TRAINING SESSION NAME

Firewall

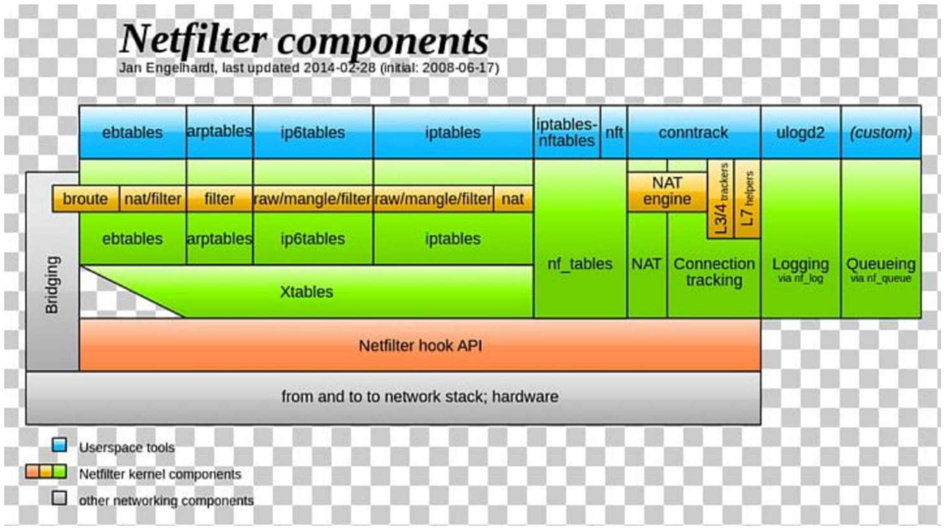
Table of contents

A. Ulogd

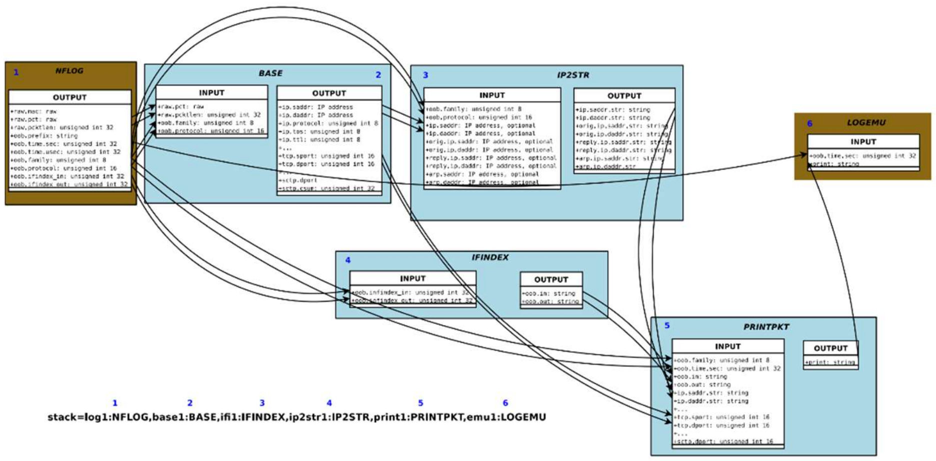
1.Introduction

1.1 Overview

ulogd is a userspace logging daemon for netfilter/iptables related logging. This includes per-packet logging of security violations, per-packet logging for accounting, per-flow logging and flexible user-defined accounting.



1.2 Stack concept


1 NFLOG
OUTPUT
+raw.mac: raw
+raw.pct: raw
+raw.pcktlen: unsigned int 32
+oob.prefix: string
+oob.time.sec: unsigned int 32
+oob.time.usec: unsigned int 32
+oob.family: unsigned int 8
+oob.protocol: unsigned int 16
+oob.ifindex_in: unsigned int 32
+oob.ifindex_out: unsigned int 32
BASE 2
INPUT
+raw.pct: raw
+raw.pcktlen: unsigned int 32
+oob.family: unsigned int 8
+oob.protocol: unsigned int 16
OUTPUT
+ip.saddr: IP address
+ip.daddr: IP address
+ip.protocol: unsigned int 8
+ip.tos: unsigned int 8
+ip.ttl: unsigned int 8
+...
+tcp.sport: unsigned int 16
+tcp.dport: unsigned int 16
+...
+sctp.dport
+sctp.csum: unsigned int 32
3 IP2STR
INPUT
+oob.family: unsigned int 8
+oob.protocol: unsigned int 16
+ip.saddr: IP address, optional
+ip.daddr: IP address, optional
+orig.ip.saddr: IP address, optional
+orig.ip.daddr: IP address, optional
+reply.ip.saddr: IP address, optional
+reply.ip.daddr: IP address, optional
+arp.saddr: IP address, optional
+arp.daddr: IP address, optional
OUTPUT
+ip.saddr.str: string
+ip.daddr.str: string
+orig.ip.saddr.str: string
+orig.ip.daddr.str: string
+reply.ip.saddr.str: string
+reply.ip.daddr.str: string
+arp.ip.saddr.str: string
+arp.ip.daddr.str
6 LOGEMU
INPUT
+oob.time.sec: unsigned int 32
+print: string
4 IFINDEX
INPUT
+oob.infindex_in: unsigned int 32
+oob.infindex_out: unsigned int 32
OUTPUT
+oob.in: string
+oob.out: string
5 PRINTPKT
INPUT
+oob.family: unsigned int 8
+oob.time.sec: unsigned int 32
+oob.in: string
+oob.out: string
+ip.saddr.str: string
+ip.daddr.str: string
+...
+tcp.sport: unsigned int 16
+tcp.dport: unsigned int 16
+...
+sctp.dport: unsigned int 16
OUTPUT
+print: string
1 2 3 4 5 6

```
stack=log1:NFLOG,base1:BASE,ifi1:IFINDEX,ip2str1:IP2STR,print1:PRINTPKT,emu1:LOGEMU
```

Each plugin has:

- Input keys : It defined the entry which are needed (mandatory or not) by the module to be able to work (ie output other keys)
- Output keys : The module outputs key→value association with key in the output list.

1.3 Type plugin

ulogd-2.x wants to provide a flexible, almost universal logging daemon for netfilter logging. This encompasses both packet-based logging (logging of policy violations) and flow-based logging, e.g. for accounting purpose.

ulogd consists of a small core and a number of plugins. All the real power lies in the plugins, and in the user who configures the interactions between those plugins.

+ Input Plugins: Input plugins acts data source. They get data from somewhere outside of ulogd, and convert it into a list of ulogd keys.

+ Filter Plugins: Filter plugins interpret and/or filter data that was received from the Input Plugin. A good example is parsing a raw packet into IPv4 / TCP / ... header information.

+ Output Plugins: Output plugins describe how and where to put the information gained by the Input Plugin and processed by one or more Filter 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 IETF IPFIX language.

Input plugins:

- **ULOG**: get packet select via the iptables target ULOG.
- **NFLOG**: get packet from NFLOG target which is the successor of ULOG.
- **NFCT**: get flow information from Netfilter conneciton tracking via libnetfilter_conntrack.

