Coccigrep: a semantic grep for C language

Éric Leblond

Stamus Networks

April 27, 2014

Eric Leblond a.k.a Regit

- French
- Network security expert
- Free Software enthousiast
- NuFW project creator (Now ufwi), EdenWall co-founder
- Netfilter developer:
  - Maintainer of ulogd2: Netfilter logging daemon
  - Misc contributions:
    - NFQUEUE library and associates
    - Port of some features iptables to nftables
- Currently:
  - co-founder of Stamus Networks, a company providing Suricata based network probe appliances.
  - Suricata IDS/IPS funded developer

Coccinelle, a program matching and transformation engine

Matching and transformation

- A command line tool for matching and transformation
- Understand C semantic
- Used for
  - Find and fix bug in code
  - Update code and API
  - Search code

Some facts

- Heavily used in Linux kernel
  - Real impact on Linux: `http://coccinelle.lip6.fr/impact_linux.php`
  - Semantic patches are included in Linux source tree
- Developped in ocaml

Semantic patches

Definition

A transformation language based on the patch syntax, extending patches to semantic patches.

Interest

- A single small semantic patch can modify hundreds of files
- semantic patch is generic and is abstracting away the specific details and variations at each code site
- don't care of spacing or variable name

Support of isomorphisms

- Coding style can vary and multiple expression are equivalent
- For example:

$$if(!y) \equiv if(y == NULL) \equiv if(NULL == y)$$

Simple real life patching

Use macro instead of direct access

- *action* field in Packet structure must not be accessed directly
- For test, we need to use a macro named TEST_PACKET_ACTION

A semantic patch

```
@@
Packet *p;
expression E;
@@
- p->action & E
+ TEST_PACKET_ACTION(p, E)
```

Patching the code

```
spatch -sp_file action.cocci -in_place src/*c
git diff --stat
 12 files changed, 77 insertions(+), 76 deletions(-)
```

Coccinelle for testing

Detect invalid usage

- All project have coding rules and internal API
- Coccinelle can be used to enforce via
  - Matching
  - Alerting

Use Python for output

```
@script:python@
p1 << zeroed.p1;
@@

print "Err at %s:%s" % (p1[0].file, p1[0].line)
import sys
sys.exit(1)
```

Coccinelle for testing

Detecting direct usage of action field

```
@action@
typedef Packet;
Packet *p;
position p1;
@@
p->action@p1

@ script:python @
p1 << action.p1;
@@

print "Invalid usage of p->action at %s:%s" % (p1[0].file, p1[0].line)
import sys
sys.exit(1)
```

Coccigrep

A semantic grep for C

- Search in C code
- For specific usage of a structure
- With a tool understanding the code

Examples

```
$ coccigrep  -t  Packet  *h
./tm-threads.h:135  (Packet  *p):          tv->tmqh_out(tv,  p);
./tm-threads.h:140  (Packet  *p):          TmqhOutputPacketpool(tv,  p);
./tm-threads.h:164  (Packet  *extra_p):  if  (extra_p  ==  NULL)
./tm-threads.h:168  (Packet  *extra_p):  r  =  TmThreadsRun(tv,  extra_p);
```

Coccigrep

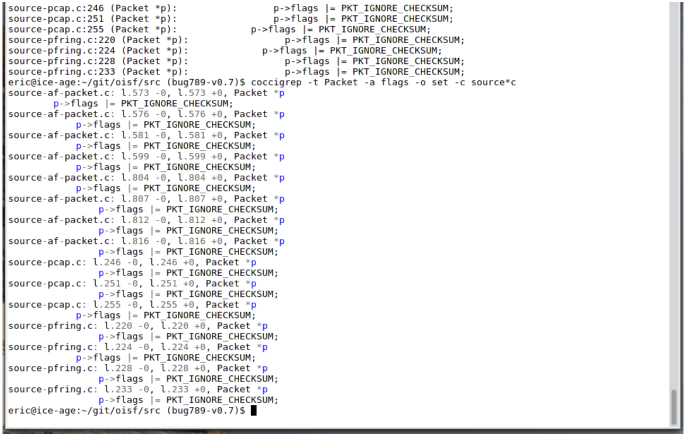
Operations

- set: Search where a given attribute of structure 'type' is set
- used: Search all usage of 'type' structure
- static: Search where a given attribute of structure 'type' is set
- func: Search for function having a struct 'type' as argument
- test: Search where a given attribute of 'type' structure is used in test.
- deref: Search for usage of a given attribute for a 'type' structure

