NetsniffNg

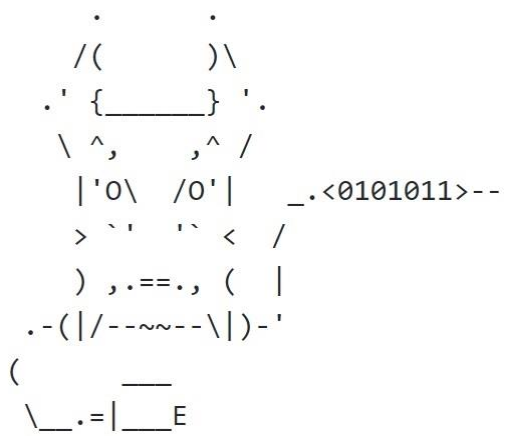
Information Security Inc.

Contents

- About NetsniffNg
- Testing Environment
- Installing NetsniffNg
- Using NetsniffNg
- References

About NetsniffNg

- Netsniff-ng is a free, performant Linux network analyzer and networking toolkit
- The Swiss army knife for network packets



Testing Environment

- Kali Linux 2017

```
root@WarBerry:~# cat /etc/*rel*
DISTRIB_ID=Kali
DISTRIB_RELEASE=kali-rolling
DISTRIB_CODENAME=kali-rolling
DISTRIB_DESCRIPTION="Kali GNU/Linux Rolling"
PRETTY_NAME="Kali GNU/Linux Rolling"
NAME="Kali GNU/Linux"
ID=kali
VERSION="2017.3"
VERSION_ID="2017.3"
ID_LIKE=debian
ANSI_COLOR="1;31"
HOME_URL="http://www.kali.org/"
SUPPORT_URL="http://forums.kali.org/"
BUG_REPORT_URL="http://bugs.kali.org/"
```

Installing NetsniffNg

- Installing dependencies

```
root@WarBerry:~# apt-get install ccache flex bison libnl-3-dev \
>   libnl-genl-3-dev libnl-route-3-dev libgeoip-dev \
>   libnetfilter-conntrack-dev libncurses5-dev liburcu-dev \
>   libnacl-dev libpcap-dev zlib1g-dev libcli-dev libnet1-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
libnl-3-dev is already the newest version (3.2.27-2).
libnl-genl-3-dev is already the newest version (3.2.27-2).
libpcap-dev is already the newest version (1.8.1-5).
zlib1g-dev is already the newest version (1:1.2.8.dfsg-5).
```

Installing NetsniffNg

- Cloning GitHub repository

```
root@WarBerry:~# git clone https://github.com/netsniff-ng/netsniff-ng.git
Cloning into 'netsniff-ng'...
remote: Counting objects: 5485, done.
remote: Total 5485 (delta 0), reused 0 (delta 0), pack-reused 5485
Receiving objects: 100% (5485/5485), 2.68 MiB | 2.00 MiB/s, done.
Resolving deltas: 100% (3628/3628), done.
```

Installing NetsniffNg

- Installing from source

```
cd netsniff-ng/
./configure
make
make install
history
```