Output plugins:

- **LOGEMU**: log packet/flow into a file
- **OPRINT**: log packet/flow to a file in multiline format
- **SYSLOG**: log packet/flow to syslog system
- **MYSQL**: log packet/flow to a MYSQL database
- **PGSQL**: log packet/flow to a PGSQL server
- **SQLITE3**: log packet to a SQLITE3 file
- **PCAP**: log packet to a Pcap file
- **IPFIX**: log flow via IP Flow Information Export
- **NACCT**: log flow to a nacct compatible format (accounting)
- **JSON** : log packet to a JSON file

Below is various represent of plugin with its input key and output key.

+ With plugin **HWDHD**

```
/ # ./usr/sbin/ulogd --info ./usr/lib/ulogd/ulogd_filter_HWHDR.so

Name: HWHDR
Input keys:
        Key: raw.type (unsigned int 16, optional)
        Key: oob.protocol (unsigned int 16)
        Key: raw.mac (raw data, optional)
        Key: raw.mac_len (unsigned int 16, optional)
        Key: raw.mac.saddr (raw data, optional)
        Key: raw.mac.addrlen (unsigned int 16, optional)
Output keys:
        Key: raw.type (unsigned int 16)
        Key: oob.protocol (unsigned int 16)
        Key: mac.saddr.str (string)
        Key: mac.daddr.str (string)
        Key: mac.str (string)
```

+ Correspond with source code of ulogd (with input and output)

```
enum input_keys {
        KEY_RAW_TYPE,
        KEY_OOB_PROTOCOL,
        KEY_RAW_MAC,
        KEY_RAW_MACLEN,
        KEY_RAW_MAC_SADDR,
        KEY_RAW_MAC_ADDRLEN,
};

static struct ulogd_key mac2str_inp[] = {
        [KEY_RAW_TYPE] = {
                .type = ULOGD_RET_UINT16,
                .flags = ULOGD_RETF_NONE|ULOGD_KEYF_OPTIONAL,
                .name = "raw.type",
        },
        [KEY_OOB_PROTOCOL] = {
                .type = ULOGD_RET_UINT16,
                .flags = ULOGD_RETF_NONE,
                .name = "oob.protocol",
        },
        [KEY_RAW_MAC] = {
                .type = ULOGD_RET_RAW,
                .flags = ULOGD_RETF_NONE|ULOGD_KEYF_OPTIONAL,
                .name = "raw.mac",
        },
        [KEY_RAW_MACLEN] = {
                .type = ULOGD_RET_UINT16,
                .flags = ULOGD_RETF_NONE|ULOGD_KEYF_OPTIONAL,
                .name = "raw.mac_len",
        },
        [KEY_RAW_MAC_SADDR] = {
                .type = ULOGD_RET_RAW,
                .flags = ULOGD_RETF_NONE|ULOGD_KEYF_OPTIONAL,
                .name = "raw.mac.saddr",
        },
        [KEY_RAW_MAC_ADDRLEN] = {
                .type = ULOGD_RET_UINT16,
                .flags = ULOGD_RETF_NONE|ULOGD_KEYF_OPTIONAL,
                .name = "raw.mac.addrlen",
        },
};


enum output_keys {
        KEY_MAC_TYPE,
        KEY_MAC_PROTOCOL,
        KEY_MAC_SADDR,
        START_KEY = KEY_MAC_SADDR,
        KEY_MAC_DADDR,
        KEY_MAC_ADDR,
        MAX_KEY = KEY_MAC_ADDR,
};

static struct ulogd_key mac2str_keys[] = {
        [KEY_MAC_TYPE] = {
                .type = ULOGD_RET_UINT16,
                .flags = ULOGD_RETF_NONE,
                .name = "raw.type",
        },
        [KEY_MAC_PROTOCOL] = {
                .type = ULOGD_RET_UINT16,
                .flags = ULOGD_RETF_NONE,
                .name = "oob.protocol",
        },
        [KEY_MAC_SADDR] = {
                .type = ULOGD_RET_STRING,
                .name = "mac.saddr.str",
        },
        [KEY_MAC_DADDR] = {
                .type = ULOGD_RET_STRING,
                .name = "mac.daddr.str",
        },
        [KEY_MAC_ADDR] = {
                .type = ULOGD_RET_STRING,
                .name = "mac.str",
        },
};
```

+ With plugin **LOGEMU**

```
/ # ./usr/sbin/ulogd --info ./usr/lib/ulogd/ulogd_output_LOGEMU.so

Name: LOGEMU
Config options:
        Var: file (String, Default: /var/log/ulogd_syslogemu.log)
        Var: sync (Integer, Default: 0)
Input keys:
        Key: print (string)
        Key: oob.time.sec (unsigned int 32, optional)
Output keys:
        Output plugin, No keys
```

+ Correspond with source code of ulogd (with input and no output and configure)

```
static struct ulogd_key logemu_inp[] = {
        {
                .type = ULOGD_RET_STRING,
                .name = "print",
        },
        {
                .type = ULOGD_RET_UINT32,
                .flags = ULOGD_KEYF_OPTIONAL,
                .name = "oob.time.sec",
        },
};

static struct config_keyset logemu_kset = {
        .num_ces = 2,
        .ces = {
                {
                        .key     = "file",
                        .type    = CONFIG_TYPE_STRING,
                        .options = CONFIG_OPT_NONE,
                        .u       = { .string = ULOGD_LOGEMU_DEFAULT },
                },
                {
                        .key     = "sync",
                        .type    = CONFIG_TYPE_INT,
                        .options = CONFIG_OPT_NONE,
                        .u       = { .value = ULOGD_LOGEMU_SYNC_DEFAULT },
                },
        },
};
```

+ With plugin **NFLOG**

```
/ # ./usr/sbin/ulogd --info ./usr/lib/ulogd/ulogd_inppkt_NFLOG.so

Name: NFLOG
Config options:
        Var: bufsize (Integer, Default: 150000)
        Var: group (Integer, Default: 0)
        Var: unbind (Integer, Default: 1)
        Var: bind (Integer, Default: 0)
        Var: seq_local (Integer, Default: 0)
        Var: seq_global (Integer, Default: 0)
        Var: numeric_label (Integer, Default: 0)
        Var: netlink_socket_buffer_size (Integer, Default: 0)
        Var: netlink_socket_buffer_maxsize (Integer, Default: 0)
        Var: netlink_qthreshold (Integer, Default: 0)
        Var: netlink_qtimeout (Integer, Default: 0)
Input keys:
        Input plugin, No keys
Output keys:
        Key: raw.mac (raw data)
        Key: raw.pkt (raw data)
        Key: raw.pktlen (unsigned int 32)
        Key: raw.pktcount (unsigned int 32)
        Key: oob.prefix (string)
        Key: oob.time.sec (unsigned int 32)
        Key: oob.time.usec (unsigned int 32)
        Key: oob.mark (unsigned int 32)
        Key: oob.ifindex_in (unsigned int 32)
        Key: oob.ifindex_out (unsigned int 32)
        Key: oob.hook (unsigned int 8)
        Key: raw.mac_len (unsigned int 16)
        Key: oob.seq.local (unsigned int 32)
        Key: oob.seq.global (unsigned int 32)
        Key: oob.family (unsigned int 8)
        Key: oob.protocol (unsigned int 16)
        Key: oob.uid (unsigned int 32)
        Key: oob.gid (unsigned int 32)
        Key: raw.label (unsigned int 8)
        Key: raw.type (unsigned int 16)
        Key: raw.mac.saddr (raw data)
        Key: raw.mac.addrlen (unsigned int 16)
        Key: raw (raw data)
```

