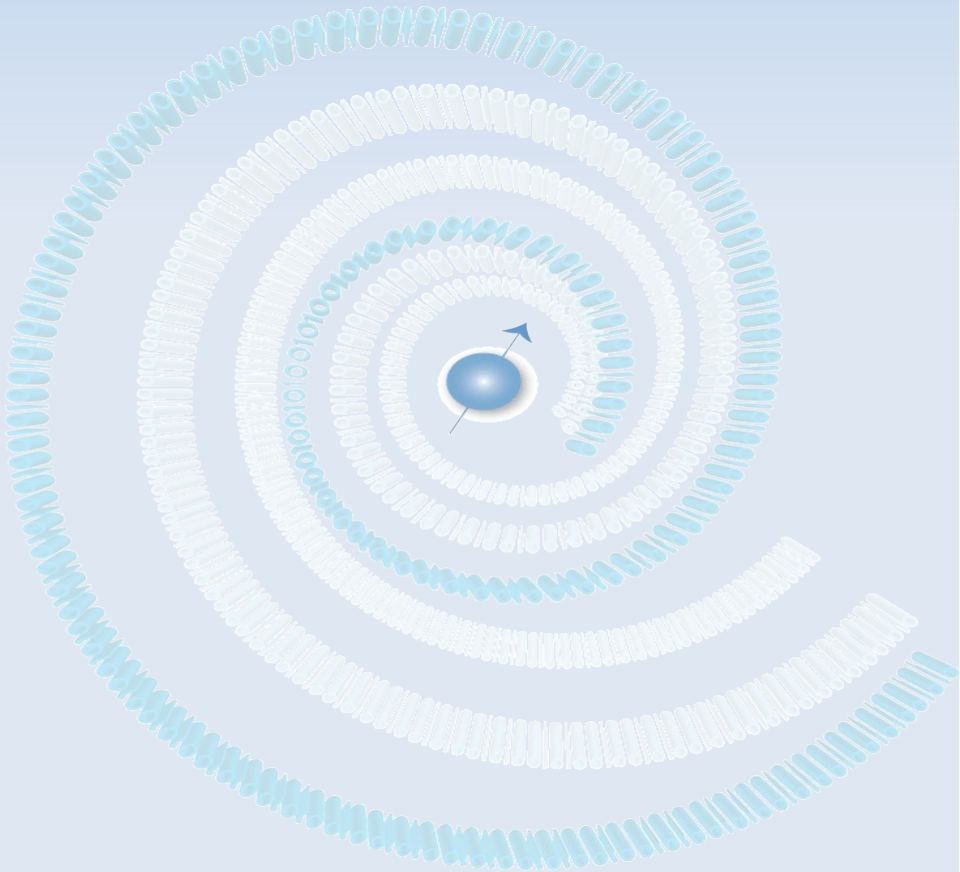
Xen and Intrusion Detection & Recovery with Linux

Bill Broadley
<bill@cse.ucdavis.edu>
Computational Science
and Engineering
UC Davis



Aug 20th, 2007
LUGOD

6:30-9:00pm

Threats

- Users
- Programmers
- Vendors
- Poor programming languages
- Applications
- Operating systems.

Who can you trust?

- If you don't have physical security?
- If you do have physical security?
- Logs? Auditing?
- Users?
- OS binaries? Application binaries?
- The file system?
- The network? Kernel?

Privilege Escalation

- What boundaries can not be broken?
  - Non-user -> user?
  - user -> root?
  - Application -> root?
  - root -> kernel?
  - DomU -> Dom0 (para)?
  - DomU -> Dom0 (HVM)?

Ideal world?

- Perfect physical security
- No network
- No users
- No untrusted binaries
- Trusted kernel
- Logs and auditing you can trust.

Realistic world

- Simple environment (kernel+sshd)
- Private network
- Maintain the high ground.
- Trustworthy binaries
- Trustworthy kernel
- No 3rd party applications
- All users, applications, and untrustworthy kernels on a virtual machine

Attack model

- Any open port (buffer overflows)
- Any user account (web or shell)
- Escalate any privileges
- Hide tracks (including tripwire)
- Install back doors
- Collect any valuable info (identity, CC,...)
- Collect recon for additional attacks

Old school obvious signs of compromise

- Unusually full disks and/or busy networks.
- Complaints from other networks you are attacking/scanning
- Complaints from other networks that you are spamming
- Defaced web pages

Often ego motivated.
Getting much less common

New signs of compromise

- Ever more invasive hackers make a mistake and break something
- Audit, maintenance, or upgrades uncover compromise, often months after the compromise
- Giant list of user/pass, ss numbers, or credit card numbers appear.
- Embarrassing press releases
- The scariest of all..... nothing.

Attackers are more organized, more professional, and are often profit instead of ego motivated.

Old school intrusion detection

- ps, lsof, /proc
- local syslog
- tripwire – (usually local/insecure), hacker scripts go over updating tripwire databases and/or crafting replacements that give the all clear
- The truly paranoid do tripwire “right” which requires downtime and is labor intensive. Rare.

Old school counter ID

- rootkits (hack ps, lsof, ls, du, etc.)
- strange filenames
- obscure directories
- zero wtmp/utmp/syslog
- bastion hosts
- known bad checksums (NEVER current)
- chkrootkit, rkhunter
- Anti-ID forensics to discover how to update database.

Current anti-ID

- Encryption (no more sniffing hackers)
- Anonymous networks (tor)
- P2P bot nets for DoS, Phishing, and spamming.
- Kernel rootkits (solaris, linux, *bsd, and windows) with or without kernel modules.
- Application rootkits (attacked in memory)
- Port knocker based backdoors.

Kernel rootkits

- Hard to detect from inside. Removes common methods of discovery like using trusted bins.
- Does NOT require modules to be enabled.
- OS bootstrap process is very complex, any binary, module, or script can compromise kernel.
- /sbin/init often attacked, post boot checksums will reflect the original checksum
- Very effective at covering back doors, sniffers, promiscuous mode, trojans, and port knockers.
- Can even hide parts of files.
- Often readdir (get next file) will not work, but open and exec work with exact name

Kernel rootkits part II

- Will happily lie to ps, du, top, lsof, and tripwire
- Open or stat will return uncompromised version
- Exec will result in compromised version being run
- Extremely common, current traditional rootkits are becoming hard to find.
- Watching the low hanging fruit (ps, du, ls, nmap, lsof and friends) no longer effective. Can be triggered from 1000's of files.

Rootkits Part III

- First gen used kernel module and insmod, played with the syscall table.
- 2nd gen access /dev/kmem, doesn't require modules, visible to /proc
- 3rd gen tweaks lower level kernel structures like the VFS layer, harder to detect, installs via /dev/mem
- 4th generation virtualizes the kernel, very hard to detect and survives “reboots and reinstalls”

Current ID

- Dom0 monitoring DomU
- Logical volumes (snapshots)
- Off host logging
- Maintaining the high ground (private networks, dom0, secure simple hosts, and a trusted boot sequence)
- Known good checksums (dramatically better than known bad.)
- Vendor supplied checksums/signatures.

Next gen anti-ID

- Get ring 0, take the high ground, virtualize the system
- Nasty tricks:
  - Pretend to turn off, but instead sleep
  - maintain backdoor system even on a power cycle with full reinstall from trusted media
  - Fake BIOS and POST
  - Mostly undetectable from the domU (tricky timing might help in some cases).

