.

The preferred method for reporting bugs is the netfilter bugzilla system, available at .