+ Correspond with source code of ulogd (with no input and output and configure)

```
enum nflog_keys {
        NFLOG_KEY_RAW_MAC = 0,
        NFLOG_KEY_RAW_PCKT,
        NFLOG_KEY_RAW_PCKTLEN,
        NFLOG_KEY_RAW_PCKTCOUNT,
        NFLOG_KEY_OOB_PREFIX,
        NFLOG_KEY_OOB_TIME_SEC,
        NFLOG_KEY_OOB_TIME_USEC,
        NFLOG_KEY_OOB_MARK,
        NFLOG_KEY_OOB_IFINDEX_IN,
        NFLOG_KEY_OOB_IFINDEX_OUT,
        NFLOG_KEY_OOB_HOOK,
        NFLOG_KEY_RAW_MAC_LEN,
        NFLOG_KEY_OOB_SEQ_LOCAL,
        NFLOG_KEY_OOB_SEQ_GLOBAL,
        NFLOG_KEY_OOB_FAMILY,
        NFLOG_KEY_OOB_PROTOCOL,
        NFLOG_KEY_OOB_UID,
        NFLOG_KEY_OOB_GID,
        NFLOG_KEY_RAW_LABEL,
        NFLOG_KEY_RAW_TYPE,
        NFLOG_KEY_RAW_MAC_SADDR,
        NFLOG_KEY_RAW_MAC_ADDRLEN,
        NFLOG_KEY_RAW,
};


static struct ulogd_key output_keys[] = {
        [NFLOG_KEY_RAW_MAC] = {
                .type = ULOGD_RET_RAW,
                .flags = ULOGD_RETF_NONE,
                .name = "raw.mac",
        },
        [NFLOG_KEY_RAW_MAC_SADDR] = {
                .type = ULOGD_RET_RAW,
                .flags = ULOGD_RETF_NONE,
                .name = "raw.mac.saddr",
                .ipfix = {
                        .vendor = IPFIX_VENDOR_IETF,
                        .field_id = IPFIX_sourceMacAddress,
                },
        },
        [NFLOG_KEY_RAW_PCKT] = {
                .type = ULOGD_RET_RAW,
                .flags = ULOGD_RETF_NONE,
                .name = "raw.pkt",
                .ipfix = {
                        .vendor = IPFIX_VENDOR_NETFILTER,
                        .field_id = IPFIX_NF_rawpacket,
                },
        },
        [NFLOG_KEY_RAW_PCKTLEN] = {
                .type = ULOGD_RET_UINT32,
                .flags = ULOGD_RETF_NONE,
                .name = "raw.pktlen",
                .ipfix = {
                        .vendor = IPFIX_VENDOR_NETFILTER,
                        .field_id = IPFIX_NF_rawpacket_length,
                },
        },raw.mac.addrlen
        [NFLOG_KEY_RAW_PCKTCOUNT] = {
                .type = ULOGD_RET_UINT32,
                .flags = ULOGD_RETF_NONE,
                .name = "raw.pktcount",
                .ipfix = {
                        .vendor = IPFIX_VENDOR_IETF,
                        .field_id = IPFIX_packetDeltaCount,
                },
        },
        [NFLOG_KEY_OOB_PREFIX] = {
                .type = ULOGD_RET_STRING,
                .flags = ULOGD_RETF_NONE,
                .name = "oob.prefix",
                .ipfix = {
                        .vendor = IPFIX_VENDOR_NETFILTER,
                        .field_id = IPFIX_NF_prefix,
                },
        },
        [NFLOG_KEY_OOB_TIME_SEC] = {
                .type = ULOGD_RET_UINT32,
                .flags = ULOGD_RETF_NONE,
                .name = "oob.time.sec",
                .ipfix = {
                        .vendor = IPFIX_VENDOR_IETF,
                        .field_id = IPFIX_flowStartSeconds,
                },
        },
        [NFLOG_KEY_OOB_TIME_USEC] = {
                .type = ULOGD_RET_UINT32,
                .flags = ULOGD_RETF_NONE,
                .name = "oob.time.usec",
		.ipfix = {
			.vendor = IPFIX_VENDOR_IETF,
			.field_id = IPFIX_flowStartMicroSeconds,
		},
	},
	[NFLOG_KEY_OOB_MARK] = {
		.type = ULOGD_RET_UINT32,
		.flags = ULOGD_RETF_NONE,
		.name = "oob.mark",
		.ipfix = {
			.vendor = IPFIX_VENDOR_NETFILTER,
			.field_id = IPFIX_NF_mark,
		},
	},
	[NFLOG_KEY_OOB_IFINDEX_IN] = {
		.type = ULOGD_RET_UINT32,
		.flags = ULOGD_RETF_NONE,
		.name = "oob.ifindex_in",
		.ipfix = {
			.vendor = IPFIX_VENDOR_IETF,
			.field_id = IPFIX_ingressInterface,
		},
	},
	[NFLOG_KEY_OOB_IFINDEX_OUT] = {
		.type = ULOGD_RET_UINT32,
		.flags = ULOGD_RETF_NONE,
		.name = "oob.ifindex_out",
		.ipfix = {
			.vendor = IPFIX_VENDOR_IETF,
			.field_id = IPFIX_egressInterface,
		},
	},
	[NFLOG_KEY_OOB_HOOK] = {
		.type = ULOGD_RET_UINT8,
		.flags = ULOGD_RETF_NONE,
		.name = "oob.hook",
		.ipfix = {
			.vendor = IPFIX_VENDOR_NETFILTER,
			.field_id = IPFIX_NF_hook,
		},
	},
	[NFLOG_KEY_RAW_MAC_LEN] = {
		.type = ULOGD_RET_UINT16,
		.flags = ULOGD_RETF_NONE,
		.name = "raw.mac_len",
	},
	[NFLOG_KEY_RAW_MAC_ADDRLEN] = {
		.type = ULOGD_RET_UINT16,
		.flags = ULOGD_RETF_NONE,
		.name = "raw.mac.addrlen",
	},

	[NFLOG_KEY_OOB_SEQ_LOCAL] = {
		.type = ULOGD_RET_UINT32,
		.flags = ULOGD_RETF_NONE,
		.name = "oob.seq.local",
		.ipfix = {
				.vendor = IPFIX_VENDOR_NETFILTER,
				.field_id = IPFIX_NF_seq_local,
			},
		},
		[NFLOG_KEY_OOB_SEQ_GLOBAL] = {
			.type = ULOGD_RET_UINT32,
			.flags = ULOGD_RETF_NONE,
			.name = "oob.seq.global",
			.ipfix = {
				.vendor = IPFIX_VENDOR_NETFILTER,
				.field_id = IPFIX_NF_seq_global,
			},
		},
		[NFLOG_KEY_OOB_FAMILY] = {
			.type = ULOGD_RET_UINT8,
			.flags = ULOGD_RETF_NONE,
			.name = "oob.family",
		},
		[NFLOG_KEY_OOB_PROTOCOL] = {
			.type = ULOGD_RET_UINT16,
			.flags = ULOGD_RETF_NONE,
			.name = "oob.protocol",
		},
		[NFLOG_KEY_OOB_UID] = {
			.type = ULOGD_RET_UINT32,
			.flags = ULOGD_RETF_NONE,
			.name = "oob.uid",
		},
		[NFLOG_KEY_OOB_GID] = {
			.type = ULOGD_RET_UINT32,
			.flags = ULOGD_RETF_NONE,
			.name = "oob.gid",
		},
		[NFLOG_KEY_RAW_LABEL] = {
			.type = ULOGD_RET_UINT8,
			.flags = ULOGD_RETF_NONE,
			.name = "raw.label",
		},
		[NFLOG_KEY_RAW_TYPE] = {
			.type = ULOGD_RET_UINT16,
			.flags = ULOGD_RETF_NONE,
			.name = "raw.type",
		},
		[NFLOG_KEY_RAW] = {
			.type = ULOGD_RET_RAW,
			.flags = ULOGD_RETF_NONE,
			.name = "raw",
		},
};
```