Using NetsniffNg

- NetsniffNg options

```
root@WarBerry:~# netsniff-ng -h
netsniff-ng 0.6.3+, the packet sniffing beast
http://www.netsniff-ng.org

Usage: netsniff-ng [options] [filter-expression]
Options:
  -i|-d|--dev|--in <dev|pcap|->  Input source as netdev, pcap or pcap stdin
  -o|--out <dev|pcap|dir|cfg|->  Output sink as netdev, pcap, directory, trafgen, or stdout
  -C|--fanout-group <id>         Join packet fanout group
  -K|--fanout-type <type>        Apply fanout discipline: hash|lb|cpu|rnd|roll|qm
  -L|--fanout-opts <opts>        Additional fanout options: defrag|roll
  -f|--filter <bpf-file|-|expr>  Use BPF filter from bpfc file/stdin or tcpdump-like expression
  -t|--type <type>               Filter for: host|broadcast|multicast|others|outgoing
  -F|--interval <size|time>      Dump interval if -o is a dir: <num>KiB/MiB/GiB/s/sec/min/hrs
  -R|--rfraw                     Capture or inject raw 802.11 frames
  -n|--num <0|uint>              Number of packets until exit (def: 0)
  -P|--prefix <name>             Prefix for pcaps stored in directory
  -T|--magic <pcap-magic>        Pcap magic number/pcap format to store, see -D
  -w|--cooked                    Use Linux "cooked" header instead of link header
  -D|--dump-pcap-types           Dump pcap types and magic numbers and quit
  -B|--dump-bpf                  Dump generated BPF assembly
  -r|--rand                      Randomize packet forwarding order (dev->dev)
  -M|--no-promisc                No promiscuous mode for netdev
  -A|--no-sock-mem               Don't tune core socket memory
  -N|--no-hwtimestamp            Disable hardware time stamping
  -m|--mmap                      Mmap(2) pcap file I/O, e.g. for replaying pcaps
  -G|--sg                        Scatter/gather pcap file I/O
```

Using NetsniffNg

- Capturing ten packets

```
root@WarBerry:~# netsniff-ng -i eth0 -n 10
Running! Hang up with ^C!

> eth0 134 1511259216s.484360101ns #1
 [ Eth MAC (b8:27:eb:34:e6:98 => 80:fa:5b:10:12:a0), Proto (0x0800, IPv4) ]
 [ Vendor (Raspberry Pi Foundation => CLEVO CO.) ]
 [ IPv4 Addr (192.168.86.15 => 192.168.86.1), Proto (6), TTL (64), TOS (16), Ver (4), IHL (5), Tlen (120), I
   D (29020), Res (0), NoFrag (1), MoreFrag (0), FragOff (0), CSum (0x9bb2) is ok ]
 [ TCP Port (22 (ssh) => 2578), SN (0xf8e36298), AN (0x4af762a9), DataOff (5), Res (0), Flags (PSH A
   CK), Window (433), CSum (0x2dcc), UrgPtr (0) ]
 [ Chr .4D..s.3.w....D.gz*.S.(..1..b...R..C...E..eb.G.....=w....F.f...3...{.b>.v...X..B ]
 [ Hex  ba 34 44 86 bd 73 1b 33 ac 77 85 f2 d3 c4 44 90 67 7a 2a d9 53 ca 28 e2 f7 31 b9 fc 62 14 1d f7 52 f
   e ff 43 e2 9f 9b 45 dd e5 65 62 f8 47 dd 97 b0 b1 e8 3d 77 e2 d0 89 89 46 1a 66 d5 a6 b8 33 f6 0b 19 7b c
   9 62 3e 92 76 ce af f3 58 94 a1 42 ]

< eth0 60 1511259216s.524974754ns #2
 [ Eth MAC (80:fa:5b:10:12:a0 => b8:27:eb:34:e6:98), Proto (0x0800, IPv4) ]
 [ Vendor (CLEVO CO. => Raspberry Pi Foundation) ]
 [ Eth trailer 000000 ]
 [ IPv4 Addr (192.168.86.1 => 192.168.86.15), Proto (6), TTL (128), TOS (0), Ver (4), IHL (5), Tlen (40), ID
   (26907), Res (0), NoFrag (1), MoreFrag (0), FragOff (0), CSum (0x6453) is ok ]
 [ TCP Port (2578 => 22 (ssh)), SN (0x4af762a9), AN (0xf8e362e8), DataOff (5), Res (0), Flags (ACK),
   Window (2049), CSum (0x66dd), UrgPtr (0) ]
```