Examples

```
$ coccigrep  -t Packet -a datalink -o set  source*c
source-af-packet.c:562 (Packet *p): p->datalink = ptv->datalink;
source-napatech.c:273 (Packet *pt):  pt->datalink = LINKTYPE_ETHERNET;
source-pcap-file.c:141 (Packet *p): p->datalink = pcap_g.datalink;
```

Demonstration



```
source-pcap.c:246 (Packet *p):                 p->flags |= PKT_IGNORE_CHECKSUM;
source-pcap.c:251 (Packet *p):                 p->flags |= PKT_IGNORE_CHECKSUM;
source-pcap.c:255 (Packet *p):             p->flags |= PKT_IGNORE_CHECKSUM;
source-pfring.c:220 (Packet *p):                 p->flags |= PKT_IGNORE_CHECKSUM;
source-pfring.c:224 (Packet *p):             p->flags |= PKT_IGNORE_CHECKSUM;
source-pfring.c:228 (Packet *p):                 p->flags |= PKT_IGNORE_CHECKSUM;
source-pfring.c:233 (Packet *p):                 p->flags |= PKT_IGNORE_CHECKSUM;
eric@ice-age:~/git/oisf/src (bug789-v0.7)$ coccigrep -t Packet -a flags -o set -c source*c
source-af-packet.c: l.573 -0, l.573 +0, Packet *p
        p->flags |= PKT_IGNORE_CHECKSUM;
source-af-packet.c: l.576 -0, l.576 +0, Packet *p
            p->flags |= PKT_IGNORE_CHECKSUM;
source-af-packet.c: l.581 -0, l.581 +0, Packet *p
            p->flags |= PKT_IGNORE_CHECKSUM;
source-af-packet.c: l.599 -0, l.599 +0, Packet *p
            p->flags |= PKT_IGNORE_CHECKSUM;
source-af-packet.c: l.804 -0, l.804 +0, Packet *p
            p->flags |= PKT_IGNORE_CHECKSUM;
source-af-packet.c: l.807 -0, l.807 +0, Packet *p
                p->flags |= PKT_IGNORE_CHECKSUM;
source-af-packet.c: l.812 -0, l.812 +0, Packet *p
                p->flags |= PKT_IGNORE_CHECKSUM;
source-af-packet.c: l.816 -0, l.816 +0, Packet *p
                p->flags |= PKT_IGNORE_CHECKSUM;
source-pcap.c: l.246 -0, l.246 +0, Packet *p
                p->flags |= PKT_IGNORE_CHECKSUM;
source-pcap.c: l.251 -0, l.251 +0, Packet *p
                p->flags |= PKT_IGNORE_CHECKSUM;
source-pcap.c: l.255 -0, l.255 +0, Packet *p
            p->flags |= PKT_IGNORE_CHECKSUM;
source-pfring.c: l.220 -0, l.220 +0, Packet *p
                p->flags |= PKT_IGNORE_CHECKSUM;
source-pfring.c: l.224 -0, l.224 +0, Packet *p
            p->flags |= PKT_IGNORE_CHECKSUM;
source-pfring.c: l.228 -0, l.228 +0, Packet *p
                p->flags |= PKT_IGNORE_CHECKSUM;
source-pfring.c: l.233 -0, l.233 +0, Packet *p
                p->flags |= PKT_IGNORE_CHECKSUM;
eric@ice-age:~/git/oisf/src (bug789-v0.7)$ 
```