1.4 Configuration

1. enable plugins

You will have to choose the input and output plugins according to your setup. NFLOG is present in recent kernels (and iptables installation), and should be preferred if possible.

Edit the ulogd config file in `/etc/ulogd.conf`

```
######################################################################
# PLUGIN OPTIONS
######################################################################

# We have to configure and load all the plugins we want to use

# general rules:
# 1. load the plugins _first_ from the global section
# 2. options for each plugin in seperate section below

#plugin="/usr/lib/ulogd/ulogd_LOGEMU.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_inppkt_NFLOG.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_inppkt_ULOG.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_inppkt_UNIXSOCK.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_inpflow_NFCT.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_IFINDEX.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_IP2STR.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_IP2BIN.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_IP2HBIN.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_PRINTPKT.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_HWHDR.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_PRINTFLOW.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_MARK.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_LOGEMU.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_SYSLOG.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_XML.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_SQLITE3.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_GPRINT.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_NACCT.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_PCAP.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_PGSQL.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_MYSQL.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_DBI.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_raw2packet_BASE.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_inpflow_NFACCT.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_GRAPHITE.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_JSON.so"
######################################################################
```

2. Build Stack

Enable the correct stack: (ensure all other stacks are commented out unless you need them for some other reason).

```
# this is a stack for logging packet send by system via LOGEMU
```

```
stack=log1:NFLOG,base1:BASE,ifi1:IFINDEX,ip2str1:IP2STR,print1:PRINTPKT,emu1:LOGEM
U
```

Set the path for your iptables log file under `[emu1]`

3. Configure input plugin

```
[log1]
group = 1
```

4. Configure output plugin

```
[emu1]
file="/var/log/ulogd/iptables.log"
sync=1
```

After creating the configuration file, ensure that `ulogd2` is restarted and that the directory `/var/log/ulog` exists.