Next gen ID

- Trusted boot sequence (BIOS -> boot loadloader -> kernel -> modules)
- Hardware support (IBM's TPM chip)
- Vendor signed binaries and modules
- Improvement in practices like actually pulling the plug before installs.
- Virtualize first.
- Careful LED watching (sleep vs suspend)
- Xebek – Honeypot tool

Xebek

- Whitehat rootkit for Xen.
- Designed for high interaction honeypots
- Excellent at crossing the semantic gap
- Intercepts system calls and keystrokes
- Does not use network stack for logging
- Dom0 does not have to be network visible
- Requires kernel source patch (whitehat)

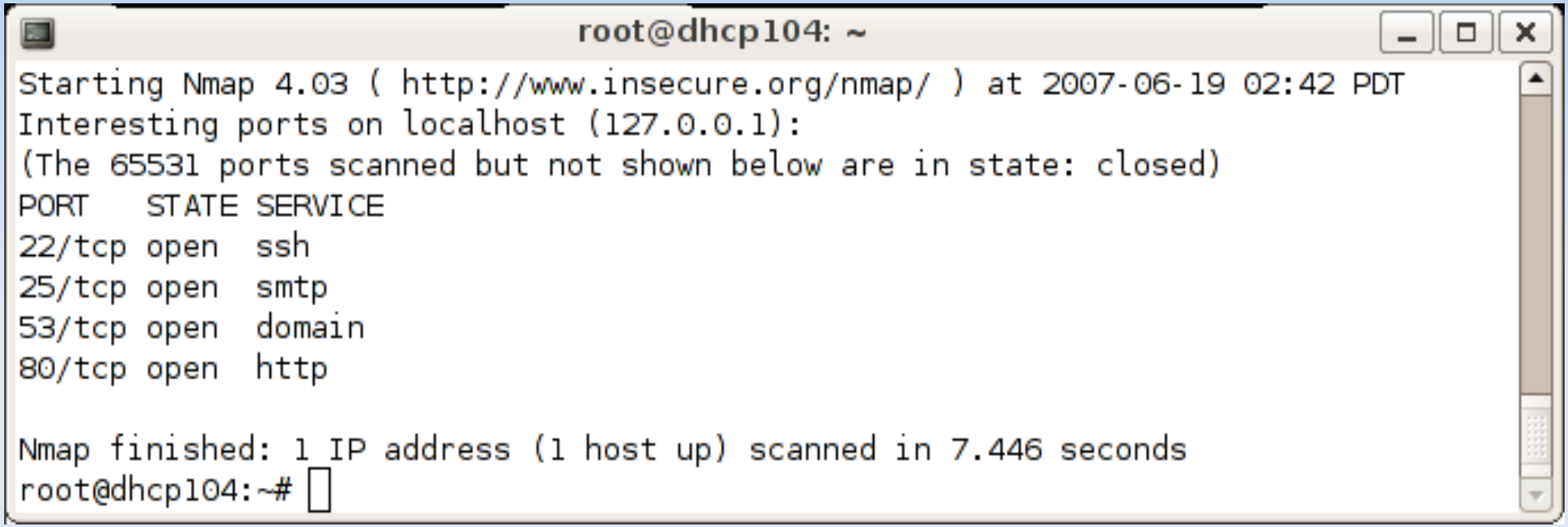
End of overview

- Discussion?
- Everyone understand?

Details, tools, and examples

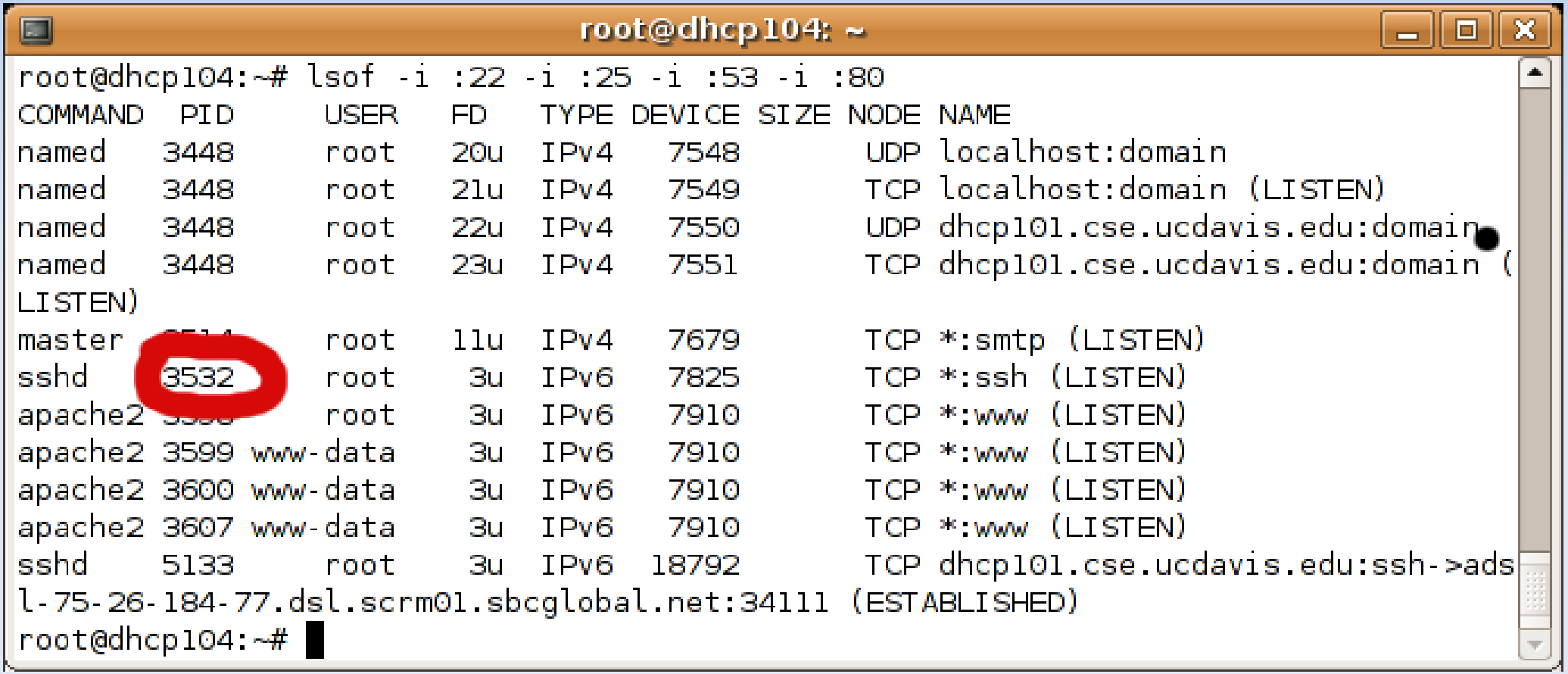
- Step by step analysis of compromised system
- Example tools
- Example backdoors
- Example trojans
- CDR – Checksums Done Right
- Dom0 vs DomU

