Ulogd2, Netfilter logging reloaded

Eric Leblond

NFWS2013, Copenhagen
Introduction
- Netfilter logging history
- Ulogd2

Connection tracking

Ulogd2 Architecture

Using Ulogd2

Conclusion
Some word about me

Eric Leblond

- French
- Previously, co-founder and CTO of EdenWall (RIP)
- Now, Contractor
- Suricata IDS/IPS developer
- @Regiteric on Twitter
### Eric Leblond
- French
- Previously, co-founder and CTO of EdenWall (RIP)
- Now, Contractor
- Suricata IDS/IPS developer
- @Regiteric on Twitter

### regit@netfilter.org
- Netfilter Coreteam Member
- Working on:
  - some kernel stuff
  - libnetfilter_queue and userspace library
  - ulogd2 maintainer
At the beginning was syslog

Pre Netfilter days

- Flat packet logging
- One line per packet
  - A lot of information
  - Non searchable
At the beginning was syslog

Pre Netfilter days
- Flat packet logging
- One line per packet
  - A lot of information
  - Non searchable

Not sexy

```
INPUT DROP IN=eth0 OUT= MAC=00:1a:92:05:ee:68:00:b0:8e:83:3b:f0:08:00 SRC=62.212.121.211 DST=91.121.73.151 LEN=60 TOS=0x00 PREC=0x00 TTL=58 ID=35342 DF PROTO=TCP SPT=59261 DPT=113 WINDOW=5440 RES=0x00 SYN URGP=0
```
```
INPUT IN=eth0 OUT= MAC=d4:be:d9:69:d1:51:00:11:95:63:c7:5e:08:00 SRC=31.13.80.7 DST=192.168.11.3 LEN=40 TOS=0x00 PREC=0x00 TTL=244 ID=37732 DF PROTO=TCP SPT=443 DPT=48875 WINDOW=0 RES=0x00 ACK RST URGP=0
```
```
INPUT IN=eth0 OUT= MAC=d4:be:d9:69:d1:51:00:11:95:63:c7:5e:08:00 SRC=31.13.80.23 DST=192.168.11.3 LEN=40 TOS=0x00 PREC=0x00 TTL=243 ID=33964 DF PROTO=TCP SPT=80 DPT=49617 WINDOW=0 RES=0x00 ACK RST URGP=0
```
```
INPUT IN=eth0 OUT= MAC=d4:be:d9:69:d1:51:00:11:95:63:c7:5e:08:00 SRC=31.13.80.7 DST=192.168.11.3 LEN=40 TOS=0x00 PREC=0x00 TTL=244 ID=62292 DF PROTO=TCP SPT=80 DPT=60462 WINDOW=0 RES=0x00 ACK RST URGP=0
```
```
INPUT IN=eth0 OUT= MAC=d4:be:d9:69:d1:51:00:11:95:63:c7:5e:08:00 SRC=31.13.80.7 DST=192.168.11.3 LEN=40 TOS=0x00 PREC=0x00 TTL=244 ID=22480 DF PROTO=TCP SPT=443 DPT=50876 WINDOW=0 RES=0x00 ACK RST URGP=0
```
Ulogd days

**ULOG**

- Netfilter introduces ULOG target
  
  ```
  iptables -A INPUT -p tcp -j ULOG --ulog-prefix "bad packet"
  ```

- Communication via a netlink socket
  - Special type of socket
  - used for kernel userspace bidirectionnal communication
**ULOG**

- Netfilter introduces ULOG target
  ```bash
  iptables -A INPUT -p tcp -j ULOG --ulog-prefix "bad packet"
  ```
- Communication via a netlink socket
  - Special type of socket
  - used for kernel userspace bidirectional communication

---

**Ulogd, a ULOG logging daemon**

- Syslog and file output
- SQL output: PGSQL, MySQL, SQLite
Netfilter introduces NFnetlink

- Rewrote userspace interaction
- For logging, queueing and connection tracking
- Multiple communication on a single netlink socket

New libraries

- libnetfilter_queue: userspace decision
- libnetfilter_log: logging
- libnetfilter_conntrack: connection tracking handling
# Ulogd2: an ulogd generalisation

## Ulogd2
- Interact with the new libraries
- Rewrite of ulogd

## libnetfilter_log (generalized ulog)
- Packet logging
- IPv6 ready
- Few structural modification

## libnetfilter_conntrack (new)
- Connection tracking logging
- Accounting, logging

## libnetfilter_nfacct (added recently)
- High performance accounting
Netfilter connection tracking

- Netfilter maintains a connection table
- Valid for "all" protocols
  - For flow-oriented protocol: TCP, SCTP
  - For protocol without state: UDP
- Support both IPv4 and IPv6
Network Address Translation