1.5 Options

Belows is options that we can select based on our purpose.

```
# Example configuration for ulogd
# Adapted to Debian by Achilleas Kotsis <achille@debian.gr>

[global]
######################################################################
# GLOBAL OPTIONS
######################################################################


# logfile for status messages
logfile="/var/log/ulogd.log"

# loglevel: debug(1), info(3), notice(5), error(7) or fatal(8) (default 5)
# loglevel=1

######################################################################
# PLUGIN OPTIONS
######################################################################
```

```
# We have to configure and load all the plugins we want to use

# general rules:
# 1. load the plugins _first_ from the global section
# 2. options for each plugin in seperate section below


plugin="@pkglibdir@/ulogd_inppkt_NFLOG.so"
#plugin="@pkglibdir@/ulogd_inppkt_ULOG.so"
#plugin="@pkglibdir@/ulogd_inppkt_UNIXSOCK.so"
plugin="@pkglibdir@/ulogd_inpflow_NFCT.so"
plugin="@pkglibdir@/ulogd_filter_IFINDEX.so"
plugin="@pkglibdir@/ulogd_filter_IP2STR.so"
plugin="@pkglibdir@/ulogd_filter_IP2BIN.so"
#plugin="@pkglibdir@/ulogd_filter_IP2HBIN.so"
plugin="@pkglibdir@/ulogd_filter_PRINTPKT.so"
plugin="@pkglibdir@/ulogd_filter_HWHDR.so"
plugin="@pkglibdir@/ulogd_filter_PRINTFLOW.so"
#plugin="@pkglibdir@/ulogd_filter_MARK.so"
plugin="@pkglibdir@/ulogd_output_LOGEMU.so"
plugin="@pkglibdir@/ulogd_output_SYSLOG.so"
plugin="@pkglibdir@/ulogd_output_XML.so"
#plugin="@pkglibdir@/ulogd_output_SQLITE3.so"
plugin="@pkglibdir@/ulogd_output_GPRINT.so"
#plugin="@pkglibdir@/ulogd_output_NACCT.so"
#plugin="@pkglibdir@/ulogd_output_PCAP.so"
#plugin="@pkglibdir@/ulogd_output_PGSQL.so"
#plugin="@pkglibdir@/ulogd_output_MYSQL.so"
#plugin="@pkglibdir@/ulogd_output_DBI.so"
plugin="@pkglibdir@/ulogd_raw2packet_BASE.so"
plugin="@pkglibdir@/ulogd_inpflow_NFACCT.so"
plugin="@pkglibdir@/ulogd_output_GRAPHITE.so"
#plugin="@pkglibdir@/ulogd_output_JSON.so"

# this is a stack for logging packet send by system via LOGEMU
#stack=log1:NFLOG,base1:BASE,ifi1:IFINDEX,ip2str1:IP2STR,print1:PRINTPKT,emu1:LOGE
MU

# this is a stack for packet-based logging via LOGEMU
#stack=log2:NFLOG,base1:BASE,ifi1:IFINDEX,ip2str1:IP2STR,print1:PRINTPKT,emu1:LOGE
MU

# this is a stack for ULOG packet-based logging via LOGEMU
#stack=ulog1:ULOG,base1:BASE,ip2str1:IP2STR,print1:PRINTPKT,emu1:LOGEMU

# this is a stack for packet-based logging via LOGEMU with filtering on MARK
#stack=log2:NFLOG,mark1:MARK,base1:BASE,ifi1:IFINDEX,ip2str1:IP2STR,print1:PRINTPK
T,emu1:LOGEMU

# this is a stack for packet-based logging via GPRINT
#stack=log1:NFLOG,gp1:GPRINT

# this is a stack for flow-based logging via LOGEMU
#stack=ct1:NFCT,ip2str1:IP2STR,print1:PRINTFLOW,emu1:LOGEMU

# this is a stack for flow-based logging via GPRINT
#stack=ct1:NFCT,gp1:GPRINT
```

```
# this is a stack for flow-based logging via XML
#stack=ct1:NFCT,xml1:XML

# this is a stack for logging in XML
#stack=log1:NFLOG,xml1:XML

# this is a stack for accounting-based logging via XML
#stack=acct1:NFACCT,xml1:XML

# this is a stack for accounting-based logging to a Graphite server
#stack=acct1:NFACCT,graphite1:GRAPHITE

# this is a stack for NFLOG packet-based logging to PCAP
#stack=log2:NFLOG,base1:BASE,pcap1:PCAP

# this is a stack for logging packet to MySQL
#stack=log2:NFLOG,base1:BASE,ifi1:IFINDEX,ip2bin1:IP2BIN,mac2str1:HWHDR,mysql1:MYS
QL

# this is a stack for logging packet to PGsql after a collect via NFLOG
#stack=log2:NFLOG,base1:BASE,ifi1:IFINDEX,ip2str1:IP2STR,mac2str1:HWHDR,pgsql1:PGS
QL

# this is a stack for logging packet to JSON formatted file after a collect via
NFLOG
#stack=log2:NFLOG,base1:BASE,ifi1:IFINDEX,ip2str1:IP2STR,mac2str1:HWHDR,json1:JSON

# this is a stack for logging packets to syslog after a collect via NFLOG
#stack=log3:NFLOG,base1:BASE,ifi1:IFINDEX,ip2str1:IP2STR,print1:PRINTPKT,sys1:SYSL
OG

# this is a stack for logging packets to syslog after a collect via NuFW
#stack=nuauth1:UNIXSOCK,base1:BASE,ip2str1:IP2STR,print1:PRINTPKT,sys1:SYSLOG

# this is a stack for flow-based logging to MySQL
#stack=ct1:NFCT,ip2bin1:IP2BIN,mysql2:MYSQL

# this is a stack for flow-based logging to PGSQL
#stack=ct1:NFCT,ip2str1:IP2STR,pgsql2:PGSQL

# this is a stack for flow-based logging to PGSQL without local hash
#stack=ct1:NFCT,ip2str1:IP2STR,pgsql3:PGSQL

# this is a stack for flow-based logging to SQLITE3
#stack=ct1:NFCT,sqlite3_ct:SQLITE3

# this is a stack for logging packet to SQLITE3
#stack=log1:NFLOG,sqlite3_pkt:SQLITE3

# this is a stack for flow-based logging in NACCT compatible format
#stack=ct1:NFCT,ip2str1:IP2STR,nacct1:NACCT

# this is a stack for accounting-based logging via GPRINT
#stack=acct1:NFACCT,gp1:GPRINT

[ct1]
```

```
#netlink_socket_buffer_size=217088
#netlink_socket_buffer_maxsize=1085440
#netlink_resync_timeout=60 # seconds to wait to perform resynchronization
#pollinterval=10 # use poll-based logging instead of event-driven
# If pollinterval is not set, NFCT plugin will work in event mode
# In this case, you can use the following filters on events:
#accept_src_filter=192.168.1.0/24,1:2::/64 # source ip of connection must belong
to these networks
#accept_dst_filter=192.168.1.0/24 # destination ip of connection must belong to
these networks
#accept_proto_filter=tcp,sctp # layer 4 proto of connections

[ct2]
#netlink_socket_buffer_size=217088
#netlink_socket_buffer_maxsize=1085440
#reliable=1 # enable reliable flow-based logging (may drop packets)
hash_enable=0

# Logging of system packet through NFLOG
[log1]
# netlink multicast group (the same as the iptables --nflog-group param)
# Group O is used by the kernel to log connection tracking invalid message
group=0
#netlink_socket_buffer_size=217088
#netlink_socket_buffer_maxsize=1085440
# set number of packet to queue inside kernel
#netlink_qthreshold=1
# set the delay before flushing packet in the queue inside kernel (in 10ms)
#netlink_qtimeout=100

# packet logging through NFLOG for group 1
[log2]
# netlink multicast group (the same as the iptables --nflog-group param)
group=1 # Group has to be different from the one use in log1
#netlink_socket_buffer_size=217088
#netlink_socket_buffer_maxsize=1085440
# If your kernel is older than 2.6.29 and if a NFLOG input plugin with
# group 0 is not used by any stack, you need to have at least one NFLOG
# input plugin with bind set to 1. If you don't do that you may not
# receive any message from the kernel.
#bind=1

# packet logging through NFLOG for group 2, numeric_label is
# set to 1
[log3]
# netlink multicast group (the same as the iptables --nflog-group param)
group=2 # Group has to be different from the one use in log1/log2
numeric_label=1 # you can label the log info based on the packet verdict
#netlink_socket_buffer_size=217088
#netlink_socket_buffer_maxsize=1085440
#bind=1

[ulog1]
# netlink multicast group (the same as the iptables --ulog-nlgroup param)
nlgroup=1
#numeric_label=0 # optional argument
```

```
[nuauth1]
socket_path="/tmp/nuauth_ulogd2.sock"

[emu1]
file="/var/log/ulogd_syslogemu.log"
sync=1

[op1]
file="/var/log/ulogd_oprint.log"
sync=1

[gp1]
file="/var/log/ulogd_gprint.log"
sync=1
timestamp=1

[xml1]
directory="/var/log/"
sync=1

[json1]
sync=1
#file="/var/log/ulogd.json"
#timestamp=0
# device name to be used in JSON message
#device="My awesome Netfilter firewall"
# If boolean_label is set to 1 then the numeric_label put on packet
# by the input plugin is coding the action on packet: if 0, then
# packet has been blocked and if non null it has been accepted.
#boolean_label=1

[pcap1]
#default file is /var/log/ulogd.pcap
#file="/var/log/ulogd.pcap"
sync=1

[mysql1]
db="nulog"
host="localhost"
user="nupik"
table="ulog"
pass="changeme"
procedure="INSERT_PACKET_FULL"
# backlog configuration:
# set backlog_memcap to the size of memory that will be
# allocated to store events in memory if data is temporary down
# and insert them when the database came back.
#backlog_memcap=1000000
# number of events to insert at once when backlog is not empty
#backlog_oneshot_requests=10

[mysql2]
db="nulog"
host="localhost"
user="nupik"
table="conntrack"
pass="changeme"
```

```
procedure="INSERT_CT"

[pgsql1]
db="nulog"
host="localhost"
user="nupik"
table="ulog"
#schema="public"
pass="changeme"
procedure="INSERT_PACKET_FULL"
# connstring can be used to define PostgreSQL connection string which
# contains all parameters of the connection. If set, this value has
# precedence on other variables used to build the connection string.
# See http://www.postgresql.org/docs/9.2/static/libpq-connect.html#LIBPQ-
CONNSTRING
# for a complete description of options.
#connstring="host=localhost port=4321 dbname=nulog user=nupik password=changeme"
#backlog_memcap=1000000
#backlog_oneshot_requests=10
# If superior to 1 a thread dedicated to SQL request execution
# is created. The value stores the number of SQL request to keep
# in the ring buffer
#ring_buffer_size=1000

[pgsql2]
db="nulog"
host="localhost"
user="nupik"
table="ulog2_ct"
#schema="public"
pass="changeme"
procedure="INSERT_CT"

[pgsql3]
db="nulog"
host="localhost"
user="nupik"
table="ulog2_ct"
#schema="public"
pass="changeme"
procedure="INSERT_OR_REPLACE_CT"

[pgsql4]
db="nulog"
host="localhost"
user="nupik"
table="nfacct"
#schema="public"
pass="changeme"
procedure="INSERT_NFACCT"

[dbi1]
db="ulog2"
dbtype="pgsql"
host="localhost"
user="ulog2"
table="ulog"
```

```
pass="ulog2"
procedure="INSERT_PACKET_FULL"

[sqlite3_ct]
table="ulog_ct"
db="/var/log/ulogd.sqlite3db"
buffer=200

[sqlite3_pkt]
table="ulog_pkt"
db="/var/log/ulogd.sqlite3db"
buffer=200

[sys2]
facility=LOG_LOCAL2

[nacct1]
sync = 1
#file = /var/log/ulogd_nacct.log

[mark1]
mark = 1

[acct1]
pollinterval = 2
# If set to 0, we don't reset the counters for each polling (default is 1).
#zerocounter = 0
# Set timestamp (default is 0, which means not set). This timestamp can be
# interpreted by the output plugin.
#timestamp = 1

[graphite1]
host="127.0.0.1"
port="2003"
# Prefix of data name sent to graphite server
prefix="netfilter.nfacct"
Footer
© 2023 GitHub, Inc.
Footer navigation
Terms
Privacy
Security
Status
Docs
Contact GitHub
Pricing
API
Training
Blog
About
```