The basics: nmap



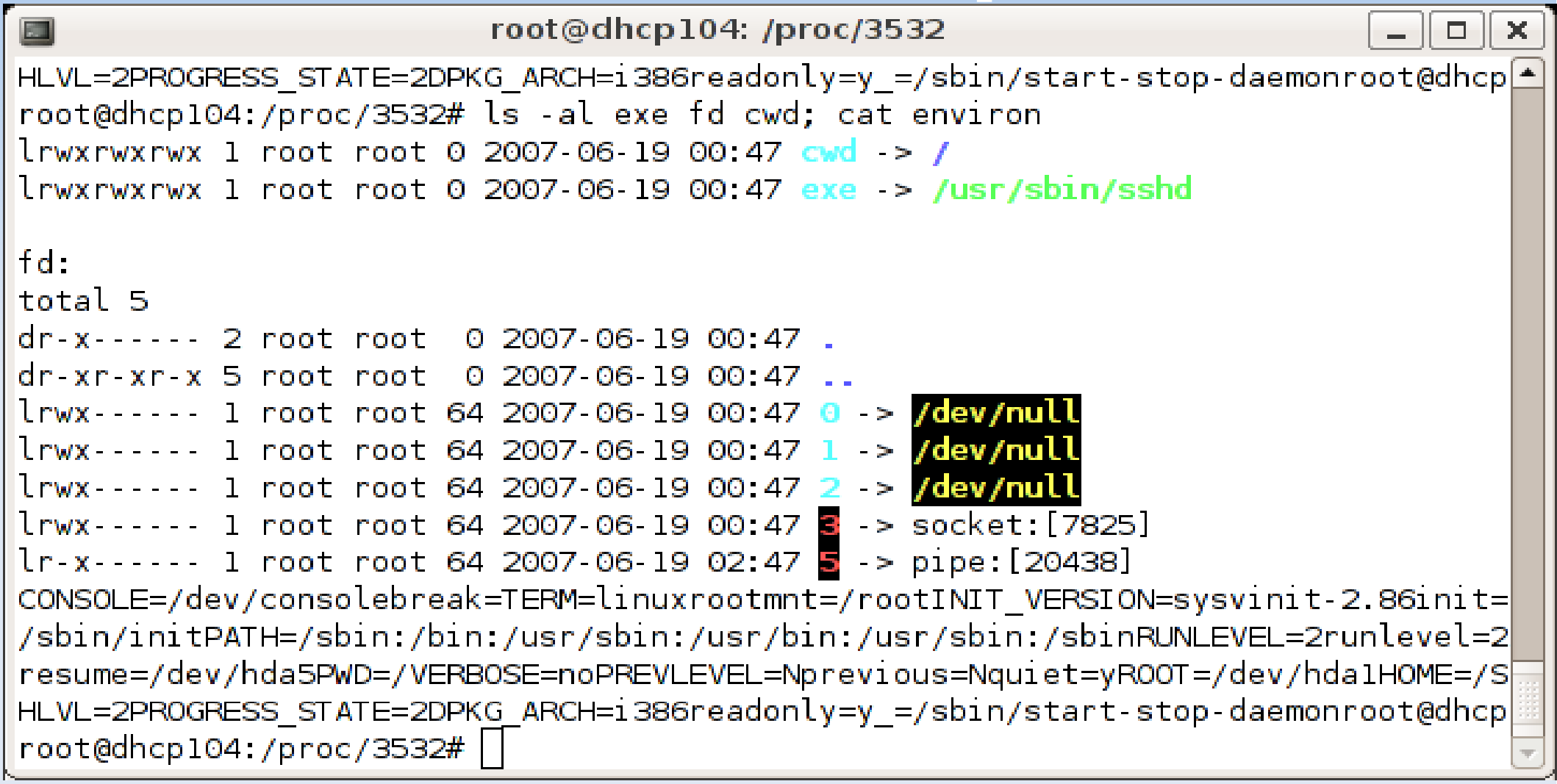
Note the explicit list of ports, 4 services running (great)

The basics: lsof



Looks sane, network connections look reasonable, can we trust the COMMAND? Lets check to see if we should trust PID 3532.

The basics: /proc



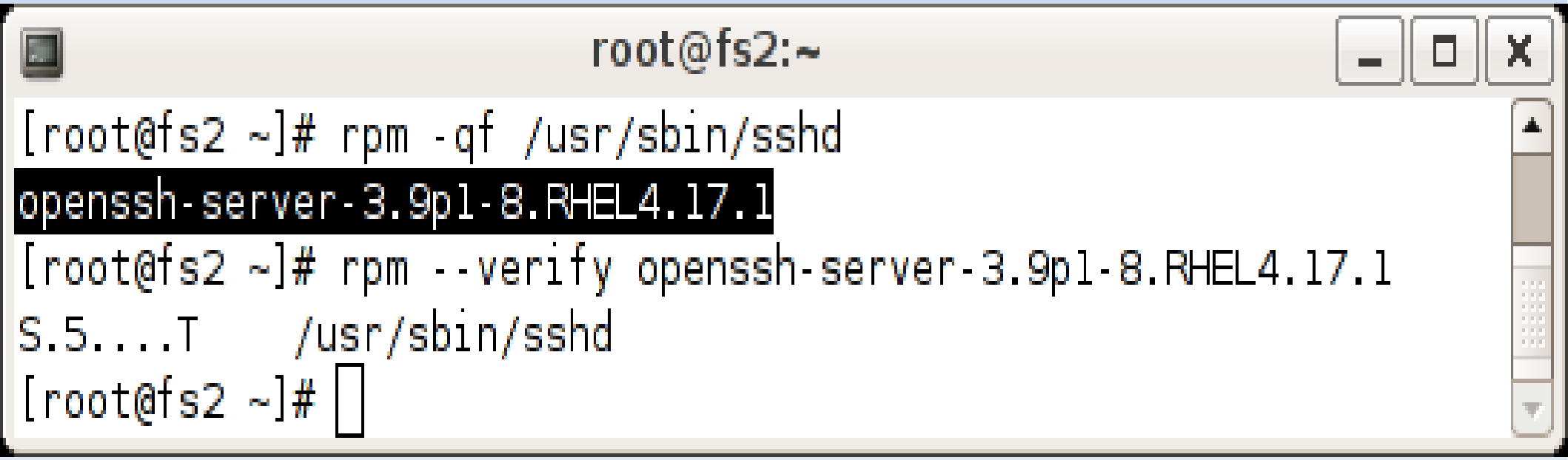
Exe points where expected (and can even recover deleted binaries) FD = file handles, no open files to strange places like /var/tmp/.foo/sniffer
envirion and cwd look reasonable. Is /usr/sbin/sshd valid?