- Private Network can’t go to internet
- Firewall has to modify packet to show its address
- Two way of seeing a connection
  - From inside
  - From outside
- Conntrack keep track of the correspondance

```
tcp 6 431996 ESTABLISHED src=192.168.1.131 dst=91.121.73.151 sport=52964 dport=22\ 
packets=13 bytes=772 src=91.121.73.151 dst=192.168.1.131 sport=22 dport=52964 \ 
packets=11 bytes=7548 [ASSURED] mark=0 secmark=0 use=1 \
```
## Interrogation
- Connections listing
- Retrieve information about a connection
  - IP information
  - Accounting statistics
- Event mode

## Modification
- Create new entry
- Change or fix timeout
- Change mark
- Destruction of entries
Send all significative connection related events to userspace:
- NEW: connection creation
- ESTABLISHED: Switch from NEW to ESTABLISHED connection
- DESTROY: connection destruction

Make possible to maintain a connection history in userspace

Accounting information

NAT decision history
Ulogd2, a modular daemon

Able to use multiple entries
- Packet logging
- Flow logging
- Accounting

And multiple output
- Text based
- DB based

Plugin based architecture
- Entry
- Output
- Filters
图9.3 Vi的三种模式

一般模式：
光标移动，搜索、替换、删除字符、删除整行、复制整行、粘贴整行

编辑模式：
可输入任何字符

命令行模式：
读取、存储文件以及其他附加功能
Ulogd2, schema of architecture

Diagram showing the architecture of Ulogd2, including NFLOG, NFCT, INPUT, Core, DB, FILE, Filter 1, Filter 2, Filter 3, Filter n, INPUT 1, INPUT 2, OUTPUT 1, and OUTPUT 2.
The stack concept

Workflow based configuration: stack

- Choose an input
- Describe transformation and filter to apply
- Choose an output

Based on key value propagation through the stack

```
stack=n1:NFLOG,bs1:BASE,i1:IFINDEX,ip2s:IP2STR,pp:PRINTPKT,emu1:LOGEMU
stack=ct1:NFCT,mark1:MARK,ip2str1:IP2STR,pgsql2:PGSQL
```
The stack concept: plugin

Plugin

- Each plugin has:
  - Input keys
  - Output keys
  - Optional configuration keys

Plugin structure

```bash
# ulogd --info /usr/lib/ulogd/ulogd_filter_IP2STR.so
Name: IP2STR
Input keys:
  Key: oob.family (unsigned int 8)
  Key: oob.protocol (unsigned int 16)
  Key: ip.saddr (IP addr)
  Key: ip.daddr (IP addr)
  [...]
Output keys:
  Key: ip.saddr.str (string)
  Key: ip.daddr.str (string)
  [...]
```
Packet logging

- Compatible with old kernel
- IPv4 support:
  - ULOG
  - NFLOG
- IPv6 support:
  - NFLOG only
- Hardware information:
  - Network interfaces
  - Hardware header
Connection tracking event logging

- libnetfilter_conntrack based
- IPv4 and IPv6
- Listen to events
- Contains the two IP tuples
  - Orig IP header
  - Reply IP header
Netfilter accounting

Principles

- High performance accounting
- A library libnetfilter_acct and an utility nfacct
- nfacct is used to create counters
- counters are referenced as match in iptables rules

Examples

```
nfacct add ipv4.http
nfacct add ipv6.http
ip6tables -I INPUT -p tcp --sport 80 -m nfacct --nfacct-name ipv6.http
ip6tables -I OUTPUT -p tcp --dport 80 -m nfacct --nfacct-name ipv6.http
iptables -I INPUT -p tcp --sport 80 -m nfacct --nfacct-name ipv4.http
iptables -I OUTPUT -p tcp --dport 80 -m nfacct --nfacct-name ipv4.http
```
Various output plugin

**File-based**
- Syslog
- File
- PCAP
- NACCT

**Databases**
- PGSQL
- MySQL
- Sqlite

**Network**
- IPFIX
- GRAPHITE
## Treatment and filtering

### Treatment plugins
- Decoding plugins: BASE, IFINDEX
- Conversion plugins: IP2STR, IP2BIN, MAC2STR

### Filtering
- Decide if treatment has to be continued
- MARK plugin: stop propagation through stack if there is no match on mark

### Multiplexing
- Reusing INPUT data
- Multiple logging
Really use databases

- Let database work to the database
- Use database capability
  - Procedure for insertion
  - Extensible schemas
- Optimize schema
  - Avoid empty fields
  - Index on most frequent request
- Autoconfiguration
  - ulogd calls a procedure
  - params are taken from field name in a table
  - no need to recompile ulogd if we change the DB
Easy modification

- Procedure can do different things with data
- Provided procedure
  - Insertion of all available data in DB
  - For connection tracking
  - For packet logging
- Possible extension
  - Arbitrary accounting
  - Statistics
Extensible database schemas