Using NetsniffNg

- Tcpdump_like filtering

```
root@WarBerry:~# netsniff-ng -i eth0 -f "host 192.168.86.15 and not port 22"
Running! Hang up with ^C!



< eth0 98 1511260251s.514417308ns #1
 [ Eth MAC (00:1b:54:01:4c:44 => b8:27:eb:34:e6:98), Proto (0x0800, IPv4) ]
 [ Vendor (Cisco Systems, Inc => Raspberry Pi Foundation) ]
 [ IPv4 Addr (192.168.20.2 => 192.168.86.15), Proto (1), TTL (61), TOS (0), Ver (4), IHL (5), Tlen (84), ID
   (47004), Res (0), NoFrag (1), MoreFrag (0), FragOff (0), CSum (0x9aaa) is ok ]
 [ ICMP Type (8), Code (0), CSum (0xe287) is ok ]
 [ Chr [..Z.....r..................... !"#$%&'()*+,-./01234567 ]
 [ Hex  5b 00 14 5a 00 00 00 00 f5 72 06 00 00 00 00 00 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 2
   1 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37 ]

> eth0 98 1511260251s.514649964ns #2
 [ Eth MAC (b8:27:eb:34:e6:98 => 00:1b:54:01:4c:44), Proto (0x0800, IPv4) ]
 [ Vendor (Raspberry Pi Foundation => Cisco Systems, Inc) ]
 [ IPv4 Addr (192.168.86.15 => 192.168.20.2), Proto (1), TTL (64), TOS (0), Ver (4), IHL (5), Tlen (84), ID
   (50001), Res (0), NoFrag (0), MoreFrag (0), FragOff (0), CSum (0xcbf5) is ok ]
 [ ICMP Type (0), Code (0), CSum (0xea87) is ok ]
 [ Chr [..Z.....r..................... !"#$%&'()*+,-./01234567 ]
 [ Hex  5b 00 14 5a 00 00 00 00 f5 72 06 00 00 00 00 00 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 2
   1 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 36 37 ]
```

Using NetsniffNg

- Tcpdump_like filtering

```
< eth0 66 1511260486s.577116274ns #3
 [ Eth MAC (00:1b:54:01:4c:44 => b8:27:eb:34:e6:98), Proto (0x0800, IPv4) ]
 [ Vendor (Cisco Systems, Inc => Raspberry Pi Foundation) ]
 [ IPv4 Addr (192.168.20.2 => 192.168.86.15), Proto (6), TTL (61), TOS (0), Ver (4), IHL (5), Tlen (52), ID
   (37274), Res (0), NoFrag (1), MoreFrag (0), FragOff (0), CSum (0xc0c7) is ok ]
 [ TCP Port (40400 => 8000 (http-alt)), SN (0x2a401f37), AN (0x3de389ed), DataOff (8), Res (0), Flag
   s (ACK), Window (229), CSum (0x5ea0), UrgPtr (0) ]
 [ Chr .....U...9V. ]
 [ Hex  01 01 08 0a 08 55 fe 04 00 39 56 ea ]

< eth0 148 1511260486s.577913304ns #4
 [ Eth MAC (00:1b:54:01:4c:44 => b8:27:eb:34:e6:98), Proto (0x0800, IPv4) ]
 [ Vendor (Cisco Systems, Inc => Raspberry Pi Foundation) ]
 [ IPv4 Addr (192.168.20.2 => 192.168.86.15), Proto (6), TTL (61), TOS (0), Ver (4), IHL (5), Tlen (134), ID
   (37275), Res (0), NoFrag (1), MoreFrag (0), FragOff (0), CSum (0xc074) is ok ]
 [ TCP Port (40400 => 8000 (http-alt)), SN (0x2a401f37), AN (0x3de389ed), DataOff (8), Res (0), Flag
   s (PSH ACK), Window (229), CSum (0x7570), UrgPtr (0) ]
 [ Chr .....U...9V.GET / HTTP/1.1..Host: 192.168.86.15:8000..User-Agent: curl/7.56.1..Accept: */*.... ]
 [ Hex  01 01 08 0a 08 55 fe 05 00 39 56 ea 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 2
   0 31 39 32 2e 31 36 38 2e 38 36 2e 31 35 3a 38 30 30 30 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 63 75 7
   2 6c 2f 37 2e 35 36 2e 31 0d 0a 41 63 63 65 70 74 3a 20 2a 2f 2a 0d 0a 0d 0a ]
```

References

- Wikipedia
https://en.wikipedia.org/wiki/Netsniff-ng

- Kali Linux 2017
https://www.kali.org/downloads/