Coccigrep and Vim

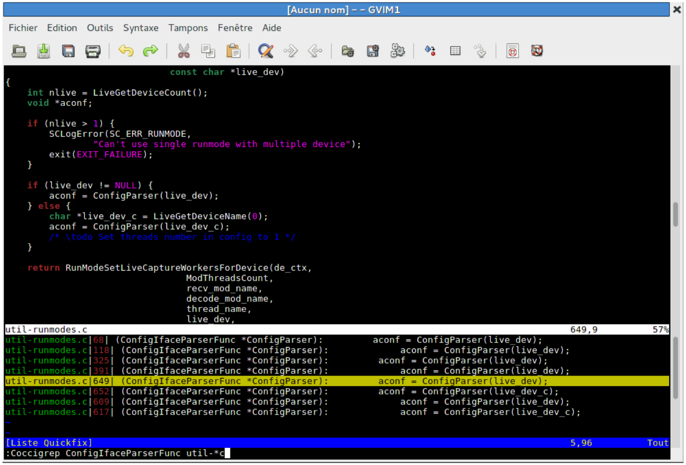
A vim plugin

- Wrapper for coccigrep command
- Get a grep like output and jump on line
- An Emacs plugin exists too

Examples

```
:Coccigrep
:Coccigrep Packet datalink source-*.c
:Coccigrep Packet datalink set src
```

Demonstration



```
                              const char *live_dev)
{
    int nlive = LiveGetDeviceCount();
    void *aconf;

    if (nlive > 1) {
        SCLogError(SC_ERR_RUNMODE,
                "Can't use single runmode with multiple device");
        exit(EXIT_FAILURE);
    }

    if (live_dev != NULL) {
        aconf = ConfigParser(live_dev);
    } else {
        char *live_dev_c = LiveGetDeviceName(0);
        aconf = ConfigParser(live_dev_c);
        /* \todo Set threads number in config to 1 */
    }

    return RunModeSetLiveCaptureWorkersForDevice(de_ctx,
                                 ModThreadsCount,
                                 recv_mod_name,
                                 decode_mod_name,
                                 thread_name,
                                 live_dev,
util-runmodes.c                                                                649,9          57%
util-runmodes.c|68| (ConfigIfaceParserFunc *ConfigParser):        aconf = ConfigParser(live_dev);
util-runmodes.c|118| (ConfigIfaceParserFunc *ConfigParser):           aconf = ConfigParser(live_dev);
util-runmodes.c|325| (ConfigIfaceParserFunc *ConfigParser):       aconf = ConfigParser(live_dev);
util-runmodes.c|391| (ConfigIfaceParserFunc *ConfigParser):           aconf = ConfigParser(live_dev);
util-runmodes.c|649| (ConfigIfaceParserFunc *ConfigParser):       aconf = ConfigParser(live_dev);
util-runmodes.c|652| (ConfigIfaceParserFunc *ConfigParser):       aconf = ConfigParser(live_dev_c);
util-runmodes.c|609| (ConfigIfaceParserFunc *ConfigParser):           aconf = ConfigParser(live_dev);
util-runmodes.c|617| (ConfigIfaceParserFunc *ConfigParser):           aconf = ConfigParser(live_dev_c);
~
~
[Liste Quickfix]                                                               5,96           Tout
:Coccigrep ConfigIfaceParserFunc util-*c
```

Conclusion

Coccinelle is a powerful tool

- Patch can be written for really complex changes
- It can save you time
- And is more accurate than manual changes
- But difficult to master
    - Terminoly is important
    - A lot of subtilities

Coccigrep for search tasks

- Coccigrep is here to help you do simple search tasks
- Easy to understand syntax

Questions

Thanks

- A **huge** one to Julia Lawall
- My father

More information

- coccinelle: `http://coccinelle.lip6.fr/`
- gallery of semantic patches: `http://coccinellery.org/`
- coccigrep: `https://home.regit.org/software/coccigrep/`
- Coccinelle for the newbie: `https://home.regit.org/technical-articles/coccinelle-for-the-newbie/`