TCP
+TCP_ID
+TCP Header

UDP
+UDP_ID
+UDP headers

ICMP
+ICMP_ID
+ICMP headers

Mac
+MAC_ID
+mac_saddr
+mac_daddr

+ID: Integer
+IP header
+Netfilter data
+MAC_ID
Extensible database schemas

- Easy to extend
  - Add table with your custom field
  - Link ID of the new table with ulog2 ID.
Use VIEW for usage ease

- VIEW can be built for common task

TCP quad view

```sql
CREATE OR REPLACE VIEW view_tcp_quad AS
SELECT ulog2._id, ulog2.ip_saddr_str, tcp.tcp_sport, ulog2.ip_daddr_str, tcp.tcp_dport
FROM ulog2 INNER JOIN tcp ON ulog2._id = tcp._tcp_id;
```

- and provide easy select

TCP quad select

```sql
ulog2=> SELECT ip_saddr_str, tcp_dport FROM view_tcp_quad;
  ip_saddr_str | tcp_dport
---------------+-----------
    148.60.18.179 |   1194
    148.60.18.179 |   1194
```

Eric Leblond ()

Ulogd2, Netfilter logging reloaded

NFWS2013, Copenhagen 26 / 40
Analysed dropped traffic
- Attack attempt
- Scans
- Worms or trojan traffic
- Detect invalid configuration

Analyse authorized traffic
- Keep a trace of access to critical data
- Forensic on successful attack
- Work with other security subsystem
Kernel event logging

**Activate kernel event logging**

```bash
echo 255 > /proc/sys/net/ nf_conntrack_log_invalid
```

**Display used log modules**

```bash
cat /proc/net/netfilter/nf_log
2 ipt_LOG (ipt_LOG, nfnetlink_log)
10 ip6t_LOG (ip6t_LOG, nfnetlink_log)
```

**Activate nfnetlink_log (group 0) on IPv4 and IPv6**

```bash
echo "nfnetlink_log" > /proc/sys/net/netfilter/nf_log/2
echo "nfnetlink_log" > /proc/sys/net/netfilter/nf_log/10
```
Djedi

A dashboard application

- DJango Extended Dashboard highly Interactive
- Provides an ulogd2 application
- [https://www.wzdftpd.net/redmine/projects/djedi](https://www.wzdftpd.net/redmine/projects/djedi)
Djedi: Demonstration

Video
Security interest

- Advantages of logging flow over logging packet
  - Start time
  - End time
  - Volume information

- Better view of severity of the event
  - Duration information
  - Data volume
  - NAT information
Recover internal IP from external data

Connection logging contains
  - Orig IP tuple
  - Reply IP tuple

Someone from outside asks you information about an attack:
  - Extern world only knows the Reply tuple
  - Connection logging lead you to the IP at the origin of an attack
Per-flow accounting

Each connection logging contains:
- bytes usage
- packet usage

Summing usage lead you to global statistic
- Using any IP criteria (per port or per IP bandwidth)
- Or using external information (per user bandwidth)

May need to activate conntrack extension

```
echo "1" >/proc/sys/net/netfilter/nf_conntrack_acct
echo "1" >/proc/sys/net/netfilter/nf_conntrack_timestamp
```
- Data visualisation tryout
- Represent both packet and connection on a graph
- Link packet to their corresponding connection
- Connections are displayed in a GANTT fashion
NF3D: SSH brute force

From 09:48:29 to 09:58:29

SRC: 207.36.180.132:46002
DST: 192.168.1.2:22
PROTO: TCP
IN: 2400, OUT: 1204 bits
DURATION: 122,000000 sec
Video


**Netfilter accounting**

### Prerequisite
- nfacct and libnetfilter_acct
- Ulogd 2.0.2 for Graphite output

### Create counters
- `nfacct add ipv4.http`
- `nfacct add ipv6.http`

### Select data to account
- `ip6tables -I INPUT -p tcp --sport 80 -m nfacct --nfacct-name ipv6.http`
- `ip6tables -I OUTPUT -p tcp --dport 80 -m nfacct --nfacct-name ipv6.http`
- `iptables -I INPUT -p tcp --sport 80 -m nfacct --nfacct-name ipv4.http`
- `iptables -I OUTPUT -p tcp --dport 80 -m nfacct --nfacct-name ipv4.http`
Ulogd and Graphite

Activate and setup the stack

```
stack=acct1:NFACCT,graphite1:GRAPHITE

[acct1]
pollinterval = 2

[graphite1]
host="127.0.0.1"
port="2003"
```
A complete logging system

A full-featured logging daemon for Netfilter

- Packet logging
- Connection logging
- Accounting

Easy to extend

- Via plugin
- Via database modification
Questions

Contacts

- Directly: eric@regit.org
- Mailling List: netfilter-devel@vger.kernel.org

References

- Djedi: https://www.wzdftpd.net/redmine/projects/djedi
- NF3D: https://home.regit.org/software/nf3d
- My blog: https://home.regit.org/