Table of Contents

current situation

CERN-CC-netlog is a custom kernel module for SLC3.

It provides the following data (based on DESIGN document from CERN-CC-netlog package):

- connect {start|stop} <proto> <local> <direction> <remote> <sid> <pid> <uid> <command>
- listen {start|stop} <proto> <local> <sid> <pid> <uid> <command>
- where
  - <proto> is "TCP" or "UDP"
  - <local> is the numerical address and port number of the local end
  - <direction> is "->" (outgoing) or "<-" (incoming)
  - <remote> is the numerical address and port number of the remote end
  - <sid> is the session id
  - <pid> is the process id
  - <uid> is the user id
  - <command> is the command name
- Note that processes can pass sockets to each other so the process that starts a connection is not necessarily the one that closes it.
- Examples:

```
Apr 24 11:11:39 localhost kernel: netlog: info: listen start TCP 0.0.0.0:22 10416 10416 0
Apr 24 11:12:50 localhost kernel: netlog: info: connect start TCP 128.141.47.174:22
```

- The current module works by overloading system calls and is often (mis-)identified as a kernel rootkit

Current sources:
`/afs/cern.ch/project/linux/redhat/cern/addon/cc/SLC3/SRPMS/CERN-CC-netlog-1.0-1`
(this actually contains the prebuild binary modules, not a real source RPM..)

solutions investigated for SLC4

porting module to 2.6.x

- a lot of work (many networking/locking/VFS changes since 2.4.x)
- most likely porting will have to be repeated for the next system version (SLC5)
- export for syscall table got removed from 2.6.x so will have to introduce some hacky way (ala AFS) to patch syscall table with wrapper functions
- high maintanance costs (module has to be recompiled/verified against each new kernel version)

iptables

- the major problem is that logging is done per packet not per connection => **MUCH** greater overhead (CPU+mem/disk)
    - ♦ we may (?) be able to use existing in-kernel connection tracking engine
        - ◊ Iptables Connection Tracking - TCP by James Stephens: http://www.sns.ias.edu/~jns/wp/2006/01/12/iptables-connection-tracking-tcp/1/
        - ◊ conntrack@netfiler: http://www.netfilter.org/projects/conntrack/index.html
            - · requires 2.6.14 so RHEL4 is no go (unless RH patched their kernel)
            - · still under development
    - ♦ doing own connection tracking too error-prone / inefficient
- we don't have direct information about lifetime of the connection
    - ♦ for TCP we may (?) use connection tracking and calculate (approximate) connection lifetime
    - ♦ for UDP this is not a problem since we consider each packet as independent connection anyway
- linux-2.6.9/net/ipv4/netfilter/ipt_LOG.c module provides too little information to be used directly but is a good start base
    - ♦ we can get 'struct iphdr' from 'struct sk_buff'
        - ◊ iphdr->saddr: source address
        - ◊ iphdr->daddr: destination address
        - ◊ iphdr->protocol: protocol (IPPROTO_TCP for TCP and IPPROTO_UDP for UDP)
            - · can get 'struct tcphdr' (for TCP) or 'struct udphdr' (for UDP) from 'iphdr'
            - · {tcp,udp}hdr->source: source port
            - · {tcp,udp}hdr->dest: destination port
    - ♦ we can get direction of the connection by checking pointers to input and output network devices
        - ◊ · "->": out && in == NULL
            - · "<-": in && out == NULL
    - ♦ enable ipt_LOG module by doing as root ('-I' is needed to insert rule at the head of the chain):
        - ◊ iptables -I INPUT -j LOG
        - ◊ iptables -I OUTPUT -j LOG
    - ♦ ipt_LOG output (from /var/log/messages):
        - ◊ Mar 29 17:19:28 it-adc-test08 kernel: IN=eth0 OUT= MAC=00:0c:f1:a0:44:e7:0a:00:30:89:20:01:08:00 SRC=217.17.41.88 DST=137.138.32.52 LEN=52 TOS=0x00 PREC=0x00 TTL=48 ID=40771 DF PROTO=TCP SPT=8074 DPT=35437 WINDOW=2939 RES=0x00 ACK URGP=0
        - ◊ Mar 29 17:20:35 it-adc-test08 kernel: IN= OUT=eth0 SRC=137.138.32.52 DST=137.138.251.10 LEN=93 TOS=0x00 PREC=0x00 TTL=64 ID=28187 PROTO=UDP SPT=7001 DPT=7000 LEN=73
- linux-2.6.9/net/ipv4/netfilter/ipt_owner.c is a good example on how to get the rest of needed info
    - ♦ we can get from 'struct sk_buff' to 'struct file' (skb->sk->sk_socket->file), in 'struct file'
        - ◊ file->f_uid: owner UID

- having file we can also get PID, SID and command by scanning list of tasks and checking if our file is in the list of files belonging to the task, however this is unsafe on SMP as stated by comment in ipt_owner.c [1] and indeed support for IPT_OWNER_{PID,SID,COMM} has been removed in recent kernels
- we really need proper SMP support so this means no PID/SID/command

[1]:

```
#ifdef CONFIG_SMP
   /* files->file_lock can not be used in a BH */
   if (((struct ipt_owner_info *)matchinfo)->match
       & (IPT_OWNER_PID|IPT_OWNER_SID|IPT_OWNER_COMM)) {
      printk("ipt_owner: pid, sid and command matching is broken "
             "on SMP.\n");
      return 0;
   }
#endif
```

audit subsytem

- available since RHEL4U2,
- we can can get most of the information which we get from our SLC3 kernel module by auditing system calls
- basic functionality implemented, it currently works in this way (root account needed):
  - start auditing syscalls:
    - # /etc/init.d/audit start
  - set proper audit rules:
    - # ./lkm.sh
  - parse audit log file and print connections
    - # ./parse-audit-log.pl -l /var/log/audit.log
- problems (please see CERN RH bug #82057 - **now closed?** for more details):
  - Finding address:port pair for cases when address and/or port is automatically assigned to the socket during i.e. connect() call.
  - Inability to log usage of non-blocking sockets (when accept system call fails with -EINPROGRESS error). For handling this situation I added logging of getsockopt() system call and checking values passed by/to user (idea invented after reading -EINPROGRESS section of 'man 2 connect') but I can only get value of 'optval' pointer which is useless because what is actually needed is the value not the pointer to it.
- above problems can be fixed by fixing kernel and auditd to provide needed information (in cooperation with RH or/and upstream maintainer(s) so we don't have to carry fixes forever) or by dynamically patching kernel using kernel module (created with SystemTap?)

SystemTap

(alone or as a convenient way to add custom data structures)

This topic: LinuxSupport > IDSNetConnectionLogger
Topic revision: r3 - 2007-01-12 - JanIven