Binary verification (old way)

root@dhcp104: /tmp/check

```
root@dhcp104:/tmp/check#  dpkg-query --search /usr/sbin/sshd
openssh-server: /usr/sbin/sshd
root@dhcp104:/tmp/check# dpkg-query --list openssh-server
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-files/Unpacked/Failed-config/Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name           Version        Description
+++-==============-==============-============================================
ii  openssh-server 4.2p1-7ubuntu3 Secure shell server, an rshd replacement
root@dhcp104:/tmp/check# locate openssh-server | grep deb
/var/cache/apt/archives/openssh-server_1%3a4.2p1-7ubuntu3.1_i386.deb
root@dhcp104:/tmp/check#  cp /var/cache/apt/archives/openssh-server_1%3a4.2p1-7u
buntu3.1_i386.deb .
root@dhcp104:/tmp/check#  ar x openssh-server_1%3a4.2p1-7ubuntu3.1_i386.deb cont
rol.tar.gz
root@dhcp104:/tmp/check# tar xvzf control.tar.gz ./md5sums
./md5sums
root@dhcp104:/tmp/check# cat md5sums | grep "usr/sbin/sshd"
c4ca1492918f2754f885173eb3dbe8ac  usr/sbin/sshd
root@dhcp104:/tmp/check# md5sum /usr/sbin/sshd
22695d4cb5682e5f8f153324063dfb35  /usr/sbin/sshd
root@dhcp104:/tmp/check#
```

redhat binary verify

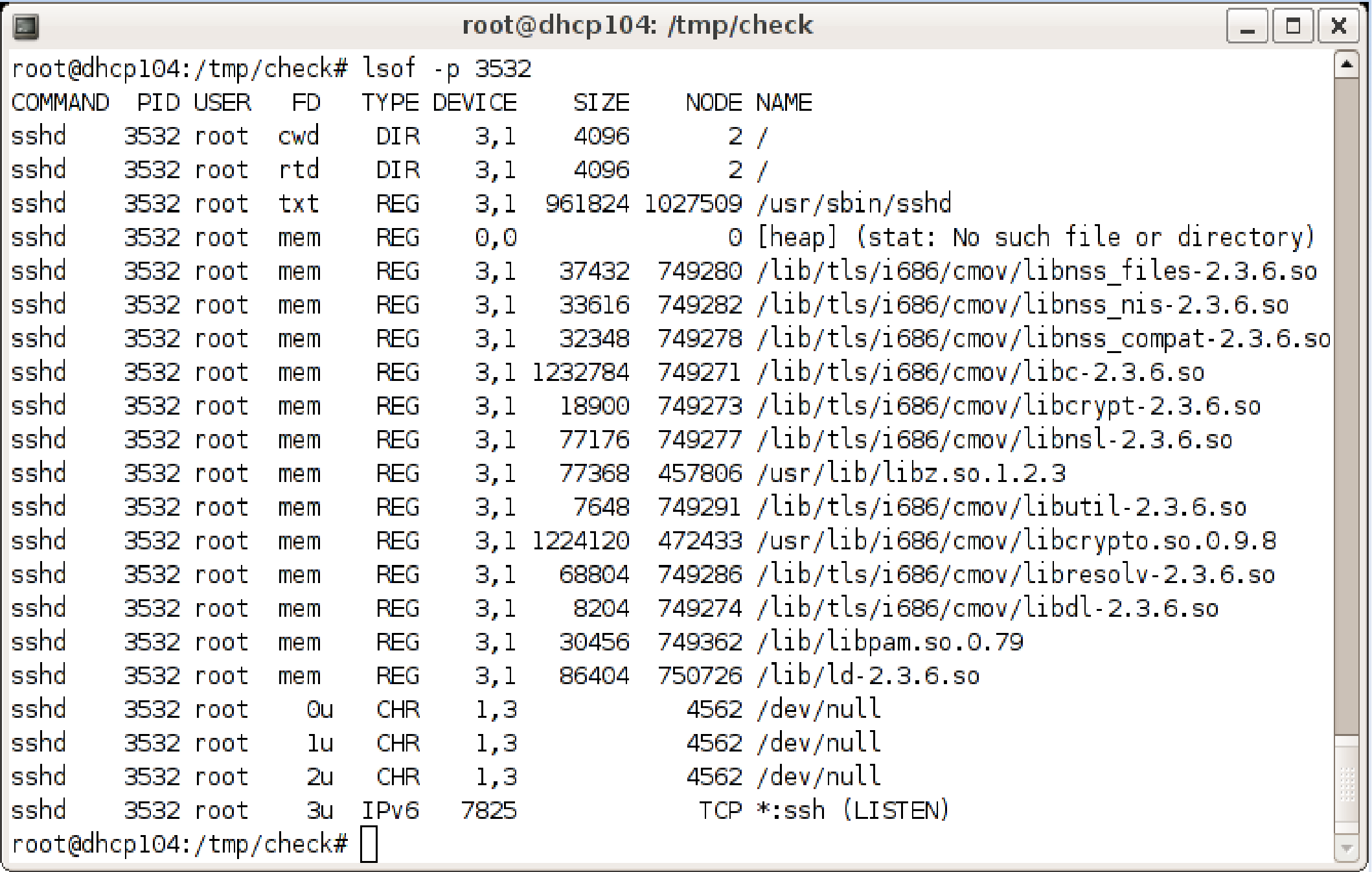


Who are we trusting?

Binary verification part II

- Binary does not verify
- Extremely suspicious
- Painful to verify (someone should automate this)
- Who are we trusting here?
- Looks like a functional trojan
- What evil could it do?
- How would we find out?