2. Demo

With JSON log file we need libjannson.so to ulogd can export log file JSON. Therefor, we need added jansson.bb file and change a few little ulogd.bb file.

- Setup jansson version 2.9

```
SUMMARY = "Jansson is a C library for encoding, decoding and manipulating JSON
data"
HOMEPAGE = "http://www.digip.org/jansson/"
LICENSE = "GPLv2+"
LIC_FILES_CHKSUM = "file://LICENSE;md5=8b70213ec164c7bd876ec2120ba52f61"

SRC_URI = "http://www.digip.org/jansson/releases/${BPN}-${PV}.tar.gz"

SRC_URI[md5sum] = "84abaefee9502b2f2ff394d758f160c7"
SRC_URI[sha256sum] =
"0ad0d074ca049a36637e7abef755d40849ad73e926b93914ce294927b97bd2a5"

inherit autotools pkgconfig
```

- Change ulogd.bb file according below:

+ Adding "jansson" into "DEPENDS" and "RDEPENDS..".

+ At "EXTRA_OECONF", if have "–without-jansson" then deleting it.

```
SUMMARY = "Netfilter userspace logging daemon"
DESCRIPTION = "Userspace logging daemon for netfilter/iptables related logging"
HOMEPAGE = "http://www.netfilter.org/projects/ulogd/index.html"
LICENSE = "GPLv2+"
LIC_FILES_CHKSUM = "file://COPYING;md5=c93c0550bd3173f4504b2cbd8991e50b"
DEPENDS = "libnfnetlink libmnl libnetfilter-log libnetfilter-conntrack jansson"
RDEPENDS_${PN} = "libnfnetlink libmnl libnetfilter-log libnetfilter-conntrack
jansson"
PR = "r2"

S = "${WORKDIR}/${PN}-${PV}"

SRC_URI = " \
    http://www.netfilter.org/projects/ulogd/files/ulogd-${PV}.tar.bz2;name=tar \
    file://ulogd.service \
    file://ulogd.conf.in.user \
    file://ulogd.logrotate.user \
"
SRC_URI[tar.md5sum] = "2bb2868cf51acbb90c35763c9f995f31"
SRC_URI[tar.sha256sum] =
"990a05494d9c16029ba0a83f3b7294fc05c756546b8d60d1c1572dc25249a92b"

EXTRA_OECONF = "\
    --disable-nfacct \
    --without-dbi    \
    --without-sqlite \
    --without-pgsql  \
    --without-mysql  \
    --without-pcap \
"
```

- Edit ulogd.conf.in.user file

  - Input plugin: NFLOG

  - Output: JSON and LOGEMU

```
########################################################################
# PLUGIN OPTIONS
########################################################################

# We have to configure and load all the plugins we want to use

# general rules:
# 1. load the plugins _first_ from the global section
# 2. options for each plugin in seperate section below

#plugin="/usr/lib/ulogd/ulogd_LOGEMU.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_inppkt_NFLOG.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_inppkt_ULOG.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_inppkt_UNIXSOCK.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_inpflow_NFCT.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_IFINDEX.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_IP2STR.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_IP2BIN.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_IP2HBIN.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_PRINTPKT.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_HWHDR.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_PRINTFLOW.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_filter_MARK.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_LOGEMU.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_SYSLOG.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_XML.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_SQLITE3.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_GPRINT.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_NACCT.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_PCAP.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_PGSQL.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_MYSQL.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_DBI.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_raw2packet_BASE.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_inpflow_NFACCT.so"
plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_GRAPHITE.so"
#plugin="/usr/lib/x86_64-linux-gnu/ulogd/ulogd_output_JSON.so"
```

```
######################################################################
2. Build Stack

Enable the correct stack: (ensure all other stacks are commented out unless you
need them for some other reason).

# this is a stack for logging packet send by system via LOGEMU
stack=log1:NFLOG,base1:BASE,ifi1:IFINDEX,ip2str1:IP2STR,print1:PRINTPKT,emu1:LOGEM
U
Set the path for your iptables log file under [emu1]

3. Configure input plugin

[log1]
group = 1
4. configure output plugin



[emu1]
file="/var/log/ulogd/iptables.log"
sync=1
After creating the configuration file, ensure that ulogd2 is restarted and that
the directory /var/log/ulog exists.

Example: Use format log is JSON  and LOGEMU
Edit ulogd.conf.in.user file

         - Input plugin:  NFLOG

         - Output: JSON and LOGEMU

[global]
######################################################################
# GLOBAL OPTIONS
######################################################################

# logfile for status messages
logfile="/data/log_data/ulogd.log"

# loglevel: debug(1), info(3), notice(5), error(7) or fatal(8) (default 5)
# loglevel=1

######################################################################
# PLUGIN OPTIONS
######################################################################

# We have to configure and load all the plugins we want to use

# general rules:
# 1. load the plugins _first_ from the global section
# 2. options for each plugin in seperate section below

plugin="@pkglibdir@/ulogd_inppkt_NFLOG.so"
plugin="@pkglibdir@/ulogd_filter_IFINDEX.so"
plugin="@pkglibdir@/ulogd_filter_HWHDR.so"
plugin="@pkglibdir@/ulogd_filter_IP2STR.so"
```

```
plugin="@pkglibdir@/ulogd_filter_PRINTPKT.so"
plugin="@pkglibdir@/ulogd_raw2packet_BASE.so"
plugin="@pkglibdir@/ulogd_output_LOGEMU.so"
plugin="@pkglibdir@/ulogd_output_JSON.so"

# full log
stack=nflog0:NFLOG,base0:BASE,ifi0:IFINDEX,hwhdr0:HWHDR,ip2str0:IP2STR,print1:PRIN
TPKT,full0:LOGEMU
stack=nflog1:NFLOG,base0:BASE,ifi0:IFINDEX,hwhdr0:HWHDR,ip2str0:IP2STR,print1:PRIN
TPKT,full0:LOGEMU
stack=nflog2:NFLOG,base0:BASE,ifi0:IFINDEX,ip2str0:IP2STR,mac2str1:HWHDR,json1:JSO
N

[nflog0]
group=29

[nflog1]
group=30

[nflog2]
group=31

[full0]
file="/data/log_data/ulogd/full.log"
sync=1

[json1]
file="/data/log_data/ulogd/firewall_log.json"
sync=1
```