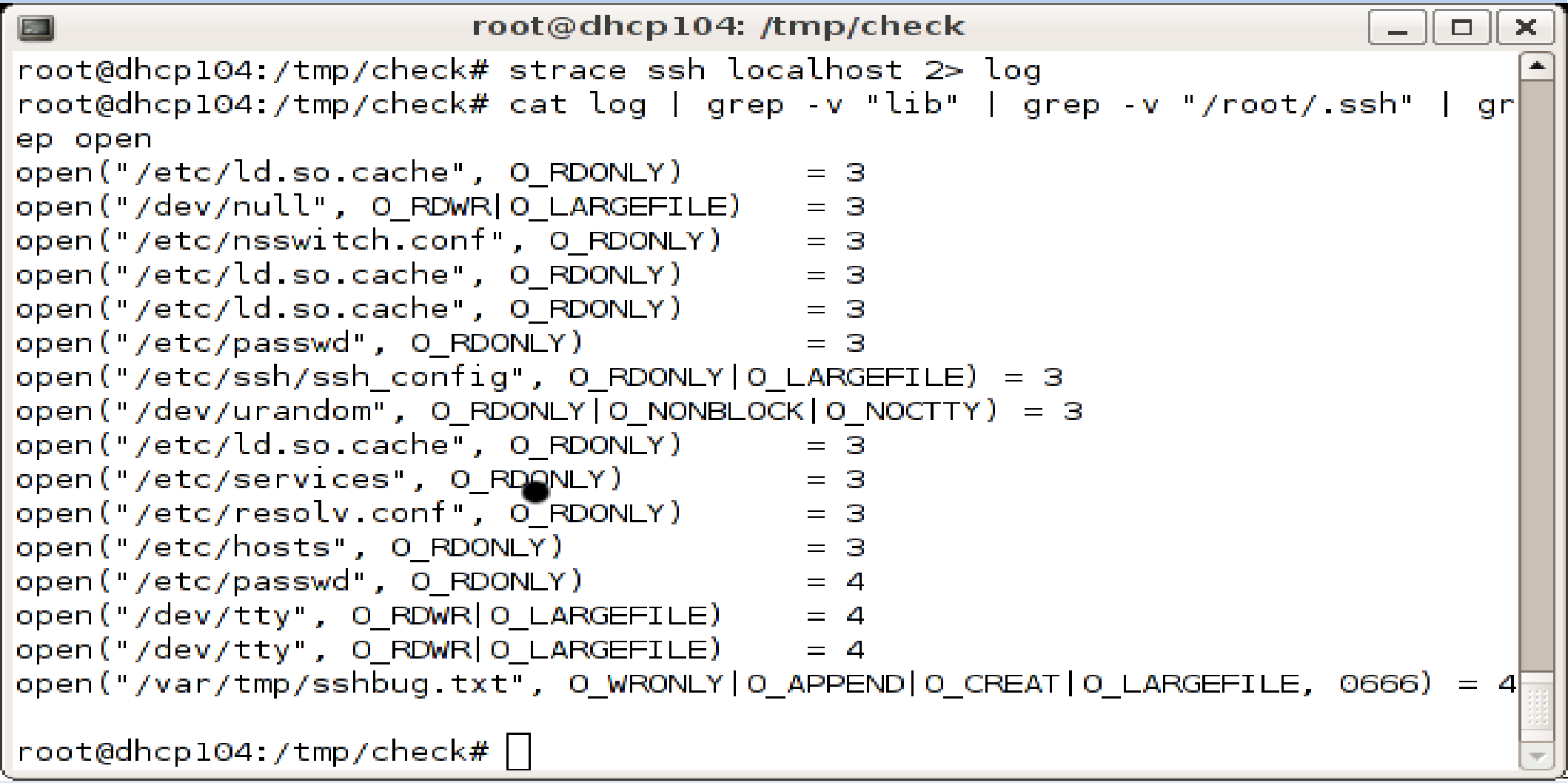
lsof part II


root@dhcp104: /tmp/check
root@dhcp104:/tmp/check# lsof -p 3532
COMMAND PID USER   FD   TYPE DEVICE   SIZE   NODE NAME
sshd   3532 root cwd   DIR   3,1   4096   2 /
sshd   3532 root rtd   DIR   3,1   4096   2 /
sshd   3532 root txt   REG   3,1 961824 1027509 /usr/sbin/sshd
sshd   3532 root mem   REG   0,0   0 [heap] (stat: No such file or directory)
sshd   3532 root mem   REG   3,1   37432 749280 /lib/tls/i686/cmov/libnss_files-2.3.6.so
sshd   3532 root mem   REG   3,1   33616 749282 /lib/tls/i686/cmov/libnss_nis-2.3.6.so
sshd   3532 root mem   REG   3,1   32348 749278 /lib/tls/i686/cmov/libnss_compat-2.3.6.so
sshd   3532 root mem   REG   3,1 1232784 749271 /lib/tls/i686/cmov/libc-2.3.6.so
sshd   3532 root mem   REG   3,1   18900 749273 /lib/tls/i686/cmov/libcrypt-2.3.6.so
sshd   3532 root mem   REG   3,1   77176 749277 /lib/tls/i686/cmov/libnsl-2.3.6.so
sshd   3532 root mem   REG   3,1   77368 457806 /usr/lib/libz.so.1.2.3
sshd   3532 root mem   REG   3,1   7648 749291 /lib/tls/i686/cmov/libutil-2.3.6.so
sshd   3532 root mem   REG   3,1 1224120 472433 /usr/lib/i686/cmov/libcrypto.so.0.9.8
sshd   3532 root mem   REG   3,1   68804 749286 /lib/tls/i686/cmov/libresolv-2.3.6.so
sshd   3532 root mem   REG   3,1   8204 749274 /lib/tls/i686/cmov/libdl-2.3.6.so
sshd   3532 root mem   REG   3,1   30456 749362 /lib/libpam.so.0.79
sshd   3532 root mem   REG   3,1   86404 750726 /lib/ld-2.3.6.so
sshd   3532 root   0u   CHR   1,3   4562 /dev/null
sshd   3532 root   1u   CHR   1,3   4562 /dev/null
sshd   3532 root   2u   CHR   1,3   4562 /dev/null
sshd   3532 root   3u IPv6   7825   TCP *:ssh (LISTEN)
root@dhcp104:/tmp/check#

lsof part III

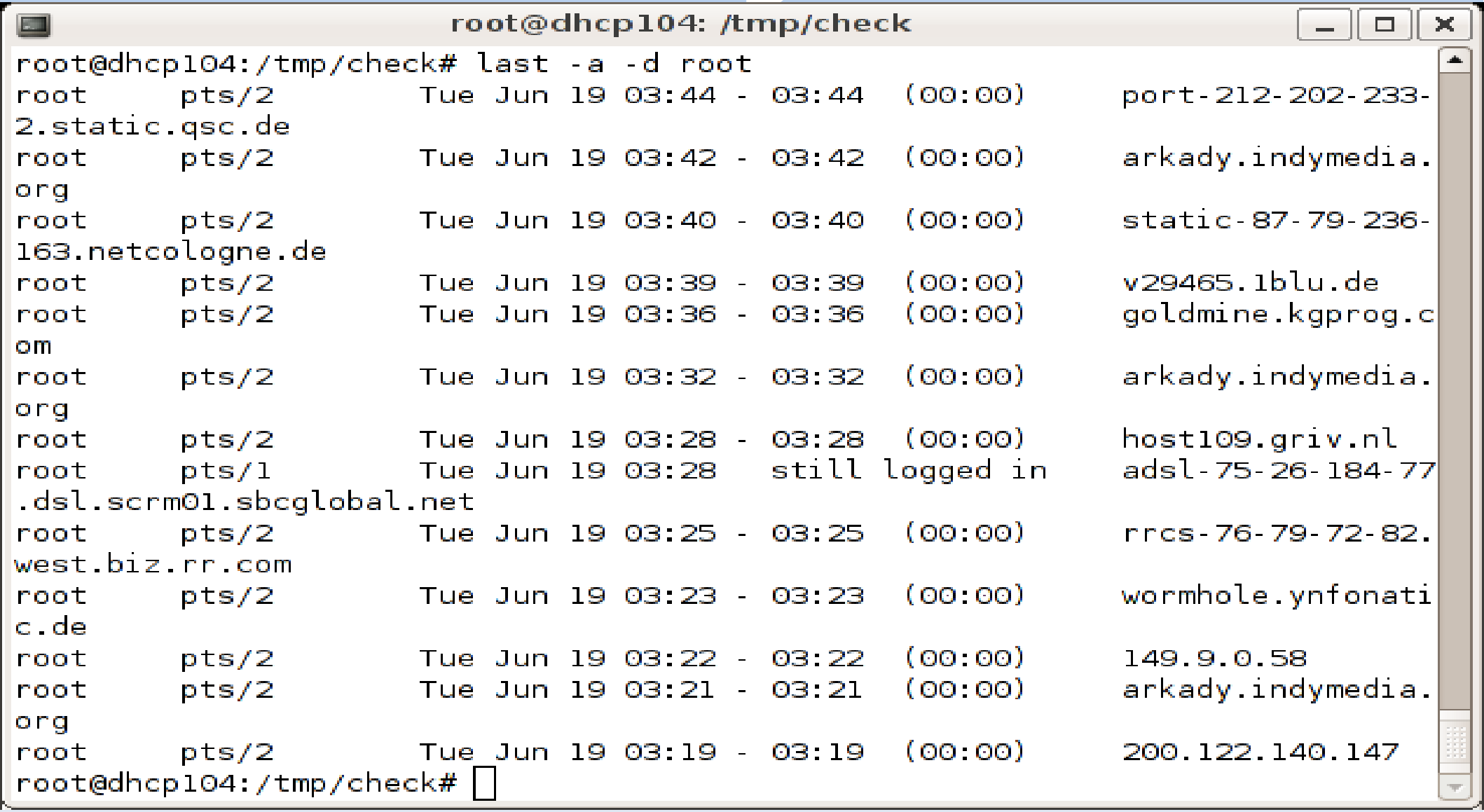
- No strange libraries
- No strange sockets
- No hint of anything strange

Basics: strace



Looks good.... mostly.

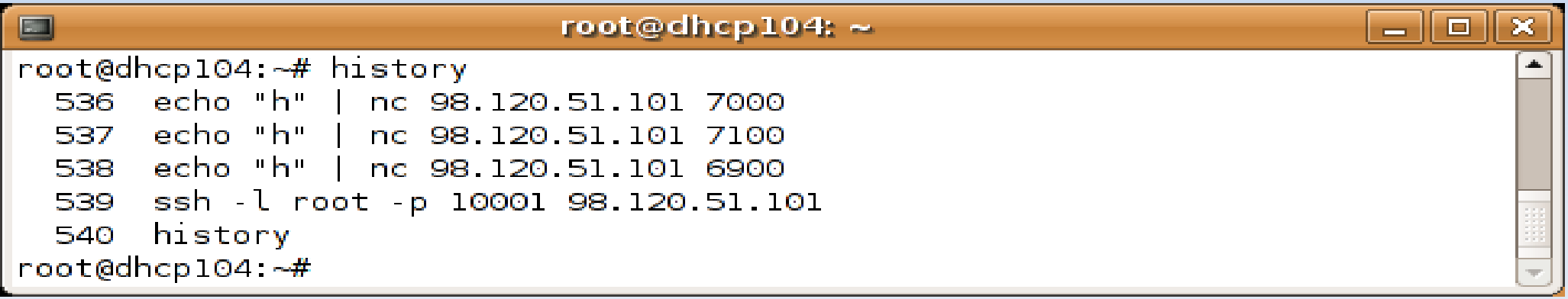
Logs



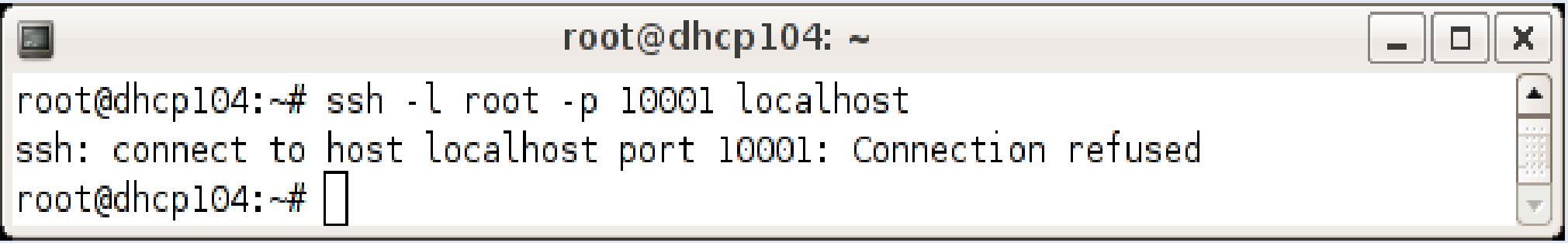
Not a good sign.

Logs part II

Sometimes you get lucky:



I wonder if that works here:



Hrm, what would the echo | nc commands be for?

ssh part II

Lets try that exactly:

```
# echo "h" | nc 128.120.51.101 7000
(UNKNOWN) [128.120.51.101] 7000 (afs3-fileserver) : Connection refused
# echo "h" | nc 128.120.51.101 7100
(UNKNOWN) [128.120.51.101] 7100 (font-service) : Connection refused
# echo "h" | nc 128.120.51.101 6900
(UNKNOWN) [128.120.51.101] 6900 (?) : Connection refused
Last login: Tue Jun 19 04:47:03 2007 from 204.13.236.244
root@dhcp104:~#
```

Bad...

root@dhcp104: ~

```
root@dhcp104:~# (lsof -i :10001 -i :22; nmap -p 22,10001 localhost) &> log

COMMAND  PID USER   FD   TYPE DEVICE SIZE NODE NAME
sshd    3532 root    3u  IPv6   7825       TCP *:ssh (LISTEN)

Starting Nmap 4.03 ( http://www.insecure.org/nmap/ ) at 2007-06-19 04:54 PDT
Interesting ports on localhost (127.0.0.1):
PORT      STATE  SERVICE
22/tcp    open   ssh
10001/tcp closed unknown

Nmap finished: 1 IP address (1 host up) scanned in 0.073 seconds
root@dhcp104:~#
```

Port knocking

- Can't be detected by lsof
- Ignores host firewalls (why do we have host firewalls again?)
- Breaks the mapping between ports and processes (making finding the culprit much harder).
- Can be used to start, stop, or trigger any sequence of packets (UDP or TCP), hitting a sequence of ports, optionally included or excluded with fin, syn, rst, psh, ack or urg flags.

Dom0 ID Advantages

- Relatively straight forward to secure (no applications, users, or external network)
- Very hard to escape from a paravirtualized domU
- Even harder to escape from a hardware virtualized domU
- Can transparently snapshot domU storage and can't be lied to about the FS.

Dom0 ID Disadvantages

- Semantic GAP is substantial
  - DomU sees TCP connections, files, sessions, processes.
  - Dom0 sees packets, blocks, and raw memory.
- Tools are starting to bridge the gap
  - New Xen-3.1 API
  - subvirt (google for king06-1.pdf)
  - Xebek (xen aware sebek descendant)
- Can't easily view DomU kernel structures, processes, data structures, or kernel memory.

Checksum pitfalls

- Need to trust kernel, libraries, and binaries.
- Patching is arduous (auditing changes)
- Need to trust (often local) database
- Database is hard to securely update (is it read only or not?)
- System needs to be secure in the first place. scans can compromise unboxed machines before the first patch finishes.
- Checksums are a moving target

Checksums Done Right

- Known good database (not based on machines preexisting state)
- Run with a trustworthy kernel, libraries, binaries, and database
- With virtualization and snapshots, checksums can be done with zero downtime
- Checksum process invisible to users and attackers.
- LVM snapshots are the low hanging fruit for crossing the semantic gap.
- Patching is easy, less effort helps insure things are done securely.