Where:

@pkglibdir@ is determined in Makefile.in

- Configure iptables with number groups: 29,30,31

```
-A MC_OUT   -m limit    --limit 1/m -j NFLOG    --nflog-prefix "0x04"  --nflog-
group 29
-A OUTPUT   -m limit    --limit 1/m -j NFLOG    --nflog-prefix "0x05"  --nflog-
group 30
-A INPUT    -i rmnet_data+             -j NFLOG    --nflog-prefix "INPUT" --nflog-
group 31
```

- Format output log actually is getted.

+ full.log file

```
Aug 25 19:55:27 0x05 IN=bridge0 OUT= MAC=
xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx SRC=0.0.0.0 DST=224.0.0.1
LEN=32 TOS=00 PREC=0xC0 TTL=1 ID=0 DF PROTO=2 MARK=0
Aug 25 19:55:27 0x05 IN=bridge0 OUT= MAC=
xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx SRC=xx::xx:xx:xx:xx
DST=ff02::1 LEN=72 TC=0 HOPLIMIT=1 FLOWLBL=0 PROTO=ICMPv6 TYPE=130 CODE=0 MARK=0
```

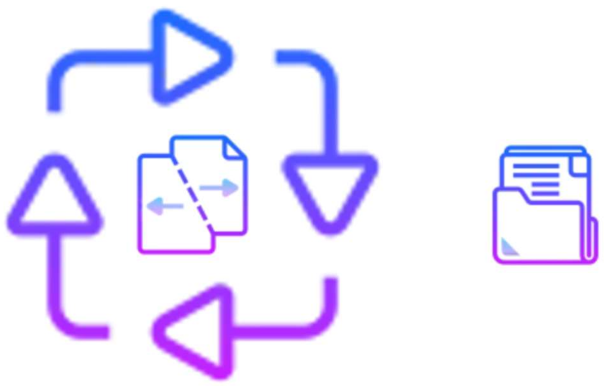
```
Aug 25 19:56:11 0x05 IN=wlan10 OUT= MAC=
xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx SRC=0.0.0.0
DST=255.255.255.255 LEN=328 TOS=00 PREC=0x00 TTL=64 ID=17722 PROTO=UDP SPT=68
DPT=67 LEN=308 MARK=0
Aug 25 19:56:12 0x05 IN=wlan10 OUT= MAC=
xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx SRC=0.0.0.0
DST=255.255.255.255 LEN=328 TOS=00 PREC=0x00 TTL=64 ID=27220 PROTO=UDP SPT=68
DPT=67 LEN=308 MARK=0
Aug 25 19:56:19 0x05 IN=wlan10 OUT= MAC=
xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx SRC=192.168.xx.xx
DST=255.255.255.255 LEN=201 TOS=00 PREC=0x00 TTL=64 ID=0 DF PROTO=UDP SPT=58325
DPT=7423 LEN=181 MARK=0
Aug 25 19:56:20 0x05 IN=wlan10 OUT= MAC=ff:
xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx SRC=192.168.xx.xx
DST=192.168.xx.xx LEN=194 TOS=00 PREC=0x00 TTL=64 ID=0 DF PROTO=UDP SPT=2190
DPT=2190 LEN=174 MARK=0
Aug 25 19:56:24 0x05 IN= OUT=wlan10 MAC= SRC=192.168.xx.xx DST=172.16.xx.xx LEN=37
TOS=00 PREC=0x00 TTL=64 ID=44412 DF PROTO=UDP SPT=xxxx DPT=xxxx LEN=17 UID=0 GID=0
MARK=0
Aug 25 19:56:24 0x05 IN= OUT=wlan10 MAC= SRC=192.168.xx.xx DST=172.16.xx.xx LEN=37
TOS=00 PREC=0x00 TTL=64 ID=44413 DF PROTO=UDP SPT=xxxx DPT=xxxx LEN=17 UID=0 GID=0
MARK=0
Aug 25 19:57:07 0x04 IN= OUT=wlan10 MAC= SRC=xx::xx:xx:xx:xx DST=ff02::2 LEN=56
TC=0 HOPLIMIT=255 FLOWLBL=0 PROTO=ICMPv6 TYPE=133 CODE=0 MARK=0
```

+ firewall_log.json file

```
{"oob.prefix": "INPUT", "oob.time.sec": 1679390587, "raw.mac_len": 18,
"oob.protocol": 2048, "ip.protocol": 6, "ip.tos": 0, "ip.ttl": 0, "ip.totlen": 45,
"ip.ihl": 5, "ip.csum": 9568, "ip.id": 0, "ip.fragoff": 0, "src_port": 11111,
"dest_port": 11111, "tcp.seq": 0, "tcp.ackseq": 0, "tcp.window": 8192,
"tcp.offset": 0, "tcp.reserved": 0, "tcp.urg": 0, "tcp.ack": 0, "tcp.psh": 0,
"tcp.rst": 0, "tcp.syn": 1, "tcp.fin": 0, "tcp.res1": 0, "tcp.res2": 0,
"tcp.csum": 19570, "src_ip": "192.168.xx.xx", "dest_ip": "192.168.xx.xx",
"mac.str": "xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx.xx"}
{"oob.prefix": "OUTPUT", "oob.time.sec": 1679390587, "oob.ifindex_out": 9,
"oob.protocol": 2048, "ip.protocol": 6, "ip.tos": 0, "ip.ttl": 64, "ip.totlen":
40, "ip.ihl": 5, "ip.csum": 42340, "ip.id": 0, "ip.fragoff": 16384, "src_port":
17687, "dest_port": 17686, "tcp.seq": 0, "tcp.ackseq": 6, "tcp.window": 0,
"tcp.offset": 0, "tcp.reserved": 0, "tcp.urg": 0, "tcp.ack": 1, "tcp.psh": 0,
"tcp.rst": 1, "tcp.syn": 0, "tcp.fin": 0, "tcp.res1": 0, "tcp.res2": 0,
"tcp.csum": 36913, "src_ip": "192.168.xx.xx", "dest_ip": "192.168.xx.xx"}
```

B. Logrotate

1.Introduction



Logrotate is used to manage & rotate the logs based on the age of the file or the file size. It automatically archives old logs, deletes them after a certain number of logs and also creates a new log file according to the given configuration.

Why to rotate logs? Logs are files that grow over time. Since they keep on filling over time. If you don't maintain them they will end up filling your mount point which you never want to see in production! Means, to save your disk space logs should be rotated. Logs are always useful for troubleshooting but yes you don't need pretty old logs keeping your disk space on toss! Logs can be purged, compressed, or moved manually. But to save your time from manual tasks you can automate log management using this tool.

In general, log files are residing under `/var/log` directory in Linux but it's not limited and can be placed anywhere.

Important Logrotate files can be found in the following location's:

- **/etc/logrotate.conf :** The main logrotate configuration file

- **/etc/logrotate.d :** It contains the application-specific configuration files.

2. Usage

Configuration

ulogd.logrotate.user file

```
/data/log_data/ulogd.log /data/log_data/ulogd/*.log
/data/log_data/ulogd/firewall_log.json {
missingok
size 1M
rotate 5
nodateext
sharedscripts
postrotate
/usr/bin/killall -HUP ulogd 2> /dev/null || true
endscript
}
```

Where:

- **/data/log_data/ulogd.log /data/log_data/ulogd/*.log /data/log_data/ulogd/firewall_log.json** - Format file will be matched to logrotate.
- **missingok** – Ignoring output error if logfile is deleted.
- **size 1M** - Log file is rotated only if it grows bigger than 1Megabyte.
- **rotate 5** - It keeps a 5 weeks backup of all log files.
- **nodateext** - Do not archive old versions of log files with date extension (this overrides the **dateext** option).
- **sharedscripts** - This command tells logrotate to check all the logs for that configuration block before running the `postrotate` script. If one or both of the logs is rotated, the `postrotate` script runs only once. If none of the logs is rotated, the `postrotate` script doesn't run.
- **postrotate/endscript** - The lines betweenpostrotateandendscript(both of which must appear on lines by themselves) are executed after the log file is rotated.

3. Advanced Usage

Once your **logrotate** configuration has been implemented, the cron scheduler on your system will invoke the command automatically so that **logrotate** can do its job. Ordinarily, users will not need to execute the **logrotate** command manually. However, the **logrotate** command can still be run manually in terminal and has a few different options you can use. Check the examples below to see how that works.

Example 1: You can experiment with the log rotation by forcing an execution of the **logrotate** in the absence of any log files to rotate. To do so, use the `-f` option and specify the configuration file you wish to use. With logrotate configure is rotate 5, hence max file were stored 5 file as below.

```
 # logrotate -f /etc/logrotate.d/ulogd.logrotate
 # ls -lah
total 48
drwxr-xr-x    2 root     root        1.0K Apr 26 19:47 .
drwxr-xr-x    4 root     root         720 Apr 26 19:47 ..
-rw-r--r--    1 root     root           0 Apr 26 19:47 firewall_log.json
-rw-r--r--    1 root     root           0 Apr 26 19:47 firewall_log.json.1
-rw-r--r--    1 root     root           0 Apr 26 19:47 firewall_log.json.2
-rw-r--r--    1 root     root           0 Apr 26 19:43 firewall_log.json.3
-rw-r--r--    1 root     root           0 Apr 26 19:42 firewall_log.json.4
-rw-r--r--    1 root     root           0 Apr 26 19:42 firewall_log.json.5
-rw-r--r--    1 root     root           0 Apr 26 19:47 full.log
-rw-r--r--    1 root     root           0 Apr 26 19:47 full.log.1
-rw-r--r--    1 root     root           0 Apr 26 19:47 full.log.2
-rw-r--r--    1 root     root         744 Apr 26 19:46 full.log.3
-rw-r--r--    1 root     root           0 Apr 26 19:42 full.log.4
-rw-r--r--    1 root     root           0 Apr 26 19:42 full.log.5
```

Example 2: If you experience any problems, and wish to debug, you can use the `-d` option with **logrotate**. This will simulate a "test run" and not actually make any changes. Instead, it will only output debug messages to help with troubleshooting.

```
# logrotate -d /etc/logrotate.d/ulogd.logrotate
reading config file /etc/logrotate.d/ulogd.logrotate
Reading state from file: /var/lib/logrotate.status
Allocating hash table for state file, size 64 entries
Creating new state
Creating new state
Creating new state

Handling 1 logs

rotating pattern: /data/log_data/ulogd.log /data/log_data/ulogd/*.log
/data/log_data/ulogd/firewall_log.json  1048576 bytes (5 rotations)
empty log files are rotated, old logs are removed
considering log /data/log_data/ulogd.log
  Now: 2023-04-26 20:21
  Last rotated at 2023-04-26 19:47
  log does not need rotating (log size is below the 'size' threshold)
considering log /data/log_data/ulogd/full.log
  Now: 2023-04-26 20:21
  Last rotated at 2023-04-26 19:47
  log does not need rotating (log size is below the 'size' threshold)
considering log /data/log_data/ulogd/firewall_log.json
  Now: 2023-04-26 20:21
  Last rotated at 2023-04-26 19:47
  log does not need rotating (log size is below the 'size' threshold)
not running postrotate script, since no logs were rotated
```

Example 3: Use the `-v` option to turn on verbosity. This will display messages during rotation so you can see exactly what's going on.

```
/ # logrotate -v /etc/logrotate.d/ulogd.logrotate
reading config file /etc/logrotate.d/ulogd.logrotate
Reading state from file: /var/lib/logrotate.status
Allocating hash table for state file, size 64 entries
Creating new state
Creating new state
Creating new state

Handling 1 logs

rotating pattern: /data/log_data/ulogd.log /data/log_data/ulogd/*.log
/data/log_data/ulogd/firewall_log.json 1048576 bytes (5 rotations)
empty log files are rotated, old logs are removed
considering log /data/log_data/ulogd.log
Now: 2023-04-26 20:27
Last rotated at 2023-04-26 19:47
log does not need rotating (log size is below the 'size' threshold)
```

```
considering log /data/log_data/ulogd/full.log
Now: 2023-04-26 20:27
Last rotated at 2023-04-26 19:47
log does not need rotating (log size is below the 'size' threshold)
considering log /data/log_data/ulogd/firewall_log.json
Now: 2023-04-26 20:27
Last rotated at 2023-04-26 19:47
log does not need rotating (log size is below the 'size' threshold)
not running postrotate script, since no logs were rotated
```

C. Questions, Exercises

TBD

D. References

No.	Info	Link/ file/ name of ebook
1		https://linuxconfig.org/logrotate-8-manual-page
2		https://adamtheautomator.com/logrotate-linux/#:~:text=Logrotate%20uses%20configuration%20files%20to,file%20and%20%2Fetc%2Flogrotate.
3		https://netslovers.com/post/understanding-logrotate-on-centos-part-1/
4		https://rlworkman.net/howtos/ulogd.html