CDR (Checksums Done Right)

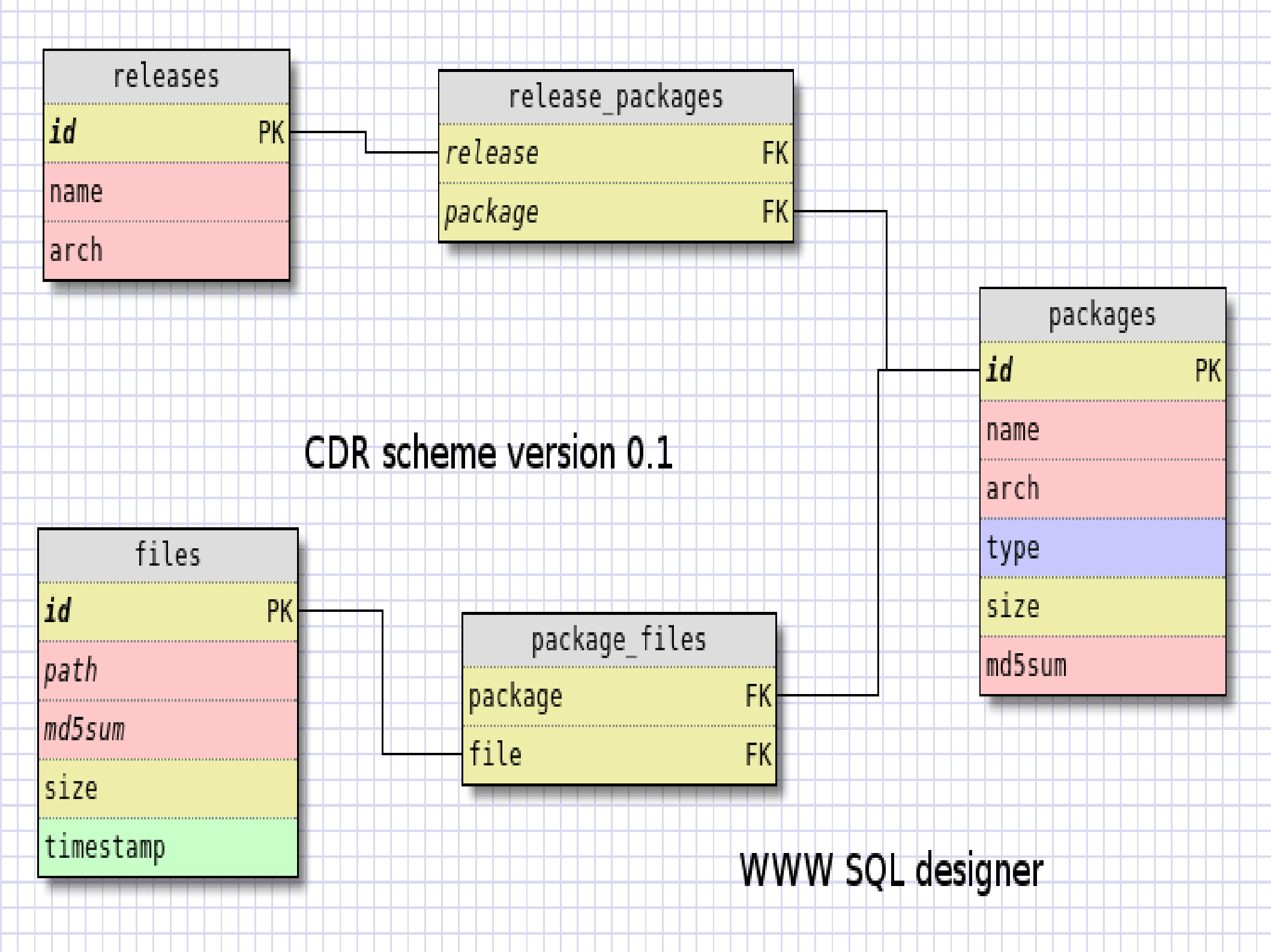
- Built on campus Centos/Ubuntu mirror
- Automatically slurps checksums from all releases, patches, and updates
- Currently 4 million files/checksums, 115k packages (should double soon)
- Client Intended to run on a Dom0 with a DomU LVM Snapshot
- Uses Official (distro provided) checksums (for better or worse)

CDR part II

- Will accept connections from on campus
- If you hammer our server too hard we might ask you to setup MySQL replications (which we have working) and hammer your own server
- Is opensource, will share source code (subversion and trac) to allow other folks to setup similar systems.
- Will allow replication from other UCs

CDR part III

- No near term plans for Microsoft OS's (is it even legal?) Willing to help those interested.
- Clients just stream checksums and filenames over ssh and get a response:
  - Path and checksum are in database and known good (currently this defaults to silent)
  - Path is in database and checksum is bad
  - Path and checksum are unknown.


releases
id PK
name
arch
release_packages
release FK
package FK
packages
id PK
name
arch
type
size
md5sum
CDR scheme version 0.1
files
id PK
path
md5sum
size
timestamp
package_files
package FK
file FK
WWW SQL designer

CDR part IV

- Database is 3.5G (likely to at least double soon).
- Server handles 12,004 requests in 4.5 seconds
- My Ubuntu desktop has around 12k binaries, libraries, and kernel modules
- Single server should easily handle 5k daily full system scans.
- Replicated servers should scale to handles as many clients as needed.

Unintentional benefits

- Change tracking
- Detecting bad disks
- Detecting system admins mistakes
- Detecting Kernel/OS/RAID controller errors.
- RAID scrubbing.
- Tracking side effects from badly behaved applications/installers.

CDR downsides

- Not all packages include MD5sums (should we add them?)
- MD5sum while unintentional collisions are very rare, intentional collisions have been documented. 2^128 is smaller than it used to be.
- OSX and windows still unsupported, the status of package signatures, and checksums for binaries is unknown.

Why not MD5?

```
bill@shell:~/imp/md5$ od --width=24 -x  file0
0000000 31d1 02dd e6c5 c4ee 3d69 069a af98 5cf9 ca2f 87b5 4612 ab7e
0000030 0440 3e58 fbb8 897f ad55 0634 f409 02b3 e483 8388 7125 5a41
0000060 5108 e825 cdf7 9fc9 1dd9 f2bd 3780 5b3c 0b96 d11d 41dc 9c7b
0000110 d8e4 f497 655a d555 7335 c79a ebf0 0cfd 2930 66f1 09d1 8fb1
0000140 2775 797f d530 eb5c e822 baad cc79 5c15 74ed ddcb c55f 6dd3
0000170 9bb1 d80a cc35 e3a7
0000200
bill@shell:~/imp/md5$ od --width=24 -x  file1
0000000 31d1 02dd e6c5 c4ee 3d69 069a af98 5cf9 ca2f 07b5 4612 ab7e
0000030 0440 3e58 fbb8 897f ad55 0634 f409 02b3 e483 8388 f125 5a41
0000060 5108 e825 cdf7 9fc9 1dd9 72bd 3780 5b3c 0b96 d11d 41dc 9c7b
0000110 d8e4 f497 655a d555 7335 479a ebf0 0cfd 2930 66f1 09d1 8fb1
0000140 2775 797f d530 eb5c e822 baad 4c79 5c15 74ed ddcb c55f 6dd3
0000170 9bb1 580a cc35 e3a7
0000200
bill@shell:~/imp/md5$ ls -al file?; sum file?; md5sum file?
-rw-r--r-- 1 bill bill 128 2004-08-18 01:15 file0
-rw-r--r-- 1 bill bill 128 2004-08-18 01:15 file1
31682     1 file0
27570     1 file1
a4c0d35c95a63a805915367dcfe6b751  file0
a4c0d35c95a63a805915367dcfe6b751  file1
bill@shell:~/imp/md5$
```

Why not MD5?

```
bill@shell:~/imp/md5$ od --width=24 -x  file0
0000000 31d1 02dd e6c5 c4ee 3d69 069a af98 5cf9 ca2f 87b5 4612 ab7e
0000030 0440 3e58 fbb8 897f ad55 0634 f409 02b3 e483 8388 7125 5a41
0000060 5108 e825 cdf7 9fc9 1dd9 f2bd 3780 5b3c 0b96 d11d 41dc 9c7b
0000110 d8e4 f497 655a d555 7335 c79a ebf0 0cfd 2930 66f1 09d1 8fb1
0000140 2775 797f d530 eb5c e822 baad cc79 5c15 74ed ddcb c55f 6dd3
0000170 9bb1 d80a cc35 e3a7
0000200
bill@shell:~/imp/md5$ od --width=24 -x  file1
0000000 31d1 02dd e6c5 c4ee 3d69 069a af98 5cf9 ca2f 07b5 4612 ab7e
0000030 0440 3e58 fbb8 897f ad55 0634 f409 02b3 e483 8388 f125 5a41
0000060 5108 e825 cdf7 9fc9 1dd9 72bd 3780 5b3c 0b96 d11d 41dc 9c7b
0000110 d8e4 f497 655a d555 7335 479a ebf0 0cfd 2930 66f1 09d1 8fb1
0000140 2775 797f d530 eb5c e822 baad 4c79 5c15 74ed ddcb c55f 6dd3
0000170 9bb1 580a cc35 e3a7
0000200
bill@shell:~/imp/md5$ ls -al file?; sum file?; md5sum file?
-rw-r--r-- 1 bill bill 128 2004-08-18 01:15 file0
-rw-r--r-- 1 bill bill 128 2004-08-18 01:15 file1
31682     1 file0
27570     1 file1
a4c0d35c95a63a805915367dcfe6b751  file0
a4c0d35c95a63a805915367dcfe6b751  file1
bill@shell:~/imp/md5$
```

CDR plans

- polish support for Ubuntu/Centos x86-64 and ia32
- Possible server side GUI for managing reports and differences.
- Currently in subversion and trac
- Requests?

DomU vs Dom0

- Make snapshot:
  - lvcreate --size 1G --snapshot --name snap /dev/virt/dapper
  - mount /dev/virt/snap /mnt/snap
- Ask CDR:

```
DomU# md5sum sbin/init | ssh -T filecheck@web
DomU#
```

```
Dom0# md5sum sbin/init | ssh -T filecheck@web
sbin/init: Entries exist but the md5sum doesn't match
Dom0#
```

Fun with Rootkits

```
DomU# ./ava I
Adore 1.56 installed. Good luck.
ELITE_UID: 2618748389, ELITE_GID=4063569279, ADORE_KEY=fgjgg
DomU# ./ava
Usage: ./ava {h,u,r,R,i,v,U} [file or PID]
       I print info (secret UID etc)
       h hide file
       u unhide file
       r execute as root
       R remove PID forever
       U uninstall adore
       i make PID invisible
       v make PID visible
DomU# ./ava h /root/hacked
File '/root/hacked' is now hidden.
DomU# ls /etc/*ko
ls: /etc/*ko: No such file or directory
DomU# ls -al /root
total 24
drwxr-xr-x  5 root root 4096 2007-06-19 18:23 .
drwxr-xr-x 21 root root 4096 2007-06-19 18:15 ..
-rw-------  1 root root 8623 2007-06-19 17:56 .bash_history
drwx------  2 root root 4096 2007-06-18 20:42 .ssh
DomU#
```

```
Dom0# ls -al /root /root/hacked/ /root/.urhack3d/ /etc/*ko > log
-rw-r--r-- 1 root        root       13212 2007-06-18 22:12 /etc/enyelkmOCULTAR.ko

/root:
total 36
drwxr-xr-x  5 root        root        4096 2007-06-19 18:35 .
drwxr-xr-x 21 root        root        4096 2007-06-19 18:36 ..
drwxr-xr-x  8 2618748389 4063569279 4096 2007-06-19 18:20 hacked
-rw-r--r--  1 root        root           0 2007-06-18 22:20 OCULTAR_credit_cards
drwx------  2 2618748389 4063569279 4096 2007-06-19 04:29 .urhack3d

/root/hacked/:
total 2096
drwxr-xr-x 8 2618748389 4063569279    4096 2007-06-19 18:20 .
drwxr-xr-x 4         500 users          4096 2007-06-18 22:41 adore-ng
-rw-r--r-- 1 root        root         21751 2007-06-18 22:33 adore-ng-0.56.tgz
drwxr-xr-x 3         500         500    4096 2007-06-18 22:29 enyelkm-1.2
-rw-r--r-- 1 root        root         12758 2007-05-24 10:53 enyelkm-1.2.tar.gz
drwxr-xr-x 5 root        root          4096 2007-06-18 20:46 mood-nt
-rw-r--r-- 1 root        root         36881 2007-06-18 20:44 mood-nt_2.3.tgz
drwxr-xr-x 6 root        root          8192 2007-06-19 00:35 openssh-4.5p1
-rw-r--r-- 1 root        root       1005183 2006-11-16 09:22 openssh-4.5p1_backd
oored.tar.gz
drwxr-xr-x 2 root        root          4096 2007-06-18 23:45 PHoss
-rw-r--r-- 1 root        root         10431 2001-06-24 08:54 PHoss_src.tar.gz

/root/.urhack3d/:
total 28
drwx------ 2 2618748389 4063569279 4096 2007-06-19 04:29 .
drwxr-xr-x 5 root        root        4096 2007-06-19 18:35 ..
-rw------- 1 root        root         339 2007-06-19 04:09 authorized_keys
-rw-r----- 1 root        root         240 2007-06-19 04:29 knockd.conf
-rw-r--r-- 1 root        root        4191 2007-06-19 11:22 knockd.log
-rw-r--r-- 1 root        root        1930 2007-06-19 04:24 sshd_config
```

Take home messages

- Automatic known good patterns are much easier to track than known bad.
- Virtualization allows powerful methods for monitoring a system, use it before the attacker gets the high ground (see subvirt)
- Checksums done right (dom0, snapshots, and a current database) can be easy, fast, cheap, and effective. There's nothing particularly hard about it.

Should you be scared?

```
+----------------------+---------------------+-----------------------------------------+
| path                 | timeat              | hostname                                |
+----------------------+---------------------+-----------------------------------------+
| /~bill/virt/virt.pdf | 2007-08-15 12:56:29 | host-141-116-142-28.ptr.hqda.pentagon.mil |
| /~bill/virt/virt.pdf | 2007-08-15 12:56:29 | host-141-116-142-28.ptr.hqda.pentagon.mil |
+----------------------+---------------------+-----------------------------------------+
2 rows in set (0.00 sec)

mysql>
```

Credits

- Thanks to Scott Beardsley for his help with MySQL, DB Schema, and CDR related python work.
- Adam Getchell for the reference to an excellent virtualization paper: http://www.eecs.umich.edu/virtual/papers/king06.pdf
- Computational Science and Engineering for the support to work on the infrastructure needed for doing clusters right.
- Xen ... material for another talk.

Discussion

- Source available at https://svn.cse.ucdavis.edu/trac/cdr/
- Feedback forms
- Slides at http://cse.ucdavis.edu/~bill/virt

Xen and Intrusion Detection &
Recovery with Linux
by Bill Broadley bill@cse.ucdavis.edu

Thanks for coming!